UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Traitement automatique des langues et génération automatique d'exercices de grammaire

THÈSE

présentée et soutenue publiquement le 25 juin 2020

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Émilie Colin

Composition du jury

Rapporteurs :	Thierry Poibeau	Directeur de Recherches CNRS, Lattice/ENS, Montrouge
	Emmanuel Schang	Maître de conférences, Université d'Orléans
Examinateurs :	Farah Benamara Zitoune	Maîtresse de conférences, Université Paul Sabatier, Toulouse
	Guy Perrier	Professeur émérite, Université de Lorraine, Nancy
	Claire Gardent	Directrice de Recherches CNRS, LORIA Nancy

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

Je voudrais tout d'abord remercier Claire Gardent, ma directrice de thèse. Son professionnalisme, ses conseils, sa connaissance aiguë du domaine m'ont aidée tout autant que guidée durant ces trois années qui pour diverses raisons ont été tumultueuses. Je salue son travail, aussi exigeant que fourni, et me permets d'exprimer le fait que je suis impressionnée par la solidité de son accompagnement sans faille.
J'ai envie d'adresser mes remerciements à tous ceux qui ont contribué à l'élaboration de ce document. L'équipe SYNALP a été un support fondamental, m'a accompagnée avec bienveillance et finesse. En particulier Yannick Parmentier, Christophe Cerisara, Hoa Thien Le, Anastasia Shimorina, Jean-Charles Lamirel, Timothee Mickus, Aurore Coince. Merci à vous pour le temps, l'humanité, l'intérêt et la confiance accordés.
Je remercie ici aussi mon grand frère, Christophe Colin, qui a intégralement relu ce manuscrit, pour m'aider à le corriger avant le grand final, ainsi que Guy Perrier et Emmanuel Schang - membres du Jury - qui ont su déceler coquilles et imperfections résiduelles. J'espère d'ailleurs ce manuscrit clair et précis, tel est son but.

Ces travaux s'inscrivent dans une histoire personnelle, et je remercie aussi Alain Dutech et Yann Boniface, pour leur accompagnement constructif pendant un stage de master. Ce stage, inscrit dans une approche connexioniste qui m'a passionnée, a solidifié mes capacités d'apprentissages. Ils ont solidifié ma confiance en moi pour partir de ce que je suis, non de ce que j'aurais pu imaginer devoir être pour progresser. Je remercie aussi Nazim Fatès pour sa dédiabolisation des modélisations mathématiques de systèmes, qui a été fondamentale. Je remercie aussi Manuel Rebuschi pour m'avoir montré qu'il était possible d'avancer dans la compréhension du monde à des endroits où je ne l'aurais pas envisagé possible, pour m'avoir montré la joie qu'il y avait dans la rigueur. J'aime la rigueur, mais la joie dans la rigueur, c'est tout de même plus festif.

Je remercie Denis Jouvet, pour sa disponibilité et son soutien : membre du comité de thèse, à l'instar de Guy Perrier, mais dans un axe "ressources humaines". Il a suivi ma thèse qui s'est déroulée - scientifiquement et professionnellement, sans anicroche.

Je me vois difficilement clore cette petite partie sans remercier ma famille, qui a éveillé et/ou nourri l'envie de faire et de faire comprendre la science : ma mère, Françoise Berger, mon père, Michel Colin, mes enfants Charlotte Démathieu, Elyo et Luna Lallement, leurs pères respectifs, Emmanuel et David. Leurs respects, confiances, tolérances, appétits propres pour ce monde m'ont été fondamentaux.

Ce manuscrit a une part technique, bien entendu. Derrière les myriades de questions qui le fondent, il y a eu un support logistique issu d'un laboratoire magnifique. J'aurai participé à la vie de ce dernier durant cinq années. Je remercie le loria, et son directeur Jean-Yves Marion. Trois mois de contrat auront pu m'être octroyés pour clore ce travail. Les déplacements et les démarches administratives m'auront continuellement été

facilités, par l'entremise active et saluable de Delphine Hubert, Martine Kuhlmann et Josiane Reffort. Merci à eux quatre. Merci aussi à Caroline, pierre angulaire de la cafétéria.

Je tiens aussi à adresser des pensées chaleureuses à Anne Boyer. Sans son travail et son implication, cette modeste participation au projet METAL n'aurait pas eu lieu, parce que METAL n'aurait pas existé. Son soutien aura été comme un rayon de soleil. Merci à elle et à l'équipe KIWI.

Merci aux amis qui n'ont pas été ici encore mentionnés, que j'ai eu tant de plaisir à côtoyer, avec qui les échanges scientifiques et professionnels ont été importants. Par ordre d'apparition, Laura Perez-Beltrachini, Mariem Mahfoudh, Laura Infante-Blanco, Alejandra Lorenzo, Imen Sayar. Je suis souvent prise dans ma bulle, mais je vous aime.

Ma reconnaissance va aussi à l'ensemble de l'équipe SYNALP. Aux membres du jury non encore mentionnés : Thierry Poibeau et Farah Benamara Zitoune qui ont évalué ce travail avec rigueur, ouverture et bienveillance. Vos retours, ainsi que ceux de Emmanuel Schang et de Guy Perrier, m'ont été enrichissants et ont renforcé mon envie de continuer.
Parce qu'ils m'ont permis de tenir face aux aléas : À Nadine et Francis Marchand, qui m'ont encouragée. À mes autres amis, Patrick et Marie Falgas, François, Sherley Timon et leurs enfants, Marie-Sol Ortola. À Grégory Berchoux-Merlinc. Son soutien en particulier technique m'a permis une soutenance un peu publique grâce à une retransmission sur grand écran, ce dans mon humble salon.
Merci aussi à l'école doctorale, l'IAEM, qui a permis tout cela, en particulier à Souad Boutaguermouchet.

Enfin, je profite de l'occasion pour remercier, en saluant leurs mémoires :
Bertrand Russell, qui nous dit que la vérité des choses est indépendante de nos moyens pour l'atteindre.
Karl Popper, pour qui le critère de la scientificité d'une théorie réside dans la possibilité de l'invalider, de la réfuter ou encore de la tester.
Chercher avec considération pour ces positions me paraît essentiel.

Merci, et, je le souhaite, bonne lecture.

Dans la courbure du temps,
Je dédie cette thèse aux miens, à mes parents, à mes frères, à mes amours, à mes enfants.

Table des matières

Table des figures

Liste des équations

Introduction générale

> Notre langage peut être vu comme une ville ancienne : un dédale de petites rues et de places, de maisons vieilles et neuves, et de maisons avec des adjonctions qui datent d'époques différentes ; et tout cela entouré par une multitude de banlieues récentes avec des rues droites et régulières et avec des maisons uniformes.
>
> Wittgenstein

Cette thèse, intitulée *Traitement automatique des langues et génération automatique d'exercices de grammaire* étudie l'utilisation de méthodes neuronales pour la génération de phrases qui puissent servir de base à la création d'exercices de grammaire.

Table 1, vous pouvez observer deux lexicalisations au contenu sémantique identique mais avec des syntaxes différentes : une juxtaposition de phrases et une relative sujet. Comme l'illustrent ces exemples, être capable de réaliser un contenu sous différentes formes syntaxiques permet de construire des exercices de grammaire : chaque phrase réalisant une structure syntaxique donnée peut être exploitée pour créer un exercice de grammaire permettant de pratiquer cette structure.

Instruction :	Souligne **toutes les relatives sujet**.
Entrée :	John voit une femme qui dort.
Solution :	John voit une femme ***qui dort***.
Instruction :	Combine les deux phrases en utilisant **une relative sujet**
Entrée :	John voit une femme. La femme dort.
Solution :	John voit une femme ***qui dort***.

TABLE 1 – Exercices typiques

Les réseaux de neurones esquissent une promesse, celle de produire des sorties plus naturelles que celle issue de l'usage automatisé de gabarits ou même de grammaires respectant la compositionnalité de la langue. Cette promesse est solide, ainsi que l'ont montré Sutskever et al. (2014); Vaswani et al. (2017); Bahdanau et al. (2014a) : les modèles sont puissants... mais ne sont pas ou pas encore la panacée. Parmi les erreurs classiques de génération avec réseaux de neurones, il y a *absence de mots attendus* et *excès de mots attendus*. Les performances se dégradent avec la taille de la phrase à traiter (Bastings et al., 2017). Qui plus est, les métriques d'évaluation de qualité des

textes générés ne sont pas corrélées pas avec les évaluations humaines (van der Lee et al., 2019).

Tout en étant conscients de ces limites, nous voulons explorer les réseaux de neurones et mettre la génération en langue naturelle au service de l'apprentissage de la grammaire. En grammaire, deux grands courants s'imposent : descriptif et prescriptif. Par exemple, l'accord du participe passé ne va pas de soi. Clément Marot l'a introduit au XVIème siècle (italianisation de la langue), Abbé d'Olivet a défini au XVIIIème la règle actuelle, et elle a été imposée à l'école au XIXème...
Pour le circonflexe, installé relativement solidement, le choix est d'étudier l'histoire. Cette histoire est bien visible (hôpital/hospitalier).
Pour le reste, l'approche est relativement dogmatique, et participe à l'établissement d'une langue commune, normée, à l'échelle de notre territoire.
Le contrôle grammatical est essentiel, d'abord pour limiter les erreurs, les détecter, mais aussi pour imposer des normalisations. À cette heure, les cours de grammaire ne sont pas des cours d'histoire ou de linguistique, mais donnent à apprendre une vue abstraite et simplifiée de la langue et de sa structure. Nous sommes en traitement automatique des langues, et dans cette thèse explorons les ressources et techniques au service de cette vue abstraite.

Une thèse appliquée

Cette thèse est une thèse appliquée parce qu'elle se propose de participer au développement de technologies innovantes pour l'apprentissage de la grammaire française. Ces technologies visent l'assistance aux enseignants et la personnalisation des exercices : idéalement la génération automatique d'exercices en fonction des besoins de l'élève dans le cadre de la progression gérée par l'enseignant. Metal[1], projet e-Fran, nous permet de fournir le travail ici présenté. e-Fran vise le rapprochement de l'Éducation et de la Recherche. Cette dynamique respecte à la lettre le plan e-Fran :

> « Déployés à l'échelle des territoires, ces projets qui sont de nature à transformer l'École, e-FRAN expérimente de nouvelles manières d'enseigner et d'apprendre, à partir de dispositifs pédagogiques et numériques innovants dans un cadre scientifique rigoureux. »

source : https://www.gouvernement.fr/e-fran-l-ecole-change-avec-le-numerique

Une thèse exploratoire

L'exploration de la génération de texte en langue naturelle est le fondement scientifique du travail ici présenté. Perez-Beltrachini et al. (2012) ont démontré la faisabilité de ce sujet d'intérêt public qu'est la génération automatique d'exercices pour l'apprentissage du français. Dans le cadre du projet INTERREG IV Allegro, elles ont permis l'exploitation de leur travail sur une plate-forme en ligne. Le travail en question requiert

1. Metal (Modèles et Traces au service de l'Apprentissage des Langues) : http://metal.loria.fr/, consulté le 6 février 2020

une grammaire rédigée manuellement par des linguistes, pré-requis coûteux en terme de temps, exigeant en terme de compétences, y compris sur le long terme (évolutivité complexe).

Parallèlement, la génération sur support neuronal est en plein développement. L'apprentissage profond, qui regroupe un ensemble de méthodes d'apprentissage automatique a fait ses preuves en classification.

En génération automatique de texte - ici langue naturelle sous contraintes syntaxico-sémantiques - le possible est ouvert, mais :

- des erreurs se produisent et sont difficiles à détecter,
- les métriques sont problématiques,
- la création de ressources est en plein essor, mais reste loin d'être suffisante.

Les réseaux de neurones tendent à joindre méthodologiquement des thématiques variées, en génération, et plus généralement en traitement automatique des langues tels le développement d'outils pédagogiques, l'aide à la décision, la traduction automatique, la synthèse et l'explication automatisées.

Ces domaines s'appuyaient jusqu'alors sur des méthodes différentes (systèmes experts (faits, règles, inférences), statistiques, ...) ou n'en étaient qu'à des stades peu avancés.

1 Objectifs et résultats

Cette thèse n'a pas eu vocation à créer un exerciseur ni à faire évaluer des exercices par des enseignants, des apprenants.
Pour qu'existe un outil capable de gérer la production automatique d'exercices de grammaire, il est nécessaire de générer des phrases de bonne qualité, sémantiquement correctes et dont nous contrôlons la syntaxe : notre objectif est d'étudier la faisabilité technique, de la mettre à l'épreuve (qualité, diversité des phrases), ce sur du français, langue moins bien dotée que l'anglais. Cet objectif de recherche a évolué sur deux étapes. D'abord, nous avons généré des phrases en anglais à partir de données RDF (voir page 36) et du corpus WebNLG (Gardent et al. (2017)), ensuite nous avons généré des phrases en français à partir de données textuelles.

À noter pour les évaluations, quand nous utilisons BLEU (Papineni et al., 2002), il s'agit systématiquement d'un calcul BLEU-4, qui compare les successions de mots quatre à quatre.

1.1 Génération à partir de triplets RDF.

En vue de générer des paraphrases syntaxiquement distinctes, nous avons formulé la tâche de génération comme une tâche de prédiction conditionnée à la fois par une entrée I et une contrainte syntaxique k.

Pour ce premier travail, nous avons utilisé le corpus WebNLG (Gardent et al., 2017). Ce corpus, à l'époque disponible uniquement en anglais, associe à des triplets RDF connectés des lexicalisations mais pas d'information syntaxique. Nous avons donc procédé à son enrichissement et créé, à partir des instances RDF/Texte du corpus WebNLG, des

Entrée	Sortie
I : RDF et/ou Texte, k : contrainte syntaxique.	une phrase qui réalise la contrainte entrée.

TABLE 2 – Modèles de la première expérience, vue d'ensemble

instances de la forme (RDF, k, Texte) où k est une structure syntaxique réalisée dans Texte. Ces instances forment un corpus de 4099 entrées pour 26 403 textes.

À partir de ce corpus enrichi (D2T, Data vers Texte), nous avons proposé un premier modèle neuronal pour la génération de textes sous contrainte syntaxique, un modèle qui apprend à générer des phrases réalisant à la fois le contenu sémantique véhiculé par les triplets RDF (I est alors du RDF) et la contrainte syntaxique (k) donnée en entrée.

Nous avons par ailleurs créé, à partir de ce premier corpus et en prenant les lots de triplets RDF comme pivots, trois autres corpus pour la génération de phrases sous contrainte syntaxique :

- un corpus (T2T) de triplets (t, k, t') où t est une paraphrase de t' contenue dans WebNLG (t et t' sont associés, dans WebNLG, au même lot de triplets RDF). Nous appuyant sur ce corpus, nous apprenons un modèle de génération de textes à partir de textes, un modèle de paraphrasage.
  Ce corpus compte 9 968 entrées pour 8 703 textes.

- un corpus (T+r2T) de tuplets de la forme (r, t, k, t') tel que WebNLG contient les instances (R, t) et $(\{R, r\}, t')$. En d'autres termes, t' est un texte verbalisant le lot de triplets RDF $\{R, r\}$ ou, de façon équivalente, une verbalisation du texte t augmenté du texte verbalisant le triplet RDF r. À partir de ce corpus, nous apprenons un modèle de génération de textes qui, à partir d'un texte et d'un triplet RDF, étend ce texte avec la verbalisation du contenu encodé dans le triplet RDF.
  Ce corpus compte 22 386 entrées pour 6 758 textes.

- un corpus (T-r2T) est similaire au second mais dans ce cas, le texte cible verbalise la soustraction du triplet RDF du texte de départ.
  Ce corpus compte 18 904 entrées pour 8 063 textes.

La Table 3 montre des exemples d'entrée/sortie pour les quatre corpus créés.

Étendre, réduire, produire, et transformer une phrase sous contrainte a impliqué dans notre expérience une approche encodeur/décodeur (expliquée page 29) avec mécanisme d'attention (expliqué page 32), quatre modèles résumés Table 2.

Nous avons enfin mené, au-delà d'une évaluation automatique (ex : BLEU-4), une évaluation humaine sur 150 textes, ce qui nous permet d'évaluer la diversité syntaxique.

La Figure 1 montre que combiner les modèles offre une grande diversité (analyse humaine). Nous vous présentons des résultats de métriques automatiques Table 4.

D2T$_{syn}$	RDF, $k \Rightarrow T_o$ avec $mg(T_o) =$ RDF, $k \in K(T)$	
	RDF	{ (Aarhus Airport operatingOrganisation Aktieselskab), (Aarhus Airport runwayLength 2776), (Aarhus Airport runwayName "10L/28R") }
	k	Coordination
	T_o	*Aarhus Airport is operated by the Aktieselskab. Its runway name is "10L/28R" and the length is 2776.*
T2T$_{syn}$	$T_i, k \Rightarrow T_o$ avec $mg(T_i) = mg(T_o), k \in K(T)$	
	T_i	*Afonso Pena International Airport is located in São José dos Pinhais. The runway length is 2215 and the runway name is 11/29*
	k	Transitive
	T_o	*Afonso Pena International Airport in São José dos Pinhais has a runway known as 11/29 with a length of 2215.*
T-r2T$_{syn}$	$T_i, k, t \Rightarrow T_o$ avec $mg(T_o) = mg(T_i) \cup \{t\}, k \in K(T)$	
	T_i	*Adolfo Suárez Madrid–Barajas Airport is located in Madrid, part of the Community of* ***Madrid in Spain*** *where the leader party is Ahora Madrid.*
	k	possessif
	RDF	{ (**Madrid country Spain**) }
	T_o	*Madrid is part of Community of Madrid whose leader party at Madrid is the Ahora Madrid. The Adolfo Suárez Madrid–Barajas Airport is located there.*
T+r2T$_{syn}$	$T_i, k, t \Rightarrow T_o$ avec $mg(T_i) = mg(T_o) \cup \{t\}, k \in K(T)$	
	T_i	***Al Asad Airbase*** *is located at "Al Anbar Province, Iraq" and operated* ***by the United States Air Force****. The base 's runway called "08/26" and 3990 meters long.*
	k	Relative sujet
	RDF	{ (**Al Asad Airbase operatingOrganisation United States Air Force**) }
	T_o	*Al Asad Airbase (in "Al Anbar Province, Iraq"), has a runway named "08/26" and a runway that is 3990 metres long.*

TABLE 3 – Tâches de génération syntaxiquement contrainte (D2T : données vers texte, T2T : texte vers texte, T+r2T : expansion , T-r2T : simplification, $mg(T)$: représentation d'un texte T, T_i, T_o : Texte$_i$ en entrée, Texte$_o$ en sortie (*input/output text*), k : contrainte syntaxique, RDF : représentation sémantique structurée en un ensemble de triplets RDF, $mg(T)$: représentation d'un texte T, $K(T)$: contraintes syntaxiques réalisées par le texte T), X_{syn} : modèle sous contrainte syntaxique

Modèles combinés :	
haut niveau linguistique BLEU-4 [2]	62.87 (6.21 et 4.71 pour les modèles de base).
haute diversité syntaxique :	6.3 phrases syntaxiquement distinctes par sens,
haute adéquation syntaxique :	91% de couverture syntaxique, contrainte k pour I correctement identifiée

TABLE 4 – Résumé résultats modèles combinés : première expérience

2. métrique mesurant la qualité d'un texte (Papineni et al., 2002) sur une échelle de 0 (pire) à 100 (meilleur).

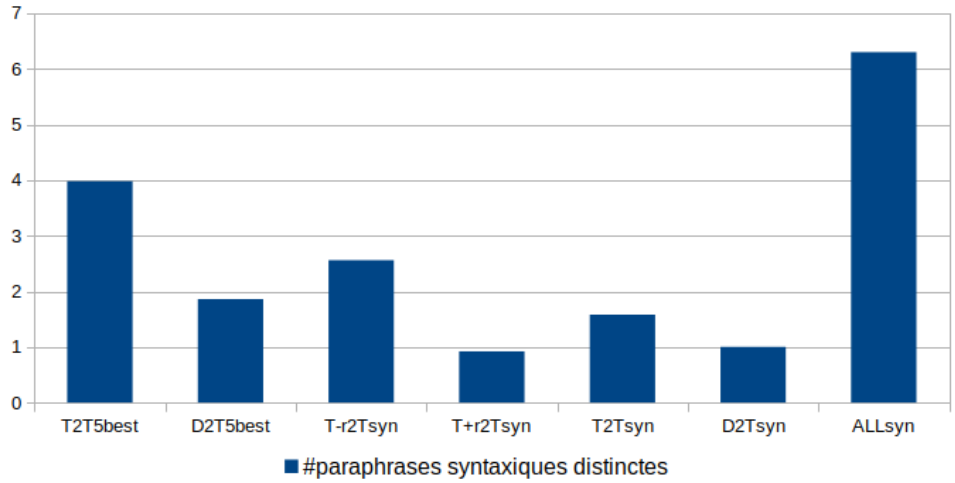


FIGURE 1 – Diversité syntaxique évaluée manuellement sur 150 textes

Diversité : nombre moyen de phrases syntaxiquement distinctes par sens,
T2T : texte vers texte, D2T : Données vers Texte,
X_{syn} : modèle sous contrainte syntaxique, ALL$_syn$ tout modèle syntaxiquement contrôlé,
X_{best} : modèle sans contrainte syntaxique, dont la diversité est obtenue par l'usage de *beamsearch* : les cinq phrases différentes les plus plausibles statistiquement générées par entrée.

Ce travail est relié à l'existence de WebNLG, corpus en langue anglaise, or l'objectif du projet est de travailler sur le français. Dans ce cadre, l'idéal serait selon nous qu'un enseignant puisse un jour fournir un texte, et qu'un outil en fournisse plusieurs reformulations en indiquant sa composition syntaxique (ex : présence et position d'une relative sujet). Ce résultat serait directement exploitable par un exerciseur.

1.2 Génération à partir de représentations sémantiques.

Pour étendre cette approche au français, une possibilité serait de traduire les textes anglais de WebNLG en français et de créer ainsi un corpus d'apprentissage sur lequel la même méthode pourrait être appliquée : le travail précédent devrait pouvoir être applicable au français. Cette expérience aurait représenté un intérêt scientifique réduit. Par contre, travailler sur du texte brut - en français - nous a paru beaucoup plus riche. Nous nous sommes attelée à la production automatisée de représentations sémantiques dans le but de les exploiter en tant que support à la génération de textes en français.

Pour cette seconde partie, nous avons donc effectué nos recherches autour de l'analyse automatique de ressources du domaine public (œuvres littéraires). Être capable de générer automatiquement une représentation sémantique du contenu de phrases permettrait - théoriquement - de pouvoir générer des phrases contenant tout ou partie de l'information contenue dans le texte initial, selon que l'on fournit au réseau la représentation

sémantique complète ou partielle.

Nous avons procédé à l'analyse automatique de centaines de milliers de phrases issues de la littérature (domaine public [3]). Chaque phrase a été soumise à trois *parseurs* : *Grew*[4] (Guillaume et al., 2012), *Talismane*[5] and le *Stanford Dependency Parser*[6]. Les phrases dont les analyses concordent et qui font moins de soixante-dix mots ont été conservées : des phrases trop longues n'étant guère exploitables par les réseaux de neurones (Bastings et al., 2017). En croisant les analyses avec les descriptions verbales du LVF (Les Verbes Français, Dubois et Dubois-Charlier (1997)), nous construisons l'information illustrée Figure 2.

Ce travail a impliqué un développement d'outils fonctionnant en cascade : téléchargement, lecture de données aux encodages variés, analyse en dépendance multi-parseurs, recoupement automatisé des concordances inter-parseurs, détection des temps, des accords, analyse sémantique profonde, construction d'une représentation et exploitation de celle-ci par un modèle neuronal pour générer du texte.

Nous repérons dans les phrases ce qui est structurant, fonctionnel, et obtenons une architecture lexicale construite autour des verbes. Les entités lexicales identifiées sont remplacées par une étiquette les anonymisant. La Table 5 illustre cette anonymisation et liste les informations obtenues de la première phrase présentée Figure 2.

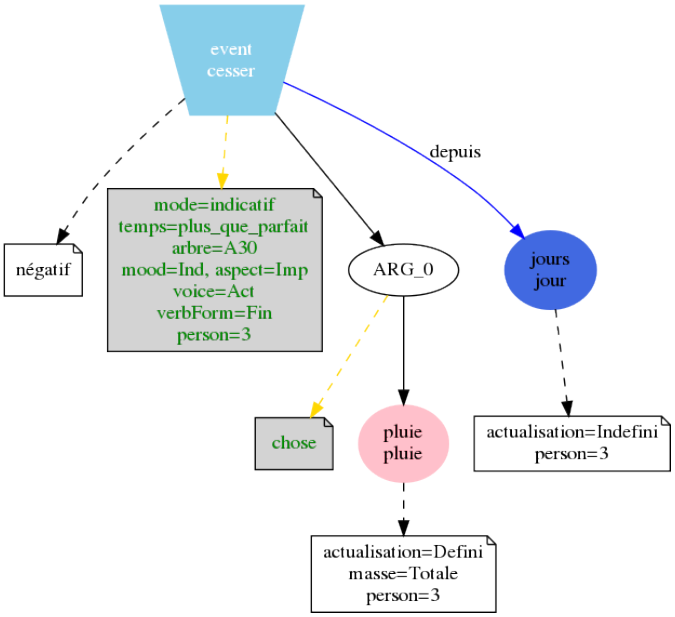
(a) La pluie n' avait pas cessé depuis plusieurs jours .
(b) La argevent_1 n' avait pas event_1 depuis plusieurs argevent_2 .

étiquette	lemme	nature	genre	nombre	pers.	mode	temps	arg_0	traits
argevent_1	pluie	nom	fem	sing	3				défini
event_1	cesser	verbe				indicatif	plus que parfait	argevent_1	
argevent_2	jour	nom	masc	plur	3				indéfini
depuis	(event_1, argevent_2)								

TABLE 5 – Représentation de sens et anonymisation : seconde expérience

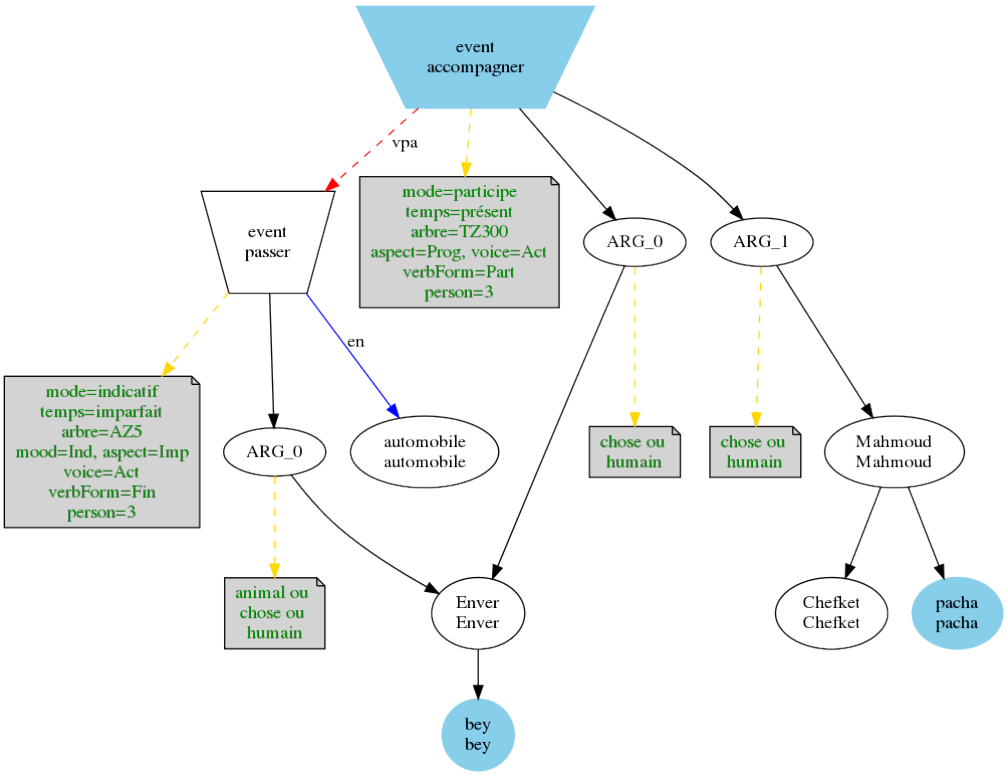
3. gutenberg : http://www.gutenberg.org/
4. http://grew.fr/
5. http://redac.univ-tlse2.fr/applications/talismane.html
6. https://nlp.stanford.edu/software/lex-parser.shtml

La couleur bleue ou rose indique un genre attribué masculin ou féminin, claire au singulier, foncée un pluriel.



La pluie n'avait pas cessé depuis plusieurs jours.
source : Mémoires du général baron de Marbot, Baron de Marbot

Analyse automatique de texte brut : exemple 1



Enver bey passait en automobile, accompagnant Mahmoud Chefket pacha.
source : L'Illustration, No. 3649, 1 Février 1913, Auteur inconnu

Analyse automatique de texte brut : exemple 2

FIGURE 2 – Analyse automatique de texte brut : deuxième expérience

Nous avons proposé une méthode permettant une obtention automatisée de représentations du sens de textes. Nous avions dans le corpus WebNLG repéré les entités non prédicatives au sein des textes pour réaliser une anonymisation poussée. Nous avons reproduit la même opération, considérant les objets/sujets des arguments verbaux, les littéraux modifiant éventuellement ceux-ci, ainsi que les prédicats - nos verbes.
Cette opération aura permis d'évaluer et de démontrer qu'une anonymisation complète des verbes et éléments non fonctionnels permettait d'améliorer nettement les résultats. Les éléments non délexicalisés étaient les mots fonctionnels, les adverbes et les auxiliaires. La Figure 2 illustre une délexicalisation. Pour permettre un fléchissement correct des formes et une re-lexicalisation, nous avons exploité des traits.

La Table 6 présente la part principale de nos évaluations, sachant que nous avons aussi mené une analyse qualitative retrouvable en annexe. Le rappel - qui marque la présence des mots attendus non présents, indique que 13% des phrases générées ne sont pas relexicalisables, et le BLEU-4 sur notre modèle est de 68.31, soit 3.96 de plus que sur données non anonymisées.
Pour donner un ordre d'idée, les BLEU-4 sur les réalisations de surface réalisées dans le cadre du E2E challenge (Dušek et al., 2020) allaient de 42.02 à 66.19 sur base de test [7] avec une médiane de 60.15 pour les 21 modèles en compétition.

Modèle	BLEU-4	C-R	FW-F1
MR→texte	64.35	0.59	0.906
Dual (Test :Anon.)	**68.31**	**0.87**	**0.937**

Table 6 – Seconde expérience, résultats

MR→texte : de la représentation de sens vers du texte,
Dual (Test :Anon.) : entraînement sur données anonymisées et non anonymisées testé sur données anonymisées,
C-R : rappel sur le vocabulaire lexical attendu (sous forme anonymisée ou non),
FW-F1 : F-mesure sur le vocabulaire fonctionnel,

BLEU-4 : évaluation linguistique de 0 (pire) à 100 (meilleure) VS 0 à 1 pour C-R et FW-F1.

L'évaluation humaine a consisté en une analyse de cinquante textes générés prélevés au hasard. 34% des phrases sont identiques à la référence, 78% sont correctes grammaticalement (voir annexes E. 1). Le fléchissement des verbes (temps et accords) était sans erreur, un souci de non conformité déterminant/nom commun a été trouvé, mais lié à une mauvaise analyse automatique et donc à un mauvais trait associé à l'élément correspondant dans la représentation de sens.

Perspectives.

La Table 7 montre un exemple de représentation de sens (b) d'une phrase (a). L'objectif de recherche de génération de (a) à partir de (b1) + (b2) fait logiquement suite

7. E2E : 42061 textes (t) et 4862 représentation de sens (MR) pour l'entraînement, test sur 630 MR/4693t ,14.68 *tokens* par phrases)
nous : 392486 phrases et MR pour l'entraînement, 49062 pour le test avec des phrases de 13.55 *tokens* en moyenne.

à notre premier travail. En effet, une représentation sémantique associée à des textes permet des jeux de génération, il est possible de mettre en place une série de modèles répondant à notre projet global : générer sous contrainte syntaxique.
Si, dans la continuité de notre première expérience, nous générons à partir d'un jeu sur les représentations b (ex : ((b1) + (b2)) ou (b1) ou (b2)), associons des contraintes aux sorties, nous ouvrons la porte à reproduire certains modèles de la première partie (RDF, extension, réduction).

(a0.1)	Norbert qui mangeait une pomme avait consulté sa montre .			
(a0.2)	Norbert avait consulté sa montre . Il mangeait une pomme .			
(a0.3)	Sa montre avait été consultée par Norbert mangeant une pomme .			
(a0.3)	Norbert, mangeant une pomme, avait consulté sa montre.			
(a1)	Norbert avait consulté sa montre .			
(a2)	Norbert mangeait une pomme .			
(b1)	consulter	arg_0	norbert	(+ traits)
	consulter	arg_1	montre	(+ traits)
(b2)	manger	arg_0	norbert	(+ traits)
	manger	arg_1	pomme	(+ traits)

TABLE 7 – Représentation de sens : perspectives

Entrée I : représentation de sens anonymisée ou non.
Sortie O : une phrase qui réalise la représentation.

TABLE 8 – Modèles de la seconde expérience, vue d'ensemble

(a)	Depuis une heure, Norbert avait consulté sa montre .
(b)	Depuis une mod_1, suj_1 avait verb_1 sa obj_1 .

TABLE 9 – Représentation de sens et anonymisation : perspectives

Note sur les réseaux de neurones et leur exploration

Nous n'avons pas choisi d'explorer tous les réseaux de neurones. Bien des expériences ont été menées à ce niveau. Les réseaux convolutifs sont plus exploités pour la classification (Hu et al., 2014; Er et al., 2016; Poria et al., 2016; Qin et al., 2016), et bien entendu pour la génération à partir d'images (Ma et al., 2016). Nous nous sommes limitée aux réseaux de neurones récurrents, plus précisément séquence vers séquence qui correspondaient fonctionnellement à nos besoins. Nous avons centré nos approches d'abord (Chapitre 1) sur la diversification des types d'entrées (texte, représentations logiques, hybridations), puis (Chapitre 2) sur le texte brut et sa représentation sémantique.

2 Plan de la thèse

Ce document est divisé en deux grandes parties, chacune étant à son tour subdivisée en deux sous-parties. La première partie situe nos travaux dans leur contexte scientifique (Génération de textes et Méthodes neuronales) ; il s'agit de l'état de l'art, ce qui permet de tirer les fils de la recherche en linguistique computationnelle, ceux avec lesquels nos travaux se tissent. La deuxième partie explore et expose nos recherches : génération sous contrôle syntaxico-sémantique à partir d'un corpus parallèle (représentations RDF [8] et lexicalisations), puis constitution automatisée d'un corpus parallèle (représentations sémantiques/lexicalisations) et apprentissage d'un modèle de génération de phrases en français à partir de représentations sémantiques.

Première partie

Méthodes neuronales

La première partie de notre état de l'art est consacrée aux réseaux de neurones, Chapitre 1.

Les premiers modèles neuronaux sont les plus simples, ce sont eux qui ouvriront le bal : racines des réseaux de neurones et perceptrons : naissance et fonctionnement global (Sections I. 1. 1 et I. 1. 2). Le perceptron initial ne permettait absolument pas d'espérer générer des phrases. Ne serait-ce que pour apprendre à classifier, à organiser l'information, il aura d'abord fallu mettre en place la rétro-propagation du gradient. À ce moment là, l'entropie devint une alliée (Section I. 1. 3). Le traitement non linéaire des données ne fut possible qu'avec les réseaux de neurones récurrents. Ceux-ci permettent le traitement d'entrées de taille variable dont l'ordonnancement temporel fait sens (Section I. 1. 4). Néanmoins, les réseaux récurrents sont fragiles pour gérer les dépendances de long terme. Le LSTM (*Long Short Term Memory*) et Gru (*Gated recurrent unit*), sa version simplifiée, prennent finement en compte les données historiques (Section I. 1. 4, I. 1. 5). Nous arriverons enfin à l'architecture encodeur-décodeur, une architecture adaptée au traitement des séquences (une phrase : une séquence de mot, de *tokens* [9]) (Section I. 1. 6), et clôturerons cette partie sur le mécanisme d'attention. Avec ce mécanisme, les éléments de séquence déjà traités peuvent être exploités en temps réel (Chapitre I. 1. 7).

Traitement des langues naturelles : génération

La génération en langue naturelle qui transforme une entrée a en un texte b prend plusieurs formes. Nous présenterons successivement, Chapitre 2, les différents types d'entrées sur lesquels la recherche se focalise, et pour chacun, ses spécificités (buts poursuivis, spécificités, état de la recherche).

8. RDF : expliqué en détail page 36
9. *tokens* : éléments constitutifs d'une séquence ; dans une phrase *tokenisée* en mots, les tokens comprennent les mots et la ponctuation

Une entrée type a visant la production d'un texte en langue naturelle b peut consister en :

- du RDF à verbaliser (*Resource Description Framework*), le RDF étant une représentation logique de données formalisée dans le cadre du web sémantique présentée Section I. 2. 1,
- du texte (Section I. 2. 2) :
  - à traduire,
  - simplifier (rendre plus facile à comprendre),
  - compresser (rendre plus synthétique),
  - moduler (paraphraser).
- de l'AMR (*Abstract Meaning Representation*, Section I. 2. 3) : une représentation logique issue du formalisme s'intéressant à la représentation du sens dont les mots rendent compte.

Deuxième partie

Le Chapitre 1 de la deuxième partie est consacré à une contribution importante de notre thèse, l'exploration des variations sur support syntaxique. Le second chapitre de cette même partie (Chapitre 2, Représentation automatique du sens) porte sur l'automatisation des traitements sur texte brut.

Variations sur support syntaxique

Après l'exposé des motivations (Section II. 1. 1) et la présentation des travaux directement liés (Section II. 1. 2), nous aborderons nos contributions quant à la génération de paraphrases syntaxiques (II. 1. 3). Nous présenterons d'abord notre corpus, WebNLG (Gardent et al., 2017) qui associe RDF et texte, II. 1. 3.1. Nous avons enrichi ce corpus avec des informations syntaxiques et en localisant les entités (sujet, objet) des triplets RDF, ce Section II. 1. 3.2. Enfin, nous avons généré des paraphrases syntaxiques à partir de triplets RDF, ce que nous présentons Section II. 1. 3.4, puis à partir de textes et de données hybrides Texte/RDF (Section II. 1. 3.5).

Les scripts ayant permis ce travail sont disponibles ici : `https://gitlab.inria.fr/colineem/rdf2nn`

Automatisation traitements sur texte brut

Nous exposons tout d'abord nos motivations quant à ce pôle de notre travail (Section II. 2. 1) et la présentation des travaux liés (Section II. 2. 2), nous abordons ensuite nos contributions quant à l'automatisation de traitements sur texte brut.

Nous avons construit automatiquement des représentations de sens (MR, *Meaning Representation*) à l'aide de ressources externes : analyseurs (*Grew* (Guillaume et al., 2012), *Talismane* (Urieli, 2013) et *Stanford Dependency Parser* (Manning et al., 2014)) et thésaurus (LVF (Les Verbes Français, Dubois et Dubois-Charlier (1997))).

Nous exposons nos choix quant à la création d'un corpus parallèle MR/Texte Section II. 2. 3 : filtrage des analyses, création des représentations, puis présentation du corpus final obtenu.
Section II. 2. 4 nous vous présentons le modèle utilisé pour générer des phrases à partir de ces données parallèles. Enfin, après avoir posé nos méthodes d'évaluation Section II. 2. 5, nous détaillons nos résultats Section II. 2. 6.

Les scripts ayant permis ce travail sont disponibles ici : https://gitlab.inria.fr/colineem/analyseurs

Nous concluons ensuite sur une synthèse de notre travail avant d'aborder les conclusions générales où nous dresserons les perspectives envisagées.

3 Contributions

Les contributions de cette thèse sont les suivantes :

- Proposition d'un modèle neuronal pour la génération de phrases sous contraintes syntaxiques.
  Le modèle est conditionné à la fois sur un contenu sémantique (un ensemble de triplets RDF par exemple) et une contrainte syntaxique (ex : relative sujet). Nous introduisons plusieurs métriques permettant d'évaluer d'une part l'adéquation sémantique des sorties et d'autre part leur adéquation syntaxique (la phrase générée contient-elle bien la contrainte syntaxique donnée en entrée ?). Ce travail a fait l'objet d'une publication à EMNLP 2018 Colin et Gardent (2018).
- Introduction de quatre tâches de génération et exploitation de ces tâches pour la génération de paraphrases syntaxiques :
  - à partir de RDF : verbalisation de l'information contenue,
  - à partir de texte : reformulation,
  - à partir de texte et de RDF :
    - reformulation du texte avec intégration d'une information portée par un triplet RDF,
    - reformulation du texte avec suppression d'une information portée par un triplet RDF,

  L'usage conjoint des quatre modèles permet l'obtention de plus de paraphrases.
- Introduction d'une méthode de délexicalisation généralisée permettant de mieux abstraire des données lexicales et d'améliorer l'apprentissage d'un modèle de génération de texte à partir de représentations sémantiques. Ce travail a fait l'objet d'une publication à INLG 2019 Colin et Gardent (2019).
- Création et mise à disposition de corpus pour la génération de textes en anglais, sous contraintes syntaxiques et avec entrées de différents types (RDF, Texte, RDF+Texte, RDF-texte)

- Création et mise à disposition de corpus pour la génération de textes en français à partir de représentations sémantiques. L'anonymisation des mots non fonctionnels augmente la qualité des sorties.

4 Publications

- E. Colin et C. Gardent. "Generating Text from Anonymised Structures". International Conference on Natural Language Generation (INLG). Oct. 29 – Nov. 1, 2019. Tokyo, Japon.
- E. Colin et C. Gardent. "Generating Syntactic Paraphrases". Conference on Empirical Methods in Natural Language Processing (EMNLP), Oct. 31 – Nov. 4, 2018. Bruxelles, Belgique. Papier court.

Première partie

Notions préalables

Chapitre 1

Les réseaux de neurones

Nous allons donc maintenant effleurer le domaine de l'intelligence artificielle, en nous centrant dans un premier temps sur les perceptrons, fruit du travail des connexionnistes[10]. Ils sont à la racine des réseaux de neurones actuels. Nous reprenons ici une part d'un travail précédent (Colin, 2015).
Nous étudierons ensuite l'architecture encodeur/décodeur (auto-encodeur), que nous avons utilisée, puis clôturerons en nous concentrant sur le mécanisme d'attention, un mécanisme particulièrement efficace dans le domaine de la génération de texte.

I. 1. 1 Aux racines des réseaux de neurones

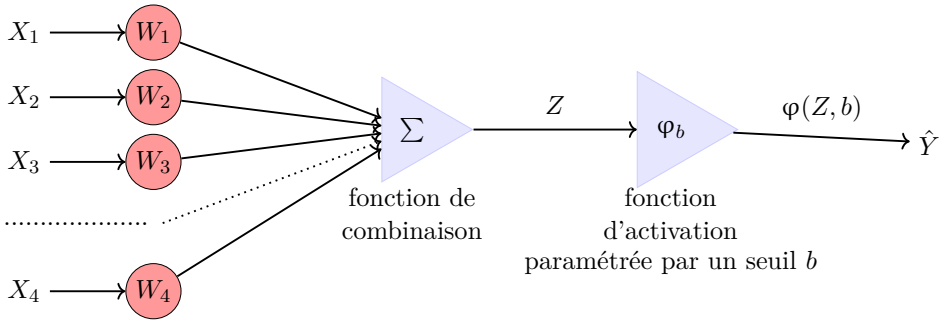
Tout a démarré au cœur de la révolution cybernétique, en 1943, avec une proposition de McCulloch & Pitts, celle du concept de neurone formel. Avec ce neurone, on crut trouver la clé de la réalisation de la machine de Turing (ce qui fut affirmé par McCulloch et Pitts (1943)). Il reste - de toute manière - que la machine de Turing est un modèle abstrait qui peut calculer tout ce qui est calculable... : tant qu'elle est abstraite, elle n'a pas de souci mémoire.

Les neurones, minoritairement situés dans notre cerveau (voir Annexe A), sont le socle inspirant de (McCulloch & Pitts). Un neurone formel est une représentation mathématique d'un véritable neurone, et un neurone artificiel une implémentation, le plus souvent informatique.

McCulloch, neurologue, Pitts, psychologue, offrent une représentation mathématique des échanges dans le cerveau, sept ans après l'expérience de pensée proposée par Turing : les neurones s'activent si l'activité combinée qu'ils reçoivent est suffisamment élevée.

En 1949, soit six années plus tard, Hebb entra en scène, proposant un réseau implémentable.
La représentation formelle dote les synapses de poids, liant des neurones par des valeurs numériques modulables : la voie de l'apprentissage est ouverte.

10. connexioniste : approche des phénomènes mentaux modélisés comme systémiques, émergeant des interactions entre unités simples.



Chaque valeur de X_i est multipliée par son poids W_i. Les i nombres obtenus sont additionnés, ce qui nous donne Z. Z est stocké.
La somme Z est modifiée par une fonction φ qui selon un seuil b nous donne $\hat{Y}$. $\hat{Y}$ vaut 0 ou 1 : à 1 le neurone est dit *activé*.

FIGURE I. 1. 1 – Structure d'un neurone artificiel post-hebbien

La règle de Hebb :
When an axon of cell A is near enough to excite B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased, Hebb et Hebb (1949).

Cette règle permit la naissance du perceptron, première implémentation informatique d'un réseau de neurones.

$$W_i' = W_i + \alpha(Y.X_i) \tag{I. 1. 1}$$

Équation de Hebb, dite règle de Hebb

Hebb nous dit en 1949 que « les neurones qui déchargent ensemble forment entre eux des circuits préférentiels » : W_i et W_i' sont un poids synaptique à un temps différent, W' est un nouvel état de W, Y et X_i des neurones. X_i est l'un des i neurones connecté à Y ; quand X_i s'active en même temps que Y, W_i est majoré d'une portion α de $Y.X_i$. Un neurone voit son activité comprise entre 0 et 1.

Franck Rosenblatt mit au point cette règle d'apprentissage :

$$W_i' = W_i + \alpha(\hat{Y} - Y)X_i \tag{I. 1. 2}$$

Équation de Rosenblatt

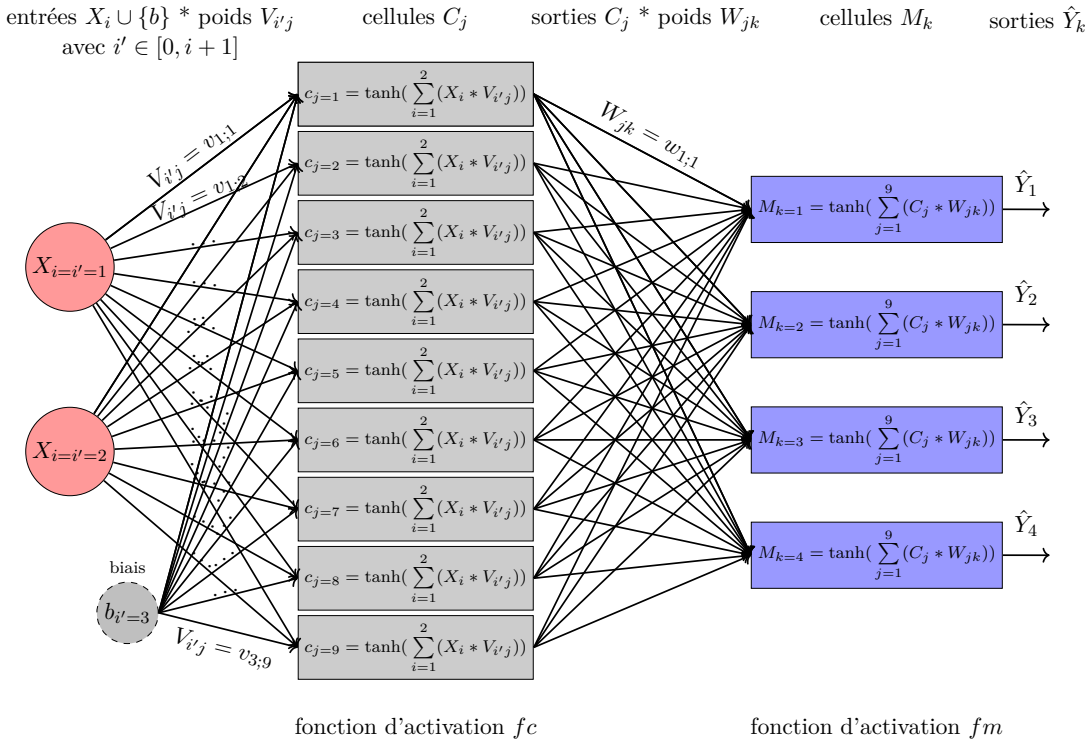
Modélisées par W_i, i synapses (voir annexe A. 1) pondèrent les entrées X : une par valeur X_i permettant l'obtention d'Y.
W_i est le poids i courant modulant l'entrée X_i, W_i' le nouveau poids qui tient compte de l'erreur d'apprentissage $\hat{Y} - Y$ (valeur obtenue $\hat{Y}$ moins valeur attendue Y) à hauteur du pas d'apprentissage α.

Le perceptron repose sur ce modèle. Cet apprentissage est supervisé.
Un ensemble de e exemples (X_i, Y_j) est présenté à un percepton. Les sorties obtenues $\hat{Y}_j$ tendent au fil de l'apprentissage à être le reflet Y_j des entrées X_i, en partant du principe que $X_{ie} \Rightarrow Y_{je}$.
Rosenblatt positionne les poids de manière à ce que ce reflet, issu d'un espace de transition des valeurs, réduise les erreurs en sortie modulo ce que les exemples d'apprentissage offrent.

Hebb est psychologue, mais aussi neuropsychologue. À l'heure des balbutiements informatiques [11], sa thèse est une irruption dans le monde des idées. L'idée formalisée dans l'équation initiale de ce scientifique (éq. 1.1) était une des prémisses à la mathématisation du monde sensible.
Les savoirs scientifiques ont évolué, et il est désormais accessible de remarquer que l'équation de Hebb est insuffisante pour créer un algorithme permettant un apprentissage. La notion de réajustement n'est pas là. Elle n'était pas encore pensable quand Hebb soumet sa théorie. Elle l'est seulement devenue avec Rosenblatt (éq. 1.2). L'erreur (écart entre valeur obtenue et valeur espérée) est dès lors utilisée pour moduler à la baisse ou à la hausse les poids, ce au prorata de chaque élément de X_i ; Cette décision est arbitraire... l'idée est pertinente, le résultat insuffisant : on altère les poids en espérant la convergence, positivement, négativement, anarchiquement.
Inventé en 1957 par Rosenblatt, le perceptron fut clairement affiné grâce aux travaux sur la rétro-propagation du gradient de Werbos (1975), puis de Parker & LeCun (1985) et de Rumelhart & McClelland (1986). Ces travaux ont rendu possibles et fonctionnels les perceptrons que nous allons maintenant aborder.

11. 1945 : architecture de von Neumann : premier modèle d'ordinateur

I. 1. 2 Les perceptrons



Chaque valeur de X_i et $\hat{Y}_k$ est un nombre réel : nous nous situons plus dans la prédiction, non dans l'activation booléenne. *tanh* est utilisée. Cette fonction est dérivable, elle va nous permettre le calcul du gradient (but : rétropropager l'erreur au prorata des valeurs poids qui y ont abouti (algorithme en annexe B)).

FIGURE I. 1. 2 – Modèle de perceptron à couche cachée

La Figure I. 1. 1 présente un perceptron, le plus simple existant : il est mono couche, toutes ses entrées (X_n) sont connectées à sa seule et unique sortie. Chaque entrée x_1 à x_n est multipliée par son poids n respectif (w_1 à w_n). Les nombres obtenus sont additionnés les uns aux autres. Cette somme z subit l'application d'une fonction d'activation f telle que :

$$f(z) = \begin{cases} 0 & \text{si} \quad z \leq 0 \\ 1 & \text{si} \quad z > 0 \end{cases} \tag{I. 1. 3}$$

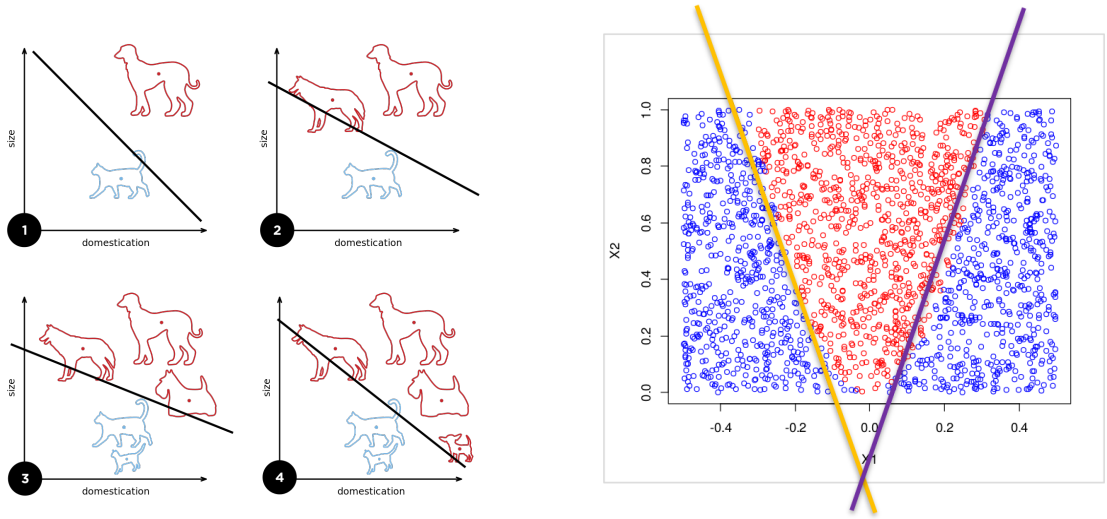
Fonction d'activation

Les entrées sont classiquement des nombres réels. $f(z)$ est une fonction booléenne (`vrai` si $f(z) = 1$, `faux` sinon). f est ici une fonction de Heaviside : cette fonction impose un seuil de zéro puisque $f(z)$ sera `vrai` ou `faux` en fonction du rapport de z au seuil b.

Vous pouvez observer Figure I. 1. 2 une représentation générique de perceptron multicouches. Celui-ci présente une seule couche cachée (blocs grisés de la colonne centrale, C_j).

Le perceptron reçoit des entrées (X_1 et X_2 sur la Figure I. 1. 2), elles permettent le calcul des valeurs de sortie ($\hat{Y}$: $\hat{Y}_1$ à $\hat{Y}_4$ ici). Tout l'enjeu est que ses valeurs s'approchent de valeurs de référence (Y : Y_1 et Y_4)

Le seuil b Figure I. 1. 1 a une valeur fixée qui revient à définir la frontière du classifieur, car le perceptron est un classifieur plus ou moins complexe, ainsi que l'illustre les Figures I. 1. 3a et I. 1. 3b.
Modifier la valeur de b modifie la seuil de déclenchement de l'activation du neurone artificiel.
Le biais Figure I. 1. 2 agit de la même manière, exception faite que son influence est apprise : il est lui aussi lié à chaque cellule de la couche cachée par des poids. L'ensemble de poids du biais peut permettre une segmentation fine des exemples.



(a) Perceptron simple

De Elizabeth goodspeed, wikimedia

(b) Perceptron multicouches

Source : http://eric.univ-lyon2.fr/~ricco/cours/slides/reseaux_neurones_perceptron.pdf, consulté le 19 novembre 2019

FIGURE I. 1. 3 – Perceptrons mono et multicouches : séparation des valeurs

Les poids sont initialisés aléatoirement. Nous avons des valeurs en entrée, nous voudrions des valeurs en sortie. Les valeurs entrée/sortie d'exemples constituent une base d'apprentissage qui doit voir ces deux pôles (entrées/sorties) fortement corrélés. Sinon le système n'apprend pas correctement.

Au début de l'apprentissage, nous présentons des valeurs en entrée, celles -ci sont multipliées par des poids ($v_{i'j}$ et v_{jk} sur la Figure I. 1. 2) et soumises de façon intermédiaire à une fonction dérivable. Les poids étant initialisés aléatoirement, les valeurs récupérées en sortie ne sont pas celles attendues. Ce n'est pas grave. Cette sortie va permettre, en

comparant valeurs attendues avec valeurs obtenues, de calculer une erreur. Cette erreur va pouvoir être répercutée sur les poids afin qu'au final, le perceptron nous fournisse les sorties désirées.

Des principes sont fondamentaux : celui de la fonction d'activation participant à l'obtention du résultat, et celui de la rétro-propagation du gradient.

La fonction utilisée, tangente hyperbolique sur la Figure I. 1. 2, doit être dérivable. Elle doit aussi être continue, bornée et à seuil. Des algorithmes nous permettent de calculer l'erreur (différence Y $\hat{Y}$), afin de corriger les poids.
Lorsqu'un poids est modifié, il est modulé à l'aide d'un tout petit pourcentage de chaque erreur jusqu'à ce que l'apprentissage soit terminé : le perceptron s'adapte à un ensemble de situations dont la base d'apprentissage se doit d'être représentative.

Nous appellerons *signal* - par référence aux travaux de Shannon (Shannon et Weaver, 1949) - les valeurs X_i issues d'un ensemble d'échantillons. De McCulloch & Pitts à Rosenblatt, en passant par Shannon, à travers les réseaux de neurones nous sommes en plein paradigme cybernétique : l'information est un signal, et ce signal circule.
Elle circule tout d'abord vers l'avant, depuis les nœuds d'entrée, à travers les couches cachées et arrive aux nœuds de sortie.
À ce stade, une fonction objectif (*objective function* ou *loss function*) est en charge de calculer l'erreur. RosenBlatt utilise une fraction α de l'écart entre valeur produite et valeur attendue (voir éq. 1.2).

Dans un système idéal, nous avons $W = Y/X$ puisque l'on désire - en simplifiant notre représentation en couches - $X * W = Y$. Rosenblatt fait osciller le poids à l'aide d'une petite part de l'erreur, ce proportionnellement à la valeur de X.
Plus l'erreur est faible, moins le poids oscille. Quand le pas α ou quand l'erreur $\hat{Y} - Y$ est égal(e) à zéro, on arrête l'apprentissage. La décrémentation du pas assure artificiellement l'arrêt de l'apprentissage tout autant que sa "finesse". Théoriquement, l'erreur est faible en fin d'apprentissage, forte initialement. Un pas plus faible implique une correction marginale (plus fine), alors qu'un pas élevé implique une correction elle même plus élevée (plus brutale aussi : toute correction se rapporte à une part très limitée du corpus d'apprentissage [12]). Seuls des essais peuvent garantir que le pas fixé et que les choix de décrémentation sont les bons. On tente alors des prédictions sur corpus de test. La baisse du taux d'erreur est elle aussi mesurable durant l'apprentissage, il suffit d'en tracer l'évolution.
Il y a là un choix arbitraire et bien d'autres solutions sont envisageables. Celle-ci fut largement suffisante pour prouver le concept de réseaux de neurones, son efficience, et lancer les recherches plus ou moins fructueuses qui la suivront.

12. correction par *batch* : répercution de l'erreur moyenne liée à plusieurs exemples.

I. 1. 3 Rétropropagation du gradient

La rétropropagation du gradient sera fondamentale pour faire effectuer un bond aux réseaux de neurones. Pour reprendre wikipédia dans sa page *Dérivée*, « la dérivée d'une fonction d'une variable réelle mesure l'ampleur du changement de la valeur de la fonction (valeur de sortie) par rapport à un changement de son argument (valeur d'entrée). », Le gradient, toujours pour citer wikipédia dans la page éponyme, est lui « la généralisation à plusieurs variables de la dérivée d'une fonction d'une seule variable. ». Le concept de rétropropagation du gradient est de rapporter à chaque poids "sa" part de l'erreur : plus le poids contribue à l'erreur, plus il sera corrigé.
Cette erreur est définie par la fonction dite de *loss*, ou fonction de perte : la fonction dont l'objectif est de minimiser l'erreur. Nous nous sommes éloignés de Rosenblatt. L'erreur est calculable de plusieurs façons. Nous avons par exemple la classique MSE (*Mean Square Error*, erreur moyenne au carré (voir éq. 1.4)) ou encore l'erreur logarithmique (voir éq. 1.5, exemple pour une classification binaire).

$$\text{MSE} = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{I. 1. 4}$$

Erreur moyenne au carré (*Mean Square Error*)

$$\text{CE} = -(\text{y}\ \log(\hat{\text{y}}) + (1 - \text{y})\ \log(1 - \hat{\text{y}})) \tag{I. 1. 5}$$

Erreur logarithmique binaire (*Logarithmic loss for 2 classes* : y vaut 0 ou 1)

Nous calculons cette erreur sur les résultats issus de fonctions dérivables (`tanh`, Figure I. 1. 2). Elles sont dérivables : elles sont situées sur une pente de la fonction concernée. Nous pouvons donc calculer réduire ou augmenter la valeur de notre poids pour repositionner celui-ci correctement afin que l'erreur tende à baisser.
Cette direction de descente (nombre négatif ou positif permettant la minimisation de l'erreur), proportionnelle dans son impact au poids qui a mené à l'erreur, vient multiplier un pas α pour tenir compte de manière modeste de chacun des exemples. Tant que l'erreur n'est pas satisfaisante, on vient modifier notre poids courant avec ce delta.
Il s'agit de la rétropropagation du gradient, dont les valeurs sont propagées couche après couche.

La rétropropagation du gradient intervient pour faire appliquer les correctifs afférents aux couches intermédiaires. Il s'agit de déterminer la part d'un poids donné dans l'erreur globale, et de le moduler en conséquence.
Un algorithme est disponible en annexe (B), et pour approfondir le sujet, *Les réseaux de neurones artificiels : introduction au connexionnisme*, Touzet (1992), est une bonne ressource.

I. 1. 4 Les réseaux de neurones récurrents

Nous utilisons un encodeur décodeur de type séquence vers séquence (*seq2seq*). Il s'agit d'un réseau récurrent similaire à celui illustré Figure I. 1. 11.
Les réseaux récurrents sont des réseaux au fonctionnement fondamentalement approprié au traitement automatique des langues : ils permettent de travailler avec des entrées à taille variable. Néanmoins, ainsi que Pouget-Abadie et al. (2014) le démontrent, plus l'entrée est longue, plus il est difficile d'avoir de bons résultats. Il est possible d'accroître la qualité des sorties (cf. Sutskever et al. (2014)) en utilisant une architecture à effet mémoire telle que LSTM (*Long Short-Term Memory*).

LSTM (*Long Short-Term Memory*) et Gru (Gated recurrent unit)

Nous avons plus précisément choisi le LSTM parce que celui-ci fait face avec succès à deux problèmes désormais bien connus du *seq2seq* : l'explosion et la disparition du gradient (voir p.26).

Les cellules de type GRU (Gated recurrent unit) offrent un mécanisme similaire au LSTM : apprendre à oublier.

Pour aborder LSTM (et GRU), il nous faut aborder les notions de disparition et d'explosion du gradient. Lorsque nous assurons la rétropropagation du gradient (voir p.23), nous sommes en pratique confronté à deux effets contraires. Ces effets sont néfastes, mais heureusement limités (si faculté d'oublier).

Retour sur le gradient Si nos données d'entrées X/sorties Y sont corrélées, il est possible de transformer X en son image Y, tout du moins en une approximation de son image : $\hat{Y}$. Les poids W ont pour charge d'être les leviers manipulables de cette transformation. Le gradient est le moteur de cette transformation $f : X \to \hat{Y}$. L'apprentissage consiste à construire dynamiquement une fonction f telle que $f(X, W) = \hat{Y}$.
Le gradient de cette fonction f sera noté $\overrightarrow{\nabla} f$.
Le gradient est un vecteur de composantes $\partial f / \partial X_i$ $(i = 1, 2, \ldots, n)$.
Le gradient est un vecteur d'une taille identique au nombre de n variables, ici de poids W_n.
Chaque élément d du vecteur est égal à l'accroissement de la variable lui correspondant entre sa valeur et sa valeur prédécédente (parmi les bornes de définition de ses variations par f).

Nous avons vu que tanh comme sigmoïde sont les fonctions les plus classiquement utilisées. En les utilisant pour obtenir $\hat{Y}$, nous devons normaliser les valeurs Y pour que la plage des valeurs possibles soit couverte et ce exclusivement par notre réseau.
Tanh et sigmoïde sont croissantes et continues. Donc si nous obtenons une valeur $\hat{Y}_i$ plus grande que le Y_i de référence, notre erreur $Y_i - \hat{Y}_i$ est négative. En utilisant la valeur

de la pente de tanh ou de sigmoïd qui y ont mené, pente qui a donc une valeur positive (voir annexes page VI), nous modifierons f de manière ce que $\hat{Y}$ converge vers Y.

Rétropropager le gradient consiste à affecter les W pour qu'en fonction de X nous approchions $\hat{Y}$ menant d'une cellule à une autre (entrées, biais, cellules couches cachées, sorties...).

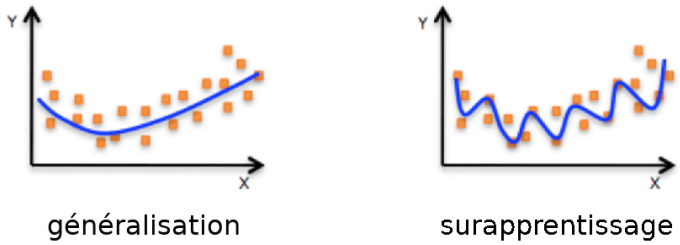


FIGURE I. 1. 4 – Surapprentissage

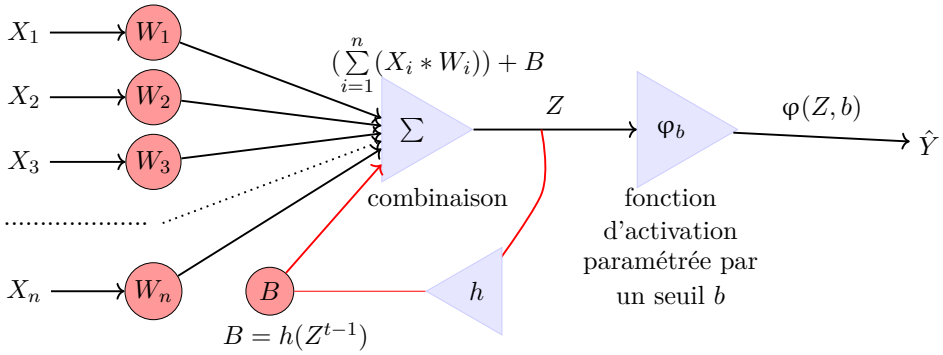
D'après : https://www.quora.com/What-are-the-key-trade-offs-between-overfitting-and-underfitting, consulté le 13 novembre 2019

Ainsi que le dit wikipédia, l'« algorithme du gradient a pour but de converger de manière itérative vers une configuration optimisée des poids synaptiques. Cet état peut être un minimum local de la fonction à optimiser et idéalement, un minimum global de cette fonction (dite fonction de coût). »

Le but de la rétropropagation du gradient est donc d'amener les poids W modifiant les valeurs entrées (X_i) de manière à ce que les valeurs de sorties $(\hat{Y}_i)$ soient proches de valeurs d'apprentissage (Y_i). La trop grande proximité entre valeurs obtenues et valeurs attendues s'appelle surapprentissage (*overfitting*). Nous souhaitons au contraire une généralisation. La Figure I. 1. 4 illustre ces deux situations.
Nous cherchons donc à moduler la sortie $\hat{Y}$ par une action sur les poids W pour que $\hat{Y}/X$ soit la valeur d'équilibre entre les exemples Y/X.

Le LSTM vise à prendre en charge la gestion de ces problèmes. Introduit par Hochreiter et Schmidhuber (1997), le LSTM est un réseau récurrent dont les cellules disposent d'une entrée, d'une sortie, et innovation, d'une porte d'oubli (respectivement en anglais *input gate*, *output gate*, *forget gate*) que l'on peut voir Figure I. 1. 7. Une notion fondamentale régit ce réseau récurrent, celle de *timestep*.

Réseau récurrent Un réseau récurrent est modelé de façon similaire à un réseau de type perceptron illustré Figure I. 1. 1, à ceci près qu'une récurrence dans le traitement de l'information est introduite, ainsi que le montre la Figure I. 1. 5.



Chaque valeur de X_i est multipliée par son poids W_i. Les i nombres obtenus sont additionnés à un vecteur initialisé marquant le début des traitements, et on obtient Z.
Z est une représentation des variables combinées, elle est stockée en tant que Z^{t-1} : il s'agit de z au *timestep* précédent. Z^{t-1} est modifiée par une fonction d'activation h qui nous donne B.
B sera ensuite utilisé en lieu et place du primo-vecteur pour nous donner Z lors des entrées suivantes. La somme z est modifiée par une fonction φ qui selon un seuil b nous donne $\hat{Y}$. $\hat{Y}$ vaut 0 ou 1 : à 1 le neurone est dit *activé*.

FIGURE I. 1. 5 – Récurrence opérée avant la fonction d'activation φ

Si la sortie d'une couche d'un réseau non récurrent (*feedforward neural network* ou réseau de neurones à propagation avant) peut se représenter $\varphi(\sum_{i=1}^{n}(X_i * W_i))$, celle d'un réseau tel que montré Figure I. 1. 5 aura pour équation :

$$\hat{Y} = \varphi(\sum_{i=1}^{n}(X_i * W_i) + h(\sum_{i=1}^{n}(X_i^{t-1} * W_i))) \quad \text{(I. 1. 6)}$$

Réseau récurrent : valeur de $\hat{Y}$

Il est tout à fait possible de calculer les *timesteps* (pas d'apprentissage) après la fonction d'activation, de les pondérer par des poids.

Les réseaux récurrents sont tout à fait adaptés au traitement de séquences à longueur variable : ils prennent en compte l'historique au sein de séquence. Néanmoins, ils font face à deux problèmes : disparition et explosion du gradient (Bengio et al., 1994).

Disparition du gradient Il y a plusieurs manières de calculer (et faire diminuer) l'erreur, comme montré page 23. Les méthodes pour affiner $\hat{Y}$ sont elles aussi des choix (un classique est l'utilisation de la dérivée seconde, qui mesure l'évolution des taux de variations).
Les méthodes dérivationnelles, que l'on doit à Werbos (1975) sont extrêmement efficaces, et à ce jour optimisées. Néanmoins, la rétropropagation du gradient souffre d'un gros inconvénient : la disparition de ce dernier (*vanishing gradient*, Hochreiter (1998)). En effet, répercuter la correction des erreurs devenues très faibles, agir quand la pente a une valeur près de zéro... quand bien même l'erreur serait forte, paralyse l'apprentissage.

Explosion du gradient Le problème de l'explosion du gradient (Pascanu et al., 2012) est inverse. Pour traiter l'erreur dans chaque couche c supérieure à celle précédemment

traitée c^{-1}, nous réalisons un produit des gradients de c^{-1} (voir Figure I. 1. 2 en imaginant une couche supplémentaire et se rapporter à l'algorithme d'apprentissage présenté en annexe B. 1). Si le produit implique trop de termes avec une valeur élevée, les gradients explosent : la convergence ne se fera pas. Les *timesteps* sont très problématiques à calculer dans les réseaux récurrents.

I. 1. 5 LSTMs et Grus

LSTM (*Long Short Term Memory*) désigne à la fois un type de réseau séquence vers séquence (*seq2seq*) et une unité constitutive de couche (cellule) de *seq2seq*. Gru (*Gated recurrent unit*) désigne un type de cellule de *seq2seq*.
Si Gru a été introduit en 2014 par Cho et al. (2014), le LSTM date lui de 1997 comme vu précédemment.

Bien qu'optimisés (on applique du calcul vectoriel sur les matrices), les modèles à base de LSTMs et de Grus illustrent aisément les besoins des réseaux de neurones : puissance de calcul, espace mémoire et temps sont des éléments incontournables. Les ordinateurs actuels sont suffisamment puissants pour que n'importe quel PC récent soit à même d'instancier un petit réseau.

Les cellules LSTM et Gru sont composées de petits perceptrons appelés "portes".

La Figure I. 1. 6 représente un réseau récurrent global.

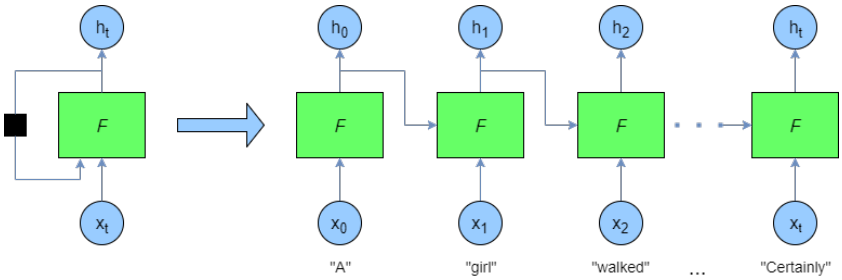


FIGURE I. 1. 6 – Réseau récurrent, vue globale

Issu de :https://adventuresinmachinelearning.com/recurrent-neural-networks-lstm-tutorial-tensorflow/, consulté le 18 novembre 2019

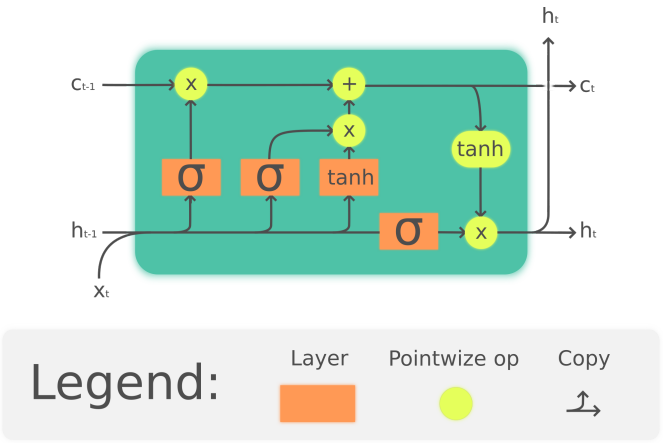


FIGURE I. 1. 7 – Cellule LSTM

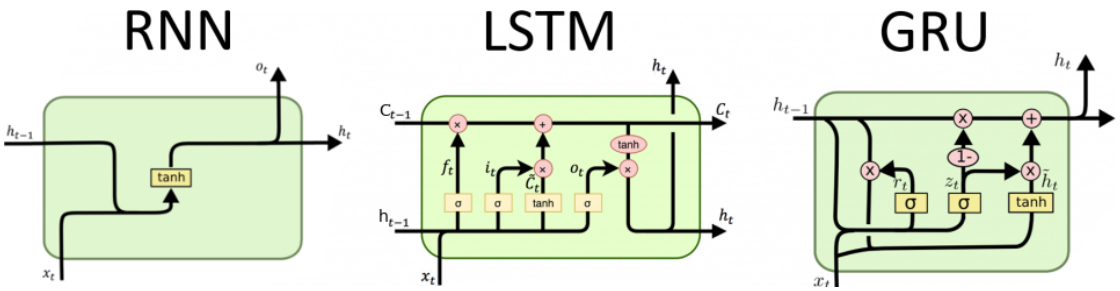


FIGURE I. 1. 8 – Cellules de RNN simple, de LSTM et cellule Gru

D'après http://dprogrammer.org/rnn-lstm-gru, consulté le 18 novembre 2019

Gru est une version simplifiée de LSTM. Les représentations sont très codifiées. C représente l'usage d'un vecteur contextuel, f la fonction propre à la `forget gate`, h pour un vecteur d'état intermédiaire (`hidden`, caché), i pour *input*, l'entrée, o pour *output*, la sortie. σ représente classiquement une fonction d'activation quelconque, comme *sigmoïde*.

Sur la représentation Figure I. 1. 6, chaque trio (valeur d'entrée X_n, espace fonctionnel F, sortie h_n) est une cellule.

Temps après temps (*timestep after timestep*), les gradients sont additionnés, les erreurs sont additionnées... les mots d'une séquence sont successivement traités (un *timestep* est égal au traitement d'un mot).

Si l'erreur à un temps t est $E(y_t, \hat{y}_t)$, qui, si on utilise l'entropie croisée (voir équation 1.5, p.23)), vaut $-y_t * log(\hat{y}_t)$, l'erreur dans un RNN est : $\sum_t E(y_t, \hat{y}_t)$.

Nous calculons le gradient pour chacun des temps t.

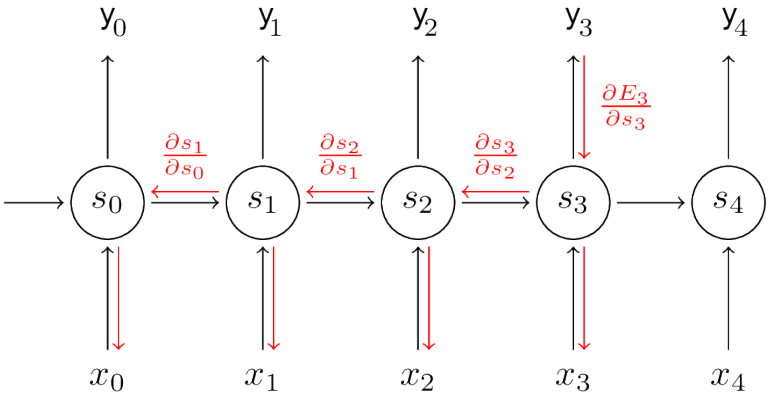


FIGURE I. 1. 9 – Backpropagation dans un RNN

D'après http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/, consulté le 19 novembre 2019

Sachant U les poids de l'entrée x_t à l'état s_t, W ceux de s_t à s_{t+1}, et V ceux de s_t à y_t, nous avons $S_t = tanh(U * x_t + W * s_{t-1})$ et $\hat{y}_t = softmax(V * s_t)$. Avec l'entropie croisée pour calculer l'erreur, nous avons $L(y, \hat{y}) = -1/N \sum_t y_t log(\hat{y}_t)$.

Tout comme dans le modèle perceptron, nous calculons les dérivées partielles au niveau de chacun de nos paramètres et les sommons simplement. Une part de cette erreur répercutée sur les paramètres (entendez les poids) permet au modèle de converger... si les données permettent cette convergence.

I. 1. 6 L'architecture encodeur/décodeur

Nos travaux ont exploité des réseaux séquence vers séquence (dits *seq2seq*), qui sont basés sur une architecture encodeur/décodeur. Cette architecture leur étant essentielle, nous allons maintenant la présenter. Pour cette explicitation, nous nous sommes largement inspirée de *Sequence-to-Sequence Models* (John Hewitt & Reno Kriz, Université de Pennsylvanie) [13], qui donne une excellente vue d'ensemble de leur fonctionnement.

Afin de traiter des séquences de mots, chaque mot du vocabulaire est représenté par un vecteur de valeurs réelles potentiellement ajustées dynamiquement lors de l'apprentissage. Ce vecteur peut être - ou non - commun aux entrées et sorties. Dans notre travail, nous avons utilisé un vecteur par mot pour les entrées, un vecteur par mot pour les sorties, de façon très arbitraire. Ce n'est pas fondamental pour les modalités que nous explorons. Les résultats sont néanmoins fortifiables par des ajustements réfléchis.

13. document en ligne https://nlp.stanford.edu/~johnhew/public/14-seq2seq.pdf, dernière consultation le 28 octobre 2019.

La Figure I. 1. 11 donne une représentation schématique de l'architecture *seq2seq*. La fonction softmax est utilisée en sortie. Vous pouvez l'observer plus en détail Figure I. 1. 10. Les données réceptionnées par la dernière couche, des nombres réels (`a`) , sont traitées pour donner n probabilités pour chacun des n mots de vocabulaire. Par exemple dans le schéma I. 1. 10 (`b`), quand le mot *dort* est fourni au réseau de neurones, sa représentation vectorielle (*word embedding*) est en quelque sorte convertie en un faisceau de probabilités. Apprentissage réalisé, c'est ici *endormi* qui a la plus forte d'entre elles.

L'encodeur/décodeur présenté Figure I. 1. 11 est idéal pour associer ces deux termes pendant l'apprentissage. Le mot "endormi" est prédit à partir de l'état courant et du dernier mot généré ("est"). La fonction softmax est formalisée ainsi :

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ pour tout } j \in \{1, \dots, K\} \tag{I. 1. 7}$$

Softmax

Softmax prend en entrée un vecteur Z composée de K nombre réels et sort un vecteur $\sigma(\mathbf{z})$ de K nombres réels strictement positifs dont la somme vaut 1.

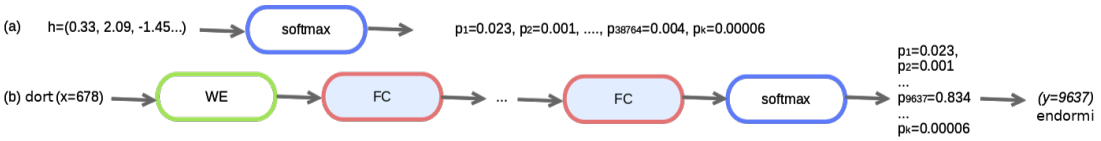


FIGURE I. 1. 10 – Softmax : le fonctionnement au sein d'un réseau

WE : *Word Embedding* (représentation vectorielle d'un mot, par exemple le mot dort sur la figure est le mot n°678 du vocabulaire. Il est representé par un ensemble de nombres réels appelé *embedding* du dit mot),
FC : *Full Connected*, ici une couche cachée,
SoftMax : couche de sortie.
Ici le réseau est entraîné à une tâche de traduction, il prédit que le mot le plus problable pour *dort* est *endormi* qui est à l'index 9637 et a une probabilité de 0.834.

D'après https://weave.eu/mecanisme-dattention-simple-astuce-mecanisme-universel/, consulté le 20 novembre 2019

Pendant l'apprentissage, on attend donc que l'indice du mot le plus probable corresponde à celui du mot attendu : le softmax est une distribution de probabilités sur l'ensemble du vocabulaire. L'erreur est calculée de manière à orienter les états internes vers l'écart le plus faible avec ce qui est espéré.

Figure I. 1. 11, chaque jonction vecteur mot / vecteur mémoire (ou *état)* est le fruit d'une combinaison : classiquement l'addition du produit scalaire de chacun des vecteurs avec des poids.
Les poids - une fois l'apprentissage achevé - reflètent la meilleure solution trouvée pour associer les deux vecteurs.

L'équation permettant la combinaison des états cachés (vecteurs mémoire) et des vecteurs mot, est :

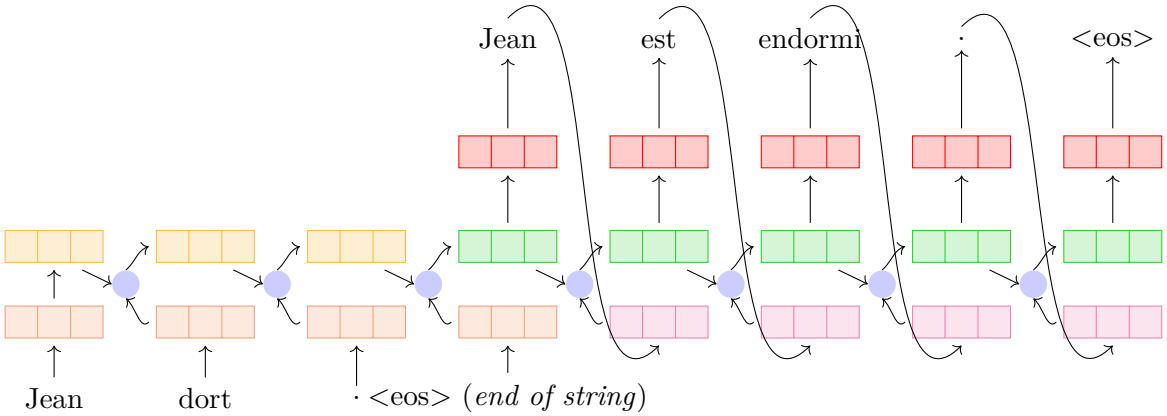
$$\bigcirc \ : \ h_t = \tanh(W^{hx} X_t + W^{hh} h_{t-1} + b^h)$$

Sachant

h_t l'état caché du réseau à un temps t ou

X_t le vecteur mot d'entrée à un temps t ou

W^{hx} les poids associés au vecteur mot d'entrée
W^{hh} les poids associés aux informations issues de l'état précédent (vecteur mémoire)
b^h un biais.



vecteurs d'un mot : représentation du mot traité (Jean, dort,. , <eos>),

vecteurs mémoire (ou état) : parts du *contexte* encodeur.

vecteurs mémoire (ou état) : parts du *contexte* décodeur.

vecteur d'un mot : représentation (Jean, est, endormi,. ,<eos>),

vecteurs d'un mot sélectionné par une fonction softmax.

fonction du réseau combinant le vecteur de mot et l'état précédent ou l'état initial pour créer un nouvel état.

FIGURE I. 1. 11 – Séquence vers séquence, une architecture encodeur/décodeur

Inspiré de :
https://tex.sIlfaudraitexpliciter/préciserlespointssuivants.tackexchange.com/a/362197/85146,
consulté le 28 octobre 2019

I. 1. 7 Le mécanisme d'attention

Dans une architecture basique encodeur/décodeur, l'encodeur fournit au décodeur une représentation de la séquence d'entrée, et ce dans un vecteur de taille fixe. La première case verte Figure I. 1. 11 correspond à ce vecteur. Il est attendu dans l'exemple qu'il permette la sélection du mot *Jean* en sortie et fournisse, avec ce mot, le contexte des mots suivants.

Cette représentation sous forme de vecteur à taille fixe de textes à longueurs variées est censée mener à un texte. En pratique, les dépendances à long terme sont problématiques. Une dépendance à long terme est présente par exemple quand le sujet se trouve éloigné de son objet dans une phrase particulièrement longue. Si ce cas ne se présente qu'une fois sur dix, l'apprentissage est difficile : le vecteur sortant de l'encodeur est un goulet d'étranglement qui ne s'adapte pas à la taille des entrées : les performances décroissent avec la longueur des données (Bahdanau et al., 2014b).

Le mécanisme d'attention vise à fournir une représentation qui varie en fonction de chacun des états (vecteurs mémoire) des mots précédemment traités : au lieu de n'avoir "que" le vecteur représentant le mot précédent adossé au contexte par le biais d'une fonction de combinaison (case jaune ou verte Figure I. 1. 11 et I. 1. 13), le réseau dispose de surcroît de la représentation de chacun des états cachés ayant été mis en œuvre (case bleue Figure I. 1. 13).

Notre idée est de donner l'intuition du fonctionnement de ce mécanisme (Vaswani et al., 2017).

$$\text{score}(h_t, \bar{h}_s) = h_t^\top \mathbf{W}_a \bar{h}_s \tag{I. 1. 8}$$

Scoring pour l'attention, Luong et al. (2015)

$\mathbf{W}_a$ est une matrice entraînable dans la couche d'attention : lors de l'apprentissage les poids sont modifiés. Le modèle assigne un score $\text{score}(h_t, \bar{h}_s)$ permettant la construction d'un vecteur de contexte pour chaque mot attendu en sortie à partir de l'état caché h_t. Il est l'évaluation de la qualité de l'alignement entre les entrées et le mot en cours de génération : chaque étape de l'encodage génère un score. Les scores sont utilisés durant le décodage pour améliorer la génération.
h_t est l'état caché courant,
$\bar{h}_s$ est la succession des états cachés déjà calculés pour la séquence s.

$$a_t(s) = \frac{e(\text{score}(h_t, \bar{h}_s))}{\sum_{s'} e(h_t, \bar{h}_{s'})} \tag{I. 1. 9}$$

Attention globale : vecteur d'alignement, Luong et al. (2015)

a_t : ce vecteur permet l'alignement aux dimensions variables de la séquence entrée.
h_t est l'état caché courant,
$\bar{h}_s$ est la succession des états cachés déjà calculés pour la séquence s.

$$c_t = \frac{\sum_{i=1}^{t} a_t h_{s'_i}}{\sum_{i=1}^{t} a_t} \tag{I. 1. 10}$$

Vecteur contextuel permettant la mise à jour de l'état courant global

c_t est un vecteur contextuel de taille fixe : ce vecteur permet l'alignement aux dimensions variables de la séquence entrée.
Le vecteur d'alignement a est utilisé comme poids pour le calcul du vecteur contextuel c_t. Le vecteur contextuel c_t correspond à une forme d'état caché pondéré qui est relatif à l'ensemble des états cachés précédemment calculés.

Le vecteur contextuel (somme des états pondérés par leur probabilité), équation 1.10, est fourni en entrée des cellules suivantes pour calculer le nouvel état. Ce contexte est calculé à partir des probabilités que chaque mot de l'entrée a (équation 1.8) d'impacter la sortie selon la situation courante. Ces probabilités sont synthétisées (équation 1.9). La Figure I. 1. 13 illustre une intégration du mécanisme d'attention à une architecture encodeur/décodeur.

Le mécanisme d'attention présente aussi l'intérêt de pouvoir suivre les corrélations établies par le réseau. Il suffit d'enregistrer et matérialiser par un graphique les contextes lors des étapes de génération, comme le montre la Figure I. 1. 12.

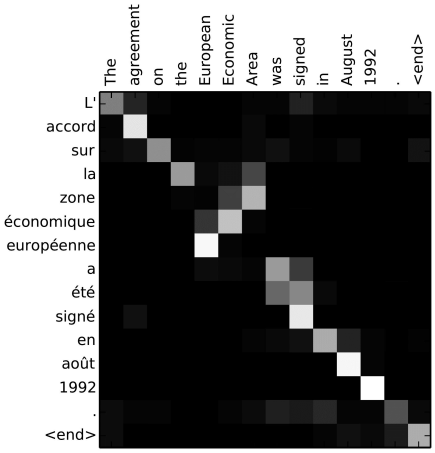


FIGURE I. 1. 12 – Mécanisme d'attention, illustration corrélations rendues exploitables par le mécanisme d'attention

source : Bahdanau et al. (2014b)

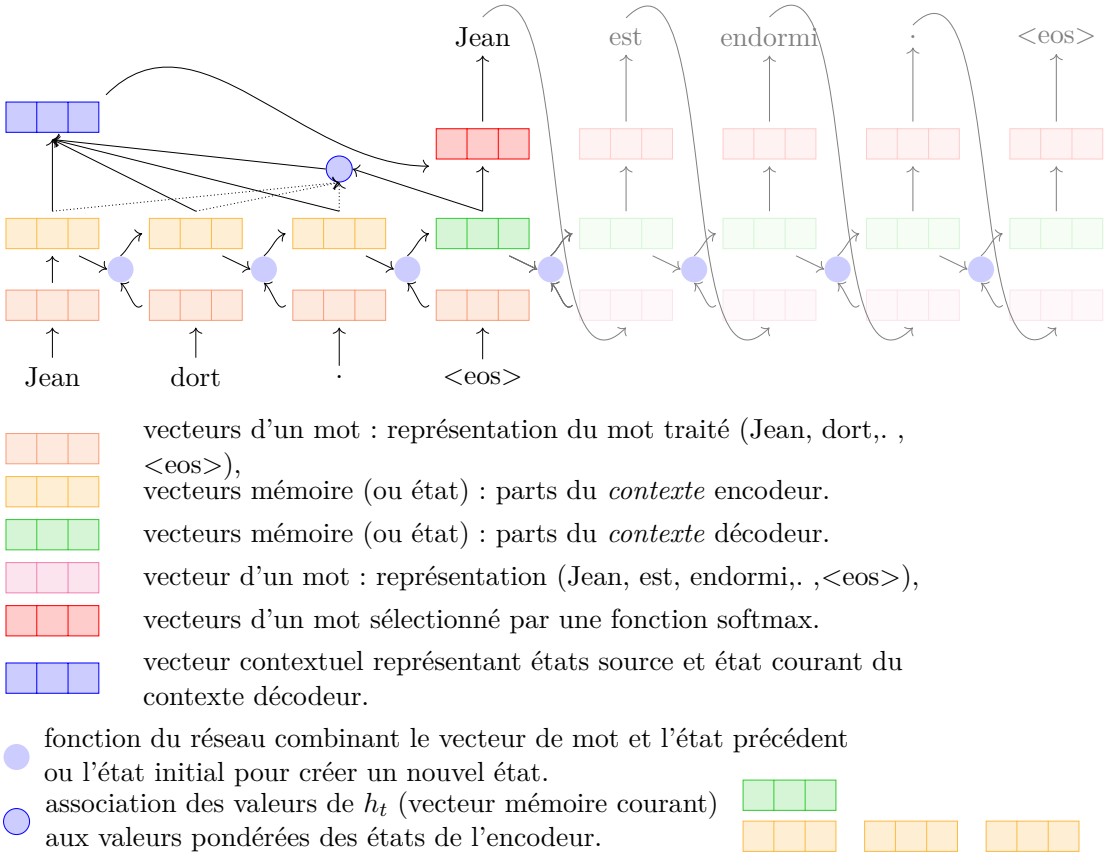


FIGURE I. 1. 13 – Mécanisme d'attention (*general attention*) dans une architecture encodeur/décodeur

La littérature offre maintes implémentations des mécanismes d'attention. Nous avons sélectionné la *general attention* qui calcule le vecteur contextuel (Luong et al., 2015).

Ce vecteur est de dimension fixe (celle des états de l'encodeur). Il est calculé autant de fois qu'il y a d'étapes de décodage.

Un softmax (voir page 30) permet ensuite de mettre en exergue pour le réseau l'importance de chacun des éléments de la source pour le terme qui sera à générer. Cette information vient moduler le contexte courant afin d'affiner les sorties et d'améliorer les performances du réseau dans le traitement des dépendances à long terme.

Chapitre 2

Génération en langue naturelle : la situation.

La génération en langue naturelle transforme une entrée *a* en un texte *b*.

Cette entrée *a* peut être une représentation logique de données sous forme de triplets RDF (association (sujet, prédicat, objet) sur laquelle nous reviendrons), un texte (typiquement une phrase à traduire), ou une représentation de texte dans laquelle la syntaxe est abstraite pour ne laisser qu'une représentation sémantique (*Meaning Representation*).

Ces trois familles ont une raison d'être, une histoire, que nous allons maintenant tâcher d'établir. En effet, nos travaux s'appuient sur chacune d'elle, de façon isolée ou hybridée.

Avec l'émergence de l'apprentissage profond, la génération de texte à partir d'une architecture encodeur/décodeur prédomine. Néanmoins, ainsi que le résume Dušek et al. (2020), les modèles *seq2seq* - ceux utilisés avec le plus de succès - « peuvent être surpassés par des systèmes basés sur l'ingéniérie au niveau des sorties, ce en terme de qualité globale, aussi bien au niveau de la complexité, de la taille et de la diversité » [14].

Les modèles *seq2seq*, toujours d'après l'étude de Dušek et al. (2020), ont démontré leur potentiel, faisant d'excellents scores en terme de fluidité des sorties et de qualité de restitution des mots attendus.

La génération de texte nécessite une matière support à la génération. Trois grandes familles ont été évoquées. Nous allons commencer par une présentation de celles s'appuyant sur les bases de connaissances structurées, telles qu'avec descriptions en OWL (*Web Ontology Language*) à l'aide de triplets RDF (*Resource Description Framework*). Nous aborderons ensuite la transformation texte vers texte, puis clôturerons sur les représentations de sens (*Meaning Representation*, sémantiques ou syntaxico-sémantiques). Il s'agit là de l'essentiel des catégories support à la génération en langue naturelle. Elles sont fondamentales pour nos travaux, nous les avons utilisées.

14. *seq2seq models can be outperformed by hand-engineered systems in terms of overall quality, as well as complexity, length and diversity of outputs.*

I. 2. 1 RDF2Text : triplets RDF vers texte

Ainsi que le décrivent Bontcheva et Wilks (2004), le web sémantique, dont le langage est OWL (*Web Ontology Language*), cible l'étayage du langage html, fournissant une couche d'informations logique et organisée au web. Il s'agit de constituer des bases de connaissances au sein de base de données adressables (les informations, dites ressources, ont une adresse web), interconnectables et exploitables. Ces bases sont des Bases de Connaissances, ce sont des ontologies. Elles sont enrichissables par qui rédige une page web (typiquement pour un wiki sémantique [15]).
La cartouche latérale droite des pages de wikipédia voit ses informations, qui sont normalisées, stockées dans une base de connaissances, *dbpedia* (ex : nombre d'habitants d'un pays, famille d'une plante...)

La Figure I. 2. 1 illustre l'interconnectabilité de ces Bases de Connaissances.

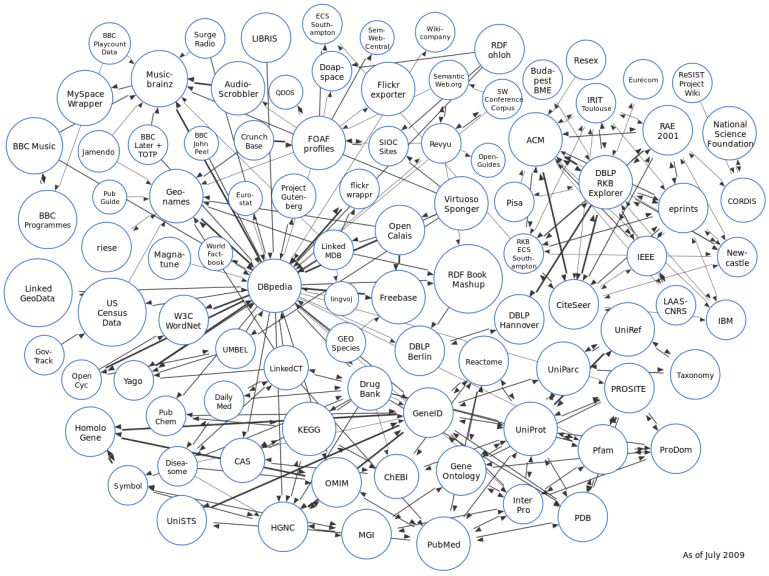


FIGURE I. 2. 1 – Représentation en carte heuristique des relations entre dbpedia et divers autres projets du web (2009).

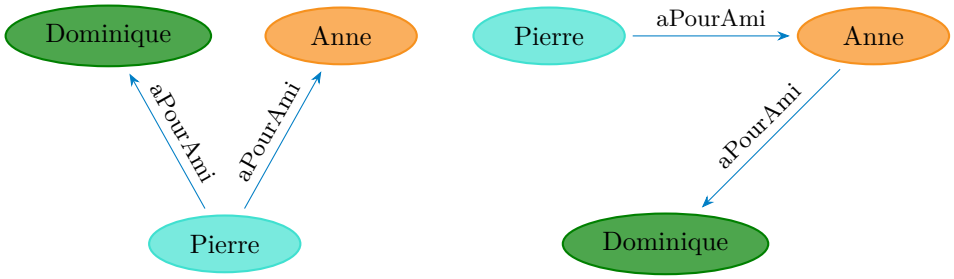
par richard cyganiak et anja jentzsch
creative commons attribution-share alike 3.0

Nous allons d'abord introduire quelques éléments permettant de décrypter la structuration des connaissances décrites en OWL avant d'évaluer l'état de l'art en matière de génération de texte depuis rdf (*resource description framework*) : la norme en matière de description.

Le RDF permet de représenter des connaissances : d'informer. Il est le socle du web sémantique, et vise le partage, la réutilisabilité des savoirs.

15. Web sémantique : extension du Web dont les normes sont un standard du World Wide Web Consortium (W3C)

Offrant une architecture logique, il autorise la représentation de l'information sous forme de graphe.



(Pierre, aPourAmi, Dominique)
(Pierre, aPourAmi, Anne)

(Pierre, aPourAmi, Anne)
(Anne, aPourAmi, Dominique)

FIGURE I. 2. 2 – Triplets RDF, représentation

I. 2. 1.1 RDF : le triplet

Un triplet RDF est composé d'un sujet, un prédicat, un objet. Le prédicat permet de décrire une relation entre des ressources. Une ressource est un objet, une représentation d'entité, un littéral x ou y comme une date, un nombre.

Des triplets sont connectables s'ils partagent au moins un même item en tant qu'objet ou sujet.

Selon la réalité à un temps t, nous pouvons avoir l'une des situations décrites en RDF Figure I. 2. 2.

RDF permet - comme la langue naturelle, une représentation du monde. Un graphe RDF est donc lexicalisable. Par exemple, le premier graphe de la Figure I. 2. 2 peut se voir traduit ainsi : « Pierre, qui a pour ami Dominique, est aussi ami avec Anne. » Le second peut recevoir cette lexicalisation : « Pierre a pour amie Anne, elle étant amie avec Dominique. »

Le RDF est le langage formel de ce qu'on appelle le *Web sémantique*, une part du Web formalisée par le *World Wide Web Consortium* (W3C) visant à fournir la connaissance sous forme de ressources décrites, annotées et manipulables. Il date de 1999.

L'ensemble des données des cartouches (blocs latéraux) sur wikipédia est disponible au format RDF dans une base nommée DBPedia. Il s'agit des informations socles (capitales pays, dates de naissance d'artistes, descriptions géographiques...). Librement accessibles, elles sont aisément exploitables informatiquement, et peuvent autoriser des calculs logiques (si Anne est tante de Béatrice, alors Béatrice est nièce de Anne).

Lexicaliser automatiquement ces données est donc un enjeu d'utilité publique et/ou commerciale que pose en préambule chaque équipe de chercheurs dont nous allons parler.

OWL fournit au RDF des foncteurs : ce sont des prédicats particuliers qui permettent d'ordonner et de hiérarchiser l'information. Nous avons par exemple les prédicats *type*, *classOf*, *subClassOf* qui permettent de définir des appartenances. La Figure I. 2. 3 nous montre un exemple d'association. Elle utilise du vocabulaire *foaf* qui permet d'informer des ressources correspondant à des êtres humains [16].

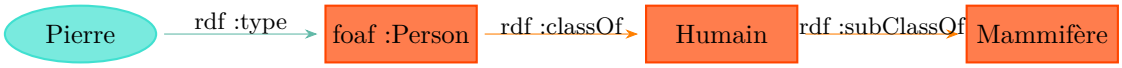


FIGURE I. 2. 3 – Triplets RDF, les functeurs

Les foncteurs permettent de poser des axiomes structurels.

I. 2. 1.2 RDF et génération

Ainsi que l'indiquent Berant et Liang (2014), « un défi central de l'analyse sémantique est de prendre en charge la myriade de possibilités que peut revêtir l'expression des prédicats issus des bases de connaissances ».

La génération depuis le RDF s'inscrit dans quelques tâches typiques du traitement naturel des langues, telles que :

- systèmes de dialogue : les triplets RDF contiennent des réponses que des humains se posent (Bordes et al., 2014; Zou et al., 2014),
- enrichissement : un système d'information, un texte, peut être enrichi automatiquement grâce à la manipulation de RDF (Zerva et Kopaneli, 2012),
- traduction automatique : un graphe RDF peut être socle à une expression dans autant de langues que gérées (Moussallem et al., 2018).

Le RDF vise à exprimer l'information sous une forme exploitable informatiquement, et de fait, il lève les ambiguïtés sémantiques, s'abstrait des ambiguïtés lexicales.

Partir du RDF pour générer du texte en langue naturelle amène à faire face aux difficultés et coûts (humains, financiers, temporels) classiques :

- avoir recours à un travail qualifié de linguiste pour
  - décrire la langue à l'aide de règles (grammaire, lexique) ou de patrons de génération (textes à trous plus ou moins améliorés),
  - gérer l'extension des règles voire gérer les incohérences entre règles.
- filtrer des phrases imparfaites (y compris issues de réseaux de neurones : ceux-ci sont prometteurs mais imparfaits (Zhu et al., 2019)).

L'usage de réseaux de neurones implique un apprentissage supervisé, donc des corpus exemples RDF vers texte.
Il est clairement plus aisé de construire des bases de connaissances RDF (Haidar-Ahmad,

16. Figure I. 2. 3 : foaf (*Friend Of A Friend*) : issu du vocabulaire permettant de décrire spécifiquement des humains, cf `http://xmlns.com/foaf/spec/` pour les spécifications.

2017) à partir d'une langue naturelle que l'inverse, ce que Gardent et al. (2017) ont contourné en faisant appel au *crowdsourcing*[17]. La création de phrases à partir d'informations librement codifiées est pour l'heure l'apanage des humains. La génération dynamique d'une codification à partir de phrases n'est pas triviale, mais est accessible.

La génération de texte à partir de bases de connaissances repose sur trois grands types d'approches :

- verbalisations de bases de connaissances OWL avec des méthodes basées sur les règles, comme Power et Third (2010),
- générations en rapprochant des triplets RDF de *templates* (textes à trous plus ou moins complexes), comme Duma et Klein (2013),
- générations avec réseaux de neurones, auxquelles Marcheggiani et Perez-Beltrachini (2018) ont participé.

Le couple web sémantique / génération de textes a une histoire dynamique. Les axiomes (voir foncteurs p. 38) ont été utilisé avec succès par Power et Third (2010) qui ont prouvé qu'un *mapping* entre des phrases en anglais et des informations génériques y correspondant permettait de traiter une grande partie des données enregistrées pour 200 ontologies (600 000 axiomes) : les 5 motifs (*patterns*) les plus fréquents dans les *templates* en langue naturelle couvrent plus de 90% des cas. La tâche (verbaliser le contenu d'une base de connaissances OWL) est donc possible.

axiome : Admiral $\sqsubseteq \exists$ CommanderOf.Fleet	(I. 2. 1)
Every admiral commands a fleet. (*Chaque amiral commande une flotte.*)	(I. 2. 2)

Verbalisation d'un axiome

source : *Complexity assumptions in ontology verbalisation* (Power et Third, 2010)

Power (2010) a montré que la complexité des termes et axiomes était le frein à la possibilité de pouvoir verbaliser le contenu d'une ontologie. La verbalisation ne peut être garantie scientifiquement. L'auteur écrit que :

- une ontologie peut comprendre des axiomes dont la verbalisation claire n'est pas envisageable (par exemple décrivant structurellement l'ontologie),
- une ontologie peut contenir des termes n'offrant pas d'entrée lexicale, comme des identifiants dépourvus de sens et non reliés à un label.

Même si les opérations logiques sont potentiellement complexes et que la complexité sémantique de la langue rend la verbalisation hasardeuse, en pratique, les ontologies sont construites par des humains qui utilisent un champ réduit de connexions entre ressources RDF : les contenus sont *de facto* relativement standardisé.

Power et Third (2010) attestent que le nombre de motifs (formes des graphes concrètement gérés) est réduit. Les ontologies sont théoriquement complexes, mais ils démontrent empiriquement que les données peuvent se révéler exploitables pour de la

17. *crowdsourcing* : intelligence des foules, il s'agit d'une production participative, ici l'externalisation du travail de lexicalisation alors confié à un grand nombre de personnes qui vont chacune écrire quelques phrases pour décrire un graphe RDF.

verbalisation.

Lampouras et Androutsopoulos (2013) utilisent eux la programmation linéaire en nombres entiers[18] pour parcourir les bases de connaissances OWL en sélectionnant, aggrégeant puis lexicalisant les données. L'ILP permet la sélection d'informations sous forme de graphes aux formes maîtrisées tout en contraignant l'importance des ressources sélectionnées (appartenant à un sous-ensemble validé, constitué d'éléments d'une fréquence minimale).
Ils ont utilisé un outil précédemment validé (Natural OWL (Galanis et Androutsopoulos, 2007)) pour générer des phrases simples et très courtes grâce à la pré-aggrégation de l'information (voir Figure I. 2. 4).

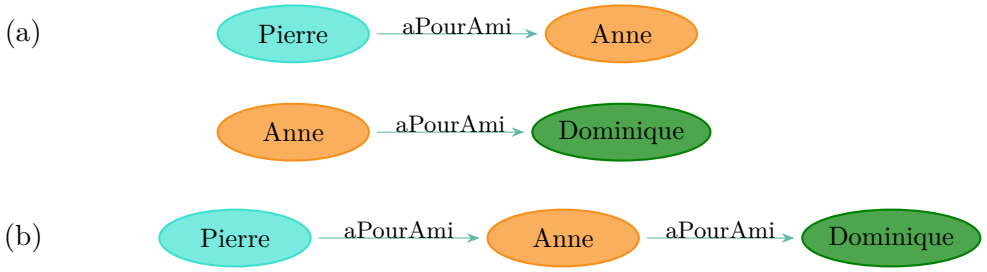


FIGURE I. 2. 4 – Aggrégation de l'information (a.*) → (b)

Galanis et al. (2009) avaient participé à une autre étude où Natural OWL a été utilisé pour générer des descriptions multi-lingues dans des univers en ligne (en l'occurence *Second Life*) pour illustrer des parcours caractéristiques de ceux des musées (ce que Galanis a aussi testé aux côtés de Vogiatzis et al. (2008)).

La génération de texte en langue naturelle cible des tâches utiles (système de dialogues, d'information du public) voire nobles, comme la production automatisée de définitions dans le domaine médical. Dans l'idée, il peut s'agir d'exploiter des informations à jour, venues de partout, pour verbaliser un savoir sur une petite problématique précise. Ainsi, Stevens et al. (2011) se sont attelés à la production automatisée d'explications depuis OWL. La conclusion est que la qualité des sorties en langue naturelle est tout à fait recevable, mais qu'elle ne peut être qu'un point de départ. Elle n'est *esthétiquement* pas acceptée (voir Figure I. 2. 5, *Machine Generated Definition* : définition informatiquement générée). Comme Lampouras et Androutsopoulos (2013), Stevens *et al.* regroupent les informations (ici pour une entité recherchée). Ils produisent ensuite des phrases puis les regroupent algorithmiquement en un paragraphe.

Pour la génération, ces chercheurs notent la facilité de production à l'aide d'outils spécifiques comme *Attempto Controlled English* (Kaljurand et Fuchs, 2007), Natural OWL évoqué précédemment ou encore MIAKT (Bontcheva et Wilks, 2004). Ces outils aux approches différentes exploitent le fait qu'un axiome OWL et une phrase en

18. *Integer Linear Programming*, ILP

langue naturelle ont une corrélation intuitive. Des lexiques permettent l'automatisation de la verbalisation. MIAKt gère l'agrégation des informations redondantes et fournit un lexique de base, une gestion de la planification du discours, il part simplement du RDF. Natural OWL est axé sur la logique, il exploite les normes OWL et le langage OWL-D qui supporte la logique de description SHOIN(D) [19]. Ces trois systèmes impliquent chacun une adaptation qui leur est spécifique (donc un coût temps utilisateur) à l'ontologie traitée. Les lexicalisations de prédicat ne peuvent pas systématiquement être dérivées automatiquement, encore moins sans vérification, des noms de prédicats (*hasBirthDay* est typiquement un prédicat à construction classique dont on peut mécaniquement obtenir un socle de verbalisation).

Attempto Controlled English est intéressant car il cible un usage générique. Il se focalise sur OWL 1.1, qui utilise des propriétés avancées par rapport au standard OWL initial.Et OWL 1.1 offre aux personnes participant à la construction des données une place normalisée pour de l'information littérale (ex : définir que *hasBirthDay* est un prédicat qui verbalisé en anglais donne *@subject has birth day @object* et en français *@subject est né le @object*), et que le prédicat inverse a pour le français la verbalisation *@object est la date de naissance de @subject*. L'article de Kaljurand conclut que la norme édictée par le W3C permet effectivement la verbalisation de façon concise et compréhensible, mais liste les problèmes encore ouverts : les conventions pour la définition des sujets objets ne permettent pas un cadrage fin de leurs spécificités morphologiques et et orthographique, ce que le développement d'outils d'aide à la saisie devrait pouvoir améliorer.

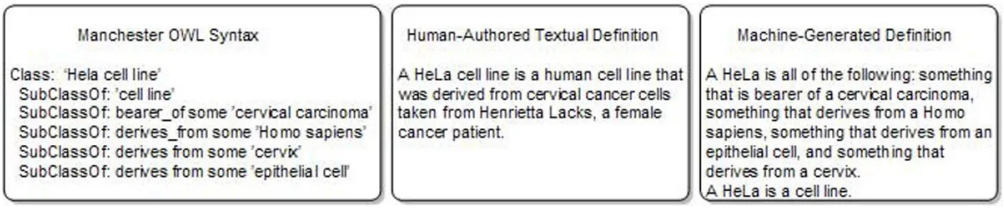


FIGURE I. 2. 5 – OWL et langage naturel, la lignée cellulaire HeLA.

source : Stevens et al. (2011)

Nous allons nous arrêter sur les travaux de Duma et Klein (2013), sur un point en particulier. Ils se demandent si « les données liées (*linked data*, concrètement les données du web sémantique) peuvent être utilisées comme point d'entrée d'un système de génération en langue naturelle. »

Ils ont utilisé le travail de Mihalcea et Csomai (2007) : le système de ceux-ci, un *wikifier* (*Wikify ! system*), fournit un couple extracteur de mots-clés/moteur de désambiguïsation qui permet, en quelque sorte, la reconnaissance d'entités nommées. La notion d'entité nommée est ici très large ; il s'agit là de toute ressource nommée susceptible de donner lieu à une page wikipédia.

19. OWL-DL Semantics :
https://www.obitko.com/tutorials/ontologies-semantic-web/owl-dl-semantics.html

Duma et Klein ont sélectionné des pages pertinentes, des triplets rdf structurés dans un graphe de profondeur 1 (voir Figure I. 2. 6), écartant les phrases dont des informations (ex : *of the Baroque period* Figure I. 2. 6)) ont échoué à être appairées avec le RDF. C'est la part la plus notable de l'étape de filtrage imposée à leurs sélections.

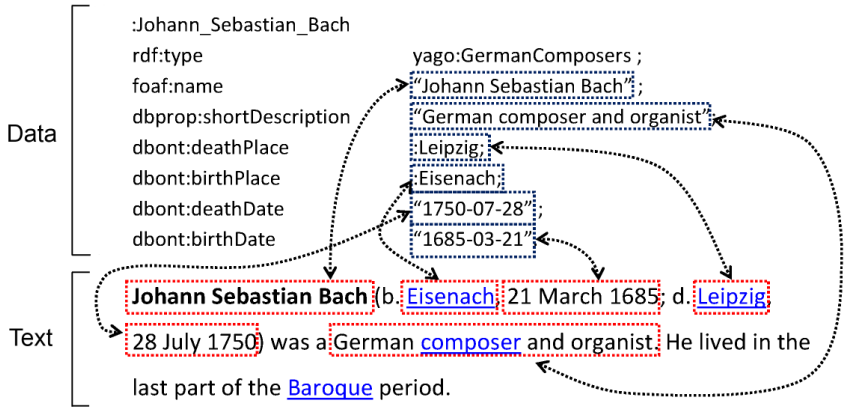


FIGURE I. 2. 6 – Aligning Data and Text, Figure 3 (Duma et Klein, 2013).

Le graphe RDF de la Figure I. 2. 6 est un graphe typique. Les informations ici fournies pour Jean-Sébastien Bach sont parallèlement disponibles pour la plupart des personnalités. Les phrases extraites, nettoyées des ressources issues du RDF (objets des prédicats pour la Figure I. 2. 6) deviennent des *templates*, des "copies vides", réutilisables pour d'autres célébrités que celle les ayant fournies. Leur approche est plus fine que ce résumé, mais l'idée est là. Ils ont aussi suivi Duboue et McKeown (2003) pour être à même de grouper les *templates* et produire des textes cohérents. Partant de 268 articles, Duma et Klein ont obtenu 199 *templates* dont seuls 74 ont passé l'étape de filtrage. Ils n'en ont jugé que 43 comme grammaticalement correctes. Les informations support à la génération ont été confiées à des humains pour qu'ils produisent des textes. Cette génération pleinement naturelle est comme habituellement nettement supérieure aux générations automatisées, la leur comprise. Néanmoins, leur système surpasse nettement la référence (où chaque triplet est réalisé dans une seule et unique phrase, travaux de Sun et Mellish (2007)) en terme de non-redondance, de structure et de cohérence, le tout en améliorant un peu la qualité grammaticale.

Les réseaux de neurones, qui appartiennent à la gamme des outils disponibles, ne sont pas le sésame magique et exclusif sur lequel comptent les chercheurs. La génération de RDF vers Texte est un domaine actif. Nous allons maintenant faire un petit bond dans le temps pour nous rendre en 2018, année de deux travaux particuliers, de Trisedya et al. (2018), de Marcheggiani et Perez-Beltrachini (2018). Les réseaux de neurones ont là pleinement fait leur entrée.

Trisedja *et al.* proposent un modèle neuronal (GTR-LSTM) sur socle des données de WebNLG (Gardent et al., 2017), un ensemble de 25 298 lots de triplets RDF associés à du texte en lexicalisant le contenu. Ils proposent une interface encodant sous forme de graphe les entrées fournies au réseau. Pour le construire, ils ont exploité Keras (Chollet et al., 2015) qui est une boîte à outils permettant une structuration modulaire de réseaux de neurones. Ils l'ont comparée à deux autres modèles. Ces modèles, ainsi que le leur, sont trois encodeurs-décodeurs, la partie décodeur étant pour chacun un simple LSTM.

(	John_Doe	,	birth_place	,	London	)

TABLE I. 2. 1 – Exemple de triplet Trisedya et al. (2018).

Dans les deux premiers modèles, les entités composants les triplets sont décomposées. Pour l'exemple Table I. 2. 1 , le prédicat *birth_place* s'établit sur deux *tokens*, le sujet *John_Doe* aussi, *London* un seul. C'est sur la partie encodeur qu'il y a différentiel, pour les modèles :

1. *Adapted Standard BLSTM Encoder* : l'entrée traite la séquence, les vecteurs mots des triplets présentés dans un ordonnancement non spécifié dans l'article, voir p.24 pour le LSTM, avec mécanisme d'attention (voir p.32 de Bahdanau et al. (2014b)) implémenté.

2. *Adapted Standard Triple Encoder* : les auteurs alignent le contenu des triplets au sein d'un espace vectoriel de même dimension : chaque triplet est complété (*zero padding*) pour que s'enchaînent en entrée des triplets de même taille.

3. GTR-LSTM : leur système phare, un *Graph-based Triple encoder* pour RDF. Cette fois, les entités composant les triplets ne sont plus décomposées, *John_Doe* donne lieu à un seul *token* : la représentation de l'objet vient se coupler avec celle du prédicat alors que le sujet du triplet en cours de traitement vient d'être traitée. La représentation vectorielle du sujet est maintenue en mémoire pour être réutilisée si elle est à nouveau nécessaire pour un couplage ultérieur au sein d'une même séquence (voir Figure I. 2. 7)

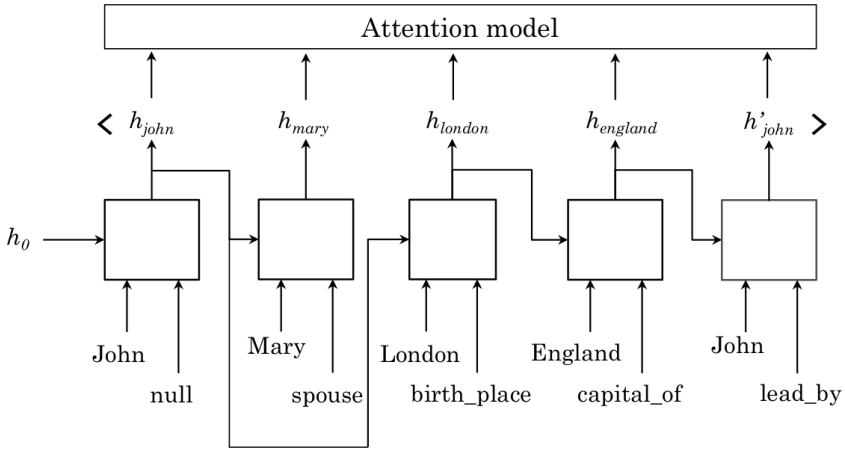


FIGURE I. 2. 7 – *GTR-LSTM triple encoder.*

source : Trisedya et al. (2018)

Les entités (comme *John_Doe*) sont anonymisées et typées [20], et si les évaluations humaines indiquent que les phrases générées sur données non vues sont loin de la perfection (phrases correctes 1.96/5, grammaire 2.04/5, fluidité 1.99/5), le modèle n°3, GTR-LSTM, surpasse nettement les autres modèles testés. Nous sommes néanmoins loin des scores qualitatifs obtenus, par exemple, en simplification par des méthodes non neuronales.

La même année, Marcheggiani et Perez-Beltrachini (2018) ont eux aussi opté pour un système encodeur/décodeur dont l'encodeur est un réseau convolutionnel pour les graphes (GCN : *Graph Convolutional Network*). Partant eux aussi des données WebNLG (Gardent et al., 2017), ils se sont intéressés à la modélisation de l'aspect structuré des données entrantes (triplets RDF en entrée, phrases en sortie). Ils ont utilisé comme les précédents un décodeur à base de LSTM.

Il n'y a pas chez eux d'évaluation humaine, mais alors que Trisedya et al. (2018) obtenaient un BLEU sur donnée non vues de 34.1, eux montent à 66.6. Ils ont comme eux anonymisé et typé les entités (avec Stanford CoreNLP (Manning et al., 2014)). Avec un modèle affiné de GCN (Scarselli et al., 2008) composé de six couches, ils ont eu l'idée de linéariser en utilisant le hasard pour le parcours de graphe. Ils parcourent le graphe en profondeur et choisissent aléatoirement les voisins à traiter. Cette méthode empêche le réseau de se baser sur la régularité de linéarisation pour gérer l'information. Trisedja *et al.* avaient eux choisi une exploration systématique : tri topologique [21] et puis parcours en largeur [22].

Les réseaux de neurones forment un espoir pour améliorer les performances de la génération RDF vers texte, il en est de même pour la génération Texte vers Texte que nous allons maintenant aborder.

20. typage réalisé avec DBPedia lookup (https://wiki.dbpedia.org/lookup), exemple tagguer "London" avec "LOCATION", "John_Doe" avec "PERSON"
21. Tri topologique :https://fr.wikipedia.org/wiki/Tri_topologique
22. Parcours en largeur :https://fr.wikipedia.org/wiki/Algorithme_de_parcours_en_largeur

I. 2. 2 Text2Text : texte vers texte

Pour le grand public, le problème typique adressé par la génération de texte depuis du texte est la traduction.

Le passage d'un texte depuis une langue vers une autre est une tâche difficile, mais une pierre de Rosette numérique serait la seule solution pour faire un passage à l'échelle du Web. 25% des internautes sont anglophones (contre 4.1% de francophones), plus de 55% des sites internet sont en anglais (en octobre 2019, source w3techs [23]). La traduction manuelle est inenvisageable pour traiter des contenus chaque jour plus nombreux, allant du sous-titrage de vidéos, au suivi d'actualités en passant par la littérature scientifique.

Nous ne nous arrêterons que brièvement sur la traduction car elle sort partiellement de notre champs d'étude.

I. 2. 2.1 La traduction

Les modèles statistiques ont longtemps été ce qui se faisait de mieux pour passer d'un texte vers un autre, l'un étant corrélé à l'autre. Les modèles statistiques combinaient un modèle d'alignement appris à partir de corpus parallèles avec un modèle de langue (Koehn (2005)) ; maintenant les modèles neuronaux prédominent. Les travaux s'intéressent particulièrement au modèle encodeur - décodeur (Sutskever et al. (2011))

Produire le texte grâce aux statistiques requiert des corpus parallèles larges. Les corpus d'apprentissage restent un souci clé pour les statistiques, et sont insuffisants.

Qui plus est, ainsi que l'indique Koehn (2005), les performances sont inégales selon la direction de la traduction (ex : anglais vers espagnol ou espagnol vers anglais). Il a travaillé à partir des données du parlement européen (corpus parallèles en 11 langues). Il note également que le manque de support pour repérer les erreurs de langue est critique.

L'évaluation de la qualité est d'ailleurs un problème majeur.

Des métriques telles que BLEU (Papineni et al., 2002) ou METEOR (Banerjee et Lavie, 2005) sont plus utiles pour comparer les travaux, se faire une idée globale, que pour comprendre les erreurs.

Ce problème est commun à toutes les classes de génération texte vers texte. Nous allons maintenant nous orienter vers les domaines de recherche plus en phase avec nos propres travaux : la simplification de phrase, la compression de phrase, et enfin la paraphrase.

I. 2. 2.2 La simplification de phrase

La simplification consiste à transformer un texte en un autre, étant attendu que celui-ci soit plus lisible, plus accessible. Elle peut aussi servir à affiner et alléger le travail des analyseurs syntaxiques (simplification de la complexité grammaticale).

Elle recoupe tout ou partie de quatre opérations :

23. https://w3techs.com/technologies/history_overview/content_language, consulté le 30/10/2019

- **découpe** : une longue phrase est découpée en plusieurs petites phrases : *Le chaton qui miaule a très faim.* devient *Le chaton miaule. Il a très faim.*
- **suppression** : un mot non essentiel est supprimé. La disparition fonctionnelle parfois opérée lors d'une découpe ne correspond pas à cette action. Ainsi, la perte du pronom *qui* dans *qui miaule* ou de tout mot-outil dans le cadre d'une restructuration n'est pas une action de simplification nommée **suppression**. *Cet homme est aussi beau que séduisant.* pourra être simplifié en *Cet homme est séduisant.*
- **réordonnancement** : il s'agit d'une opération liée au maintien, au renforcement de la cohérence du texte (ex : modification de l'ordre sujet verbe complément, CF travaux de Watanabe et al. (2009), qui ont approfondi la question),
- **substitution** : remplacer un mot complexe par une expression plus claire, par exemple remplacer *le stomatologue* par une expression plus simple : *le chirurgien dentiste.*

Zhu et al. (2010) ont mis en œuvre avec succès les quatre opérations en 2010. Pour ce faire, ils ont mis en place un système de quatre algorithmes après un travail d'alignement entre des phrases issues de pages wikipédia et de pages qui leurs étaient miroir sur *simple wikipedia*[24]. Les phrases issues du premier site étant considérées complexes, et chacune rapprochée (ou exclue des traitements) de plusieurs petites phrases successives au contenu correspondant (une phrase complexe pour une à n phrases simples, voir Zhu et al. (2010), partie *Monolingual Sentence Alignment* pour plus de détails). Zhu et al. (2010) réalisent un apprentissage des probabilités d'effectuer chacune des opérations (découpe, suppression, réordonnancement et substitution), en s'inspirant du modèle de traduction statistique de Yamada et Knight (2001), un modèle de langage visant à améliorer la grammaticalité. À partir de ce modèle, le décodeur "traduit' les phrases complexes en phrases simples en suivant une approche "*greedy*".

Une grammaire quasi synchrone a été utilisée par Woodsend et Lapata (2011) : ceux-ci ont exploité le fait que la grammaire des phrases complexes n'est pas la même que celle des phrases simples. Ils publient un travail où l'alignement des arbres grammaticaux de phrases complexes avec celui des phrases simples (une phrase complexe associée à deux simples dans leur corpus) est appris. À partir des exemples travaillés, la résolution de la découpe est évaluée comme un programme linéaire en nombres entiers : des variables décisionnelles issues des pré-traitements (touchant aussi les variations lexicales) permettent de décider ce qui est gardé, déplacé, remplacé. Leur corpus est comme celui de Zhu et al. (2010), issu de wikipédia et de son miroir simplifié, simple wikipédia. Les simplifications lexicales sont elles aussi automatiquement détectées pendant la constitution des statistiques. Le modèle de Zhu et al. (2010) obtient, notent-ils, globalement de meilleurs résultats et une meilleure couverture que le leur. Néanmoins la méthode suivie a le mérite de ne pas offrir de différences significatives en terme de simplicité, de

24. http://simple.wikipedia.org

grammaire et de sens avec les références de simple wikipédia. Avec cette méthode très automatisée, ils parviennent à surpasser Zhu *et al.* en terme de préservation du sens.

Wubben et al. (2012) utilisent un système de traduction automatique beaucoup plus simple que Zhu et al. (2010), ce dernier s'appuyant sur la syntaxe contrairement aux systèmes de traduction automatique classiques. La contribution principale de Wubben et al. (2012) est de compléter un système de traduction automatique classique (Moses) avec une procédure de *re-ranking* (re-ordonnancement des sorties) pour favoriser les sorties qui sont les plus différentes de l'entrée. Une évaluation par l'humain sur 20 phrases montre une amélioration en termes de grammaticalité et de précision sémantique par rapport à Zhu et al. (2010) et Woodsend et Lapata (2011).

Coster et Kauchak (2011) se concentrent sur le pôle suppression de mots, y voyant un levier d'amélioration des sorties. Zhu *et al.* y avaient constaté un point d'achoppement de leur méthode. La contribution principale de Coster et Kauchak (2011) est de modifier le modèle d'alignement des mots (utilisant Giza++) pour permettre les alignements nuls et ainsi favoriser les effacements nécessaires à la simplification. Les résultats montrent une légère amélioration par rapport à une approche utilisant la version basée sur les phrases de Moses excluant la procédure de gestion des suppressions.

Réutilisant le même corpus, Narayan et Gardent (2014) proposent un système hybride, basé sur une représentation sémantique profonde de l'entrée réalisée avec Boxer (Curran et al., 2007), représentation sémantique qui sert de socle pour découper les phrases et effacer des constituants. Un modèle de traduction automatique est ensuite utilisé pour les opérations de substitutions et de réordonnancement. Les résultats (évaluation humaine et automatique) montrent une amélioration par rapport à l'état de l'art.

Approches neuronales La simplification ne se limite pas à ces travaux, et nous voudrions encore vous présenter un travail : Zhang et Lapata (2017) utilisent un modèle neuronal de structure encodeur/décodeur dans lequel ils implémentent un algorithme de réenforcement. Leur base est un modèle somme toute classique avec un LSTM côté encodeur, un LSTM côté décodeur, et l'intégration du fameux mécanisme d'attention (voir p.32).

Zhang *et alii* ont mis en place un système d'agent [25] dans le cadre d'un algorithme de ré-renforcement : un système de récompense inversé qui module les poids. Évaluant les contributions neuronales aux sorties, c'est lui qui fournit les valeurs permettant la rétropropagation du gradient (rétropropagation : p.23). Les auteurs utilisent une prégénération de statistiques et une mise en œuvre progressive pour obtenir des résultats à minima équivalents à ceux d'un système encodeur/décodeur sans agent. L'algorithme calculant la récompense exploite trois métriques pour accorder les récompenses : simplicité (avec SARI (Xu et al., 2016)), conformité (*Sequence Auto-Encoder*, SAE (Dai et Le, 2015)), fluidité avec un modèle de langue entraîné sur les phrases simples.

Ils ont par ailleurs entraîné un modèle LSTM à effectuer des simplifications lexicales. L'algorithme de renforcement intègre les sorties de ce modèle (qui fournit des probabi-

25. un agent est une entité modélisée : ici l'encodeur-décodeur est vu comme un agent qui lit une phrase et dispose d'un système de règles pour intervenir sur les apprentissages.

lités de bon alignement avec des mots plus simples que ceux de l'entrée). Zhang *et al.* ont utilisé cette solution globale sur le corpus *wikipedia* (simple et complexe) avec une évaluation humaine sur 205 paires de phrases.

La méthode encodeur/décodeur pour la simplification est transposable à la génération de paraphrases mais les contraintes en simplification sont différentes de celles de notre travail (pas de suppression ni de découpage).

I. 2. 2.3 La compression

La compression consiste à transformer un texte en un autre, étant attendu que l'information non essentielle en soit écartée, avec une signification globale conservée.

Deux mécanismes sont particulièrement utiles à la compression, celui de copie, clé pour le traitement des mots rares ou absents du corpus d'apprentissage, et celui de couverture qui permet d'améliorer l'adéquation sémantique (le fait que le texte généré reflète bien le contenu donné en entrée). Le premier est issu de recherches plus fondamentales (Vinyals et al., 2015) ; le second nous vient du traitement automatique des langues (introduit par Tu et al. (2016)).

Si la simplification est un type particulier de paraphrase, la compression vise à restituer l'essentiel de l'information, mais aussi et avant tout, à sélectionner, à évaluer un seuil à partir duquel l'information est considérée comme superflue. Bien entendu, il s'agit de générer un texte correct et cohérent en langue naturelle.

La compression est du résumé automatique, qui peut prendre la forme d'un titre, comme dans le cadre des travaux de Dorr et al. (2003) (génération à base de règles). Les travaux se focalisent principalement sur la suppression de mots, de parties de phrases "inutiles". Nous allons directement aller aux approches neuronales, qui sont riches d'enseignement, avec Vinyals et al. (2015), qui introduisent le mécanisme de copie, et See et al. (2017) qui le mettent en œuvre dans un système hybride.

Nous allons nous arrêter sur les deux mécanismes précités.

Mécanisme de copie Vinyals et ses collègues proposent une technique, celle du "Pointer Networks", que l'art retient sous l'appelation *copy mecanism*. Découlant du mécanisme d'attention présenté page 32, ce mécanisme traite le vocabulaire de sortie comme restreint au vocabulaire d'entrée. C'est à dire que la taille du dictionnaire de sortie est fixe, mais restreinte aux vecteurs (représentations vectorielles) de la séquence entrée : un *softmax* (voir Figure I. 1. 10) sélectionnant le *token* à replacer en sortie.

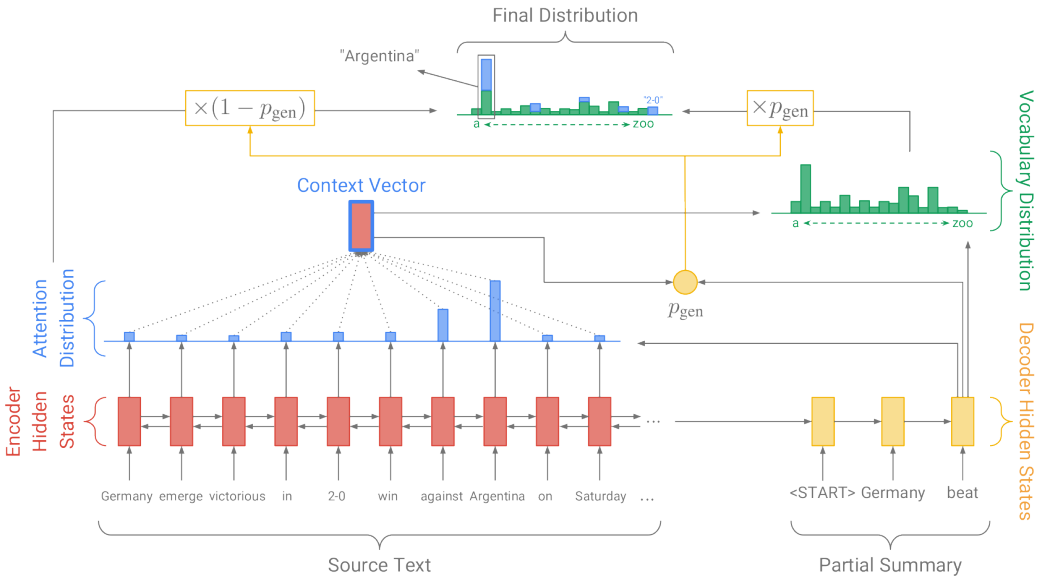


FIGURE I. 2. 8 – Mécanisme de copie.

source : *Get To The Point : Summarization with Pointer-Generator Networks* See et al. (2017)

Figure I. 2. 8, pour chaque étape de décodage (*decoder timestep*), une probabilité $p_{gen} \in [0, 1]$ est calculée, avec évaluation de la probabilité de générer des mots à partir du vocabulaire de sortie (VOC-S) et des mots du texte source. La distribution du vocabulaire et la distribution de l'attention sont pondérées et additionnées pour obtenir la distribution finale, à partir de laquelle la prédiction est faite. Les mots du VOC-S (ex : `Argentina`) et les mots hors VOC-S tels que `2-0` sont inclus dans la distribution finale.

L'intuition générale est que l'encodeur/décodeur qu'ils utilisent, aux couches composées de cellules LSTM (voir p.24), apprend à ordonnancer les entrées. Vinyals *et alii* le mettent donc à l'épreuve sur des problèmes du type "voyageur de commerce". Si cette problématique n'est pas la nôtre, les modalités d'utilisation de ce mécanisme sont transposables à notre cas. See et al. (2017) ont su démontrer que son intérêt dans la génération texte vers texte, et plus spécifiquement la compression, est grand.

Non seulement See et al. (2017) ont mis en œuvre le mécanisme de copie de manière parcellaire (ils génèrent un texte sur un dictionnaire plus étendu que celui des mots de la séquence d'entrée), mais ils l'ont adossé au mécanisme dit de couverture. Ce mécanisme vise initialement à contrecarrer deux phénomènes connus de la génération séquence vers séquence : répétitions ou *a contrario* absences de mots.

Mécanisme de couverture Les sorties du mécanisme d'attention (voir Figure I. 1. 13) sont maintenues en mémoire le temps du traitement d'une séquence, sous forme - bien entendu - d'un vecteur (taille fixe, chaque sortie vient s'additionner à la précédente).

Nous devons le mécanisme de couverture à Tu et al. (2016) qui travaillent sur la traduction automatique. Le mécanisme d'attention applique un *softmax* sur les états cachés des mots précédemment traités, ce avec une pondération apprise. La sortie du

mécanisme d'attention permet, étape après étape, de savoir quels mots ont été traités.

Ce contexte maintenu en mémoire permet de favoriser spécifiquement des sorties qui devraient être ou de pénaliser celles qui ont déjà été traitées. See et al. (2017) ont développé pour cette partie du réseau de neurones un calcul de l'erreur spécifique pénalisant les sorties répétées.

Hybridation copie/couverture Le vecteur de couverture participe étape par étape au calcul de l'attention. See *et al.* l'ont implémenté, et ont ajouté une version modifiée du mécanisme de copie de Vinyals et ses collègues : ils calculent à chaque étape les probabilités d'obtenir une erreur plus faible en copiant un mot issu de l'entrée ou en exploitant le vocabulaire de sortie. En fonction de ce calcul, c'est un mot issu de l'entrée ou un mot issu de la sortie qui est généré. Ils ont noté que la fréquence de chaque mot dans l'entrée est corrélée à son usage en sortie.

Sur les données de test, le modèle obtient 17.32% de sorties parfaitement exactes (METEOR) avec la copie et de couverture, 15.35% sans la couverture, et 11.65% quand le modèle entraîné est un séquence vers séquence dépourvu de ces deux mécanismes (tout en conservant celui de l'attention).

Une approche alternative pour une gestion des mots rares est la délexicalisation : au lieu de les chercher dans l'entrée pour les copier en sortie, on les remplace automatiquement par des marqueurs retrouvés en sortie.

I. 2. 2.4 La paraphrase

Paraphraser consiste à reformuler un texte pour en restituer le contenu sous une autre forme (lexicale et/ou syntaxique).

Le travail de recherche autour des paraphrases est relativement transversal.

Riezler et al. (2007) l'ont démontré utile pour étendre les bases de données requêtables pour des questions/réponses . De manière similaire, Wieting et al. (2015b) ciblent la production de données additionnelles pour de l'apprentissage automatique et la constitution de base de données tout comme c'est le cas en traduction automatique, ainsi que le notent Madnani et Dorr (2010) dans leur étude de l'état de l'art.

Trois types de modèles principaux ont été utilisés avant les réseaux de neurones pour générer des paraphrases : basé sur règles (McKeown, 1983), sur statistiques (Quirk et al., 2004; Zhao et al., 2008; Wubben et al., 2010) (SMT, système de traduction automatique statistique), à partir de synonymes (avec WordNet (Fellbaum, 1998)) couplé à des statistiques et des règles pour acter ou non un remplacement de termes (Bolshakov et Gelbukh, 2004)).

En 2016, Narayan et al. (2016) ont exploré la génération de paraphrases en couplant différentes techniques : PCFGs (*Probabilistic Context-Free Grammars*[26], voir illustration Figure I. 2. 9), filtrage (MT metrics (Madnani et al., 2012)). Ils se basent pour

26. PCFG : les formes que prend le texte sont analysées probabilistiquement, indépendamment du sens, et des probabilités sont associées aux structures envisageables. Une modèle de grammaire entraîné sur un ensemble de texte permet d'obtenir un arbre probable pour un nouveau texte.

l'entraînement sur le corpus Paralex qui contient 18 millions de paires de questions paraphrasées, et utilisent le couple grammaire/paraphrases de Paralex pour construire des treillis parallélisant des synonymes, ce qui leur permet ensuite d'être à même de générer de nouvelles paraphrases (voir Figure I. 2. 10).

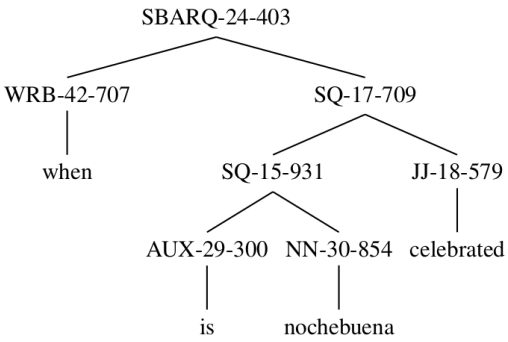
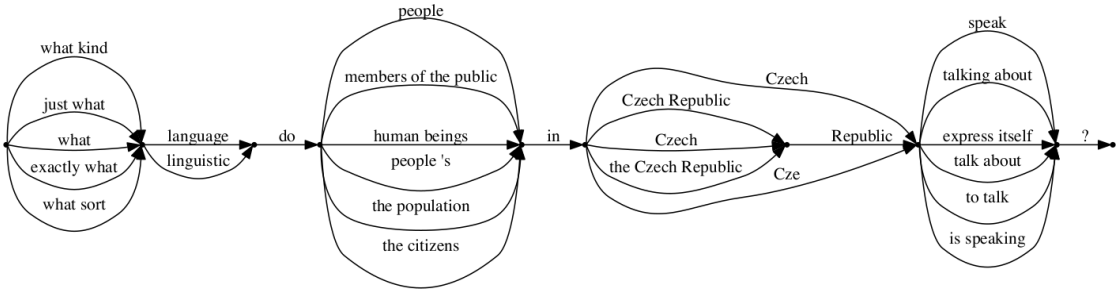


FIGURE I. 2. 9 – Analyse PCFG : exemple, *When is nochebuena celebrated*

issu de la Figure 2, *Paraphrase Generation from Latent-Variable PCFGs for Semantic Parsing* Narayan et al. *(2016)*



An example word lattice for the question *What language do people in Czech Republic speak?* using the lexical and phrasal rules from the PPDB.

FIGURE I. 2. 10 – Exemple de treillis de mots.

depuis la Figure 1, *Paraphrase Generation from Latent-Variable PCFGs for Semantic Parsing* Narayan et al. *(2016)*

Narayan et al. (2016) ne génèrent pas toutes les phrases possibles ; la jonction cartésienne des informations est bien trop généreuse pour être gérable. Ils effectuent un échantillonnage. Cet échantillonage est ensuite filtré. Les lots de paraphrases sont ensuite exploités pour réaliser une analyse sémantique, 85% des lots ayant au moins une forme labelisée donc le graphe est isomorphe à un graphe de la base de connaissances nommée *Freebase*[27]. Cette analyse sémantique est utilisée pour effectuer l'analyse d'erreur.

27. *Freebase* : 1,9 milliards de relations RDF stockées sont formes de graphes, http://rdf.freebase.com

L'année suivante, Mallinson et al. (2017) revisitent la génération paraphrastique à l'aide des méthodes de la traduction automatique neuronale, que nous évoquerons plus précisément lors de la présentation de nos travaux (voir page 73).

Dans un cadre plus général, il nous paraît plus important de présenter trois apports complémentaires Iyyer et al. (2018); Chen et al. (2019); Li et al. (2019).

La diversité syntaxique au sein des ressources dont nous disposons en traitement automatique des langues est faible, ainsi que le notent Iyyer et al. (2018) . La génération de paraphrases syntaxiques - qui doivent respecter par définition la sémantique d'origine, reste un défi pleinement ouvert. Les travaux à ce niveau restent balbutiants, travaux qui ne sont pas forcément auréolés de succès clairs et indéniables, fussent-ils intéressants. Chen et al. (2019) marquent une avancée dans le domaine, pour les questions et méthodes qu'ils avancent : proposition pour séparer syntaxe et sémantique (impact, faisabilité), possibilités techniques actuelles (quels leviers technologiques à mettre en place face au défi).

Chen et al. (2019) exploitent un auto-encodeur variationnel, introduit par Kingma et Welling (2013). Celui-ci est un encodeur/décodeur très spécifique : il fait usage de techniques probabilistiques, menant proche de l'analyse en composantes principales. Cette architecture vise la réduction dimensionnelle des informations, c'est à dire la perte d'informations ciblées. Concrètement, les auteurs utilisent deux variables latentes, sémantiques et syntaxiques. Leur but est de produire une phrase Z à partir de la sémantique d'une phrase X et de la syntaxe d'une phrase Y.

La phrase X permet la production de la variable sémantique. Un auto-encodeur de type bi-LSTM [28] traite cette phrase en entrée/sortie. Les auteurs font la moyenne des vecteurs de sorties, suivant Wieting et al. (2015a). Wieting et al. (2015a) ont observé que effectuer une moyenne sur les vecteurs de mots était efficace pour les tâches sémantiques. Il en découle pour les auteurs que cette moyenne est une représentation sémantique valide pour calculer la variable latente attendue.

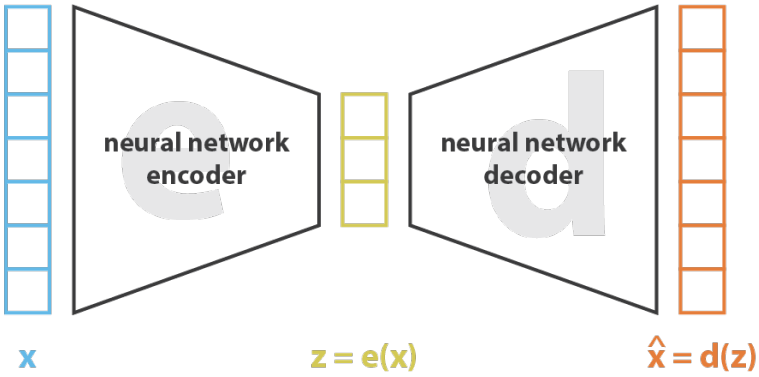
Nous allons marquer un arrêt sur les auto-encodeurs variationnels (*variational autoencoders*) afin d'approcher la technique exploitée. Pour cela, nous nous appuyons un blog didactique, *towardsdatascience* [29] (article de Joseph Rocca et Baptiste Rocca).

Chen et al. (2019) n'utilisent pas directement les vecteurs obtenus en effectuant la moyenne des vecteurs des mots (représentation sémantique de la phrase X), mais une valeur normalisée suivant une distribution von Mises-Fisher, analogue à une distribution gaussienne, si ce n'est qu'elle se situe dans un hyperespace. Il s'agit là de leur variable latente sémantique. Un auto-encodeur variationnel est chargé de la normalisation marginalisée des variables latentes.

Pour rappel, un auto-encodeur répond au fonctionnement illustré Figure I. 2. 12.

28. LSTM (voir page 24) bidirectionnel ou bi-LSTM : un LSTM dont les entrées sont traitées dans l'ordre et dans l'ordre inverse par deux couches parallèles.

29. https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73, consulté le 9 janvier 2020



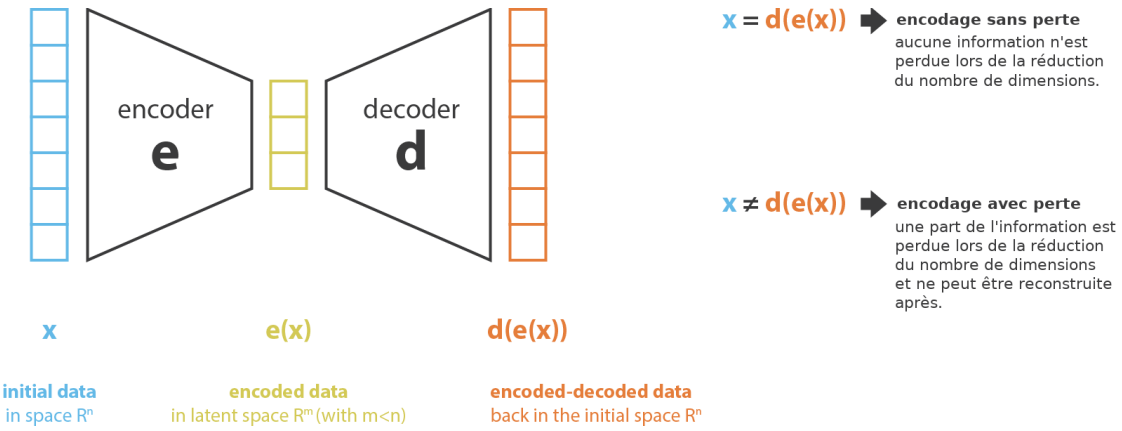
$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

x : entrées dans l'espace R^n, z : valeurs de x encodées par e, e : fonction de réduction de la dimensionalité avec x encodé dans l'espace latent R^m avec $m < n$, d : fonction d'extension de la dimensionalité avec $d(e(x))$ assurant le décodage des données (retour dans l'espace initial R), $\hat{x}$ étant une approximation de x.

FIGURE I. 2. 11 – Illustration d'un auto-encodeur avec sa fonction *loss*

D'après Joseph Rocca, *understanding variational autoencoders*

Un auto-encodeur cible la réduction de la dimensionnalité lors de l'encodage - y compris sans perte si les paramètres permettent un apprentissage "par cœur". (*overfitting*, voir page 25). Le décodage permet la reconstruction de l'information, à l'identique ou sous une autre forme ($x \rightarrow x'$).

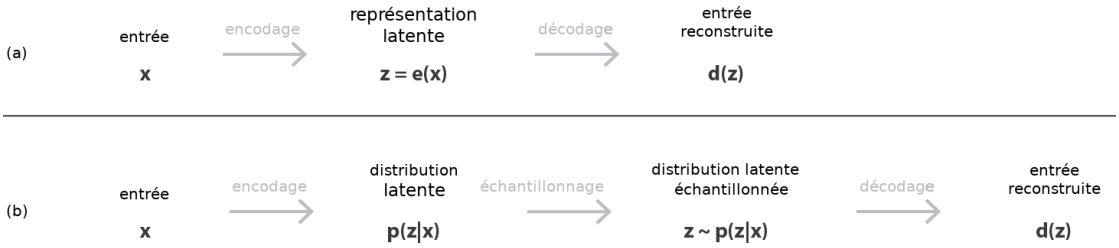


x : entrées dans l'espace R^n, z : valeurs de x encodées par e, e : fonction de réduction de la dimensionalité avec x encodé dans l'espace latent R^m avec $m < n$, d : fonction d'extension de la dimensionalité avec $d(e(x))$ assurant le décodage des données (retour dans l'espace initial R), $\hat{x}$ étant une approximation de x.

FIGURE I. 2. 12 – Illustration du principe de réduction de dimensionnalité dans un encodeur/décodeur

D'après Joseph Rocca, *understanding variational autoencoders*

Un auto-encodeur variationnel redistribue dans l'espace les valeurs issues de l'encodeur afin de les normaliser.

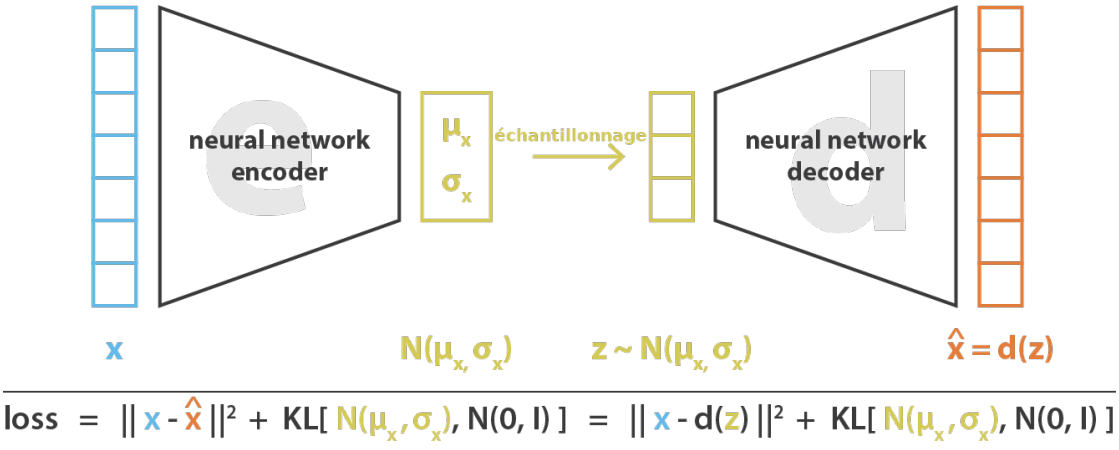


(a) et (b) x : entrées dans l'espace R^n, d : fonction d'extension de la dimensionalité assurant le décodage des données (retour dans l'espace initial R^n),
(a) z : valeurs de x encodées par e, e : fonction de réduction de la dimensionalité avec x encodé dans l'espace latent R^m avec $m < n$.
(b) $p(z|x)$: valeurs probables issues d'un espace latent normalisé R^m avec $m < n$ (Figure I. 2. 15) selon distribution étant donné dz pour x, z valeur sélectionnée dans R^m selon probabilités qu'elle permette de reconstruire x (Figure I. 2. 14).

FIGURE I. 2. 13 – Différence entre un auto-encodeur ((a) déterministe) et un auto-encodeur variationnel ((b) probabiliste)

D'après Joseph Rocca, *understanding variational autoencoders*

Les valeurs issues de l'encodeur ne sont pas communiquées directement au décodeur : elles sont utilisées pour générer une valeur normalisée qui est fonction de l'ensemble des représentations traitées.



x : entrées dans l'espace R^n, $d(z)$: reconstruction de la valeur échantillonnée dans l'espace R^n (donne $\hat{x}$, approximation de x), N espace des "observations", (μ_x, σ_x) : moyenne et covariance des "observations", z échantillon probable pour obtenir $d(x) = \hat{x}$ sachant $N(\mu_x, \sigma_x)$ pour x, KL : divergence entre la distribution retournée et une distribution gaussienne (divergence Kulback-Leibler), (O, I) : moyenne et covariance dans l'espace de référence (ici gaussien).

FIGURE I. 2. 14 – Dans les auto-encodeurs variationnels, il y a reconstruction (calcul d'erreur entrée/sortie) et régulation (pour rendre l'espace latent régulier).

D'après Joseph Rocca, *understanding variational autoencoders*

La régularisation présentée Figure I. 2. 15 permet une répartition spatiale continue et cohérente des données.

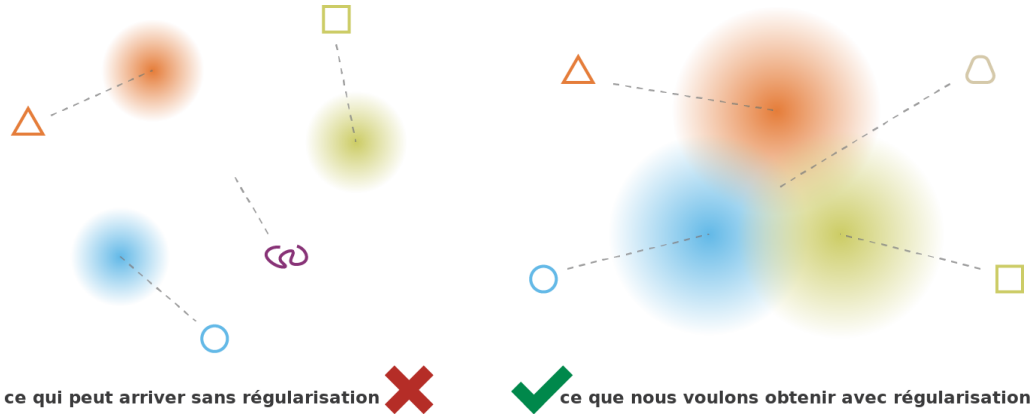


FIGURE I. 2. 15 – Les distributions retournées des VAEs doivent être régulées pour obtenir un espace latent avec de bonnes propriétés

D'après Joseph Rocca, *understanding variational autoencoders (VAEs)*

Chen et al. (2019) exploitent donc un auto-encodeur variationnel, travaillant deux espaces latents parallèles, sémantique et syntaxique. Leur but est - ainsi que dit précédemment - de produire une phrase Z à partir de la sémantique d'une phrase X et de la syntaxe d'une phrase Y.

Ces variables seront chacune un vecteur prélevé probabilistiquement dans un espace régulé (voir Figure I. 2. 14).

Pour la valeur syntaxique, Chen et al. (2019) produisent pour chacun des mots un code latent marginalisé. Ce code signe l'appartenance du mot à des classes (certaines étant surtout composées de mots fonctionnels, d'autres d'adverbes...). Chen et al. (2019) leur définissent une utilité syntaxique, car définis par leurs appartenances à des classes qui ne sont pas ou peu sémantiques.

Pour la valeur sémantique, nous l'avons vu, la moyenne des vecteurs des mots de l'entrée (représentation sémantique de la phrase X) est normalisée.

Elles sont utilisés en entrée dans les *encodeurs* parallélisés de l'expérience des auteurs. Cet exemple est donné parmi d'autres :

X	phrase support sémantique :	*with luck, it may turn out you're right.*
Y	phrase support syntaxique :	*of course, i've done better.*
Z	phrase générée :	*of course, i'll be getting lucky.*

TABLE I. 2. 2 – Exemple de génération (auto-encodeurs variationnels)

Issu de la Table 8, Chen et al. (2019)

Des problèmes de consistance des pronoms, d'omission de mots et de généralisation excessive (ponctuation des phrases par *you ?* trop fréquente) sont rapportés. Ces pro-

blèmes sont selon eux commun aux SCPNs (*syntactically controlled paraphrase networks*) que nous allons aborder ci-après.

La locution *of course*, dans la phrase générée présentée dans la Table I. 2. 2, provient, notent les auteurs, de la part syntaxique, alors que cette partie syntaxique ne vise qu'à influencer la forme de la réalisation [30]. Par cet exemple, Chen et al. (2019) illustrent un problème noté récurrent, et qui ne se produit pas avec les SCPNs proposés par Iyyer et al. (2018) [31].

Chen et al. (2019) ont conséquemment cherché à limiter l'impact de la discrimination sémantique des données voulues syntaxiques en remplaçant aléatoirement dans lesdites données certains mots par d'autres, ce sur la base d'une nature grammaticale commune (nature grammaticale : nom, verbe, adverbe. . .). Leurs résultats ne sont guère améliorés par ce choix. Les BLEU sont extrêmement faibles, ce qui est normal étant donné que les phrases sont réellement transformées. L'absence d'évaluation humaine ne permet pas d'évaluer la production. Les exemples et le point de vue des auteurs dessinent une tendance positive quant aux générations, ce qui est léger mais n'enlève rien à l'intérêt intellectuel de leur proposition.

Iyyer et al. (2018) ont proposé une architecture reposant pleinement sur des LSTMs (voir page 24) : le SCPN (*Syntactically Controlled Paraphrase Network*).

Les auteurs, à partir d'une phrase et d'une forme syntaxique cible, génèrent une paraphrase. Précédant Chen et al. (2019), ils analysent les phrases du corpus PARANMT-50M (Wieting et Gimpel, 2017) à l'aide de l'analyseur de stanford (Manning et al., 2014). Ce corpus compte 50 millions de paraphrases obtenues par *backtranslation*, un processus qui permet de générer de la paraphrase en passant par une langue pivot étrangère (en l'occurence le tchèque). Une phrase est traduite vers une autre langue puis retraduite vers la langue d'origine. Cela apporte des variations syntaxicales et lexicales du fait du non recouvrement des réalisations de surface interlangues.

Iyyer et al. (2018) ont linéarisé les arbres syntaxiques obtenus par l'analyseur syntaxique, produit une version simplifiée (*syntactic templates*) en ne gardant que les deux premiers niveaux (Figure C. 1 en annexe : l'arbre complet `(S (NP (D N)) (VP (V P NP (D N)))` a ici pour version simplifiée `S` → `NP VP`).

Leur expérience met en place un pipeline de réseaux (bi-LSTM, LSTM), avec mise en œuvre aussi bien du mécanisme de *copy* (See et al., 2017) [32] que celui de l'attention (Bahdanau et al., 2014b) [33]. Enfin ils utilisent *beamsearch* [34] pour produire dix sorties par

30. Les codes latents pour la variable syntaxique étant marginalisés (au mot), le réseau les distingue et peut hélas s'en servir comme appui (et générer un contenu sémantique lié). Les travaux de Chen et Gimpel (2019) montrent que cette méthode permet en associant les résultats de multiples *clusterisations* (ce qu'ils ont fait ici), une représentation efficiente de l'information obtenue : très légère et permettant de reconstituer les vecteurs pré-entraînés GloVe (Pennington et al., 2014).

31. Les SCPNs (*syntactically controlled paraphrase networks*) partent d'une intuition similaire à celle de Chen et al. (2019) mais utilisent des représentations d'arbre syntaxique (*parse tree*) couplées à une phrase.

32. mécanisme de copie : voir page 49.

33. mécanisme d'attention : voir page 33.

34. *beamsearch* : génération de n meilleures sorties possibles.

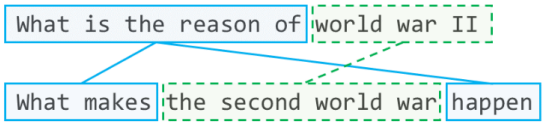
entrée. L'exemple à paraphraser est accompagné d'une représentation de son arbre syntaxique complet pour un premier test. Un second test utilise le *template* syntaxique (qui est plus succinct mais se révèlera ne pas permettre d'amélioration). Pendant la phase d'apprentissage, le *template* syntaxique de la sortie est utilisé (fourni en entrée du décodeur). Pendant la phase de test, les 30 *templates* les plus fréquents sont évalués.Un système de filtres permet de neutraliser les sorties incohérentes (demander d'une phrase longue une phrase nominale courte mène à une réalisation de surface insensée).

Les paraphrases générées ont été évaluées par des *crowdworkers*. De ce travail résulte que 63.7 % des phrases exploitant les arbres syntaxiques linéarisés sont des paraphrases grammaticalement correctes, contre 62.3 % pour celles utilisant les *templates*. Le corpus d'origine voit lui sa proportion de phrases validées s'établir à 65% (sur 100 phrases prises au hasard). Les auteurs effectuent des tests avec les exemples de deux autres corpus, SST (Socher et al., 2013) et SICK (Marelli et al., 2014). SST contient des phrases complexes à haute variance syntaxique, et SICK des phrases courtes et simples. L'une des problématiques du peu de diversité syntaxique dans les exemples d'apprentissage est que les systèmes mis au point supportent mal d'être mis à l'épreuve sur des corpus différents de ceux des apprentissages.

Iyyer et al. (2018) montrent que les SCPNs permettent des transformations complexes tout en préservant la sémantique de l'entrée et la qualité de la grammaire, même sur un corpus de domaine différent de celui du corpus d'apprentissage. Les SCPNs font moins de transformations lexicales que le système NMT-BT (entraîné sur la ligne de base des paraphrases directement générées à partir de PARANMT-50M (Wieting et Gimpel, 2017)), mais assurent plus de transformations syntaxiques. Sur SST la validité des sorties est de 77.1 contre 68.1 pour NMT-BT, et sur SICK de 77.7 contre 81.0.

Enfin, si Iyyer et al. (2018) ne fournissent pas en entrée du décodeur les *embeddings* d'un *template* cible, la probabilité d'obtenir une paraphrase se réalisant avec la même forme syntaxique que l'entrée passe de 28.7% à 38.6%.

Li et al. (2019) ont publié quasiment concomitamment à Iyyer et al. (2018). Eux s'intéressent à d'autres variables latentes que celles du couple sémantique/syntaxe : les variables liées à la granularité au sein d'un texte (voir Figure I. 2. 16).



what is the reason of $x → what makes $x happen
world war II → the second world war

FIGURE I. 2. 16 – Granularité au sein de la phrase : les auteurs segmentent ce qui est de nature phrastique (*sentence level* en bleu), plus général et abstrait, et ce qui est de l'ordre du composant phrastique (*lexical/phrase level* en vert).

D'après Li et al. (2019)

Le *transformer* est ici utilisé au sein d'une architecture encodeur-décodeur. Le réseau

est complexe : $m-$encodeurs et $m-$décodeurs indépendants (un par niveau de granularité, soit deux niveaux pour leur expérience : *sentence* et *lexical/phrase*). Ce modèle, dans sa version originale, offre deux fois six couches (six pour encoder, six pour décoder) avec une attention (voir page 32) liant l'information des entrées, dite *multi-head attention* (voir Figure I. 2. 18). Vaswani et al. (2017) ont décrit ce modèle, ils donnent tous les détails utiles, ainsi que Li et al. (2019). Ces derniers posent clairement que ce choix est flexible, ils le présentent comme arbitraire (substituable par du LSTM par exemple). Néanmoins le *transformer* paraît particulièrement adapté à leur situation.

Voici les points clés du *transformer* :

- encodeur : l'ordre dans les entrées est géré par une notion de positionnement, l'ensemble des représentations vectorielles - positionnées - sont évaluées par un mécanisme d'auto-attention [35] qui va pondérer les valeurs,
- décodeur : le mécanisme d'auto-attention est complété pour chaque décodeur d'un mécanisme d'attention classique (contexte courant) favorisant l'usage des informations pertinentes pour l'élément à générer.

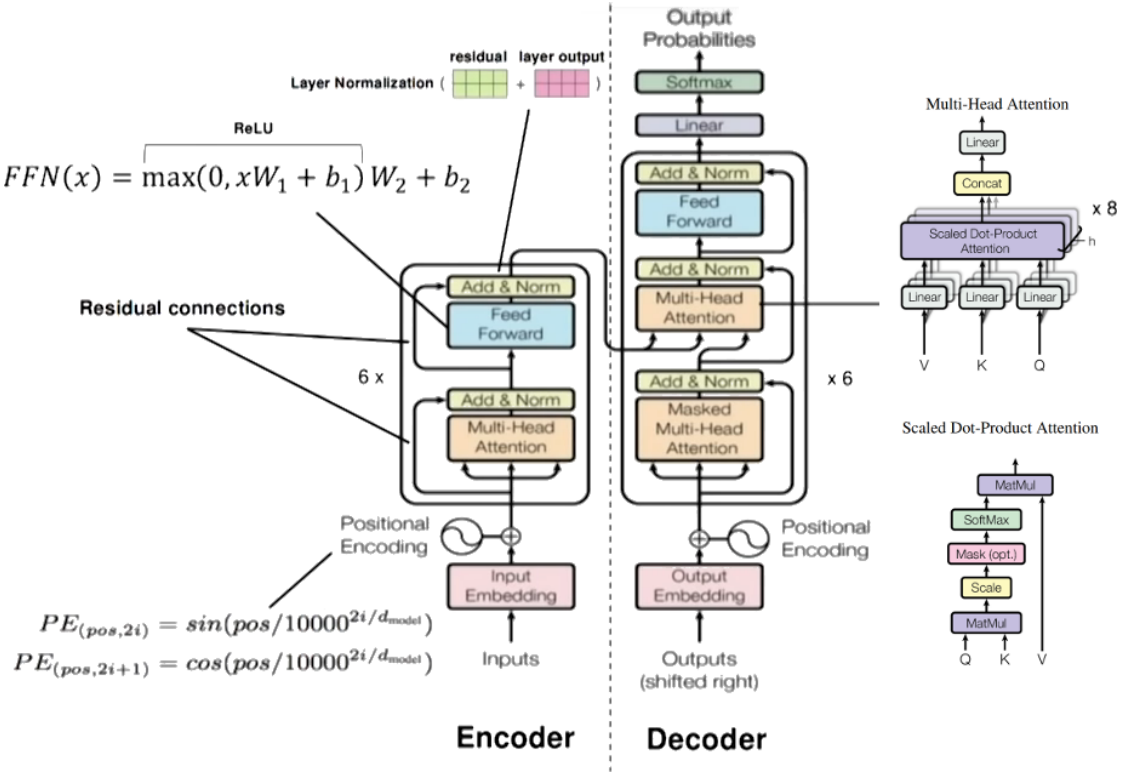


FIGURE I. 2. 17 – Représentation d'un *transformer*
Source : Ivan Bilan: Understanding and Applying Self-Attention for NLP [36], d'après Vaswani et al. (2017)

35. auto-attention (*self attention*) : chaque item de l'entrée va voir sa représentation co-construite avec les autres (voir Figure I. 2. 18)

36. https://www.youtube.com/watch?v=OYygPG4d9H0, consulté le 15 janvier 2020

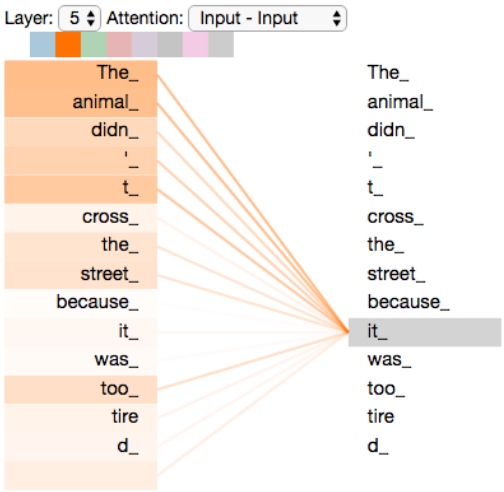


FIGURE I. 2. 18 – Processus de *self attention*, *transformer*
Quand "*it*" est traité dans l'encodeur #5 (l'encodeur supérieur de la pile), une partie du mécanisme d'attention se concentrait sur "*The animal*", et s'appuie sur la représentation de ces mots pour générer "*it*".
Source : The Illustrated Transformer[37], d'après Jay Alammar

Pour constituer le *transformer*, Li et al. (2019) n'ont utilisé que deux encodeurs et que deux décodeurs utilisés en couple encodeur/décodeur. Chaque couple cible le traitement d'un niveau de granularité (granularités Figure I. 2. 16).

Un séparateur (une pile de LSTMs) intervient avant le *transformer*, il est chargé de déterminer le dit niveau de granularité. Un aggrégateur (un LSTM) combine les sorties des décodeurs pour générer la paraphrase.

La différenciation encodeur/décodeur est la suivante :

- encodeur/décodeur des composants *lexical/phrase level* :
  - mécanisme de copie (voir page 48) ,
  - *self-attention* (*cf* Figure I. 2. 18) réduite aux trois mots adjacents pour chaque position,
  - position des termes exploitée : position réelle
- encodeur/décodeur de plus haut niveau (*sentence level*) :
  - *self-attention* couvrant le quasi ensemble de la phrase à paraphraser (masquage des items marqués *lexical/phrase level*),
  - position des termes exploitée : position relative (pour rendre le traitement insensible à la taille de la phrase).

37. https://jalammar.github.io/illustrated-transformer/, consulté le 16 janvier 2020

Une supervision légère, basée sur la faible fréquence des mots au regard du corpus complet, assure la détection des groupes de mots constitutifs de l'ensemble des composants *lexical/phrase level*. Cela a permis aux auteurs d'appliquer leur méthode à de vastes corpus qu'il serait impossible à annoter (*WikiAnswers*, 2 millions de paires de phrases et *Quora duplicate question pair*, 120 000).

Li et al. (2019) ciblent - grâce à la gestion de la granularité et au mécanisme de copie intégré, une adaptation hors domaine. Et effectivement, si leur performance intra-domaine est bonne par rapport aux modèles servant de comparaison [38], elle est tendanciellement excellente hors domaine [39].

L'évaluation humaine (sur 120 groupes de paraphrases) que les six assesseurs ont mené porte sur l'expérience hors domaine *WikiAnswers* → *Quora duplicate question pair*, sachant qu'ils ont affiné l'apprentissage de leur modèle pour qu'il soit plus performant pour les tests hors domaine, donne une note de 1.79 à leurs sorties (1 : meilleur, 4 le pire). Cette note porte sur la fluidité, la précision et la dissimilarité avec la phrase originale. La paraphrase de référence obtient elle en moyenne 1.48. le meilleur modèle adverse, MTL avec mécanisme de copie Domhan et Hieber (2017)), obtient 3.22. Non affiné, leur modèle "naïf" réalise un score sensiblement identique, à 0.09 près : 3.13.

I. 2. 3 MR2Text : représentation du sens vers texte

La représentation de texte (*MR*, *Meaning Representation* a un besoin critique de normes pour pouvoir faire l'objet de recherches. Comment comparer, vérifier, sans outils communs ? Une représentation du sens est intimement liée à la capacité d'être formel. Les *shared tasks*, ou tâches partagées, jouent un rôle fondamental dans l'établissement des normes, et sont un stimulant pour notre domaine.

D'après Anja Belz et Helen Hastie (Stent et Bangalore, 2014), la génération en langue naturelle (NLG, *Natural Language Generation*) paraît parfois être le parent pauvre de l'analyse de la langue naturelle. Toujours d'après elles, c'est l'avènement de l'évaluation comparative et compétitive qui est clé pour le statut du NLG. Effectivement, cela a même permis, ainsi qu'elles l'écrivent, que "de nombreux chercheurs extérieurs au NLG traditionnel [soient] recrutés à la suite des tâches partagées (Basile et Bos, 2011; Boyd et Meurers, 2011)". Elles ajoutent : "une évaluation comparative est de plus en plus attendue, et les challenges en génération font partie intégrante de l'alternance des événements INLG et ENLG depuis 2008." [40]

Trois grands types de représentation de sens ont été utilisés :

- les arbres de dépendances de surface et profond lors des tâches partagées sur la réalisation de surface (SR'11, SR'18, SR'19),

38. BLEU de 41.61 sur le *WikiAnswers* contre un BLEU de 39.36 pour le meilleur modèle adverse (*Pointer generator* (See et al., 2017), soit un écart de 2.25 points.

39. *WikiAnswers* → *Quora duplicate question pair* : BLEU de 16.98 avec 7.15 points d'écart sur le meilleur modèle adverse (multi-task learning (MTL avec mécanisme de copie, Domhan et Hieber (2017)), *Quora duplicate question pair* → *WikiAnswers* : BLEU de 35.12 (25.03 en intra-domaine pour *Quora*) avec 4.34 points d'écart (même modèle adverse).

40. NDR : Les extraits de Stent et Bangalore (2014) sont une traduction.

- les systèmes de dialogue lors des challenges E2E ,
- les *AMRs*, pour les tâches partagées *SemEval* portant sur la génération en langue naturelle.

Partir d'une représentation du texte (*MR*) vers du texte est un domaine de la Génération de Texte Naturel (*NLG* : *Natural Language Generation*) qui recouvre entre autres le RDF. Celui-ci permet de représenter du sens en abstrayant la sémantique et en désambiguïsant. C'est là le propre de la représentation de sens. Ayant déjà présenté dans son cadre spécifique le *RDF* (page 36), nous le laissons globalement de côté.

Nous allons présenter en grandes lignes les *share tasks* participant le plus avant à l'établissement de normes et d'un travail collectif, puis nous concentrer sur les AMRs (*Abstract Meaning Representation*). Ce sont les représentations de sens les plus proches de notre travail.

I. 2. 3.1 Tâches partagées : établissement de normes et dynamisation

Les tâches partagées contribuent à cadrer la recherche. Les trois principales sont *Surface Realisation Shared Task*, *E2E NLG Challenge* et *Semeval Shared Task on NLG from AMRs*.

Surface Realisation Shared Task (SRxx)

Initialement monolingue, la *Surface Realisation Shared Task* a été (début 2020) déclinée en trois éditions : SR'11, SR'18, SR'19. Les deux dernières éditions sont multilingues, ce qui devrait se maintenir dans l'avenir.

Ces tâches partagées se subdivisent en deux parties : génération en langage naturel depuis des arbres de dépendances (voir Figure I. 2. 19) de surface et profond.

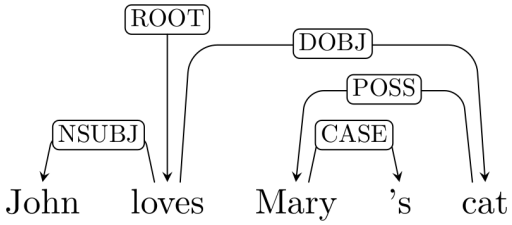


FIGURE I. 2. 19 – SR'19, *Meaning Representation of a sentence*

De Kovács et al. (2019)

Les données à exploiter pour la tâche sont stockées de façon linéaire (un *token*, une ligne) au format dit CoNLL [41].

41. CoNLL : maintenu dans le cadre de *Conference on Natural Language Learning*, le format est décrit https://universaldependencies.org/format.html : il s'agit en 2020 du format *Universal Dependencies 2*

Les phrases sont décrites par plusieurs lignes, et les lignes descriptives contiennent différents champs normés, tel que un numéro d'index correspondant à la position de chaque mot dans la phrase d'origine, la forme de réalisation du dit mot, son lemme...

Le challenge consiste à réaliser les informations sur une base réduite :

- arbre de dépendances de surface (*shallow track*) : les informations touchant à l'ordonnancement des mots dans la phrase de référence ne sont pas stockées dans l'arbre. Les mots sont lemmatisés (limité à une unité autonome non infléchie).
- arbre de dépendances profond (*deep track*) : les informations touchant à l'ordonnancement des mots dans la phrase de référence ne sont pas stockées dans l'arbre. Les mots fonctionnels sont supprimés, ainsi que les informations morphologiques et syntaxiques.

La première tâche consiste à reconstituer l'ordre et l'inflexion des mots, la seconde à réintroduire les mots fonctionnels, et les caractéristiques morpho-syntaxiques.

E2E NLG Challenge

Le challenge NLG E2E (Dušek et al., 2018) porte sur des données non alignées pour un système de bout en bout (E2E = *end-to-end*) : Nous avons 1 à n références pairée(s) à une représentation de sens. Pour E2E 2017 [42], l'exemple suivant est fourni :

MR :
```
name[The Eagle],
eatType[coffee shop],
food[French],
priceRange[moderate],
customerRating[3/5],
area[riverside],
kidsFriendly[yes],
near[Burger King]
```
NL :
"The three star coffee shop, The Eagle, gives families a mid-priced dining experience featuring a variety of wines and cheeses. Find The Eagle near Burger King."

TABLE I. 2. 3 – E2E Challenge 2017, *Meaning Representation of a sentence*

Les données décrivant un objet (ici un restaurant) proviennent d'une base de connaissances. Les notions stockées sont exprimées dans la ou les phrases de référence, mais peuvent l'être de façon indirecte et bruitées. Par exemple, l'humain qui a écrit le texte lisible Figure I. 2. 3 n'a pas jugé utile de préciser explicitement que la nourriture était française (MR `food[french]`), la variété de vins et fromages offerte à l'expérience gourmande lui paraissant l'information pertinente à exprimer.

Dans le cadre de E2E, et pour cette tâche - générer depuis une représentation de sens vers du texte, Puzikov et Gurevych (2018) concluent que le développement de systèmes

42. http://www.macs.hw.ac.uk/InteractionLab/E2E/, consulté le 17 janvier 2020

complexes et coûteux n'est pas forcément justifié, qu'un modèle de génération basé sur des gabarits (exemple : avec des phrases à trous préparées par avance) peut être suffisant. Dušek et al. (2020) qualifient même les systèmes simples de plus fiables.

Semeval Shared Task on NLG from AMRs

Les tâches partagées de SemEval (*Semantic Evaluation*) visent à générer du texte à partir de réprésentation tel qu'illustré Table I. 2. 4, à partir d'AMR (*Abstract Meaning Representation*).

MR :

```
(s / say-01
      :ARG0 (s2 / service
            :mod (e / emergency)
            :location (c / city :wiki "London"
                  :name (n / name :op1 "London")))
      :ARG1 (s3 / send-01
            :ARG1 (p / person :quant 11)
            :ARG2 (h / hospital)
            :mod (a / altogether)
            :purpose (t / treat-03
                  :ARG1 p
                  :ARG2 (w / wound-01
                        :ARG1 p
                        :mod (m / minor)))))
```

NL :

"The London emergency services said that altogether 11 people had been sent to hospital for treatment due to minor wounds."

TABLE I. 2. 4 – SemEval-2017 Task 9, *Meaning Representation of a sentence*

Nous allons accorder un espace spécifique à l'AMR, les représentations de sens dont nos travaux sont porteurs s'en approchant.

I. 2. 3.2 AMR : Représentation Abstraite de Sens

Difficile de parler de *Meaning Representation* sans faire référence à *Abstract Meaning Representation* (*AMR*), la Représentation Abstraite de Sens. La Figure I. 2. 4 en est une première illustration.

L'*AMR* résulte des travaux en linguistique formelle. Elle est documentée dans un guide d'annotations d'une soixantaine de pages [43].

43. https://amr.isi.edu/language.html

Malgré un travail linguistique poussé pour dépasser la forme, l'AMR n'est pas supra linguistique et a été pensée à partir de l'anglais [44]. Elle vise à décrire le monde de façon concise et précise, loin des aléas syntaxiques. Elle est, pour des raisons conceptuelles, particulièrement adaptée à l'anglais.

Les exemples présentés Table I. 2. 5a devraient avoir la même représentation abstraite, dans le cadre des normes définies. Palmer et al. (2014) en fournit la codification formelle (Table I. 2. 5b).

he described her as a genius.
his description of her : genius.
she was a genius.
	`(d / describe-01`
	`  :arg0 (h / he)`
	`  :arg1 (s / she)`
	`  :arg2 (g / genius)`

(a) Réalisations de surface	(b) AMR correspondante

TABLE I. 2. 5 – AMR : Représentation de sens (b) des réalisations de surface (a).

De Banarescu et al. (2013) et Palmer et al. (2014)

Les représentations *AMR* forment des graphes à racine unique, des arbres non cycliques et labellisés. Elles respectent une formalisation stricte et sont représentables sous forme logique (voir Figure I. 2. 20c).

Les étiquettes des nœuds correspondent à des mots dont le sens est identifié : verbes, des noms ou des concepts spécifiques AMR comme `describe-01`, `genius`, Table I. 2. 5.

L'un des nœuds joue le rôle de racine (nœud `d` Table I. 2. 5, nœud `a` Figure I. 2. 20). Enfin, un graphe AMR contient des arcs labelisés, qui correspondent aux rôles sémantiques PropBank (Palmer et al., 2005) ou encore à des relations type définies pour cette normalisation (comme `arg0`, `mod`...).

Les approches proposées pour la génération de texte à partir d'AMR suivent une logique continue. Un système d'encodage/décodage non neuronal est d'abord à noter chez Flanigan et al. (2016) et Pourdamghani et al. (2016), reposant sur de la linéarisation du graphe AMR pour l'encodage, et pour la partie décodage, reposant sur les travaux de Koehn et al. (2007) (voir page 47) dans une approche statistique. Ils ont utilisé le corpus présenté par Banarescu et al. (2013), associant phrases et représentations.

Les architectures neuronales ont rapidement été développées, en particulier en traitant la transformation AMR vers texte comme un problème d'apprentissage graphe vers séquence. Li et al. (2015) est la référence d'arrière plan de ceux · celles voulant gérer le graphe au plus près de sa structure, tels Beck et al. (2018); Marcheggiani et Perez-Beltrachini (2018). Li et al. (2015) décrivent l'usage de GRU (Gated Encoded Unit (Cho et al., 2014)) traitant lors de l'encodage les vecteurs représentant les nœuds du graphe en les liant par un contexte porteur des vecteurs représentant ses arêtes. Le décodage - dont on attend le traitement de la séquence de mots verbalisant l'AMR, est réalisé par des LSTMs.

44. "*AMR is heavily biased towards English. It is not an Interlingua.*" (Banarescu et al., 2013)

```
Pierre a pour ami Dominique.
```

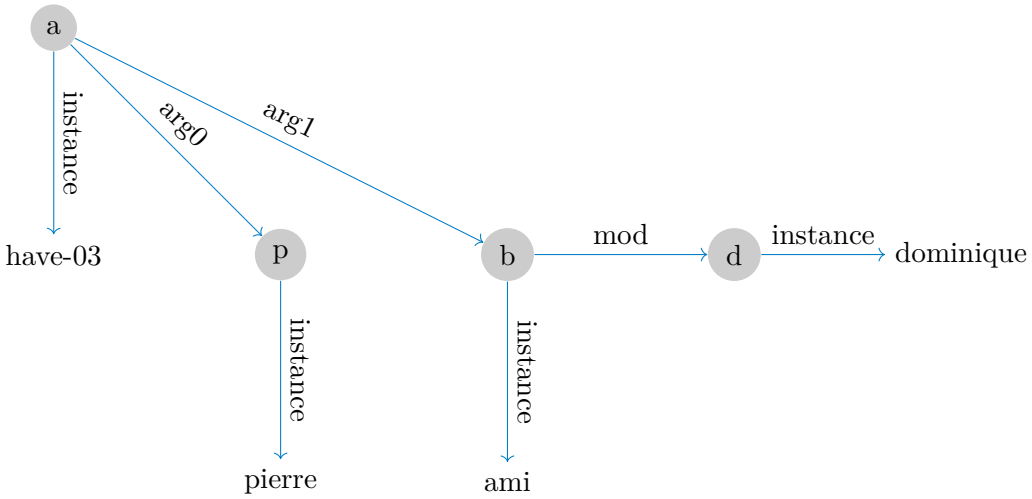
(a) Lexicalisation

```
(a / have-03
   :arg0 (p / pierre)
   :arg1 (b / ami
      :mod (d / dominique))
```

(b) AMR de la lexicalisation I. 2. 20a

```
∃ a, p, b, d:
  instance(a, have-03) ∧
    instance(p, pierre) ∧
    instance(d, dominique) ∧
    instance(a, ami) ∧
    arg0(a, p) ∧ arg1(a, b) ∧ mod(b, d)
```

(c) Forme logique de l'AMR I. 2. 20b, lexicalisation I. 2. 20a



(d) Transposition graphique de l'AMR I. 2. 20b de *Pierre a pour ami Dominique*

FIGURE I. 2. 20 – Formalisation AMR, formalisation logique et transposition graphique

Koncel-Kedziorski et al. (2019) ont exploré les *transformers*[45], similairement à Veličković et al. (2018), avec usage du mécanisme de copie. Ils ont en particulier un souci de couverture des entités nommées : 40% d'entre elles n'apparaissent pas dans les textes générés.

Konstas et al. (2017) ont eux adossé de manière prometteuse l'augmentation des données à l'usage de l'anonymisation des entités nommées. Pour augmenter les données, ils ont analysé en mode automatique des millions de phrases issues de Gigaword pré-annotées (Napoles et al., 2012). Le générateur d'AMR a ensuite été pré-entrainé sur ces données avant d'être affiné avec des données *gold* issues de la tâche partagée SemEval 2016 (SemEval, voir page 63). Le graphe est linéarisé par un parcours en profondeur. Ils tiendront le haut de l'état de l'art jusqu'à Song et al. (2018) puis Ribeiro et al. (2019).

Song et al. (2018) encodent le graphe AMR à l'aide d'un réseau récurrent où à chaque

45. *transformers* : voir travail de Li et al. (2019), page 58

transition t sont ouvertes les connexions entre le nœud courant et les nœuds auxquels il est connecté : cela se traduit par l'incorporation de la somme des états cachés des nœuds entrants et sortants en tant que contexte. Les entrées incluent les représentations des arêtes reliant les nœuds. Le mécanisme de copie est lui aussi activé, et le dictionnaire réduit : les mots sont segmentés.

Ribeiro et al. (2019) mettent eux en œuvre une représentation duale des graphes (*top-down* (G_t) et *bottom-up* (G_b), voir leur modèle Figure I. 2. 21). Les graphes AMR sont, suivant la transformation dénommée *Levi transformation* (Beck et al., 2018), adaptés : chaque arête labelisée r reliant deux nœuds v_i et v_j donne lieu à deux arêtes non labelisées : $(v_i, r)(r, v_j)$, et de fait, est intercalé entre v_i et v_j un nœud dont la représentation vectorielle r sera traitée par le réseau.

Ribeiro et al. (2019) confirment - si le doute était encore présent - que l'augmentation des données, telle que pratiquée par Konstas et al. (2017) est très favorable à la qualité des résultats.

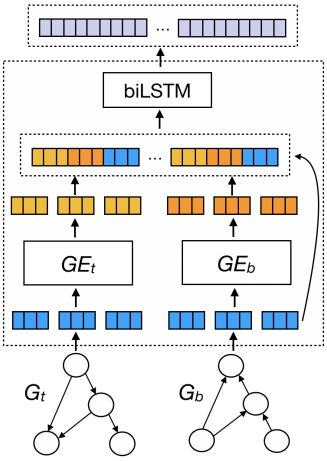


FIGURE I. 2. 21 – Dual Graph Encoder

L'encodeur reçoit deux vues du graphe (G_t et G_b) et génère des représentations structurées des nœuds qui sont utilisées par le décodeur. GE est un encodeur de graphe. Les représentations en bleu, jaune et orange sont respectivement celle du nœud courant, la représentation de ses voisins dans le graphe *top down* et la représentation de ses voisins dans le graphe *bottom up*.

Source : Ribeiro et al. (2019)

Deuxième partie

Réalisation de surface : Contributions

Chapitre 1

Paraphrases syntaxiques

Notre thèse s'intègre dans plusieurs perspectives, elle est issue du projet METAL [46] qui rapproche école et recherche. Dans ce cadre, nous souhaitons proposer des méthodes qui permettent de faciliter la génération semi-automatique d'exercices de grammaire du français. Concrètement, nous nous concentrons sur la génération automatique de phrases et proposons un modèle qui permet, à partir d'une contrainte syntaxique (e.g., 1b) et d'une entrée (e.g., 1a), de générer une phrase (e.g., 1c) réalisant cette contrainte syntaxique et verbalisant le contenu donné en entrée.

(1) a. { (Aarhus Airport operatingOrganisation Aktieselskab) }
b. Passif
c. Aarhus Airport is operated by the Aktieselskab.

Si nous sommes capable de générer des paraphrases syntaxiquement contraintes, celles-ci deviennent exploitables pour un but pédagogique précis. Elles peuvent aussi intégrer des ensembles structurés pouvant être offerts à la pratique des apprenants selon leur niveau, selon les contraintes spécifiées par les enseignants.
Les exerciseurs existants s'appuient généralement sur des ressources construites manuellement. Les ressources sont limitées en terme de volume et de caractéristiques utilisables pour produire des recommandations, s'exercer par essai/erreur. Ainsi que nous l'écrivons sur le site de présentation du projet Metal [47] :

> En fournissant une méthode pour créer semi-automatiquement des exercices, notre objectif est de permettre la création rapide de millions d'exercices adaptés aux besoins des enseignants et des apprenants.

Nous avons en conséquence construit et évalué les performances de modèles de génération de paraphrases intégrant des contraintes syntaxiques faisant l'objet d'un but pédagogique.

46. http://metal.loria.fr/, consulté le 11 février 2020
47. http://metal.loria.fr/?page_id=2165

Nous utilisons WebNLG (Gardent et al., 2017)) qui associe des lots de lexicalisations à des ensembles de triplets RDF (voir page 36). Ces triplets sont organisés en lots de taille définies. Ils sont constitués en lots de 1 à 7 unités.

Nous noterons ici A_i un lot de triplets (une description RDF A), avec i compris entre 1 et 7. Le nombre de triplets composant le lot est marqué par i. Les lexicalisations sont au nombre maximal de trois par description n triplet pour u, soit b_1A à b_3A pour les lexicalisations B liées à A.

Chaque lot est architecturé en graphe (voir Figure I. 2. 2), la plupart des graphes d'une taille donnée s'inscrivant dans les graphes de taille supérieure .

Par exemple, pour un graphe A^1 de taille 2 qui serait :
"`(x aPourAmi y) (y aime chocolat)`", nous pourrions avoir $A^{1'}$ de taille 3 :
"`(z aPourAmi k) (k aime riz) (riz natureAliment céréale)`".
WebNLG fournit aussi des exemples intégralement inclus, pour lesquels nous pouvons avoir A^2 :
"`(x aPourAmi y) (y aime chocolat)`", nous pourrions avoir le RDF de taille 3 :
"`(x aPourAmi y) (y aime chocolat) (chocolat natureAliment fève)`". Ce RDF inclut intégralement A^2.

Nous avons utilisé le RDF comme pivot pour constituer quatre corpus d'apprentissage :

- données vers texte : du RDF (linéarisé) vers la lexicalisation, le RDF en entrée est adossé à une contrainte syntaxique repérée dans la lexicalisation demandée,
- réduction de texte : lexicalisation courte b_1 en sortie, cette lexicalisation a une description RDF A_i avec i entre 1 et 6 ; pour b_1 en sortie, nous avons en entrée :
  - une contrainte syntaxique repérée dans b_1
  - une lexicalisation b_2 de description RDF A_j avec j entre 2 et 7, avec A_i appartenant à A_j, un triplet de A_j nommé a_x étant surnuméraire à A_i,
  - a_x : triplet de A_j surnuméraire à A_i : on veut que le réseau apprenne à supprimer l'information de a_x dans b_2 pour obtenir b_1
- extension de texte : lexicalisation longue b_1 en sortie, cette lexicalisation a une description RDF A_i avec i entre 2 et 7 ; pour b_1 en sortie, nous avons en entrée :
  - contrainte syntaxique repérée dans b_1
  - lexicalisation plus courte b_2 de description RDF A_j avec j entre 1 et 6, avec A_j appartenant à A_i, un triplet de A_i nommé a_x étant surnuméraire à A_j,
  - a_x : triplet de A_i surnuméraire à A_j : on veut que le réseau apprenne à ajouter l'information de a_x dans b_2 pour obtenir b_1
- texte vers texte : lexicalisation b_1 vers lexicalisation b_2, avec contrainte syntaxique repérée dans la lexicalisation b_2 demandée en entrée, et *rdf* commun aux deux lexicalisations, au moins dans sa forme et ses prédicats.

Une fois ces corpus construits, nous avons entraîné les modèles neuronaux correspondants, puis évalué leur qualité, tant du côté de la syntaxe que de leur diversité. Nous avons aussi combiné les sorties en les groupant sur critère de RDF commun.

Cette partie présente donc la génération automatique de paraphrases sur levier syntaxique. Conditionner la génération sur des contraintes permet effectivement la géné-

ration de textes syntaxiquement différents pour une même entrée. L'exploitation de différents encodages de l'information permet d'en accroître significativement le nombre.

Nous avons évalué les résultats à l'aide de BLEU, mais aussi d'une mesure de similarité basée sur Ratcliff/Obershelp, sur laquelle nous reviendrons lors de l'analyse des résultats (p. 90). Nous avons par ailleurs calculé automatiquement la proportion de phrases obtenues contenant a priori la contrainte syntaxique souhaitée ainsi que toutes les entités attendues (celles qui sont constitutives des sujets/objets des prédicats). Enfin, un linguiste expert a examiné les sorties pour cinquante significations. La combinaison des modèles d'apprentissage permet - selon le linguiste - d'obtenir 6.3 paraphrases correctes (et différentes syntaxiquement) par signification. Le taux de similarité entre les sorties est de 61%, contre 81% quand on génère à partir du seul RDF sans contrainte syntaxique. Le BLEU-4 est maximisé (83.87) en extension de texte. Le BLEU-4 est de 62.87 sur la combinaison des modèles (nous sommes à 4.71 quand on génère à partir du seul RDF sans contrainte syntaxique).

II. 1. 1 Motivations

Générer des paraphrases automatiquement (c'est à dire exprimer une même chose sous différentes formes) n'est pas utile que pour notre action. Bien des aires du Traitement Naturel des Langues y travaillent, tel que les systèmes de questions/réponses, où cette capacité peut permettre d'améliorer les requêtes (Riezler et al., 2007). Pouvoir lexicaliser dynamiquement autorise une reformulation d'une phrase générée à partir d'une grammaire et de la rapprocher de son pendant naturel (Berant et Liang, 2014).

Les paraphrases ont aussi un potentiel non négligeable en traduction (Kauchak et Barzilay, 2006) et en résumé automatique (Ganitkevitch et al., 2011), et pour consolider des représentations computationnelles de phrases (Wieting et al., 2015b). Elles permettent en effet d'augmenter la masse de données d'entraînement et/ou d'évaluation. D'un point de vue purement linguistique, la génération automatisée de paraphrases est une tâche importante du fait qu'elle démontre la capacité des techniques de traitement automatique des langues à prendre en charge cette spécificité clé des langues naturelles.

Dans ce chapitre, nous nous centrons sur la génération automatique de paraphrases syntaxiques pour des textes tels que 12a-e partageant une même signification, mais qui diffèrent syntaxiquement. Nous avons une cible applicative particulière. Si générer des paraphrases est en pratique utile pour les différentes applications précédemment listées, une application typique est la constitution d'une base matérielle pour créer des exercices de grammaire. Ainsi que l'expliquent Perez-Beltrachini et al. (2012), les supports pédagogiques pour l'apprentissage des langues incluent généralement des exercices ciblant spécifiquement une structure grammaticale particulière et sont basés sur une phrase ou un texte court illustrant le phénomène étudié. La table II. 1. 1, par exemple, montre deux lexicalisations syntaxiquement contraintes (en italique) qui évaluent la maîtrise des contraintes par l'apprenant (en gras).

Notre travail apporte les contributions suivantes. Nous montrons que conditionner la génération de langue naturelle sur support syntaxique permet de générer des paraphrases

Instruction :	Trouve et souligne la **subordonnée relative sujet**.
Texte :	John voit un chat qui dort.
Solution :	John voit un chat ***qui dort***.
Instruction :	Combine deux phrases en une seule contenant **une subordonnée relative sujet**.
Texte :	John voit une femme. La femme dort.
Solution :	John voit une femme ***qui dort***.

TABLE II. 1. 1 – Exemples de phrases illustrant un point de grammaire dans un cahier d'exercices

syntaxiques à partir d'une même entrée. Nous fournissons une exploration systématique de comment différentes tâches de génération impactent les sorties pour une même entrée, et nous montrons qu'exploiter ces différents modes de traitement permet d'augmenter le nombre de paraphrases produites pour une entrée donnée. Nous rendons par ailleurs disponible le corpus d'entraînement de chaque modalité testée : données vers texte, texte vers texte, expansion/simplification de texte. Ce corpus est disponible sur demande.

II. 1. 2 Travaux liés

Les principaux travaux sur la génération de paraphrases sont catégorisables en trois groupes.

Le premier groupe s'appuie sur des corpus monolingues, utilisant des méthodes dites « guidées par la donnée » (Lin et Pantel, 2001), par la grammaire ou sur thésaurus (Madnani et al., 2007; McKeown, 1983; Hassan et al., 2007; Kozlowski et al., 2003; Quirk et al., 2004; Zhao et al., 2008).

Contrastant avec ces approches, la méthode par pivot repose sur l'exploitation de données bilingues et de méthodes de traduction automatique pour d'abord extraire puis générer des paraphrases (Callison-Burch, 2008; Ganitkevitch et Callison-Burch, 2014; Ganitkevitch et al., 2011).
Enfin, nous trouvons les méthodes du *deep learning* reposant sur une architecture *encoder-decoder* pour apprendre des modèles aptes à fournir des déclinaisons textuelles ciblées (Mallinson et al., 2017; Prakash et al., 2016).

Notre approche est proche de ces dernières. La principale différence est qu'au lieu de limiter le problème à de la réécriture de texte, nous explorons comment des sources variées d'entrées impactent le nombre et le type de paraphrases. Une autre différence d'importance est que nous nous focalisons sur de la paraphrase syntaxique et conditionnons la génération sur des attributs grammaticaux. En ce sens, notre approche partage des similitudes avec des modèles tout à fait récents de génération contrôlée (Hu et al., 2017; Semeniuta et al., 2017). Les travaux de Hu *et al.*, Semeniuti *et al.* reposent sur des autoencoders variationnels qui intègrent une modélisation de propriétés particulières telles que le style et autres propriétés syntaxico-sémantiques. Notre travail se focalise

sur la paraphrase syntaxique et introduit une nouvelle méthode de production de texte basée sur une hybridation "données et texte" pour les entrées.

Bien que proches sur le fond, les travaux de Hu et al. (2017); Semeniuta et al. (2017) ne ciblent pas du paraphrasage au sens où nous l'entendons. Ces travaux font un usage de contraintes pour générer du texte. Semeniuta *et al.* par exemple produisent des tweets en génération libre (comme si on générait du texte en roue libre, un mot amenant le suivant). Les auteurs maintiennent un vecteur de variables latentes qui évolue au fil de ce qui est déjà sorti, et qui aide à structurer la sortie pour obtenir quelque chose de cohérent. Et cela fonctionne : ils font concrètement levier avec des informations syntaxiques [48], obtiennent du texte beaucoup plus cohérent.

Hu et al. (2017) ont développé un modèle qui permet l'apprentissage dirigé par ces variables latentes : le modèle intègre un *discriminator* qui a pour mission d'inférer les attributs de la phrase (sentiment et temps à partir de pré-analyse). Cela vise l'augmentation de données. L'idée est d'obtenir une nouvelle phrase correcte, elle sera dans le domaine de la phrase d'entrée (ie : *i guess the movie is too bland and too much* pour *this is one of the outstanding thrillers for all the time*), et sera potentiellement détachée sémantiquement, sa raison d'être n'est pas là. Le but de Hu *et al.* est que la phrase soit correcte, qu'elle ait les marquages attributifs attendus (pour eux, temps & sentiment). Ils ont un corpus de 350 000 critiques de films, ne cherchent pas la paraphrase, mais la possibilité de faire varier et de maîtriser suffisamment la variation pour augmenter les données d'apprentissage.

Notre production de paraphrases - dans sa réalisation - a plus à voir avec les travaux de Mallinson et al. (2017), précédemment évoqué. Ils ont introduit un modèle neuronal basé sur de la rétro-traduction multilingue (sur pivots donc) et montré que leur modèle surpasse un modèle de paraphrase formé avec un système de traduction automatique statistique (SMT) couramment utilisé sur trois tâches, à savoir la corrélation avec le jugement humain de la qualité de la paraphrase ; détection de paraphrase et de similarité ; et génération de paraphrases au niveau de la phrase. Plus proches encore sont les travaux de Iyyer et al. (2018). Iyyer *et al.* utilisent aussi la rétro-traduction au sein de leur modèle de générateur de paraphrases, pour produire un corpus d'apprentissage conséquent. La proximité avec notre modèle n'est pas là - nous ne faisons pas de rétro-traduction. Au-delà de ce choix démontré pertinent, Iyyer *et al.* utilisent de la syntaxe pour contrôler la génération de paraphrases. Selon un gabarit syntaxique T et une séquence d'entrée S, le modèle génère tout d'abord une analyse syntaxique complète P_T. Ensuite, cette analyse syntaxique est associée à la phrase initiale - la séquence d'entrée - pour prédire une paraphrase S réalisant la contrainte fournie par le gabarit T.

Nos travaux sont proches de ceux de Iyyer et al. (2018) mais en diffèrent. Au lieu de limiter la génération de paraphrases à un problème de réécriture de texte, nous explorons l'impact de la diversification des sources - la modulation de l'information - sur le nombre et le type de paraphrases générées. Notre approche diffère de celle de Mallinson et al. (2017) par le fait que nous nous concentrons sur les paraphrases syntaxiques à

48. Semeniuta et al. (2017) : *We also perform linear operations in the latent space to demonstrate that they result in smooth and syntactically correct transitions between generated texts.*

signification maîtrisée (nous ne sommes pas dans l'augmentation de corpus) et sur le conditionnement syntaxique explicite. Quoiqu'il en soit, notre approche partage des similitudes avec les modèles récents de génération de texte contrôlable (Hu et al., 2017; Semeniuta et al., 2017), qui utilisent des auto-encodeurs variationnels pour modéliser des propriétés holistiques de phrases telles que style, sujet et autres caractéristiques syntaxiques. Notre travail est sans doute conceptuellement plus simple, se concentre sur les paraphrases syntaxiques et introduit un nouveau mode de production de texte basé sur des données hybrides « données et texte ».

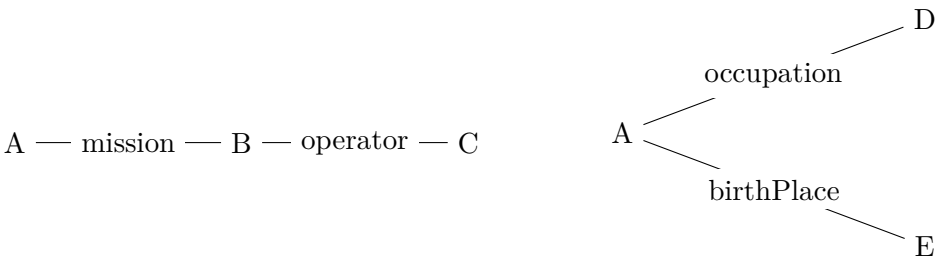
II. 1. 3 Génération de paraphrases syntaxiques

Pour présenter notre approche, nous commençons par introduire le corpus WebNLG. Nous montrons ensuite comment nous l'enrichissons avec des informations syntaxiques. Enfin nous présentons les modèles de générations proposés et les résultats obtenus.

II. 1. 3.1 Le corpus WebNLG

Le corpus WebNLG est issu du WebNLG Challenge (Colin et al., 2016). Il comptait initialement 25 298 lexicalisations pour 9674 lots de triplets RDF (voir Partie I. 2. 2). Nous avons utilisé une version intermédiaire ultérieure comptant 42 594 étiquettes correctes (voir Table II. 1. 3). Les différentes versions sont accessibles sur `https://gitlab.com/shimorina/webnlg-dataset` [49].

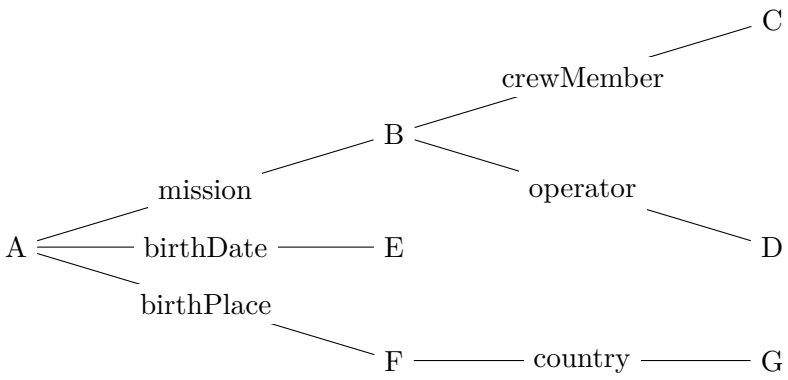
A chaque lexicalisation correspondent un à sept triplets reliés entre eux selon un motif arborescent tel qu'illustré Figure II. 1. 1 (enchaînement strict ou cousinage ou mixte, d'après Gardent et al. (2017)). Dix catégories sont couvertes : Astronautes, Universités, Monuments, Bâtiments, Personnages de *comics*, Nourriture, Aéroports, Équipes sportive, Villes, Corps célestes, Moyens de transport, Ville, Athlètes, Politiciens, Artistes et Littérature. La distribution en terme de lots de triplets au sein des catégories est disponible en Annexe D. 2.



(a) Enchaînement strict (*chain*) (b) Cousinage (*sibling*)

La Figure II. 1. 1a nous montre un motif chaîné, propice aux lexicalisations offrant des participiales ou des relatives sujet, nous pouvons avoir comme lexicalisation la phrase (2).

49. WebNLG : les différentes versions, consulté le 17 février 2020



(c) Cousinages et enchaînements

FIGURE II. 1. 1 – Triplets RDF, motifs de connexion

D'après Gardent et al. (2017)

(2) *"A participated in **C operated by B**".*

La Figure II. 1. 1b, qui met deux entités objet à même niveau (leurs triplets ont pour sujet partagé `A`, objets `D` et `E`), illustre un motif favorable à la création de juxtapositions ou de coordinations. Nous pourrions retrouver cette structure dans les lexicalisations :

(3) "*A was born in E**. She** worked as a D.*"
"*A was born in E **and** worked as a D.*"

Ces lexicalisations aux entités anonymisées sont issues de Gardent et al. (2017).

Les triplets ont été extraits de DBPedia, présentée Figure I. 2. 1 selon une procédure permettant de sélectionner un contenu varié : des arborescences diversifiées, l'intuition étant que les lexicalisations envisageables sur support de ces triplets tendraient à être syntaxiquement variées. WebNLG fournit 62 motifs différents, la Figure II. 1. 1 en illustrant trois.

Nous utilisons indifféremment les expressions "motif" et "structure arborescente". Les motifs sont transversaux aux catégories : les motifs des lots de trois triplets peuvent se retrouver au sein de structures arborescentes plus grandes (quatre triplets, cinq triplets...). Les prédicats (`mission` Figure II. 1. 1a) peuvent être propres à une catégorie (`mission`) ou non (`birthDate`). Il y a un total de 373 prédicats RDF différents.

Prédicats comme triplets sont transversaux aux différents lots de triplets. Le premier groupe de tailles des lots RDF (composés chacun d'un et un seul triplet, nommé groupe RDF 1 ci-après) est très spécifique. Tous les triplets du corpus, constitutifs un à un des lots, pour toutes les catégories, disposent tous de deux à trois lexicalisations.

C'est à dire pour le lot illustré Figure II. 1. 1a, `(A mission B) (B operator C)` verra chacun de ses triplets lexicalisé dans le groupe RDF 1 :

`(A mission B)`	*A was a crew member of B.*	
	avec par exemple	`A` égal à `Buzz Aldrin` `B` égal à `Apollo 11`.
`(B operator C)`	*The B operator is A*	*A operated B*
	avec par exemple	`B` égal à `Appolo 12` `C` égal à `NASA`.

TABLE II. 1. 2 – Lexicalisations de triplets (lot de 1)

Les données de WebNLG sont stockées au format XML, et présentées en Annexe D. Quelques statistiques générales sont données ci-après Table II. 1. 3. Nous avons entre 2 et 3 lexicalisations par lot de triplets.

triplets (lot de)	1	2	3	4	5	6	7	tous
# lexicalisations	9 022	8 306	9 184	8 646	6 257	602	577	42 594
# lots RDF	3 895	2 997	3 339	3 146	2 334	237	213	16 161

TABLE II. 1. 3 – Statistiques générales données WebNLG utilisées

Les lexicalisations ont été produites par des humains : la procédure était exemplifiée et le travail augmentait *crescendo* en difficulté (nombre de triplets à traiter).

Dans le groupe RDF 1, il était attendu d'eux que pour le RDF `(Buzz Aldrin mission Apollo 11)` ils produisent une phrase comme `Buzz Aldrin was a crew member of Apollo 11.` Ce travail était du travail à la chaîne, réalisé par des *crowdworkers*. Le travail par la foule est ce qui a permis d'obtenir autant de lexicalisations : rien que 30 secondes par phrase (sachant que les ensembles de triplets à lexicaliser comptaient jusq'à sept éléments) totalisent 350h de saisie, soit 50 jours-hommes. Diversifier les sources (le *crowdworking* permet de sélectionner un certain nombre de locuteurs) est aussi un gage de diversification des productions.

II. 1. 3.2 Enrichissement du corpus avec des informations syntaxiques

L'enrichissement du corpus a nécessité plusieurs étapes. Nous avons tout d'abord repéré les sujets et objets des triplets RDF au sein des lexicalisations. Nous voulons trouver comment l'information se réalise, autrement dit trouver quelle forme prend la relation pour un sujet et un objet donné : en vue d'une généralisation, nous anonymisons, et de fait vérifions la cohérence des données (l'entité est trouvable et ne nuit pas à l'analyse). Les textes sont préalablement analysés en constituants et en dépendances (Nivre et al., 2016). La phrase 4 illustre un souci typique de ponctuation . Les pré-traitements (anonymisation et analyse) permettent une adaptation itérative du script mis au point sur la base du recueil des échecs.

(4) *Jens Härtel is in the 1. FC Magdeburg club.*
L'entité *1. FC Magdeburg club* provoque une analyse erronée du texte considéré comme constitué de deux phrases.

Les résultats finaux sont stockés (`entities_parsed.xml`, cf Annexe Figure D. 7) ainsi que les résultats intermédiaires : phrase d'origine, phrase anonymisée, analyse en

constituants, graphes, triplets, positionnement objets et sujets des triplets dans la phrase et nature de la lexicalisation [50].

Il était fondamental pour notre expérience de savoir où se réalisaient les items mis en relation par des prédicats. Nous avons donc aussi étiqueté leurs positions au sein de la lexicalisation, qu'elle soit directe, reformulée (ce qui est typiquement le cas des dates, mais pas seulement : des prénoms peuvent se voir supprimés) ou pronominalisée.

Préparation du corpus

Les lexicalisations sont propres à un lot de triplets, à une catégorie donnée. Les sujets/objets ont un contenu de type proche de la notion d'entité nommée. Nous visons un modèle délexicalisé où ces entités ne seraient pas apprises, où nous relexicaliserions après génération.

Étant attendu que les structures arborescentes reliant du RDF impactent la formulation, nous cherchons à maximiser l'apprentissage des structures syntaxiques - en reportant le port du poids sémantique sur les prédicats.

Ce pré-traitement vise à repérer les entités (sujets/objets des prédicats) dans les lexicalisations. La complexité variable de ces entités (intégrant de la ponctuation, des verbes ou mots identifiés comme verbes, des nombres) comme `Aaron_S._Daggett` ou encore `Ministry_of_Health_,_Welfare_and_Sport_(Netherlands)` nuit à la détection de certaines contraintes, auquel cas la phrase anonymisée est utilisée comme support. Ensuite, l'anonymisation permet la création de corpus plus synthétique : `Ministry of Health , Welfare and Sport ( Netherlands )` ou `Ministry of Health , Welfare and Sport in Netherlands` occupent respectivement 10 et 9 tokens, la clef d'anonymisation un seul.

L'anonymisation augmente la transversalité de chaque exemple. Il est attendu un apprentissage détaché de la précision des entités : il n'est pas intéressant pour nous d'apprendre à parler d'une entité particulière (*Barack Obama*), mais plutôt de verbaliser avec variation autour d'un sujet (par exemple pour exprimer de façons variée *quand est-ce que x a été élu*). Estomper l'entité sujet/objet derrière une étiquette sémantiquement pauvre y contribue.

Le RDF 5-a, lexicalisé en 5-b voit les sujets/objets de ses prédicats alignés sous leurs différentes formes (5-c), c'est-à-dire repérés pendant le pré-traitement et anonymisés dès celui-ci (5-d).

50. nature de la lexicalisation : les objets et sujets des triplets peuvent prendre place sous différentes formes, et l'analyse en constituants se double d'une analyse en co-références. L'information quant à la forme de la lexicalisation est stockée et peut être :
- *different* : l'entité est trouvée sous une forme différente de celle attendue,
- *equal* : l'entité est trouvée sous la forme exacte attendue,
- *possessive pronominalization* : une référence à l'entité a été identifiée sous la forme d'un déterminant possessif,
- *pronominalization* : une référence à l'entité a été identifiée sous la forme d'un pronom possessif.

(5)) a. RDF : ("1955_Dodge" bodyStyle "Convertible") ("1955 Dodge" engine "230 cubic inches")
b. Texte : *The 1955 Dodge is a Convertible with a 230 cubic inches engine.*
c. Alignement : { (xlid0xrid,"1955_Dodge",”1955 Dodge"), xlid1xrid :"Convertible" et xlid2xrid :"230 cubic inches"}
d. Anonymisation "The xlid0xrid is a xlid1xrid with a xlid2xrid engine ."

Nous avons mis en place une procédure récursive pour identifier les sujets/objets des triplets RDF dans les textes. Les entités DBPedia sont complexes, et parfois annotées ou intégrant des redondances. Par exemple un étiquetage de ville peut comprendre le pays ou la région `(Paris, France | country | France)` ce qui, à la lexicalisation, pourrait donner en français “Paris est en France.”. Cette donne a abouti à un travail faiblement contraint en ce qui concerne la saisie des *crowdworkers*, qui ont parfois fait des erreurs de saisie, des oublis, ou des reformulations (acronymisations, inversement des extensions : *US* pouvant donner lieu à *USA*, *United States*, ...) .

Les entités attendues dans les lexicalisations - qui très exceptionnellement n’y sont pas (oubli du lexicaliseur (*crowdworker*)) - sont triées par taille, et traitées de la plus grande à la plus petite. Ainsi, "Paris, France" sera recherché avant France. Son taux de similarité le plus haut sera avec *Paris*, et laissera *France* disponible pour alignement avec l’objet du prédicat. Si une lexicalisation est du type *Paris, France, est la grande ville de ce pays*, l’objet *France* ne sera pas satisfait car le *Paris, France* de la lexicalisation s’alignera sur le sujet du triplet. L’algorithme traitera les données de manière à déjà rechercher *France* et à finaliser l’alignement de *Paris* lexicalisé avec *Paris, France* issu du triplet. Quand les alignements sont satisfaits, l’algorithme passe à la phrase suivante.

L’algorithme de Black (2004) permet d’évaluer la similarité entre deux chaînes de caractères. Nous l’utilisons pour repérer les entités par similarité. Nous partons du principe que toutes les entités sont présentes au moins une fois. Les dates sont spécifiquement identifiées à l’aide de la librairie *datefinder*[51].

Phénomènes syntaxiques

Les phénomènes linguistiques sont destinés à servir de levier, à partir de labels repères. Lors de l’expérience ici décrite, nous générons une phrase à partir de diverses entrées. Les phénomènes linguistiques attendus dans la sortie sont fournis en entrée. Leurs labels sont relativement *ad hoc*, d’abord parce que les définitions linguistiques peuvent être complexes (ex : l’apposition ne voit pas sa définition faire consensus), ensuite parce que les outils et règles définies pour les étiqueter sont perfectibles. Cette expérience vise à démontrer la faisabilité de la génération sous contrainte, avec pour première cible une génération de textes cohérents dans lesquels le phénomène syntaxique recherché est trouvé.

La détection des phénomènes syntaxiques est relativement simple. Elle couple un appui sur graphe à une couche abstraite. Un phénomène est détecté quand il répond à

51. *datefinder* : librairie python version 0.7, https://pypi.org/project/datefinder/

une règle. Nous n'avons ici que de l'algorithmie, et celle-ci s'appuie sur les graphes créés dans une première phase, stockés dans le xml dont un extrait est disponible en Annexe Figure D. 7. Un moteur très simple permet de recréer des graphes : nous avons utilisé l'API [52] de Stanford CoreNLP (Nivre et al., 2016) pour les construire. Pour construire un graphe, il suffit d'utiliser cette API. Elle délivre le graphe dans un format informatique exploitable qui nous sert de support.
Nous obtenons 32503 textes.

Pour étiqueter un texte, nous avons donc effectué une analyse avec Stanford CoreNLP dependency parser [53]. Nous avons défini un ensemble de règles permettant d'identifier les phénomènes syntaxiques recherchés. Les phénomènes rares sont sur-représentés dans le corpus analysé manuellement. L'extraction aléatoire de phrases à tester a été guidée pour assurer une présence permettant de soumettre à validation l'intégralité des contraintes prévues. Nous avons exclu des contraintes utilisées pour notre expérience les participiales, n'ayant pas réussi à les détecter avec acuité (statistiques présentées Table II. 1. 5).

Les étiquettes de dialogue stanford sont celles utilisées pour le Penn Treebank (voir en Annexe F. 1), les dépendances utilisent celles des Universal Dependencies (voir en Annexe F. 2).

La Figure II. 1. 2 illustre le graphe du texte *The 1955 Dodge is a Convertible .* Celui-ci a été anonymisé "The xlid0xrid is a xlid1xrid ." lors de la recherche des sujets/objets du triplet RDF qui y est associé (`("1955_Dodge" bodyStyle "Convertible")`). Les clefs d'anonymisation sont xlid0xrid pour `"1955_Dodge"` (*1955 Dodge* dans le texte), et xlid1xrid pour ***`Convertible`***.

Pour chaque phrase conservée, nous produisons deux graphes, un graphe sur phrase anonymisée et un graphe sur phrase d'origine. Les informations permettant la jointure des deux graphes sont connues et ont permis de construire l'illustration fournie Figure II. 1. 2.
La structure du graphe est la suivante :

- à chaque nœud correspond l'index d'un mot ou d'un signe de ponctuation de la phrase, telle qu'elle a été *tokenizée* (séparée en éléments distincts) par le Stanford parser,
- chaque nœud se voit associé deux clés : pos (*part of speech*) et word (mot tel qu'il est dans la phrase),
- chaque relation entre deux mots (analyse en dépendance) est matérialisée par un arc entre deux nœuds (*edge*).
- chaque arc est dirigé en fonction des résultats de l'analyse (ex : l'arc va du verbe au sujet, et non du sujet au verbe).
- chaque arc est étiqueté par un mot identifiant la nature de la relation unissant chaque couple (voir section F. 2)

52. API : *Application Programming Interface*, ici porte d'accès aux services d'une bibliothèque logicielle documentée par les développeurs.
53. Stanford CoreNLP dependency parser version 3.8, du 9 juin 2017, exploitant en particulier les co-références, la reconnaissance d'entité, l'analyse en dépendance simple et densifiée (*enhanced*)

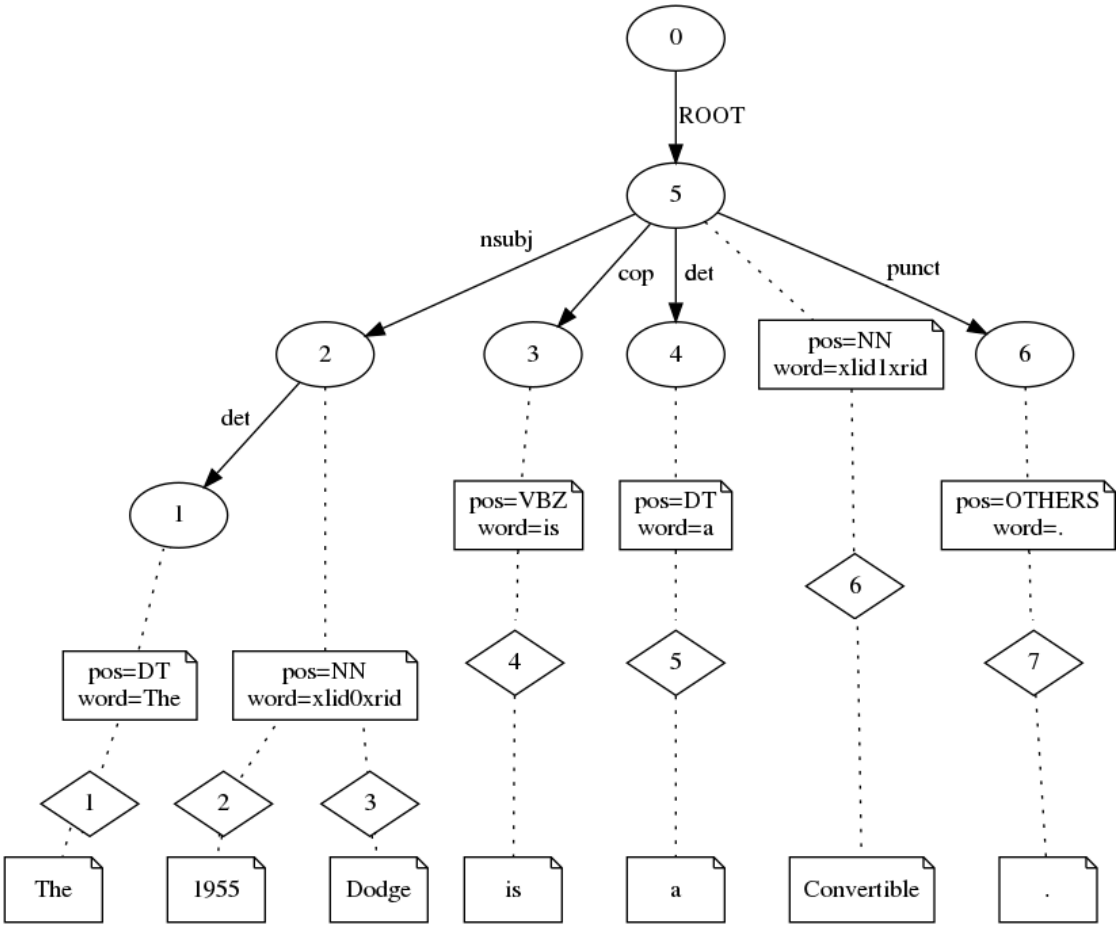


FIGURE II. 1. 2 – Exemple de graphe support à la capture de phénomène syntaxique

Nous définissons un ensemble de règles[54] ; elles permettent de circonscrire 16 situations différentes (voir Table II. 1. 5). Sur le graphe de la Figure II. 1. 2, une seule contrainte est détectée.

La règle[55] qui l'a détectée ne prend pas en compte les liens intra-entité (ex : entre les nœuds de `1955` et `Dodge` constitutifs de l'entité `1955 Dodge`).

Elle s'est appliquée car une dépendance (un arc) étiquettée `cop` (copule) lie un nœud quelconque et un nœud verbal. L'analyse complète des phrases et ce qui a permis l'identification des contraintes sont stockés dans un fichier présenté Figure D. 8 en Annexe. Le verbatim suivant donne les contraintes pour la phrase de la Figure II. 1. 2 :

```
<constraints count="1" uniq="1">
 <contrainte name="existential">
  <socle id="0">
```

54. Les règles sont présentées en détail en Annexe D. 5.1
55. existentielle, voir Annexe D. 5.7

```
    <node index_node_rule="0" rule="existential"><token index="5"/>
    </node>
    <node index_node_rule="1" pos="V.*" rule="existential">
     <token index="3"/>
    </node>
   </socle>
  </contrainte>
 </constraints>
```

Il est à noter que la précision est plus importante pour notre exercice que le rappel. En effet, il est bien plus important de présenter des données cohérentes que des données chaotiques : quand la contrainte est présentée au réseau, mieux vaut qu'elle y soit vraiment si on veut aider le réseau à construire des phrases correctes sur des bases syntaxiques.

Nous constatons dans la Table II. 1. 5 80 913 contraintes (une même contrainte peut être réalisée plusieurs fois dans une même phrase) pour 32 503 textes. Le nombre total est de 82 214 en comptant les textes sans contrainte détectée, ils fournissent aussi une information.

Des phrases exemples illustrent Table II. 1. 4 les quinze contraintes définies ainsi que l'absence de contrainte.

Apposition *apposition* (p. XXIV)	*Ska punk* , ***the genre of A. Bertram*** , originates *from Punk rock* .
Coordination verbale *coordinated clauses* (p. XXV)	*Peter Stöger is the manager of 1 FC Köln* **and** plays for *the Austria national football team* .
Coordination *coordinated full clauses* (p. XXIX)	*AC Cesena play in Serie B* ***and*** *their ground is in Cesena* .
COD *direct object* (p. XXXI)	*Aaron Bertram performs* ***Ska punk music*** *and plays for the Suburban Legends band* .
Prédicative *existential* (p. XXXI)	*Ernie Colón* ***is*** *a Puerto Ricans national* .
COI *indirect object* (p. XXXII)	*It was the All India Council for Technical Education in Mumbai that gave* ***Acharya*** **Institute of Technology** *its Technical Campus status* .
Juxtaposition *juxtaposition* (p. XXXII)	*Paulo Sousa is the manager of the ACF Fiorentina* **,** *he played at the Inter Milan club* .
Participiale *participle clause* (p. XXXIII)	***After being recruited by NASA in 1963*** , *Alan Bean spent 100305 minutes in space* .
Participiale sujet *participle clause subject* (p. XXXIV)	***Serving the city of Alderney***, *Alderney Airport's 1st runway is made from Poaceae* .
`stanford parser` →	***`Serving the city of Alderney`*** `est annotée clause sujet passif de` ***`made`***`, ce qui est discutable.`
Voix passive *Voix passive* (p. XXXV)	*The A-Rosa Luna* ***was built*** *at the Neptun Werft in Germany* .
Possessif *possessif* (p. XXXV)	*The AIDAluna has a length of 252000.0 millimetres and* ***its*** *christening date was 2009-04-04* .
Relative adverbiale *relative adverb* (p. XXXVI)	*Andrews County Airport is located in Texas* ***where the capital is Austin*** *and the largest city is Houston* .
Relative objet *relative object* (p. XXXVI)	*Albany , Oregon is a city in Oregon* ***whose capital is Salem , Oregon*** .
Relative sujet *relative subject* (p. XXXVII)	*The ALCO RS-3 was built by the American Locomotive Company* ***which is located in*** *the* ***United States .***
COD et COI *twice objects* (p. XXXVIII)	*The All India Council gave the Acharya Institute of Technology* ***the status of Technical Campus .***
sans contrainte	*They fought* .

TABLE II. 1. 4 – Contraintes syntaxiques, exemples

contrainte / *étiquette*	#corpus	#éval.	rappel	précision	F-score
Apposition *apposition* (p. XXIV)	4 669	25	0.92	0.96	0.94
Coordination verbale *coordinated clauses* (p. XXV)	4 085	22	0.71	1.0	0.83
Coordination *coordinated full clauses* (p. XXIX)	2 890	24	0.71	0.92	0.8
COD *direct object* (p. XXXI)	11 889	70	0.99	1.0	0.99
Prédicative *existential* (p. XXXI)	25 429	97	0.97	1.0	0.98
COI *indirect object* (p. XXXII)	9	9	1.0	1.0	1.0
Juxtaposition *juxtaposition* (p. XXXII)	299	14	0.92	0.86	0.89
Participiale *participle clause* (p. XXXIII)	574	14	0.82	1.0	0.9
Participiale sujet *participle clause subject* (p. XXXIV)	28	7	0.5	0.14	0.22
Voix passive *Voix passive* (p. XXXV)	14 523	79	0.98	0.82	0.9
Possessif *possessif* (p. XXXV)	5 199	54	0.96	1.0	0.98
Relative adverbiale *relative adverb* (p. XXXVI)	3 166	21	1.0	1.0	1.0
Relative objet *relative object* (p. XXXVI)	1 433	17	0.94	1.0	0.97
Relative sujet *relative subject* (p. XXXVII)	6 712	35	1.0	1.0	1.0
COD et COI *twice objects* (p. XXXVIII)	8	8	1.0	1.0	1.0
sans contrainte	1 301	10	1.0	1.0	1.0
Total	82 214	506	0.96*	0.96*	0.95*

TABLE II. 1. 5 – Évaluation de la pertinence des étiquettes des leviers, sur la base de 150 textes examinés manuellement, avec #corpus le compte d'occurrences automatiquement comptabilisées dans le corpus, #éval. le compte d'occurrences automatiquement comptabilisées dans le corpus de l'évaluation manuelle et l'étoile(*) marque un nombre moyen pondéré par #corpus.

(Le rappel est défini par le nombre de documents pertinents sélectionnés au regard du nombre de documents pertinents. La précision est définie par le nombre de documents pertinents sélectionnés au regard du nombre de documents sélectionnés).

II. 1. 3.3 Corpus d'apprentissage

Pour chaque tâche de génération d'un texte T, nous visons l'apprentissage d'un modèle qui maximise la probabilité $P(T|I;k;\theta)$ d'une entrée I, selon une contrainte k pour des paramètres propres au modèle θ. À cette fin, nous utilisons un encodeur-décodeur exploitant le mécanisme d'attention. Des tests préalables nous ont montré qu'effectivement ce mécanisme simplifiait les traitements, ainsi que l'ont démontré Vaswani et al. (2017). Avec ce mécanisme, une mise en forme logiquement structurée de nos informations logiques n'a pas d'impact. La linéarisation est à entendre dans ce qui suit comme la mise à plat d'un lot de triplet RDF, les données contenues étant mécaniquement triées par ordre alphabétique selon valeur prédicat, sujet, puis objet.

Nous comparons nos modèles avec les cinq meilleures sorties par entrée produite quand aucune contrainte syntaxique n'est demandée. L'extraction de ces cinq phrases les plus probables pour une entrée donnée est réalisée par un *beam search*. D2T_{5best} est le modèle génératif *beam search* données vers texte sans contrainte, T2T_{5best} groupe ensemble les cinq sorties produites depuis un texte qui a pu par ailleurs être utilisé lors d'une expansion, une simplification ou en génération simple.

Nous évaluons aussi l'intérêt de coupler les modèles syntaxiquement contraints (ALL_{syn}) pour parvenir à une plus grande diversité syntaxique.

Beam Search : le socle de notre ligne de base

Beam Search est un algorithme qui permet de conserver n sorties probables pour une entrée donnée, à partir du softmax (voir Figure I. 1. 10).

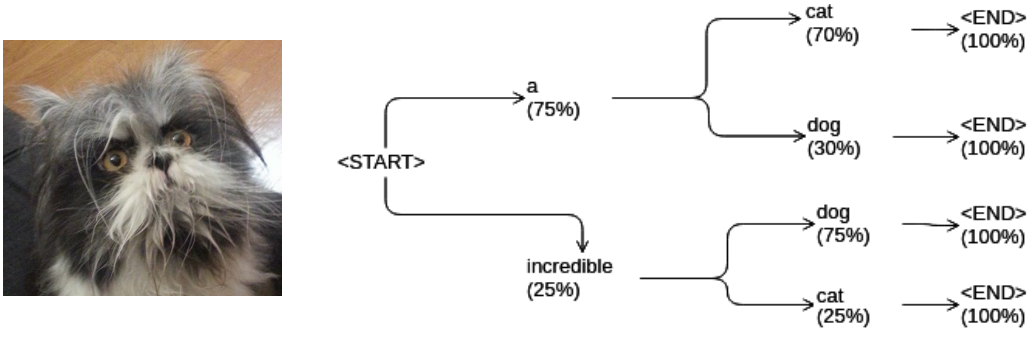


FIGURE II. 1. 3 – Beam Search, principe : ici quatre phrases les plus probables.

A l'instar des autres modèles, pour notre ligne de base, nous avons utilisé l'encodeur-décoder d'OpenNMT_{py} (Klein et al., 2017) avec les paramètres par défaut pour un en-

codage bidirectionnel avec usage du mécanisme d'attention. L'entraînement a été réalisé sur 13 époques. Il est illustré Figure II. 1. 4.

La délexicalisation

Afin d'optimiser l'apprentissage de la syntaxe portée par la langue, nous avons anonymisé les entités. Nous considérons la syntaxe - certes choisie par le locuteur - induite par la structure logique des triplets mais aussi par chacun des prédicats. Le prédicat *operatingOrganisation*, a des proximités avec *Nationality* et *Country*.

- *operatingOrganisation* : définit la société gérant un aéroport,
- *Nationality* : définit la nationalité de quelqu'un,
- *Country* : définit un rattachement territorial.

Le sujet d'*operatingOrganisation* sera tendanciellement objet dans le cas d'une existencielle, mais sujet pour *Nationality* et *Country*. La réalisation de surface diffère d'autant que les prédicats sont distincts :

Prédicat	**Exemple de lexicalisation** et **RDF**
operatingOrganisation	*Aktieselskab operates Aarhus Airport.*
	`(Aarhus Airport operatingOrganisation Aktieselskab)`
Nationality	*Aaron is american.*
	`(Aaaron Nationality American)`,
Country	*Madrid is part of Spain.*
	`(Madrid Country Spain)`

Table II. 1. 6 – Prédicats : Transversalité de la syntaxe

L'anonymisation vise à maximiser l'affranchissement des biais de corpus, l'apprentissage transversal de la syntaxe. Chaque entité non anonymisée peut être massivement utilisée dans les triplets de domaines particuliers (politiciens, sports. . .), et être tendanciellement présente dans un cadre syntaxique lui aussi particulier.

Nous avons délexicalisé les entrées à partir du RDF : les entités ont été repérées lors de la procédure de pré-traitement, les triplets RDF sont triés alphabétiquement (prédicat, sujet, objet), et les sujets/objets des triplets (les entités) renommés :

pour le sujet d'un prédicat donné : [nomPrédicat]*Sujet*[incrément]

pour l'objet d'un prédicat donné : [nomPrédicat]*Objet*[incrément]

Par exemple, le RDF 6-a verra `"1955_Dodge"` anonymisé `bodyStyleSujet1`. Si ce prédicat est présent deux fois avec deux valeurs différentes pour sujet, l'incrément est augmenté, `"Aston_Martin_V8"` prend pour valeur `bodyStyleSujet2` en 6-c2).
Cette méthode est celle utilisée pour tout le travail courant.

(6) a1. RDF : ("1955_Dodge" bodyStyle "Convertible")
b1. Texte : *The 1955 Dodge is a Convertible.*

c1. Texte anonymisé : *The bodyStyleSujet1 is a bodyStyleObjet1.*
a2. RDF : ("1955_Dodge" bodyStyle "Convertible") ("Aston_Martin_V8" bodyStyle "Convertible")
b2. Texte : *The 1955 Dodge and the Aston Martin 8 are convertibles.*
c2. Texte : *The bodyStyleSujet1 and the bodyStyleSujet2 are bodyStyleObjet1.*

Nous relexicalisons toutes les sorties pour un même lot de phrases probables issues du *beam search* avec l'information ayant servi de support pour l'entrée, que celle-ci soit du RDF ($D2T_{5best}$) ou du texte ($T2T_{5best}$).

$D2T_{5best}$: exemples de sorties en test

RDF : (Alan Bean , birthPlace , Wheeler , Texas),
(Alan Bean , occupation , Test pilot)

(7) Alan Bean was born in Wheeler, Texas and served as a Test pilot.
Alan Bean, who was born in Wheeler, Texas, served as a Test pilot.
Alan Bean was born in Wheeler, Texas and is a Test pilot.
Alan Bean, a Test pilot, was born in Wheeler, Texas.
Alan Bean was born in the Wheeler, Texas and served as a Test pilot.

$T2T_{5best}$: exemples de sorties en test

RDF : (Adare Manor , location , Adare),
(Adare Manor , location , County Limerick)

(8) Adare Manor is located in Adare, County Limerick.
Adare Manor is in County Limerick, Adare.
Adare Manor is in Adare, County Limerick.
Adare Manor is found in Adare, County Limerick.
Adare Manor is located in County Limerick, Adare.

Corpus : vue d'ensemble

Aux sorties de chacun de nos modèles correspondent des phrases. Une même phrase de sortie peut être liée à plusieurs instances d'apprentissage, chaque entrée préfixée avec une étiquette syntaxique différente.

Utilisant le corpus WebNLG ainsi enrichi, nous avons construit quatre corpus d'entraînement basés sur les lots de triplets RDF, exploités en pivots rattachés à des paraphrases. Au-delà, nous avons imposé que, dans les modèles impliquant du texte en entrée, avec k une contrainte, T_i un texte en entrée et T_o un texte en sortie, toute instance $\langle k, T_i, T_o \rangle$ T_o diffère de T_i sur la base d'exactement un label syntaxique[56]. Des statistiques plus détaillées sont fournies en annexe (D. 6). La base de test est construite

56. $K(T_i) = (K(T_i) \cap K(T_o)) \cup \{k\}$ et $K(T_o) = (K(T_i) \cap K(T_o)) \cup \{k'\}$ pour tout $k \neq k'$.

de telle sorte que pour toute entrée $\langle k, I \rangle$ y prenant place, $\langle k, I \rangle$ est absent des données d'apprentissage (tel que I est unité de donnée ou un texte).

Les statistiques descriptives de nos corpus d'apprentissage sont fournies Table II. 1. 7.

Corpus	Base	T2T$_{syn}$	T+r2T$_{syn}$	D2T$_{syn}$	T-r2T$_{syn}$	D2T$_{5best}$	T2T$_{5best}$	ALL$_{syn}$
#entrées	32,503	18,904	22,386	4,099	9,968	4,106	26,621	3,266
#sorties	0	8,703	6,758	26,403	8,063	27,144	26,621	1,997
Contraintes								
Apposition	4,669	401	1,206	4,390	1,178	NA	NA	223
Coordination verbale	4,085	287	894	3,837	857	NA	NA	186
Coordination	2,890	162	622	2,751	669	NA	NA	105
COD	11,889	1,708	1,408	9,938	1,281	NA	NA	451
Prédicative	25,429	4,023	1,609	18,352	1,304	NA	NA	480
COI	9	0	0	9	0	NA	NA	0
Juxtaposition	299	13	100	294	136	NA	NA	35
Voix passive	14,523	1,745	1,509	11,786	1,527	NA	NA	296
Possessif	5,199	232	704	4,549	887	NA	NA	138
Relative adverbiale	3,166	159	494	2,991	491	NA	NA	179
Relative objet	1,433	48	259	1,394	385	NA	NA	118
Relative sujet	6,712	351	1,097	6,279	1,543	NA	NA	216
COD et COI	8	0	0	8	0	NA	NA	0

TABLE II. 1. 7 – Tâche, Corpus et contraintes syntaxiques

Les modèles à contraintes syntaxiques sont marqués de *syn* en indice, T2T le modèle texte vers texte, T+r2T l'extension de texte et T-r2T la réduction de texte (texte+rdf en entrée), D2T le modèle RDF vers texte, ALL$_{syn}$ le modèle évaluant la conjonction des modèles contraints (à l'intersection des significations partagées comptabilisées #entrées).

II. 1. 3.4 Génération de paraphrases syntaxiques à partir de triplets RDF

D2T : encodeur-décodeur avec mécanisme d'attention

La Figure II. 1. 4 du modèle neuronal utilisé vise à en donner une représentation simplifiée.

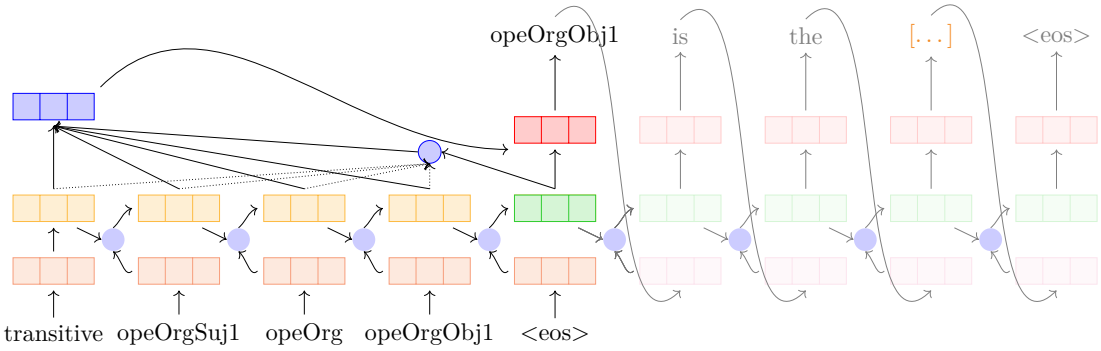


FIGURE II. 1. 4 – Données vers Texte (D2T) : modèle utilisé

Phrase support : *Aktieselskab is the operating organisation for Aarhus Airport.*,
Sur la figure, anonymisation : *opeOrgSuj1*, *Aarhus Airport*, *opeOrgObj1*, *Aktieselskab* et *opeOrg* pour *operatingOrganisation*.
RDF support : `(Aarhus Airport operatingOrganisation Aktieselskab)`,
Entrée : `transitive opeOrgSuj1 operatingOrganisation opeOrgObj1`,
Sortie attendue : *opeOrgObj1 is the operating organisation for opeOrgSuj1* .

Le modèle contraint RDF vers Texte ($D2T_{syn}$) associe une contrainte syntaxique à du RDF pour une sortie texte intégrant le phénomène linguistique recherché. La Table II. 1. 8 présente La Table II. 1. 8 présente un exemple (non délexicalisé pour plus de lisibilité).

$D2T_{syn}$		$M, k \Rightarrow T_o$ with $mg(T_o) = M, k \in K(T)$
	k	Coordination
	M	{ (Aarhus Airport operatingOrganisation Aktieselskab), (Aarhus Airport runwayLength 2776), (Aarhus Airport runwayName "10L/28R") }
	T_o	*Aarhus Airport is operated by the Aktieselskab. Its runway name is "10L/28R" and the length is 2776.*

TABLE II. 1. 8 – Tâche de génération syntaxiquement contraintes

(D2T : data-to-text, T_o : texte en sortie, k : contrainte syntaxique, M : représentation signifiante, un lot de triplets RDF, $mg(T)$: signification d'un texte T, $K(T)$: contraintes syntaxiques réalisées dans le texte T).

L'encodeur accueille les données au sein de deux LSTMs, son entrée étant bidirectionnelle. Le décodeur est lui aussi un LSTM. L'encodeur reçoit en entrée une séquence contenant à la fois I et la contrainte syntaxique k. Pour la génération données vers texte ($D2T_{syn}$), l'entrée est, au-delà de la contrainte k, une version linéarisée et délexicalisée

d'un lot de triplets RDF représentant la signification du texte attendu en sortie, pour la génération texte vers texte (T2T$_{syn}$), l'entrée I est un texte de même signification que celui en sortie, et pour les générations à entrées hybrides, données et texte vers texte (T+r2T$_{syn}$ et T-r2T$_{syn}$), l'entrée est un texte couplé à un triplet RDF linéarisé. Nous présenterons Partie II. 1. 3.5 les modèles T2T$_{syn}$, T+r2T$_{syn}$ et T-r2T$_{syn}$.

Métrique d'évaluation et résultats

Évaluation. Au niveau des textes générés, nous avons évalué à la fois la diversité des paraphrases et l'adéquation linguistique/syntaxique.

Model	BLEU	Synt	BLEU$_{syn}$	Sim	#Txt/M	#Corpus	#SPar/Mg
T2T$_{5best}$	6.21	N/A	N/A	0.76 (0.61)	**5.85 (44.98)**	715910	**3.98 (5.5)**
D2T$_{5best}$	4.71	N/A	N/A	0.81 (0.63)	4.71 (5.98)	27156	1.86 (4.42)
D2T$_{syn}$	**46.20**	0.84	48.16	**0.81 (0.61)**	1.04 (3.87)	66595	1 (1.1)

TABLE II. 1. 9 – Résultats et tailles des corpus, {D2T}$_{syn}$

score BLEU$_{syn}$: proportion de sorties satisfaisant k et les objets attendus ; Sim : similarité moyenne paire-à-paire entre les textes sortant pour une signification donnée ; # Txt/M : nombre moyen de textes par signification (M = *meaning*) ; (Sim et Txt/M, valeurs entre parenthèses : moyennes sur le corpus) ;# Corpus : taille du corpus (80% apprentissage, 10% évaluation, 10% test) ; #SPar/Mg : Nombre de paraphrases syntaxiques par sens selon évaluation humaine (*vs.* nombre moyen de sorties considérées).

Il est remarquable dans les données de référence que le nombre de phrases par entrée est supérieur au nombre de phrases par lot de triplets RDF. En effet, la délexicalisation permet à une structure prédicative de regrouper plus de textes : par exemple, le RDF concernant un aéroport x pourra voir ses phrases utilisées pour un aéroport y pour toute structure arborescente RDF commune. Cela nous représente 44.98 phrases exprimant une même information logique en texte vers texte (T2T$_{5best}$) où un produit cartésien s'applique entre les sorties d'un même support logique.

En D2T$_{5best}$, nous obtenons une diversité de 5.98 phrases pour chaque lot de triplets linéarisé avec une sortie par paire disponible sur le corpus de référence, soir 27156 couples entrée/sortie.

Pour le modèle syntaxiquement contraint, D2T$_{syn}$, nous avons plusieurs phrases identiques pour un même sens, l'entrée se différenciant par une contrainte. Nous avons néanmoins 3.87 phrases distinctes par entrée.

Adéquation syntaxique et linguistique (BLEU, Syn, BLEU$_{syn}$) .

Pour l'ensemble des modèles, nous calculons un score BLEU qui, pour les modèles syntaxiquement contraints, s'appuie sur une référence satisfaisant la contrainte k demandée en entrée. Le score BLEU est une mesure de proximité entre une cible syntaxique et une phrase générée. Il évalue la finesse du modèle à pouvoir générer le phénomène attendu – comme le nombre de références varie d'une entrée type à l'autre, c'est au niveau de la phrase que nous le calculons (Papineni et al., 2002). Le BLEU (avec n-gram 4) est de minimum 40 points supérieur à la ligne de base.

Cette mesure est complétée par le calcul de la proportion de textes satisfaisant la contrainte syntaxique (Synt) et contenant les objets attendus. Pour le modèle RDF vers Texte contraint, la couverture syntaxique moyenne est de 84%. Si la contrainte demandée en entrée est trouvée dans la sortie, la couverture syntaxique pour la phrase est de 1, sinon de 0.

Le score BLEU pour ces sorties satisfaisant la demande a été expressément calculé (BLEU_{syn}). Il a pu l'être suite à une pré-analyse des sorties générées consistant en l'application algorithmique du détecteur de contraintes. Le BLEU gagne quasiment 2 points sur les phrases couvrant syntaxiquement la demande.

Diversité (Sim, #Txt/Mg) Pour mesurer la diversité des paraphrases obtenues, nous avons regroupé ensemble les sorties dont on attend qu'elles partagent une même signification (*meaning* : les sorties liées aux données WebNLG d'un même lot de triplets) et nous calculons le nombre de textes distincts générés (# Txt/Mg). Nous analysons plus en profondeur ces sorties en évaluant le taux moyen de proximité entre ces paires (Sim). Nous utilisons l'algorithme Ratcliff/Obershelp (Black, 2004) pour ce calcul de similarité[57]. Une similarité basse indique une plus grande diversité entre textes partageant une même signification.

La similarité en phrases générées par le modèle RDF vers Texte sous contrainte est aussi élevée que lorsqu'on génère depuis du RDF vers du texte sans contrainte : similaires à un taux de 0.81, elles sont peu variées. Le taux marqué entre parenthèse indique la diversité mesurée sur le corpus de référence. Nous n'atteignons pas la diversité du corpus de référence (0.61 pour ce modèle).

Nous obtenons seulement 1.04 phrases par signification différente, alors qu'une description RDF complète testée en offre en moyenne 3.87.

Évaluation humaine (#SPar). Pour chaque modèle, un linguiste expert a examiné, ce pour cinquante significations, un maximum de dix sorties choisies aléatoirement et enregistré le nombre moyen (#SPar) de paraphrases par entrée syntaxiquement correctes.

Le faible nombre de texte produit par ensemble RDF induit une faible diversité relevable : nous avons une phrase correcte par RDF sur le corpus de test dédié à validation humaine... sur les 1.1 phrases par RDF. En parallèle, les modèles de références (T2T_{5best} et D2T_{5best}) offrent respectivement presque 4 phrases syntaxiquement différentes pour le premier, 1.86 pour le second. Ses taux sont à mettre en regard avec le nombre de phrases soumises à évaluation par RDF : 5.5 et 4.42. Pour une représentation RDF donnée, l'ensemble des phrases différentes produites ont été présentées à l'évaluateur.

57. Le score de similarité Ratcliff/Obershelp varie entre 0 et 1 où 1 correspond à une identité de couple. Le calcul s'exprime par la formule $sim(S1, S2) = \frac{2*match(S1,S2)}{len(S1)+len(S2)}$ où une identité de couple est définie par la somme de longueur des segments identiques ramenée à la longueur totale des segments ($match(S1, S2) = \sum_{m \in overlap(S1,S2)} len(m)$.

II. 1. 3.5 Génération de paraphrases syntaxiques à partir de textes et de données hybrides Texte/RDF

Notre modèle données vers texte syntaxiquement contraint (D2T$_{syn}$) ne représentait qu'une étape expérimentale.

Nous servant des données RDF comme pivot, nous avons constitué trois autres corpus. Nous avons des lots de triplets, à chacun correspondant des phrases. Ainsi qu'expliqué page 70 et de façon plus générique dans la description de WebNLG (page 74), les lots RDF sont organisés par taille, chaque triplet se voyant verbalisé dans le premier groupe (ensemble RDF ne contenant que un seul triplet par lot). Chaque lot RDF est potentiellement inclus dans un lot ou plusieurs lots RDF de taille supérieure.

Par exemple, (9-a) est incluse dans (9-b), mais pas dans (9-c) si ce n'est grâce à la délexicalisation.

(9) a. `(Aarhus Airport operatingOrganisation Aktieselskab)` : groupe RDF 1 (lot RDF de un triplet).
b. `(Aarhus Airport operatingOrganisation Aktieselskab), (Aarhus Airport runwayName "10R/28L")` : groupe RDF 2.
c. `(Afonso Pena International Airport operatingOrganisation Infraero), (Afonso Pena International Airport location São José dos Pinhais)` : groupe RDF 2.

Le RDF (9-a) peut mener à des phrases comme :

(10) a- *Aktieselskab is the operating organisation for Aarhus Airport .*
b- *Aktieselskab operates Aarhus Airport .*
c- *Aarhus Airport is operated by the Aktieselskab organisation .*

Les phrases (10) présentent respectivement une prédicative, un complément d'objet direct, un passif.

Le RDF (9-b) mène à des phrases comme :

(11) a- *Aarhus Airport is operated by the Aktieselskab organisation and the runway name is 10L/28R.*
b- *Aktieselskab operates Aarhus Airport which has the runway known as 10L/28R.*
c- *Aktieselskab operates Aarhus Airport , and its runway is called 10L/28R.*
d- *Aktieselskab is the operating organisation for Aarhus Airport which has the runway name 10R/28L.*
e- *Aktieselskab is the operating organisation for Aarhus Airport , where the runway name is 10R/28L.*
f- *The runway name at Aarhus Airport , operated by Aktieselskab , is "10R/28L".*

Le modèle neuronal utilisé pour nos trois autres modèles, illustré Figure II. 1. 5, va autoriser l'apprentissage de la syntaxe de manière plus manifeste et plus propice que notre premier modèle. Nous atteignons un BLEU-4 sur phrases contraintes pouvant

aller jusqu'à 89.32. Mieux encore, la similarité entre phrases sortantes pour une même représentation RDF est très proche, voire identique à celle des corpus d'apprentissage (voir résultats II. 1. 11). Enfin, offrant la simple possibilité de multiplier les modes de traitement de l'information, ces trois modèles, ajoutés au premier, nous permettront d'évaluer une diversification poussée des sorties.

Nous avons dans les lexicalisations (11) du passif, de la coordination, relative sujet (introduit par *which*), adverbial (*where*)...

À l'aide de notre ressource, nous avons construit un premier modèle, texte vers texte qui transforme une phrase en une autre sous contrainte syntaxique. Cette contrainte n'est pas dans la phrase d'entrée, elle est uniquement dans la phrase de sortie. Les données d'apprentissage sont construites de telle sorte que pour toute entrée $\langle k, r, t\rangle$ y prenant place, toute sortie o, tel que t est un texte, r du RDF pouvant être égal à $\emptyset$, tel que o est un texte, k est une contrainte présente dans o et absente dans le texte de t. Plus encore, o diffère de t d'exactement une contrainte.

Une différence d'exactement une contrainte prend deux formes : soit o a une contrainte en plus que t, et c'est par celle-ci que la transformation est demandée, soit t et o ont exactement le même nombre de contraintes, auquel cas la contrainte demandée pour o n'est pas dans t, et t intègre une contrainte qui n'est pas dans o.

{T2T, T-r2T, T+r2T}$_{syn}$: encodeurs-décodeurs avec mécanisme d'attention

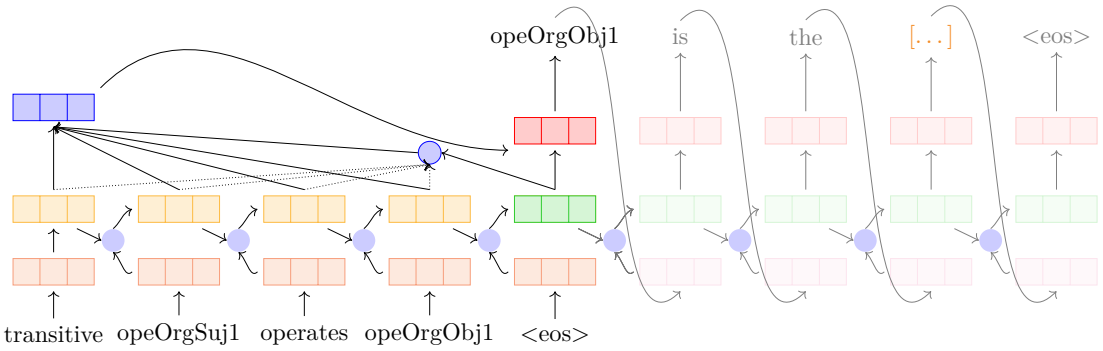


FIGURE II. 1. 5 – Texte vers Texte (T2T$_{syn}$) : modèle utilisé
Phrases supports : *Aktieselskab is the operating organisation for Aarhus Airport.*, *Aktieselskab operates Aarhus Airport.*
Sur la figure, anonymisation : *opeOrgSuj1*, *Aarhus Airport*, *opeOrgObj1*, *Aktieselskab*.
RDF support : `(Aarhus Airport operatingOrganisation Aktieselskab)`,
RDF anonymisé : `(opeOrgSuj1 operatingOrganisation opeOrgObj1)`,
Entrée : `transitive opeOrgObj1 operates opeOrgSuj1 .`,
Sortie attendue : *`opeOrgObj1 is the operating organisation for opeOrgSuj1 .`*

Nous créons trois corpus en nous basant sur les triplets RDF délexicalisés et leurs phrases correspondantes, délexicalisées elles aussi.

Les triplets sont organisés en lots rdf_1 à rdf_7, le lot de triplets rdf_1 ne contenant qu'un triplet, rdf_2 2 triplets...jusqu'à rdf_7 qui en contient 7.

Pour nos trois corpus supplémentaires, le choix des couples d'entrée sortie se fait selon l'équation II.1.1 : sur la base d'une restriction au niveau des contraintes et d'une définition RDF partagée entre entrée et sortie.

Le premier corpus est basé sur des phrases appartenant à un même lot et partageant une même définition RDF délexicalisée. Il s'agit d'un modèle texte vers texte sous contrainte syntaxique. Il associe une contrainte k à une phrase t_i et t_o respectant l'équation II.1.1.

Le second corpus associe des phrases lexicalisant un lot rdf_j vers des phrases lexicalisant un lot rdf_{j+1} avec $j \in 1, \ldots, 6$.
$\text{rdf}_j \subset \text{rdf}_{j+1}$
$\text{rdf}_{j+1} \cap \text{RDF}_j = \{RDF\} | RDF$ est un triplet RDF .
Il s'agit d'un modèle "entrée hybride vers texte" sous contrainte syntaxique. Respectant l'équation II.1.1, il associe une contrainte k à une phrase t_i représentée par rdf_j, et un triplet *rdf* tel que précédemment défini, avec une phrase t_o représentée par rdf_{j+1}.
Ce second corpus cible donc l'extension du texte t_i sous contrainte syntaxique. L'information présente de t_o et absente dans t_i est spécifié par *rdf*.

Le troisième corpus associe des phrases lexicalisant un lot rdf_j vers des phrases lexicalisant un lot rdf_{j-1} avec $j \in 2, \ldots, 7$.
$\text{rdf}_{j-1} \subset \text{rdf}_j$
$\text{rdf}_{j-1} \cap \text{RDF}_j = \{RDF\} | RDF$ est un triplet RDF .
Il s'agit d'un modèle "entrée hybride vers texte" sous contrainte syntaxique. Respectant l'équation II.1.1, il associe une contrainte k à une phrase t_i représentée par rdf_j, et un triplet *rdf* tel que précédemment défini, avec une phrase t_o représentée par rdf_{j-1}.
Ce troisième corpus cible donc la réduction du texte t_i sous contrainte syntaxique. L'information absente de t_o et présente dans t_i est spécifié par *rdf*.

$$K_i \cap K_o = \{k\} \cup \{u\} \text{ avec } k! = u, k \in K_o, u \in K_i \qquad \text{(II. 1. 1)}$$

Définition contrainte autorisée corpus

Une phrase t_o contenant une contrainte k absente d'une autre t_i mais partageant avec elle la même définition rdf_j permettra la création d'un tuple k, t_i, t_o où (k, t_i) sera une entrée, t_o une sortie, si et seulement si pour K_i, K_o l'ensemble des contraintes de respectivement t_i, t_o respecte cette définition.

La Table II. 1. 10 résume les trois autres modèles à contraintes et présente un exemple d'entrée sortie pour chacun, non délexicalisé pour plus de lisibilité. Les résultats sont présentés Table II. 1. 11.

T2T$_{syn}$		$T_i, k \Rightarrow T_o$ with $mg(T_i) = mg(T_o), k \in K(T)$
	k	**Transitive**
	T_i	*Afonso Pena International Airport is located in São José dos Pinhais. The runway length is 2215 and the runway name is 11/29.*
	T_o	*Afonso Pena International Airport in São José dos Pinhais **has a runway** known as 11/29 with a length of 2215.*
T+r2T$_{syn}$		$T_i, k, t \Rightarrow T_o$ with $mg(T_o) = mg(T_i) \cup \{t\}, k \in K(T)$
	k	**Possessif**
	T_i	*Madrid is part of Community of Madrid whose leader party is the Ahora Madrid. The Adolfo Suárez Madrid–Barajas Airport is located there.*
	t	{ (Madrid country Spain) }
	T_o	*Adolfo Suárez Madrid–Barajas Airport is located in Madrid, part of the Community **of Madrid** in Spain where the leader party is Ahora Madrid.*
T-r2T$_{syn}$		$T_i, k, t \Rightarrow T_o$ with $mg(T_i) = mg(T_o) \cup \{t\}, k \in K(T)$
	k	**Relative sujet**
	T_i	*Al Asad Airbase is located at "Al Anbar Province, Iraq" and operated by the United States Air Force. The base 's runway called "08/26" and 3990 meters long.*
	t	{(Al Asad Airbase operatingOrganisation United States Air Force)}
	T_o	*Al Asad Airbase (in "Al Anbar Province, Iraq"), has a runway named "08/26" and a runway **that is** 3990 metres long.*

TABLE II. 1. 10 – Tâches de génération syntaxiquement contraintes

T2T : *text-to-text*, T+r2T : *expansion de texte*, T-r2T : *réduction de texte*, T_i, T_o : texte en entrée/sortie, k : contrainte syntaxique, M : représentation signifiante, un lot de triplets RDF, $mg(T)$: signification d'un texte T, $K(T)$: contraintes syntaxiques réalisées dans le texte T

Métrique d'évaluation et résultats

Model	BLEU	Synt	BLEU$_{syn}$	Sim	#Txt/M	#Corpus	#SPar/Mg
D2T$_{5best}$	4.71	N/A	N/A	0.81 (0.63)	4.71 (5.98)	27156	1.86 (4.42)
T2T$_{5best}$	6.21	N/A	N/A	0.76 (0.61)	**5.85 (44.98)**	715910	**3.98 (5.5)**
T2T$_{syn}$	49.95	**0.98**	50.23	0.72 (0.68)	1.92 (18.49)	202218	1.58 (1.70)
T-r2T$_{syn}$	66.63	0.70	80.45	**0.62 (0.63)**	**2.92 (6.10)**	40936	**2.56 (3.9)**
T+r2T$_{syn}$	**83.87**	0.88	**89.32**	0.68 (0.68)	2.56 (6.55)	74844	0.92 (1.16)

TABLE II. 1. 11 – Résultats et tailles des corpus, {T2T, T-r2T, T+r2T}$_{syn}$

score BLEU, Synt : proportion de sorties satisfaisant k et les objets attendus ; BLEU$_{syn}$: score BLEU pour les sorties satisfaisant k et les objets attendus ; Sim : similarité moyenne paire-à-paire entre les textes sortant pour une signification donnée (que les entrée soient mixtes, données ou texte) ; # Txt/M : nombre moyen de textes par entrée (M = *meaning*) ; # Corpus : taille du corpus (80% apprentissage, 10% évaluation, 10% test ; #SPar/Mg : Nombre de paraphrases syntaxiques par sens selon évaluation humaine(*vs.* nombre moyen de sorties considérées).

Évaluation. Au niveau des textes générés, nous avons évalué à la fois la diversité des paraphrases et l'adéquation linguistique/syntaxique, comme pour le modèle données vers texte syntaxiquement contrôlé (D2T$_{syn}$).

Les modèles obtenus sont complémentaires, ce qui reflète l'exploitation qu'a chacun du corpus. Le modèle texte vers texte sous contrainte "marie" des exemples équivalents sous contrainte syntaxique, ce qui permet au réseau d'apprendre pleinement ce que la contrainte induit lexicalement et syntaxiquement.

Pour le modèle T2T$_{syn}$, le taux de couverture syntaxique (respect de la contrainte syntaxique analysé automatiquement) tend vers les 100%. Le BLEU de ce modèle est le plus faible du groupe, en cause la diversité des exemples appris en sortie pour une même entrée textuelle : 18.49 phrases en moyenne par signification. Ce corpus, du fait de sa construction quasiment cartésienne des exemples dispose de 202 218 couples entrée/sortie , soit près de 5 fois la taille du corpus T-r2T$_{syn}$ (réduction de texte) et 2.7 fois celle de T+r2T$_{syn}$ (expansion de texte). Il obtient un BLEU-4 supérieur de 3.75 point à celui de D2T$_{syn}$, soit 49.95 points, ce qui jouxte le BLEU-4 des phrases évaluées comme respectant la contrainte attendue (50.23). Aucune surprise ici, 98% des phrases respectant théoriquement l'attente.

Le modèle T-r2T$_{syn}$, ciblant la réduction de texte, ne maximise pas l'usage des lexicalisations dans un cadre d'extension/réduction. La distribution des lots de triplets, disponible en annexe D. 2, n'est pas égale : le corpus WebNLG offre presque 4000 lots en rdf 1, contre 450 cumulés en RDF 6 et 7, et moins de 2400 en rdf 5. Il n'y a aucune entrée en texte réduction basée sur les lexicalisations les plus simples (rdf 1) puisque nous partons au moins d'une lexicalisation dont le RDF de référence est composé de deux triplets (rdf 2) pour la réduire à une lexicalisation issue du groupe rdf 1. Il s'ensuit une base d'apprentissage *de facto* plus petite (40 936 couples entrée/sortie), mais aussi une complexité différente du modèle T+r2T$_{syn}$: le modèle apprend à aller vers des phrases moins denses ; toute l'information est dans l'entrée verbalisée, il y a donc une part de transformation de l'entrée lexicalisée vers une sortie plus simple dans laquelle le levier syntaxique permet peut-être un apprentissage de la syntaxe plus facile qu'en extension.

Dans le modèle T-r2T$_{syn}$, nous obtenons le meilleur score en terme de sorties syntaxiquement différentes pour une même signification (2.56 textes en moyenne sur les 3.9 textes par sens évalués par un linguiste). Le nombre moyen de textes obtenus par signification sur l'ensemble du corpus de test est de 2.92 (contre 6.10 pour le corpus complet). La réduction de texte est le modèle permettant d'avoir la plus grosse dissimilarité entre sortie (similarité 0.62), cette dissimilarité jouxte celle du corpus d'apprentissage (0.63 pour la similarité entre sortie de même signification sur ce dernier).

Le modèle T+r2T$_{syn}$ permet lui aussi d'apprendre correctement les variations intra-lexicalisations (similarité de 0.68 en test comme pour le corpus de référence). Ses sorties permettent moins de variétés que T-r2T$_{syn}$, puisque l'expertise humaine n'a validé que

0.92 phrases différentes par entrée. Cela signifie que non seulement il y en moyenne peu de phrases différentes sur les sorties testées (l'expert n'en a vu en moyenne que 1.16 par signification), mais que de surcroît certaines significations n'ont offert aucune sortie validable (phrase incorrecte). De bons scores sont à relever côté métrique automatique, qui se réfèrent à l'ensemble des données de test. La couverture syntaxique est de 88% (70 % pour la réduction de texte et 98 % pour le texte vers texte). Le BLEU-4 est particulièrement élevé : 89.32 sur les sorties respectant la couverture syntaxique attendue et 83.97 sur l'ensemble des données testées. À noter, nous obtenons en moyenne 2.56 textes par signification (6.55 sur le corpus de référence). Ces textes peuvent avoir une diversité syntaxique similaire, répondre à plusieurs contraintes.

Usage combiné des modèles

Une relativement grande diversité des sorties, avec contrôle syntaxique, est accessible en combinant des sources d'informations différentes sur une sémantique équivalente. Le modèle ALL_{syn} associe en effet les paraphrases obtenues lorsque nous générons du texte à partir de données, de texte, ou encore de texte couplé à des données. Cette dernière situation est couverte par les deux dernières tâches évoquées, nommées expansion et simplification. Pour chacune de ces tâches, l'entrée consiste en un texte associé à une unité de données. Pour l'expansion, la sortie est un texte verbalisant l'ensemble des informations contenues dans l'entrée, qu'elles appartiennent à la part textuelle ou au triplet RDF (l'unité de données). Réciproquement, pour la simplification, la sortie attendue est un texte verbalisant l'information contenue dans le texte d'entrée, déduction faite de celle contenue dans le triplet RDF. La table II. 1. 12 présente le modèle combiné. Quelques exemples de sorties issues de ALL_{syn} pour une entrée signifiante sont présentés par les phrases (12).

Modèle	BLEU	Synt	BLEU_{syn}	Sim	#Txt/Mg	#Corpus	#SPar/Mg
T2T_{5best}	6.21	N/A	N/A	0.76 (0.61)	5.85 (44.98)	715910	3.98 (5.5)
D2T_{5best}	4.71	N/A	N/A	0.81 (0.63)	4.71 (5.98)	27156	1.86 (4.42)
ALL_{syn}	62.87	0.91	65.11	**0.61** (0.62)	**13.25** (68.95)	NA	**6.3** (15)

TABLE II. 1. 12 – Résultats et tailles des corpus pour ALL_{syn}

score BLEU, Synt : proportion de sorties satisfaisant k et les objets attendus ; BLEU_{syn} : score BLEU pour les sorties satisfaisant k et les objets attendus ; Sim : similarité moyenne paire-à-paire entre les textes sortant pour une signification donnée (que les entrée soient mixtes, données ou texte) ; # Txt/M : nombre moyen de textes par entrée (M = *meaning*) ; # Corpus : taille du corpus (80% apprentissage, 10% évaluation, 10% test ; # SPar/Mg (%) : Nombre de paraphrases syntaxiques par sens selon évaluation humaine(*vs.* nombre moyen de sorties considérées).
Le corpus ALL_{syn} n'a pas de corpus d'apprentissage propre.

(12) **a.** Al Anderson (NRBQ band), a member of the band NRBQ, performs Country music. He also played once with The Wildweeds.
b. Country music artist Al Anderson (NRBQ band), who is part of the The Wildweeds, is associated with musical artist NRBQ.
c. Al Anderson (NRBQ band) is part of NRBQtrois and once was part of The Wildweeds. NRBQ is a Country music band.

d. Al Anderson (NRBQ band) plays Country music music and performed for the The Wildweeds. Al Anderson (NRBQ band) is also associated with NRBQ.
e. Country music artist NRBQ, is associated with Al Anderson (NRBQ band), who is associated with artist The Wildweeds.

Le modèle combiné ALL$_{syn}$ permet d'obtenir la plus grande dissimilarité entre sorties pour une même représentation. Nous sommes à un taux de similarité de 0.61, inférieur à celui du corpus. Les modèles se complètent, et nous obtenons 13.25 textes, ce qui est plus de deux fois supérieur au nombre obtenus par T2T$_{5best}$. L'évaluation manuelle compte elle 6.3 texte par signification. Le taux de couverture syntaxique reste honorable (91%) des sorties incluant la contrainte attendue, et le BLEU-4 permet d'évaluer une qualité globale respectable (65.11 sur les sorties a priori correctement contraintes).

Ce modèle tire profit de ce qui le compose. Le rapport paraphrases distinctes et phrases considérées (évaluation humaine) est quasiment aussi bas pour le modèle ALL$_{syn}$ que pour le pire des modèles non contraint syntaxiquement, D2T$_{5best}$ (voir Table II. 1. 13). L'évaluation humaine de ALL$_{syn}$ est l'évaluation des phrases à l'intersection des corpus testés. Les tests sont accomplis sur support des sens (*meanings*) : nous évaluons la richesse syntaxique par sens. Les sens regroupant les phrases offrant le plus de présence sont conséquemment plus présents à l'intersection des corpus évalués. Ce fait impacte à la hausse le nombre statistique de sorties par sens... et à la baisse le ratio paraphrases correctes par sens sur nombre de sorties considérées : plusieurs modèles auront "réussi" à générer la phrase telle que demandée. Si ces sorties sont similaires syntaxiquement, et valides, elles ne compteront que pour un.

Les sorties évaluées sont disponibles en Annexe D. 6.2.

Modèle	D2T$_{5best}$	T2T$_{5best}$	D2T$_{syn}$	T2T$_{syn}$	T-r2T$_{syn}$	R+r2T$_{syn}$	ALL$_{syn}$
#Spar/Mg	1,86	3,98	1,00	1,58	2,56	0,92	6,3
#Consid	4,42	5,5	1,10	1,7	3,9	1,16	15
Ratio	0,42	0,72	0,91	0,93	0,66	0,79	0,42

TABLE II. 1. 13 – Nombre paraphrases correctes par sens/Sorties considérées

#SPar/Mg : Nombre de paraphrases syntaxiques par sens selon évaluation humaine ; #Consid : nombre moyen de sorties considérées par sens ; Ratio : rapport entre les deux.

Le modèle Data vers Texte (évaluation Table II. 1. 9), qui offre une faible diversité de ses sorties mais une couverture des demandes honorable apporte modestement sa pierre. Le levier syntaxique est respecté en génération dans 84% des cas. La qualité de ses sorties lui permet un ratio de 0.91 entre phrases conservées (correctes) et distinctes *vs* phrases considérées (Table II. 1. 13) : peu de pertes.

Le modèle Texte vers Texte (évaluation Table II. 1. 11) - sous contrainte syntaxique a une couverture syntaxique excellente (98% des contraintes demandées retrouvées en sortie). Le ratio (Table II. 1. 13) est fort (0.93).

Le modèle T-r2T$_{syn}$ (évaluation Table II. 1. 11) offre une dissimilarité entre phrases à la hauteur de celle du corpus d'apprentissage, ce qui est accompagné d'une couverture syntaxique de la demande plus faible que pour les autres modèles (70% en évaluation

automatique), une perte illustrée avec justesse par le ratio de 0.66 entre nombre de phrases correctes, distinctes et nombre de phrases considérées par l'évaluateur.

Le modèle T+r2T$_{syn}$ (évaluation aussi présentée Table II. 1. 11) offre une diversité syntaxique faible (0.92 phrases syntaxiquement différentes sur le corpus évalué manuellement). Avec sa couverture syntaxique forte et son BLEU élevé, on note sans surprise relativement peu de pertes.

Synthèse Des modèles différents, achoppant sur des difficultés diverses, sachant que l'encodeur et le décodeur diffèrent techniquement (LSTM bidirectionnel pour l'entrée, réseau récurrent couplé à un mécanisme d'attention pour la sortie).

Un principe est commun aux différents sous-modèles à entrée texte de ALL$_syn$: la contrainte syntaxique vient d'une représentation (traitée par l'espace inter-{encodeur/-décodeur}) d'où elle est absente : la phrase d'origine ne contient pas la contrainte qui est en sortie. Elle s'en distingue - pour les contraintes traitées - par la perte d'un phénomène au profit du phénomène recherché.

- extension de texte (T+r2T$_{syn}$) : phrases plus longues en sortie, travail plus simple pour l'encodeur,
- réduction de texte (T-r2T$_{syn}$) : phrases plus longues en entrée, travail plus simple pour le décodeur,
- texte vers texte (T2T$_{syn}$), phrases de toute taille possible en entrée/ sortie, avec produit cartésien et maximisation de la possibilité pour le réseau d'apprendre où est le phénomène syntaxique,
- données vers texte (D2T$_{syn}$), phrases de toute taille possible en sortie, avec levier syntaxique guidant la sortie vers une phrase cohérente l'intégrant, ce sur de nombreux exemples.

Ces paramètres différenciant les apprentissages permettent leur complémentarité : chacun tire profit différemment du corpus initial. Cela nous permet d'offrir ce modèle qui génère le plus grand nombre de paraphrases syntaxiques pour un même signifiant : 6.3 par sens en évaluation humaine, soit plus d'un tiers de la sortie T2T$_{5best}$ (Table II. 1. 13) qui offre le plus de diversité parmi les modèles testés, en dépit de l'absence de levier syntaxique pour opérer les générations.

II. 1. 4 Conclusion

Nous avons proposé de nouveaux modèles à contrainte syntaxiques pour la génération de texte et montré que leur usage permettait effectivement la génération de paraphrases syntaxiques. Les résultats montrent des différences marquées entre la génération impliquant expansion ou simplification *versus* génération D2T et T2T.

Diverses applications sont envisageables. L'automatisation de la génération automatique de texte est source d'erreurs difficiles à détecter, un humain se relit, peut être relu ; automatiser signifie autant pouvoir générer des proportions jamais imaginées que pouvoir générer dans un cadre interactif.

L'intersection de modèles offre des possibilités de classement qualitatif des sorties. L'échec de la relexicalisation permet l'élimination directe de textes. L'absence du levier syntaxique attendu a *a priori* une validité étudiable en terme de détection d'erreur.

Enfin, si nous nous situons dans l'assistance à la création d'exercices, nous n'oublions pas que des applications sont envisageables dans le web des données. Le procédé de génération peut en effet être inversé (analyse sémantique) en vue de dériver des triplets RDF à partir de texte. Le levier syntaxique accompagné d'un texte d'entrée peut permettre le repérage dans la lexicalisation des sujets ou triplets de relations. Si les relations entre entités ont maintes manières d'être exprimées, cette expression n'est pas liée au hasard. Ce qui est mangé ne sera sujet d'un verbe signifiant "manger" que sous la forme passive, par exemple.
Une segmentation de la recherche d'information peut aussi être envisagée à l'aide de l'extension ou de la réduction. La construction d'une base d'apprentissage exemple pourrait procéder suivant cette logique à partir d'une base type WebNLG : transformation d'un texte en RDF, transformation progressive de texte en RDF (sortie d'un triplet situé syntaxiquement et d'une phrase courte elle-même réductible à un triplet suivi d'une phrase courte ou vide), la réduction, l'extension d'un texte sous de multiple formes pour en travailler l'analyse. . . , isolement de triplets. . .

Une autre direction de recherche qui reste ici ouverte concerne l'utilisation des phrases générées pour la création d'exercices d'exercices de grammaire tel qu'illustré par la Table II. 1. 1. Il serait intéressant d'évaluer *in situ* la proportion de phrases qui peuvent effectivement servir de base à la création d'exercices : étant donnée un requête utilisateur (un enseignant) pour une construction syntaxique donnée, combien des phrases générées par notre modèle sont elles acceptées par l'enseignant ? Quelles sont les causes de rejet (grammaticalité, sémantique, naturel) ? Pour les apprenants, quelle est l'impact de notre méthode : la génération automatique d'exercices de grammaire permet elle un meilleur apprentissage ? Pour les enseignants, dans quelle mesure cette automatisation représente-t'elle un gain en temps et en diversité ?

Chapitre 2

Génération de phrases en français à partir de représentations de sens

II. 2. 1 Introduction

La réalisation de surface (SR, *Surface Realisation*) consiste à générer un texte à partir d'une représentation sémantique (MR, *Meaning Representation*). Nos travaux dans ce domaine en sont très proches. Par exemple, à partir de la représentation syntaxico-sémantique abstraite (AMR, Abstract Meaning Representation) de la Figure II. 2. 1, l'objectif est de générer une phrase verbalisant cette représentation e.g., *"I beg you to excuse me"*.

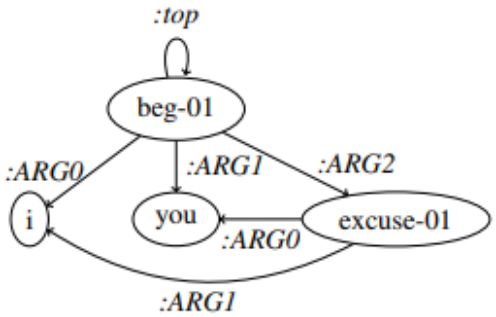


FIGURE II. 2. 1 – AMR.

La réalisation de surface (SR) est un élément clé de la génération de textes à partir de données : elle est mise en jeu pour convertir les représentations produites par les différentes phases de planification (sélection et structuration de contenu, agrégation, génération d'expressions référentielles) en phrases. Elle est également utile dans des

applications telles que la synthèse ou la traduction automatique, où l'objectif est de reformuler le contenu informationnel et où les représentations sémantiques fournissent un interlingua qui permet d'abstraire des différences de surface. Ainsi, Liao et al. (2018) montrent que l'incorporation de représentations sémantiques abstraites(AMR, *Abstract Meaning Representations*) en tant que connaissances supplémentaires permet d'améliorer un modèle neuronal de résumé automatique. De même, Takase et al. (2016) montrent que ce type d'ajout améliore la génération automatique de titres.

Dans ce chapitre, nous nous concentrons sur la réalisation de surface et proposons une nouvelle approche pour générer des phrases en français. Cette approche comprend deux grandes parties.

Premièrement, nous présentons une méthode pour créer automatiquement un ensemble de données parallèles de textes (en français) et de leurs représentations. Le corpus produit est disponible sur simple demande.

Deuxièmement, nous proposons une nouvelle méthode pour générer du texte à partir de représentations de sens. Cette approche diffère des travaux précédents en ce qu'elle repose sur une anonymisation poussée des données. L'intuition sous-jacente est que l'abstraction du contenu lexical permet de réduire la rareté des données (*data sparsity*), ce qui devrait faciliter l'apprentissage des structures linguistiques. Nous montrons que l'anonymisation étendue améliore en effet les performances. Pour aller plus loin dans l'évaluation du degré d'acquisition de la structure linguistique de notre modèle, nous fournissons une analyse de la réintroduction dans la phrase générée des mots fonctionnels qui sont absents de l'entrée.

II. 2. 2 Travaux liés

Corpus SR Différents jeux de données ont été mis au point pour permettre l'apprentissage de réalisateurs de surface.

Le travail effectué lors de l'*AMR SemEval generation shared task* (May et Priyadarshi, 2017) fournit un corpus parallèle où les représentations sémantiques constitutives, par lot, des entrées sont des AMRs (*Abstract Meaning Representation*, Banarescu et al. (2013)) : la tâche est de générer une phrase verbalisant chaque lot. Les jeux de données multilingues, dérivés MR-to-Text de l'UD (Universal Dependencies, Mille et al. (2018) [58]) offrent deux types d'entrée (pour deux tâches distinctes) : léger et profond (*shallow* et *deep*). Dans une entrée peu profonde, les nœuds de l'arbre de dépendance UD sont brouillés pour supprimer les informations d'ordre des mots et les mots sont remplacés par leurs lemmes. La tâche de génération consiste à ordonner et à infléchir les lemmes décrivant l'arborescence d'entrée. L'entrée profonde est plus proche d'un contexte applicatif et s'éloigne de la forme de la surface en supprimant des informations supplémentaires de l'arborescence UD et en remplaçant les étiquettes des liens entre noeuds syntaxiques par des étiquettes PropBank. Enfin, Novikova et al. (2017) introduisent un ensemble de données dans lequel les MR en entrée sont déduits d'éléments de dialogue.

58. http://universaldependencies.org

Ces ensembles de données ont été coûteux à construire car ils nécessitent un temps humain conséquent. Les ensembles de données AMR ont été construits en annotant manuellement des phrases avec des AMR, les ensembles de données SR ont été dérivés de banques d'arbres annotées à la main et l'extraction d'éléments de dialogue de Novikova et al. (2017) a été ajustée par le travail de *crowdsourcers*. De plus, à l'exception de la tâche de SR, ces jeux de données sont tous centrés sur l'anglais. Nous nous écartons de ce travail précédent en introduisant un nouvel ensemble de données pour le français, qui est automatiquement dérivé du texte en utilisant des analyseurs.

Modèles SR En utilisant les corpus parallèles MR-to-Text décrits précédemment, différents modèles SR ont été proposés. Nous nous concentrons ici sur les SR provenant de représentations sémantiques profondes (*AMRs and SR'18 deep track MRs*) car elles sont plus proches de notre proposition. Les premiers travaux sur la génération R-to-Text linéarisent le graphe d'entrée et utilisent diverses méthodes statistiques pour générer du texte (Flanigan et al., 2016; Song et al., 2017; Pourdamghani et al., 2016; Bohnet et al., 2010). De même, les premières approches neuronales linéarisent le graphe d'entrée et utilisent un modèle séquence à séquence. Konstas et al. (2017) obtiennent de bons résultats sur la tâche AMR en texte en utilisant le développement de données et l'anonymisation des entités tandis que Cao et Clark (2019) exploitent les informations de syntaxe pour améliorer les performances. Sur les données SR profondes, Elder et Hokamp (2018) utilisent le développement de données et un modèle factorisé seq-to-seq. Des modèles de graphes en séquences ont également été proposés en utilisant divers encodeurs et tests sur différents jeux de données (Marcheggiani et Perez-Beltrachini, 2018; Song et al., 2018; Beck et al., 2018; Koncel-Kedziorski et al., 2019; Veličković et al., 2018).

Notre approche est proche de celle de Elder et Hokamp (2018) en ce qu'ils utilisent un modèle sequence-2-sequence pour informer des nœuds capturant la structure de graphes représentant des phrases. Elle s'en différencie par le fait que nous utilisons une pleine anonymisation en entrée comme en sortie des entités lexicales complexes (hors mots fonctionnels).

II. 2. 3 Création d'un corpus parallèle MR/Texte

Créer un corpus parallèle de paires (R,T) avec T un texte et R une représentation de sens est une tâche chronophage qui demande en outre une expertise linguistique forte. Pour contourner cette difficulté, nous proposons une méthode pour la création automatique de représentations de sens à partir de texte. Cette méthode se décompose en deux grandes étapes. Premièrement, nous comparons les résultats de trois analyseurs syntaxiques en dépendances et ne conservons que les phrases pour lesquelles il existe un consensus fort. Deuxièmement, nous traitons les analyses pour en extraire une représentation, ce en croisant les informations obtenues et les descriptions issues du LVF (Les Verbes Français, Dubois et Dubois-Charlier (1997)).

II. 2. 3.1 Filtrage des analyses syntaxiques

Nous prenons en entrée le contenu de livres en accès libre diffusés par le site Gutenberg[59]). Nous avons téléchargé 2 835 livres en français (3 806 889 phrases) depuis ce site et appliqué un premier filtre qui exclut les phrases ne contenant pas au moins un nom et un verbe ainsi que celles présentant un parenthésage incorrect ou du vocabulaire étranger. Suite à ce premier filtrage, nous disposons d'un total de 1 700 000 phrases.

Afin de maximiser les chances d'une analyse syntaxique correcte, nous utilisons trois analyseurs et exploitons une procédure d'alignement sur leur sorties pour éliminer les phrases pour lesquelles la concordance entre les analyseurs est trop basse.

		Depuis	une	heure	,	Norbert	avait	consulté	sa	montre	.
token		1	2	3	4	5	6	7	8	9	10
POS	G	P	DET	NC	PONCT	NPP	V	VPP	DET	NC	PONCT
	S	ADP	DET	NOUN	PUNCT	PROPN	AUX	VERB	DET	VERB	PUNCT
	T	P	DET	NC	PONCT	NPP	V	VPP	DET	NC	PONCT
DEP	G DEP	mod	det	obj.p	ponct	suj	aux.tps	.	det	obj	ponct
	HEAD	7	3	1	7	7	7	_	9	7	7
	S DEPG	case	det	nmod	punct	nsubj	aux	root	nmod : poss	dobj	punct
	HEAD	3	3	7	7	7	7	0	9	7	7
	T DEP	mod	det	prep	ponct	suj	aux_tps	root	det	obj	ponct
	HEAD	7	3	1	3	7	7	0	9	7	10

TABLE II. 2. 1 – Exemple d'alignement (G : *grew*, S : *stanford*, T : *Talismane*) pour une phrase validée lors de la pré-analyse

Les trois analyseurs utilisés sont *Grew*[60] (Guillaume et al., 2012), *Talismane*[61] (Urieli, 2013) et le *Stanford Dependency Parser*[62] (Manning et al., 2014). Ces trois analyseurs ont chacun leur stratégie d'analyse et de tokenisation. Par exemple, *"entre autres"* est traité par Talismane et Stanford comme formant deux tokens distincts mais est analysé comme n'en formant qu'un seul par Grew (*entre_autres*). Les plus longues séquences (e.g., *entre_autres*) sont utilisées comme base pour l'alignement.

Pour les POS (*Part Of Speech* tags, parties du discours) et les relations de dépendances, l'alignement n'est pas réalisé par le biais d'une table de correspondance établie manuellement mais sur la base de statistiques. Pour ce faire, nous avons effectué un premier passage sur les données, et établi les correspondances inter-analyseurs en prenant Grew comme référence. Nous avons expérimentalement fixé les correspondances acceptées comme valides, celles dont la fréquence permet de conserver une grande partie des analyses en optimisant l'exclusion d'erreurs d'analyses patentes. La fréquence retenue est de 95% pour les POS tags et de 94% pour les relations de dépendance.

Grew est utilisé comme référence, et une correspondance POS tag/relation de dépendance de talismane et du stanford analyseur sera jugée compatible si elle y est sta-

59. http://www.gutenberg.org/robot/harvest?filetypes[]=txt&langs[]=fr
60. Version 0.48.0 http://grew.fr/
61. Version 5.1.2, http://redac.univ-tlse2.fr/applications/talismane.html
62. Version 2018-02-27, https://nlp.stanford.edu/software/lex-parser.shtml

tistiquement associée. L'ensemble des tokens, donc la phrase, ne sera conservée que si l'analyse des trois analyseurs est cohérente. Nous conservons donc les phrases qui ont vu la valeur inter-analyseurs validée (à la fois pour l'étiquetage des POS tags et des relations de dépendances). Par ailleurs, nous excluons les phrases qui contiennent plus de 70 tokens. La Table II. 2. 1 montre un exemple d'alignement. Les associations inter-analyseurs sont fournies en Annexe E. 5 et E. 6. Une phrase est validée parce que les étiquetages concordent : à ce stade, nous ne nous occupons pas des graphes ou de la cohérence de ceux-ci.

La Table II. 2. 1 montre un exemple d'alignement pour une phrase validée par l'alignement inter-analyseurs. La Table II. 2. 2 illustre les alignements ayant déterminé la phrase comme acceptable côté nature des mots (POS, *Part Of Speech*). La Table II. 2. 3 illustre les alignements ayant déterminé la phrase comme acceptable côté étiquetage des mots quant à leur rôle dans la phrase (DEP, dépendance).

POS Grew	tokens	Stanford	Talismane	pour Stanford	pour Talismane
P	1	ADP	ADP	P dans 94.48% des cas	P dans 96.75% des cas
DET	2 8	DET	DET	DET dans 92.29% des cas	DET dans 79.25% des cas
NC	3 9	NC VERB	NC	NOUN dans 93.61% des cas VERB parmi les 95% des associations les plus fréquentes.	NC dans 95,8% des cas
PONCT	4 10	PUNCT	PONCT	PUNCT dans 99,99% des cas	PONCT dans 99,99% des cas
NPP	5	PROPN	NPP	PROPN dans 90,65% des cas	PRO dans 81.82% des cas
V	6	AUX	V	VERB 72.1% des cas AUX 24,04%	V dans 97,11% des cas
VPP	7	VERB	VPP	VERB, 85,56% des cas	VPP 94,79%

TABLE II. 2. 2 – POS : correspondance pour la phrase II. 2. 1 de Grew vers Stanford et Talismane

DEP Grew	tokens	Stanford	Talismane	pour Stanford	pour Talismane
mod	1	case	mod	case 16.53 % des cas*	mod 83,35% des cas
det	2 8	det nmod : poss	det det	det 77.26% nmod : poss 16.37%	det, 96.99 %
obj.p	3	nmod	prep	nmod 75.07%	prep 94.17%
ponct	4 10	punct	ponct	punct 99.98%	ponct 100.0%
suj	5	nsubj	suj	nsubj 83,06%	suj 100%
aux.tps	6	aux	aux_tps	case 24,49% des cas*	aux_tps, 86.05%
.	7	root	root	root 17,09% des cas*	root 22,83 % des cas*
obj	9	dobj	obj	dobj 64.65%	obj dans 75,36% des cas

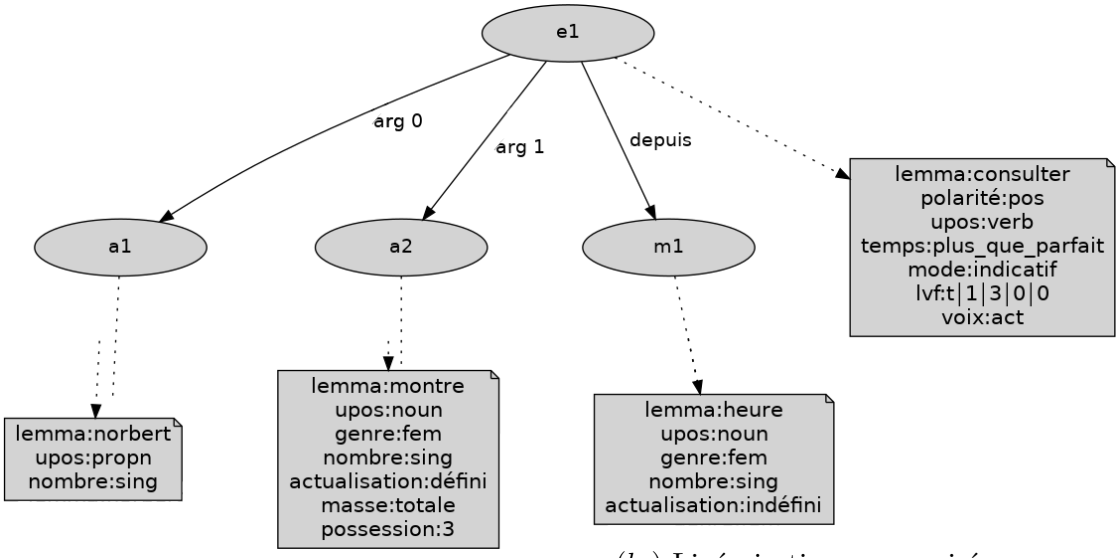
TABLE II. 2. 3 – DEP : correspondance pour la phrase II. 2. 1 de Grew vers Stanford et Talismane
*dans les 94% d'associations DEP les plus fréquentes

II. 2. 3.2 Création de représentations

Nous dérivons ensuite les représentations de l'arbre d'analyse en rapprochant les fonctions grammaticales (sujet, etc.) des fonctions sémantiques (arg0, arg1, etc.), en supprimant les mots fonctionnels (déterminants, auxiliaires, pronoms relatifs, complémentateurs, prépositions non sous-catégorisées), en lemmatisant les mots et en ne gardant que les caractéristiques qui ne peuvent pas être apprises, par exemple, le nombre et le genre d'un nom et le temps d'un verbe. L'information portée (genre, nombre, défini ou indéfini) par les déterminants est projetée sur le nom. Le déterminant *la*, par exemple, permet de déterminer que le nom commun auquel il est associé est féminin. Les contraintes d'accord (entre verbe et sujet et entre nom, déterminants et adjectifs) doivent être apprises par le modèle. Les traits apposés aux entrées permettent de guider la réalisation vers les bonnes structures linguistiques.

La Figure II. 2. 2 montre une phrase exemple (a) et ses représentations (utilisées en entrée) (b_1 avec les lemmes, b_2 sans les lemmes (entrée/sortie anonymisées)).

```
a1- Depuis une heure, Norbert avait consulté sa montre .
a2- Depuis une m1|noun|gender:fem|number:sing|.., a1|propn||number:sing|.. avait
    e1|verb||number:sing|passé|participe|person:3|.. sa a2|noun|gender:fem|number:sing|..
```



(b_1) Linéarisation :

```
( consulter|pos|verb|pqp|... arg_0
      norbert|propn|sg|... )
( consulter|pos|verb|pqp|... arg_1
      montre|noun|sg|fem|... )
depuis ( consulter|pos|verb|pqp|... <
      heure|noun|sg|fem|... )
```

(b_2) Linéarisation anonymisée :

```
( e1|pos|verb|pqp|... arg_0
      a1|propn|sg|... )
( e1|pos|verb|pqp|... arg_1
      a2|noun|sg|fem|... )
depuis ( e1|pos|verb|pqp|... <
      m1|noun|sg|fem|... )
```

FIGURE II. 2. 2 – Exemple de représentation du sens (MR) et linéarisations associées

Le mappage des relations syntaxiques sur les relations verbes/arguments utilise un lexique verbal, le LVF (Lexique des Verbes du français), qui permet d'identifier les éléments qui sont des arguments (plutôt que des modificateurs). Nous commençons par présenter ce lexique, nous explicitons ensuite comment nous l'exploitons pour déterminer les relations syntaxico-sémantiques.

Dérivation de relations syntaxico-sémantiques

Nous mappons les relations syntaxiques sur les relations sémantiques aux relations syntaxico-sémantiques décrites par le LVF (Lexique des Verbes du français). Celui-ci permet d'identifier les éléments qui sont des arguments (plutôt que des modificateurs) : sujet, objet, objet introduit par une préposition.

Nous filtrons les phrases, ne gardant que celles ayant au moins un verbe et un sujet, le verbe devant être connu du LVF.

Le filtrage a aussi été l'occasion d'exclure les phrases écrites dans d'autres langues que le français.

Le Lexique Verbal du Français (LVF)

Deux ressources décrivant les cadres syntaxico-sémantiques des verbes du français sont disponibles : Verbenet [63], qui prend modèle sur VerbNet [64] et le LVF (Les Verbes du Français [65].

Le LVF offre 12310 verbes différents et 25 610 entrées sémantico-syntaxiques, les deux tiers des verbes n'ayant qu'une seule description ; une stricte minorité - 290 verbes courants - ayant dix entrées ou plus [66]. Par entrée nous entendons usage syntaxico-sémantique (voir Table II. 2. 4).

VerbeNet compte lui 2813 verbes et 6644 entrées sémantiques. Seuls 44.2% des verbes n'ont qu'une seule entrée, 81.8% en ayant de 1 à 3. Par entrée nous entendons usage sémantique offrant une syntaxe (voir Table II. 2. 4).

VerbeNet est bien adapté pour des traitements exploitant directement des rôles thématiques ou impliquant des langues étrangères. Des ressources VerbNet ont ainsi été développées pour plusieurs langues (en particulier l'arabe, le basque et donc le français).

Le LVF fournit une description très fine des verbes du français, mais ce n'est pas une description sémantique.

La Table II. 2. 4 illustre leurs différences.

63. http://verbenet.inria.fr
64. https://uvi.colorado.edu/
65. http://rali.iro.umontreal.ca/rali/?q=fr/node/1237. Nous utilisons "Les verbes français" de Jean Dubois and Françoise Dubois-Charlier , version LVF+1.
66. http://rali.iro.umontreal.ca/LVF+1/documents/INFORMATIONS_LVF.pdf

Verbenet	Classe 11.2 E3f	
	Exemple	Max a glissé de sa chaise.
	Syntaxe	Theme V {de} Initial_Location
	Sémantique	motion(during(E), Theme) location(start(E), Theme, Initial_Location)
	glisser 12	choir de
	Exemple	Le livre glisse des mains.
	Syntaxe	type : transitif indirect, sujet : chose, compl-prep : de = N3b

TABLE II. 2. 4 – Exemple Verbenet/LVF

Dubois et Dubois-Charlier (1997) présentent leur socle linguistique :

> La **classification syntaxique des verbes français** repose sur l'hypothèse qu'il y a **adéquation** entre les schèmes syntaxiques de la langue et l'interprétation sémantique qu'en font les locuteurs de cette langue (...).

Le LVF explore méthodiquement les emplois distincts de chaque verbe. C'est un atout pour analyser les textes - littéraires du corpus Gutenberg.

La table II. 2. 5 résume les différences quantitatives entre les deux ressources et les compare au corpus.

	Nb. Verbes	% Corpus	Nb. Entrées/Verbe (avg/min/max)
LVF	12310	100%	2.08 / 1 / 61
VerbeNet	2813	58.3 %	2.36 / 0 / 25
Corpus initial	3542	95.2%*	2.72 / 1 /18

*171 verbes ont été perdus suite à échec d'appariement avec une entrée du LVF.

TABLE II. 2. 5 – Statistiques VerbeNet and LVF

Thésaurus de classes syntaxico-sémantiques, le LVF est particulièrement adapté à la jonction avec une analyse en dépendances. Chaque couple verbe/usage est doublé d'au moins un exemple et de son analyse. La vue - la classification - par verbe permet elle aussi un traitement intuitif des données, et facilite les vérifications manuelles durant la mise en place de l'expérience, sans développement d'outils *ad hoc*.

Le LVF fournit les schèmes de construction syntaxique. Cette description est codifiée sur trois à quatre caractères, cela facilite son usage informatique. Chaque description pour chaque verbe constitue une entrée.

L'information est codée successivement : usage verbal (ex :intransitif), nature du sujet (ex :humain), et quand le cas échoit, nature de l'objet (ex :humain), complément circonstanciel introduit par une préposition (codification de la préposition, ex : à, de), nature du circonstant complément (ex :locatif).

L'usage verbal est donc codé par le premier caractère : intransitif (A, exemple (13b), transitif indirect (N, (13c)), transitif (T, (13a)), pronominal (P, (13d)) [67].

La nature des sujets et objets directs est codifiée par un nombre de 1 à 9, celle du sujet pour tous les verbes, et celle de l'objet pour les transitifs et les pronominaux. Nous avons humain (1), animal (2), chose (3), complétive ou chose (4) [68].

Par exemple, l'entrée 01 du verbe croire (13) décrit un usage transitif (T), un sujet humain (1), et une complétive pour objet (4).

(13) a. *croire 01 [T1400] On croit que tu dis la vérité.*
b. *croire 06 [A10] Le chrétien croit en Dieu.*
c. *croire 09 [N1j] On croit en Pierre qui a de l'avenir.*
d. *croire 11 [P1400] On se croit aimé de tous.*

Les prépositions sont indiquées par une lettre (détails en Annexe E. 16).

Par exemple, l'entrée 1 du verbe avertir (14) décrit un usage transitif (T), un sujet humain (1), un objet humain (1) et un complément circonstanciel introduit par la préposition "de" (b).
L'entrée 9 du verbe croire (13c) décrit un usage transitif indirect (N), un complément circonstanciel introduit par la préposition "dans" (j) ici réalisé par "en".

(14) *avertir 01 [T11b0] On avertit Pierre de mon arrivée*

La nature des compléments circonstanciels est également indiquée par un chiffre [69]. Par exemple, l'entrée 4 du verbe conduire (15) décrit un usage transitif (T), un sujet chose (3), un objet humain (1) et un complément locatif (4).

(15) *conduire 04 [T3140] Ces empreintes conduisent l'enquêteur au voleur*

La codification de l'usage syntaxico-sémantique des verbes permet un usage algorithmique de l'information. Toutes les indications sont fournies par les auteurs [70] et par commodité fournies en Annexe E. 7.

Des relations de dépendances aux relations syntaxico-sémantiques

Tout verbe du LVF est détaillé en un certain nombre d'entrées, les propriétés extraites sont listées Table II. 2. 11. Nous prendrons ici pour support un des verbes de la Phrase 16, issue des Œuvres de Arthur Rimbaud, *Vers et proses*.

(16) Un rayon blanc, tombant du haut du ciel, anéantit cette comédie.

67. Le détail des codes utilisés pour la spécification des schèmes de construction syntaxique est donné dans l'Annexe E. 14.
68. Le détail des COD utilisés pour décrire la nature syntaxique ou sémantique des arguments et modifieurs verbaux est donné dans l'Annexe E. 15)
69. Détails en Annexe E. 17).
70. http://rali.iro.umontreal.ca/LVF+1/documents/INFORMATIONS_LVF.pdf, consulté le 27/03/20

Le verbe tomber compte 33 entrées[71], anéantir[72] en compte 6. Pour des raisons pratiques, nous nous intéresserons principalement au second, anéantir, dont les entrées sont synthétisées Table II. 2. 6.

entrée	exemple	code	décrytage
01	On anéantit la ville sous les bombes.	T1308	transitif direct, sujet : humain, objet : chose, circonstant : instrumental, moyen
02	L'orage anéantit la récolte.	T3300	transitif direct, sujet : humain, objet : chose
03	On anéantit une armée. On anéantit Pierre avec cette demande.	T1108	transitif direct, sujet : chose, objet : humain, circonstant : instrumental, moyen
04	Cette nouvelle anéantit ceux qui l'ont connu. Cette mort anéantit Pierre.	T3100	transitif direct, sujet : chose, objet : humain
05	On anéantit nos espoirs avec cette nouvelle.	T1308	transitif direct, sujet : humain, objet : chose, circonstant : instrumental, moyen
	Nos espoirs s'anéantissent.		pronominal, sujet : humain
06	On s'anéantit dans le désespoir. On s'anéantit devant la divinité.	P10j0	pronominal, sujet : humain, compl-prep : en, dans

Information commune aux entrées 01 à 06 : *auxiliaire : avoir (sauf si pronominal ou entrée en être)*

TABLE II. 2. 6 – Verbe exemple : anéantir, informations principales utilisées (LVF)

	mot	Un	rayon	blanc	,	tombant	du	haut	du	ciel	,	anéantit	cette	comédie	.
	token	1	2	3	4	5	6	7	8	9	10	11	12	13	14
lemme	grew	un	rayon	blanc	,	tomber	*du	haut	*du	ciel	,	anéantir	ce	comédie	.
	différentiel T						de		de						
	différentiel S														
POS	grew	D	N	A	PONCT	V	P+D	N	P+D	N	PONCT	V	D	N	PONCT
	différentiel T					VPR									
	différentiel S						ADP		ADP						
head	grew	2	11	2	2	2	5	6	7	8	2	0	13	11	11
	différentiel T				3						9				13
	différentiel S		0				7	5	9	7		2		11	2
DEP	grew	det	suj	mod	ponct	mod	mod	obj.p	dep	obj.p	ponct	root	det	obj	ponct
	différentiel T							prep		prep		root			
	différentiel S		root	amod		acl	case	nmod	case	nmod		acl		dobj	

TABLE II. 2. 7 – Analyse de la phrase support 16 (T pour Talismane, S pour Stanford).

Le LVF n'est pas exhaustif, mais il vise à l'être. Nous partons donc du principe qu'il existe une et une seule entrée correspondant à chacun des verbes de notre corpus.

En fonction des données des analyses, telles celles montrées Table II. 2. 7, nous cherchons à identifier quelle entrée du LVF est utilisée pour le verbe.

71. tomber : http://rali.iro.umontreal.ca/LVF+1/alphabetique/T/tomber.html (vu le 2/04/20)
72. anéantir : http://rali.iro.umontreal.ca/LVF+1/alphabetique/A/aneantir.html (vu 2/04/20)

`tomber` a eu pour correspondance sélectionnée **N3b** (transitif indirect, sujet :chose, complément prépositionnel :de), son sujet profond est `rayon`, il n'a pas de dépendance de_obj, mais le complément prépositionnel introduit par "du" a permis la sélection de ce cadre.

`anéantir`, toujours dans la phrase support 16, est paradoxalement plus complexe. Un objet direct (comédie) a permis d'identifier que son schème syntaxique est un schème de transitivité. Deux cadres LVF sont possibles : T3300 avec un sujet et un objet de type " chose" (Table II. 2. 6-02) et T3100 avec un sujet de type "chose" et un objet de type "humain" (Table II. 2. 6-04).

Nous avons déterminé conséquemment que :
$\{\text{rayon}\} = \{\text{chose}\} \cap \{\text{chose}\}$
$\{\text{comédie}\} = \{\text{chose} \cup \text{humain}\}$

Nous avons opté pour une gestion de l'incertitude, notant `Z` deux arbres indécidables. Ici, `comédie` est un objet soit `chose` (code 3), soit `humain` (code 1). Nous avons donc retenu comme classe "T3Z00" pour le verbe "anéantir".

Les analyses des parseurs sont un outil pour se rapprocher du LVF, mais les analyses ne correspondent pas : il peut y avoir un complément d'objet introduit par une préposition pour les parseurs, et ne pas en avoir pour le thésaurus qui est notre référent.

Voici ce que nous avons fait :

- si le verbe est lié à une dépendance "s=refl" ou "aff", il est pronominal,
- si le verbe est modifié par une préposition, il est à modifier lors de son usage par un complément prépositionnel introduit par cette préposition,
- un objet direct marque la place d'un transitif,
- nous recherchons les dépendances longue distance : dans la Phrase 17, nous détectons `balcons` comme circonstant (locatif (où on est)) pour le verbe `tourbillonnaient`.

(17) Mille flammes s' échappaient par les balcons où tourbillonnaient des flots d' étincelles.

Les circonstants étant potentiellement élidés, ils sont considérés comme facultatifs.

Les arbres potentiels sont élagués en fonction des informations disponibles. Par exemple, si le verbe est pronominal, les autres possibilités sont éliminées. Nous élaguons les cadres jusqu'à ne plus pouvoir réduire. Si un complément circonstanciel est attendu et trouvé pour un cadre donné, ce cadre aura priorité (sera retenu). Un dépendant introduit par une préposition sera classifié comme modifieur si aucun cadre ne le demande.

Le sujet sera classifié arg_0, l'objet arg_1, le complément introduit par une préposition arg_2 et le circonstant arg_3.

Vous pourrez trouver en Annexe E. 8 les résultats du pré-traitement des Phrases 16 et 17.

Les verbes de notre corpus peuvent être trouvés dans en Annexe E. 2, catégorisés par cadre syntaxico-sémantique en Annexe E. 4.

II. 2. 3.3 Corpus final

Au terme des traitements décrits ci-dessus (filtrages des phrases mal formées ou contenant d'autres langues que le français, identification des phrases dont l'analyse syntaxique obtient un bon score inter-analyseur, identification des arguments syntaxico-sémantiques sur la base du LVF), nous obtenons un corpus parallèle entrée syntaxico-sémantiques / phrase de 490 456 phrases. Les tables II. 2. 8 et II. 2. 9 résument le contenu quantitatif de ce corpus.

	entrées			sorties		
	train	dev	test	train	dev	test
Nb. instances	392,486	48,944	49,026	392,486	48,944	49,026
Nb. mots	61,547	24,599	24,609	66,526	25,852	25,891

TABLE II. 2. 8 – Corpus créé

entrées			sorties		
avg	min	max	avg	min	max
41.8	10	510	13.55	3	69

TABLE II. 2. 9 – Taille des entrées et des sorties (nombre de *tokens*)

II. 2. 4 Le modèle

Notre approche étend un modèle standard séquence à séquence avec (i) une anonymisation complète des mots pleins et (ii) une représentation factorisée des tokens qui capture les informations linguistiques et structurelles associées à chaque nœud dans le graphe d'entrée.

L'anonymisation L'anonymisation (aussi souvent nommé délexicalisation) a fréquemment été utilisée dans la génération neuronale en langue naturelle pour faciliter le traitement des mots rares ou inconnus (Wen et al., 2015; Dušek et Jurcicek, 2015; Chen et al., 2018). Les mots rares sont remplacés par des marqueurs, tant dans les données en entrée ou représentations sémantiques que dans la phrase, le texte attendu en sortie. Les modèles sont entraînés sur des données anonymisées. Après génération, une étape de post-traitement assure la relexicalisation du texte généré, en reprenant la valeur originelle des marqueurs.

Dans ces approches, l'anonymisation est limitée aux items rares (entités nommées) et utilise de simples identifieurs (les marqueurs). Dans le cadre de l'expérience ici décrite, nous appliquons l'anonymisation à tous les mots porteurs d'un contenu lexical, adverbes exceptés. Nous dérivons les représentations sémantiques des arbres analysés (cf. Section II. 2. 3), nous remplaçons les mots pleins (noms, verbes, adjectifs) par un

marqueur et nous gardons la trace des correspondances marqueur/contenu lexical. Nous utilisons ensuite cette information lors de l'étape de post-traitement pour effectuer la relexicalisation.

La Table II. 2. 10 montre l'impact important de l'anonymisation sur la taille du vocabulaire.

	entrée			sortie		
	train	dev	test	train	dev	test
Nb. mots	61,547	24,599	24,609	66,526	25,852	25,891
Nb. mots (anonymisation)	2,673	1,209	1,221	9,580	2,979	3,014

TABLE II. 2. 10 – Impact de l'anonymisation sur la taille du vocabulaire

Représentation factorisée des tokens. Nous utilisons le modèle OpenNMT (Klein et al., 2017) séquence vers séquence avec attention. Chaque *token* de l'entrée est représenté par la concaténation de 20 embeddings, où chacun des *embeddings* représente une propriété particulière (*part-of-speech* (nom, verbe...), genre, nombre, temps...,).

Nous délexicalisons les évènements verbaux, les arguments et modifieurs à l'aide de la phrase nous ayant permis de générer chaque exemple. Ainsi pour une entrée composée d'élements délexicalisés mis en relation, la sortie reprend les étiquettes délexicalisées, ne reste en texte clair que les mots fonctionnels (déterminants, auxiliaires, pronoms relatifs, complémentateurs, prépositions non sous-catégorisées par le LVF).

Les propriétés présentées Table II. 2. 11 sont utilisées - factorisées - pour les entrées. Chaque propriété est à blanc quand elle est inadaptée (ex : le temps et le mode seront à blanc pour un nom commun).

Propriété	Valeurs possibles	Exemple	Explication
élision	élidable, non élidable, spécifique-être, spécifique-aller	*Le perroquet est un oiseau*	oiseau élidable, perroquet non, être a une élision spécifique.
polarité	négative, positive	*Je ne mange pas*	manger, polarité négative.
POS	verbe, pronom, nom propre, nom, adjectif, adverbe, x, conjonction de coordination	*Il mangera du pain*	pain est un nom
mode	indicatif, conditionnel, participe, subjonctif		manger est à l'indicatif

Suite page suivante

TABLE II. 2. 11 – *Suite de la page précédente*

Propriété	Valeurs possibles	Exemple	Explication
temps	présent, passé-composé, na, passé-simple, imparfait, plus-que-parfait, passé, futur, futur-antérieur, passé-antérieur		manger est au futur
voix	passif, actif		manger est à la voix active
auxiliaire	être, avoir, être ou avoir, avoir-sauf-si-pronominal-ou-entrée-en-être		auxiliaire par défaut de manger : avoir-sauf-si-pronominal-ou-entrée-en-être
position	avant, après	c'est un joli garçon.	joli est caractéristiquement avant ce qu'il modifie.
nombre	singulier, pluriel	les chiens mangent.	chien est au pluriel
genre	masculin, féminin		chien est masculin
personne	0, 1, 2, 3		chien est un mot à la 3ème personne
actualisation	défini, indéfini	les chiens mangent un chien mange	chien est défini chien est indéfini
possession	0, 1, 2, 3	sa maison est magnifique	maison est possédée par quelqu'un (à la 3ème personne)
masse	totale, partielle, nulle	un arbre masque la forêt les arbres masque la forêt aucun arbre masque la forêt	arbre a une masse partielle arbre a une masse totale arbre a une masse nulle
type (annexe E. 14)	A, N, T, P, Z	On croit que tu dis la vérité.	croire est transitif (T)
sujet (annexe E. 15)	1, . . . , 9, Z		croire a un sujet humain (1)
objet (annexe E. 15)	1, . . . , 9, Z		croire a pour objet une complétive (4)

Suite page suivante

TABLE II. 2. 11 – *Suite de la page précédente*

Propriété	Valeurs possibles	Exemple	Explication
préposition (annexe E. 16)	a, b, c, d, e, g, i, j, k, l, m, n, q, Z	On avertit Pierre de mon arrivée	avertir attend un complément introduit par "de"
complément (annexe E. 17)	1, . . . , 8, Z	il montre l'oiseau par un geste	montrer a un complément instrumental/-moyen (4).
closure	question, exclamation	Tu manges ?	La propriété *closure* de manger prend "question"

TABLE II. 2. 11 – Propriétés encodées dans les facteurs en entrée

II. 2. 5 Paramètres expérimentaux

Métriques d'évaluation Suivant les pratiques courantes en Génération de Langue Naturelle, nous avons évalué nos expériences à l'aide de BLEU-4 [73]. Pour mieux comprendre dans quelle mesure le modèle est capable d'apprendre les structures linguistiques, nous évaluons également le rappel sur le vocabulaire lexicalisé (C-R, content words recall) et le F1-score sur les mots fonctionnels (FW-F1, function words F1). Nous définissons le vocabulaire lexicalisé comme étant les lemmes présents dans l'entrée. Les mots fonctionnels sont les autres mots présents dans la phrase à générer. Les exemples suivants illustrent le fonctionnement de ces mesures sur un exemple incorrect et un exemple correct.

Exemple de phrase incorrecte : La phrase (18REF) (délexicalisée automatiquement(18REFDélex) est une phrase attendue. (18GENDélex) est la phrase qui a été prédite. (18GEN) sa version relexicalisée. Nous avons mis en gras les mots fonctionnels.

(18) **REF :** On **ne** parlait **que de** réformes , **d'** économies : **de toutes** parts , on entrevoyait **un** heureux avenir.
REFDélex : argevent_5 **ne** event_1 **que de** argevent_4 , **d'** argevent_6 : **de toutes** argevent_2 , argevent_3 event_2 **un** mod_100 argevent_1 .
GEN : On entrevoyait **de toutes** parts **un** avenir heureux , on **ne** parlait **que de** réformes.
GENDélex : argevent_3 event_2 **de toutes** argevent_2 **un** argevent_1 mod_-100 , argevent_5 **ne** event_1 **que de** argevent_4 .

Exemple de phrase correcte : La phrase (19REF) (délexicalisée automatiquement (19REFDélex)) est une phrase attendue. (19GENDélex) est la phrase qui a été prédite.

73. Nous utilisons sacrebleu, Post (2018)

(19GEN) sa version relexicalisée. Nous avons mis en gras les mots fonctionnels générés ou attendus (les prépositions ou adverbes venant guider une modification n'en font pas partie (ex : `fâcher contre sophie` est une représentation fournie de manière brute en entrée) pour la phrase 18, `contre` est considérée comme une forme d'entité de la représentation sémantique). Les mots qui ne sont pas en gras ne sont pas considérés comme du vocabulaire fonctionnel.

(19) **REF :** Elle **se** fâcha contre Sophie , **parce_qu'** elle **avait** répandu **quelques** propos sur **son** compte.
REFDélex : argevent_3 **se** event_1 contre argevent_4 , **parce_qu'** argevent_1 **avait** event_2 **quelques** argevent_2 sur **son** mod_100 .
GEN : Elle **se** fâcha contre Sophie ; elle **avait** répandu **des** propos sur **son** compte.
GENDélex : argevent_3 **se** event_1 contre argevent_4 ; argevent_1 **avait** event_2 **des** argevent_2 sur **son** mod_100 .

Le rappel sur le vocabulaire lexicalisé (C-R) et le F1-score sur les mots fonctionnels (FW-F1) ont été calculé en macro. Nous avons recherché tous les mots attendus sur le corpus de test, et tous les mots obtenus. La Table II. 2. 12 et la Table II. 2. 13 montrent chacune un exemple.

	Phrase 1						Phrase 2					
mots fonctionnels	ne	que	de	d'	toutes	un	se	avait	son	parce qu'	quelques	des
attendus	1	1	2	1	1	1	1	1	1	1	1	
obtenus	1	1	2	0	1	1	1	1	1	0	0	1
précision	0.89											
rappel	0.67											
F1-score	0.76											

TABLE II. 2. 12 – F1-score sur les mots fonctionnels (FW-F1)

vocabulaire (`voc`)	event		argevent						mod
étiquette (`eti`)	1	2	1	2	3	4	5	6	100
(`voc_eti`) **attendus**	2	2	2	2	2	2	1	1	2
(`voc_eti`) **obtenus**	2	2	2	2	2	2	1	0	2
rappel	0.89								

ex :*argevent_3 event_2* ***de toutes*** *argevent_2* ***un*** *argevent_1 mod_100 .,On entrevoyait de toutes parts un avenir heureux.* : ici dans l'ordre d'apparition, nous avons un argument verbal numéroté 3, un évènement, étiqueté 2, un argument verbal numéroté 2, puis un autre numéroté 1 et un modifieur, numéroté 100.

TABLE II. 2. 13 – Rappel sur le vocabulaire lexicalisé (C-R) : exemple

Modèles. Le modèle de base (baseline) est un modèle séquence vers séquence (avec 20 embeddings factorisés par *token*) avec attention, sur données originelles, non anonymisées (BL dans la Table II. 2. 14). Nous comparons cette base avec le même modèle appris sur des données anonymisées (Anonymisé), mais aussi avec deux modèles appris sur un corpus comprenant à la fois les données originelles et les données anonymisées. L'intuition derrière l'utilisation à la fois de données anonymisées et non anonymisées est de voir si la combinaison des deux sources d'informations peut être utile. Le premier modèle (Dual (Test : Non Anon.)) est testé sur des données lexicalisées. L'adjonction de données anonymisées permet-elle d'améliorer la génération à partir de représentations sémantiques non anonymisées ? Le second modèle (Dual (Test : Anon.)) est testé sur des données anonymisées. L'adjonction de données non anonymisées aide-t-elle à améliorer la génération à partir de représentations sémantiques anonymisées ?

Modèle	BLEU-4	C-R	FW-F1
BL	64.35	0.59	0.906
Dual (Test : Non Anon.)	66.55 (+ 2.20)	0.62 (+ 0.30)	0.910 (+ 0.004)
Anonymisé	66.87 (+ 2.52)	0.87 (+0.28)	0.933 (+ 0.027)
Dual (Test : Anon.)	68.31 (+ 3.96)	0.87 (+0.28)	0.937 (+ 0.031)

TABLE II. 2. 14 – Résultats (les nombres entre parenthèses indiquent le delta avec BL (*Base Line*))

II. 2. 6 Résultats

Les résultats sont présentés dans le tableau II. 2. 14. Les scores BLEU montrent que l'entraînement sur des structures anonymisées donne de meilleurs résultats (+2,52 BLEU). L'utilisation à la fois de données lexicalisées et anonymisées pour l'apprentissage l'améliore encore, ce qui n'est pas surprenant étant donné que la taille des données d'entraînement a doublé. Il est intéressant de noter que le delta est légèrement meilleur lors des tests sur des données anonymisées (+3,96 contre +2,2), ce qui suggère que l'ajout d'informations lexicales aux données d'apprentissage est plus bénéfique pour la génération anonymisée que l'inverse.

Une tendance similaire peut être observée concernant le traitement des mots fonctionnels, bien que l'augmentation soit beaucoup moins importante.

Analyse qualitative Sur l'ensemble du corpus de test (49 026 phrases), une analyse automatique des résultats montre que (i) 34,76 % des phrases générées sont identiques à la phrase de référence, (ii) la relexicalisation échoue pour certains mots dans 34,07 % des phrases. et (iii) au total, 94,8 % des mots du contenu en entrée sont présents dans la sortie.

Nous avons également examiné manuellement 50 phrases générées choisies au ha-

sard [74] 34% de celles-ci sont exactement les mêmes que leur référence. 58 % sont grammaticalement et sémantiquement correctes, sachant que 78 % sont grammaticalement correctes. Tous les verbes se sont avérés être dans la forme attendue (accord et temps). Nous n'avons détecté qu'une seule erreur d'accord entre un nom et son déterminant. Une analyse détaillée des erreurs trouvées peut être trouvée elle aussi en Annexe E. 1.

exacte	(a)	Au milieu de la réprobation générale , Anicet fils ne perd pas le sentiment de sa dignité .
	(b)	Au milieu de la réprobation générale , Anicet fils ne perd pas le sentiment de sa dignité .
	(a)	Depuis longtemps l' empereur français pressait les Américains de prendre les armes .
	(b)	Depuis longtemps l' empereur français pressait les Américains de prendre les armes .
correcte	(a)	Sans doute ces animaux sont dressés à revenir au logis .
	(b)	Ces animaux sont sans doute dressés à revenir au logis .
	(a)	Le sacristain était un gros gars qui avait peut-être servi la messe aux chartreux dans son enfance , et qui désormais était dépositaire des clefs du couvent .
	(b)	Le sacristain était un gars gros qui avait peut-être servi la messe aux chartreux dans son enfance , et un gars était dépositaire des clefs du couvent .
mauvaise	(a)	Le porteur s' est dérobé , et la malle a pris un bain d' un quart d' heure .
	(b)	Le porteur s' est dérobé , et la malle a pris un bain du quart de heure .
	(a)	Le soulèvement , la séparation même d' une province ne font pas tout le sort d' un empire .
	(b)	Le tout soulèvement , la séparation d' une province ne font pas le sort même d' un empire .
échec	(a)	Ces rudes débuts ne découragèrent personne ; de suite on se mit au travail .
	(b)	on se mit au travail ; ces rudes débuts ne personne découragèrent pas .
	(a)	Le capitaine m' a expliqué que le mot hippopotame signifie " cheval de fleuve " .
	(b)	Le capitaine a expliqué que le mot signifie cheval de fleuve , le capitaine hippopotame signifie cheval de cheval .

TABLE II. 2. 15 – Phrases exemples : (a) version attendue, (b) version générée.

II. 2. 7 Conclusion

Le travail décrit dans ce chapitre a (i) permis l'introduction d'un nouvel ensemble de données (représentation sémantique, phrase) pour le français et (ii) montré qu'une anonymisation étendue aux verbes, noms et adjectifs contribue à améliorer la réalisation de la surface. La construction automatique de données parallèles à l'aide de l'analyse fournit en outre une description linguistique détaillée des phrases cibles à générer qui pourrait naturellement servir de support au développement d'un modèle de génération sous contraintes syntaxiques similaire à celui présenté dans le chapitre précédent.

Ces travaux qui permettent de générer du texte à partir de représentations sémantiques sont proches de la tâche partagée organisée dans le cadre de SemEval sur la

74. Les phrases sélectionnées sont données en annexes ainsi que des statistiques sur leur longueur.

génération de phrases à partir de représentations sémantiques abstraites (AMR) [75] ainsi que de la tâche de réalisation de surface multilingue où les entrées sont des arbres de dépendances profonds [76] (May et Priyadarshi, 2017; Mille et al., 2019). Une différence clé concerne la création du corpus d'apprentissage. Alors que ces deux tâches dérivent les données d'apprentissage à partir d'un corpus arboré construit manuellement et donc coûteux en temps et en expertise, nous n'utilisons pas de corpus arboré, mais avons choisi une analyse inter-parseurs qui certes cause des pertes, mais permet d'écarter les phrases aux analyses les plus délicates : de détecter des contradictions souvent synonyme d'erreurs. Ensuite, nous avons fait le choix d'utiliser le LVF pour référence. Cela nous a permis de bénéficier d'un support solide pour l'appariement entre les relations syntaxiques produites par les analyseurs syntaxiques et les relations sémantiques à inclure dans la représentation sémantique. Nous créons ainsi, de façon entièrement automatique, un corpus d'apprentissage "silver" de bonne qualité. Les résultats (phrases générées) montrent qu'il est possible, à partir de ces données imparfaites, d'apprendre un modèle de génération de bon niveau (génération de phrases généralement grammaticales et reflétant bien l'entrée sémantique).

Grâce aux informations syntaxiques fournies par l'analyse syntaxique, nous pouvons adapter le travail courant pour le joindre au travail précédent et générer sous contrainte syntaxique, en adaptant les régles de détection de contraintes syntaxiques développées pour l'anglais au français. Ces informations permettent également la mise au point de techniques de filtrage (ex : vérification automatique des accords, des temps attendus). Des tests en ce sens ont été amorcés, et nous notons comme travail qu'il serait possible d'apprendre la position relative des objets dans des traits prévus à cet effet. Cela permettrait de vérifier plus aisément la cohérence potentielle, les accords des auxiliaires etc. Les traits proposés ont été fournis en Annexe E. 9. Les incohérences inter-parseurs permettent de détecter des phrases mal analysées. Nous pensons que plus les informations au sein d'un texte sont cohérentes, plus il est probable qu'un humain la trouve correcte. Augmenter le support de la recherche d'incohérences est une piste utile.

Il serait aussi intéressant de renforcer l'usage du LVF : utiliser les cadres pour générer les variantes, faire tendre l'entrée vers une représentation sémantique pure (par exemple avec un travail sur les pronominaux factitifs). A côté du LVF, nous avons aussi le Dictionnaire Électronique des Mots, fournissant des informations sur les sens, les domaines. Les données pourraient être croisées, utilisées aussi pour prédire la nature des sujets, sachant que la marge d'erreur resterait grande : la langue est libre, et *une maison peut manger de la place sur un terrain.*

75. *SemEval-2017 Task 9* : http://alt.qcri.org/semeval2017/task9/, consulté le 07/04/2020.
76. *The Second Multilingual Surface Realisation Shared Task (SR'19)* : https://www.aclweb.org/anthology/D19-6301/, consulté le 07/04/2020.

Conclusion générale

Résumé

Cette thèse, démarrée le 1 février 2017, porte sur la génération de phrases pouvant servir de base à la création d'exercices de grammaire. Son cadre implique l'exploration de méthodes neuronales.

Nous avons exploré deux méthodes pour la génération de phrases utilisables pour créer des exercices de grammaire.

Dans un premier temps, nous avons généré des phrases en anglais sous contrainte syntaxique à partir des données WebNLG (Gardent et al., 2017).

Nous avons proposé quatre modèles différents, les avons combinés pour maximiser l'exploitation du corpus. Un même signifié se voit traité sous l'angle de son RDF (représentation logique), par un texte, une phrase à étendre par un triplet RDF ou au contraire à raccourcir (en supprimant l'information portée par un triplet RDF).

Pour ce travail, nous avons généré un grand nombre de paraphases syntaxiques de bonne qualité. La diversité des générations pour un même signifié est équivalente ou presque à celle du corpus d'apprentissage (similarité de 0.61 pour 0.62 sur notre combinaison de modèle). Les modèles combinés permettent de générer en moyenne 13.25 phrases différentes par sens. D'après l'étude qualitative menée, 6.3 phrases sont différentes syntaxiquement. Le meilleur modèle à ce niveau n'en produit que 3.98. Ce dernier est une des deux lignes de base où l'absence de contrôle syntaxique limite la portée des métriques automatiques (BLEU-4 de 6.21 contre un BLEU-4 de 62.87 pour les sorties des modèles combinés).

Dans un second temps, nous avons mis au point un modèle permettant de générer des phrases, en français, à partir de représentations sémantiques. Nous avons proposé une méthode de création des données d'apprentissage (représentation de sens/phrases) et étudié l'impact d'une délexicalisation étendue (hors mots fonctionnels). Nous nous sommes aussi demandé si la réalisation d'un apprentissage sur le couplage des données délexicalisées et non délexicalisées améliorait les résultats.

Alignant les analyses de trois parseurs, nous obtenons un corpus de 490 456 phrases comptant moins de 70 caractères. Chacune de ses phrases est traitée, les mots fonctionnels, modifieurs repérés, les évènements verbaux associés à des informations issues du

thésaurus LVF tout comme leurs arguments.

Générer d'après les représentations construites automatiquement nous permet de produire des phrases correctes et cohérentes dans 92% des cas (analyse qualitative). La réalisation d'un apprentissage sur le couplage des données délexicalisées et non délexicalisées offre un delta de +3.96 sur le BLEU du modèle de base (apprentissage sur représentations non délexicalisées), atteignant un BLEU-4 de 68.31. Les mots fonctionnels espérés sont présents à 93.7%.

Ces deux travaux ont chacun donné lieu à publication, le premier à EMNLP (Colin et Gardent, 2018), le second à INLG (Colin et Gardent, 2019).

Perspectives

Exercices de grammaire

Combiner les deux contributions est tout à fait envisageable. Deux approches sont possibles qui consistent à approcher :

- la première de la deuxième : traduire WebNLG en français (et reproduire la première expérience, mais sur ce "nouveau" corpus),
- la deuxième de la seconde, ce qui implique en particulier d'enrichir le corpus d'apprentissage des leviers syntaxiques présents dans les phrases.

Un test *in situ* permettrait l'évaluation de la méthode. La chaîne complète (création opérationnelle des exercices, intégration dans un exerciseur, et évaluation par les élèves et enseignants) serait relativement optimale pour démontrer finement les limites et opportunités des travaux similaires aux nôtres.

Le multilinguisme est possible, WebNLG par exemple et d'ores et déjà porté vers d'autres langues (exemple, en russe (Shimorina et al., 2019), avec traduction automatique suivie de *post editing*). La deuxième contribution peut aussi se voir portée sur d'autres langues. L'alignement et l'anonymisation sont indépendants de la langue. Une ressource locale peut permettre d'adapter les rapprochements syntaxico-sémantiques (et, optionnellement, de réécrire les représentations de sens). Les contraintes syntaxiques se doivent par contre d'être définies langue par langue.

Génération à partir de données hybrides Texte/RDF

La première contribution propose de nouvelles tâches : une expansion ou réduction de texte à partir de données hybrides (Texte/RDF), l'usage de modèles combinés.

L'expansion ouvre la possibilité d'enrichir du texte à partir de données du Web. Les applications possibles sont nombreuses : robot-journalisme, aide à la rédaction, exercices de grammaire… : quelles sont les performances des méthodes de génération actuelles sur ce type de données ?

La réduction est une forme de compression contrôlée. Elle pourrait être exploitée pour résumer, créer de nouveaux corpus d'apprentissage. Comment les méthodes pour la compression/le résumé automatique fonctionnent-elles sur ce type de données ? des adaptations sont-elles nécessaires ?

La combinaison de modèles permet de générer de nombreuses paraphrases. On peut aussi voir ce corpus comme un outil d'isomorphisme qui permettrait de choisir des espaces de travail (pour de la traduction par exemple : passer de formes impliquant des dépendances longues, des passifs, à des formes plus traduisibles).
Les paraphrases sont classifiables (présence des entités, des mots fonctionnels attendus, des contraintes présentes, du BLEU-4, de soucis d'accords (si nous ajoutons des traits en sortie, voir la proposition en Annexe E. 9)). On pourrait imaginer utiliser ces paraphrases et leur qualité pour une aide à l'écriture dans un mode de recherche participatif : une extension d'openoffice - par exemple - permettrait de proposer des n meilleures réécritures syntaxiques automatiquement pré-sélectionnées après génération (avec des *pouce vert et rouge* pour offrir des retours).

Nouvelles méthodes neuronales

Continuer l'exploration des méthodes neuronales est un dernier type de perspective. Le *transformer* (voir p. 58) est extrêmement prometteur (Li et al., 2019). Il serait intéressant de ré-évaluer notre travail avec cette technologie.
Réaliser/utiliser un pré-apprentissage affecterait nos résultats : nous pensons là à l'usage de vecteurs de mots pré-entraînés, tels que ceux de BERT (Devlin et al., 2018).
Enfin, cette thèse a avancé en même temps que les savoirs, et un retour sur les données comme sur simplement les hyperparamètres des *séquence-vers-séquence* ne serait pas sans impact. Il s'agit aussi d'une perspective : dans quelle mesure les nouvelles méthodes de génération à base de Transformer telles que celles proposées dans (Chan et al., 2019; Lawrence et al., 2019; Dong et al., 2019) permettraient-elles d'améliorer les résultats ?

Annexes

Annexe A

Notions d'anatomie

Éléments d'anatomie

Nous donnerons quelques rappels et informations touchant à la biologie, utilisons ici un travail précédemment accompli (Colin, 2015). L'idée de cette partie est de fournir à comprendre globalement ce dont l'intelligence artificielle s'inspire.

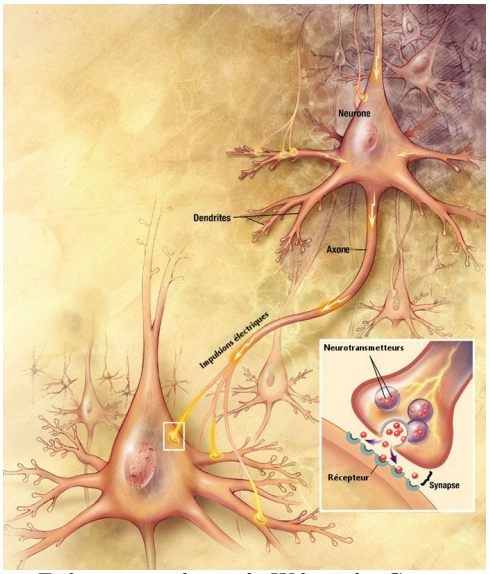
Au niveau micro

Une synapse est une zone de contact (on parle aussi de fente synaptique) entre deux neurones ou encore entre un neurone et une cellule. L'humain a moins de 20% de ses neurones dans le cortex d'après Azevedo et al. (2009).

Les neurones sont un élément de base du système nerveux. Ces éléments de base ne sont pas seuls. Des cellules gliales, estimées plus nombreuses par bien de vulgarisateurs, sont à peu près au même nombre, et donc... ne les entourent pas. Elles co-constituent le cerveau.

Azevedo et al. (2009) les ont évalué à plus de 8 milliards chaque. Ces cellules, variées, intéressent particulièrement les scientifiques, tant en biologie qu'en intelligence artificielle.

Les cellules gliales, sur lesquelles nous disserterons peu, et les neurones sont interdépendants. Parmi elles, les astrocytes, étiquetées parfois cellules en étoile du fait de leur forme.



Fichier original issu de Wikimedia Commons

FIGURE A. 1 – Synapse entre neurones

On sait désormais que leur action est régulée par l'activité synaptique et sa potentialisation de long-terme (*long-term potentiation*), c'est-à-dire le renforcement de la dite activité (Bernardinelli et al., 2014). Oberheim et al. (2009) ont eux établi la spécificité des astrocytes humains qui peuvent réunir jusqu'à deux millions de synapses dans un même domaine d'activité, propageant de façon fulgurante, au regard de celles d'autres animaux, des vagues d'ions calcium [77]. Les astrocytes se structurent en réseau et ont un impact indéniable sur les fonctions cognitives, l'activité neuronale.

Un autre élément important à apporter au niveau micro : la myéline. Constitutive des cellules de Schwann (qui sont comme les astrocytes des cellules gliales), elle leur permet de gainer les axones [78] et conditionnent, en l'accélérant, le flux. Un axone gainé de myéline est donc une connexion aussi stable que renforcée, moins plastique.

Les axones gèrent les efférences, ce qui part de la partie de référence (les neurones) vers les synapses. Les dendrites représentent l'autre versant, gérant donc les afférences. Les dendrites sont le prolongement post-synaptique des neurones. Les termes efférence et afférance sont des repères complexes dans la littérature neuroscientifique. Ils sont parfois utilisés de manière "consacrée". Par exemple, la "voie efférente", voie motrice (qui part du système nerveux central vers les muscles), a tendance à être étiquetée ainsi quelque soit le point de référence de l'écrit.

Les neurones sont, éléments essentiels parmi d'autres, constitutifs du système nerveux. Ce système biologique propre au règne animal assure gestion et transmission de l'information. Les nerfs et ganglions forment le système nerveux périphérique, le cerveau et la moelle épinière le système nerveux central. Les neurosciences sont la branche de la biologie consacrée à l'étude de cet ensemble.

77. Ion calcium (Ca++) : lié à la transmission du potentiel d'action, ce qui active ou inhibe les neurones
78. Axones : "liens" synaptiques, prolongement pré-synaptique des neurones

Annexe B

Réseaux de neurones

Algorithme de correction d'erreur

ALGORITHME apprentissage

tableau : couches[] ← couches_existantes de dimension entière $nbCouches$
//x_1, x_2 sont les valeurs entrantes, θ_1, θ_2 les valeurs désirées
réel : x_1, x_2, θ_1, θ_2

DEBUT

tableau : entrées[] ← [x_1, x_2], sorties[] ← [θ_1, θ_2]
réel : pas ← 0.5
//utilisation du perceptron en mode avant (affecte valeurs des neurones).
POUR i ALLANT_DE 0 A nbCouches-1
  entrées ← obtenir_sorties(entrées)
FIN_POUR

//les synapses sont portées par les neurones accueillant la valeur issue du
//neurone du rang précédent. Rien n'est à modifier pour couches[0], la boucle
//voit donc i s'arrêter à 1.
POUR i ALLANT_DE nbCouches-1 A 1 AVEC PAS DE -1
  entier : nb_neurones_couche_référente ← nb_neurones(couches[$i-1$])
  tableau : gradient[] ← [$0 *$ nb_neurones_couche_référente]
  entier : nb_neurones ← nb_neurones(couches[i])

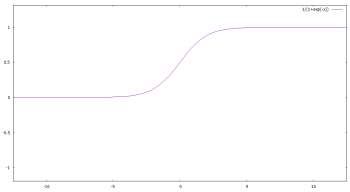
  POUR j ALLANT_DE 0 A nb_neurones
    réel : potentiel ← 0
    //le nombre de poids correspond au nombre de synapses entrantes
    POUR k ALLANT_DE 0 A nb_poids(neurone[j])
      potentiel ← potentiel + poids[k] $*$ valeur(obtenir_neurone(couche[$i-1$], k))

```
            FIN_POUR
            //sorties contient les sorties désirées au premier tour de boucle,
            // le gradient ensuite.
            réel : erreur
            SI seuil(couches[i]) = faux
                //la couche est la dernière couche
                erreur ← dérivée(potentiel) * (sorties[j]−valeur(neurone[j]))
            SINON
                erreur ← dérivée(potentiel) * sorties[j]
            FIN SI
            POUR k ALLANT_DE 0 A nb_poids(neurone[j])
                poids[k] ← poids[k] + pas ∗ erreur
                                * valeur(obtenir_neurone(couche[i − 1], k))
                gradient[k] ← gradient[k] + poids[k] ∗ erreur
            FIN_POUR
          FIN_POUR
        sorties ← gradient
        FIN_POUR
FIN
//notes : chaque couche intègre une fonction (tanh, ...), ''dérivée'' et ''valeur''
//utilisées ici exploitent la fonction définie pour ce qui est demandé.
```

FIGURE B. 1 – Algorithme d'apprentissage d'un perceptron

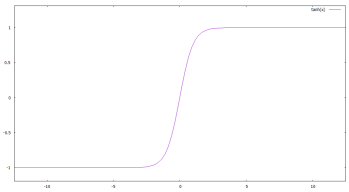
Les fonctions d'activation classiquement utilisées sont sigmoïde ou tangente hyperbolique.

Sigmoïde



Sa dérivée est :
$exp(-\text{valeur})/(1 + exp(-\text{valeur})^2)$

Tangente hyperbolique



Sa dérivée est : $1 - tanh(valeur)^2$

FIGURE B. 2 – Sigmoïde et tangente hyperbolique, deux fonctions classiques

Annexe C

Approches symboliques à partir de grammaires

LTAG : Lexicalized Tree Adjoining Grammar

TAG est un formalisme, elui des grammaires d'arbres adjoints. LTAG est un peu plus spécifique, il permet la lexicalisation des dites grammaires.

Pour entrapercevoir LTAG, reportons nous dans le passé, il y a un peu plus de vingt ans. Chandrasekar et Srinivas (1997) l'utilisent. Leur but ? détecter automatiquement les articulations au sein de phrases afin de permettre la découpe des dites phrases et leur restitution sous une forme grammaticalement plus légère.

Dans le formalisme de grammaire d'arbres adjoints LTAG, chaque feuille (éléments finaux de l'arbre) est associée à un élément lexical.

La phrase « *A fire rage in the mountains.* » représentée Figure C. 1 utilise l'évènement `e1` *rage*. *Rage* est un évènement défini dans un lexique. Chaque mot est défini dans un lexique, avec une entrée pour chacune de ses variantes (les homonymes ont chacun leur entrée).

```
*ENTRY: rage
*CAT: v
*EX: {A_fire_raged_in_the_mountains.}
```

TABLE C. 1 – extrait lexical minimaliste : *rage*

Une grammaire d'arbre adjoint permet de décrire dynamiquement un texte. Par exemple, Figure C. 1 le symbsectionole N, attribué dans le lexique aux noms, désigne des feuilles préfixables par un élan ancré avec un déterminant (D), il devient alors un groupe nominal (NP) qui lui n'est ni préfixable par un déterminant, ni par un adjectif (A).

Lors de l'analyse d'une phrase, un parseur tâche d'effectuer les attributions de symboles possibles, et ne retient - quand tout est géré par la grammaire - que celles qui permettent d'aboutir à une phrase où tous les éléments sont attribués de manière consi-

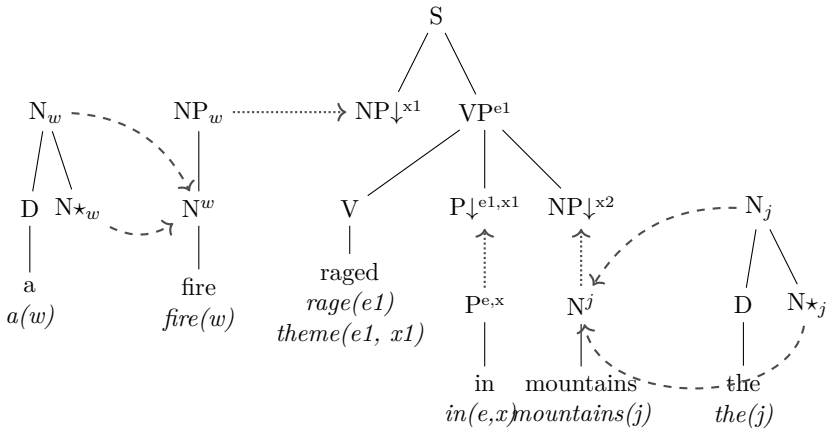


FIGURE C. 1 – Lexicalized Tree Adjoining Grammar : "A fire raged in the mountains"
.
Source : Création automatique d'une grammaire syntaxico-sémantique Colin (2017)

tante.

Par exemple, pour la structure présentée Figure C. 1, une grammaire est fonctionnelle si elle pose les règles :

- $N \rightarrow NP$ (un nom peut s'ancrer dans un groupe nominal),
- $A + N \rightarrow N$ (un adjectif peut s'ancrer dans un nom s'il est à sa gauche),
- $D + N \rightarrow NP$ (un déterminant peut sur un nom ancré dans un groupe nominal s'il est à sa gauche),
- $(NP + V) \Rightarrow S$ (un groupe nominal et un verbe forment une pharse),

Le parseur va tâcher de ranger les mots voisins au sein d'une structure arborescente selon un processus associant itérations et récursivité.

Annexe D

Paraphrases syntaxiques

D. 1 Données utilisées

Les données utilisées pour la génération de paraphrases sont disponibles sous git [79]. La structure des données est la suivante :

```
<?xml version='1.0' encoding='UTF-8'?>
<benchmark>
  <entries>
    <entry category="Astronaut" eid="Id8" size="1">
      <originaltripleset>
        <otriple>Alan_Bean | status | "Retired"@en</otriple>
      </originaltripleset>
      <modifiedtripleset>
        <mtriple>Alan_Bean | status | "Retired"</mtriple>
      </modifiedtripleset>
      <lex comment="good" jobid="954293" lid="1">Alan Bean is retired.</lex>
      <lex comment="good" jobid="954293" lid="2">Alan Bean has retired.</lex>
      <lex comment="good" jobid="904853" lid="3">Alan Bean is now retired.</lex>
    </entry>
    [...]
  </entries>
</benchmark>
```

FIGURE D. 1 – Structure des données WebNLG

Les textes sont distribués par catégorie tel que décrit par la Table D. 2. La taille des textes se mesure en nombre de *tokens*, c'est à dire d'éléments séparés par une espace (mots et ponctuation). La Figure D. 3 donne une vue d'ensemble de la taille des textes par lot de triplets.

La médiane du nombre de tokens est à 14 (moyenne à 15,71), avec le premier quartile à 9, le troisième à 21, selon la distribution montrée Figure D. 4.

79. https://gitlab.inria.fr/colineem/rdf2nn/tree/master

	Domaine							
Domaine	RDF 1	RDF 2	RDF 3	RDF 4	RDF 5	RDF 6	RDF 7	**Total**
Airport	376	241	234	259	252			**1362**
Artist	345	282	308	300	293			**1528**
Astronaut	90	57	80	107	108	114	113	**669**
Athlete	356	235	270	184	84			**1129**
Building	295	214	258	258	197			**1222**
CelestialBody	211	164	162	147	109			**793**
City	304	231	297	278	286			**1396**
ComicsCharacter	123	96	82	44	14			**359**
Food	340	348	392	405	299			**1784**
MeanOfTransportation	373	264	285	303	191			**1416**
Monument	48	40	52	60	56	45	32	**333**
Politician	374	310	323	310	188			**1505**
SportsTeam	314	213	213	188	56			**984**
University	72	49	73	91	78	78	68	**509**
WrittenWork	274	253	310	212	123			**1172**
Total	**3895**	**2997**	**3339**	**3146**	**2334**	**237**	**213**	**16161**

FIGURE D. 2 – WebNLG : Distribution lot de triplets par domaine

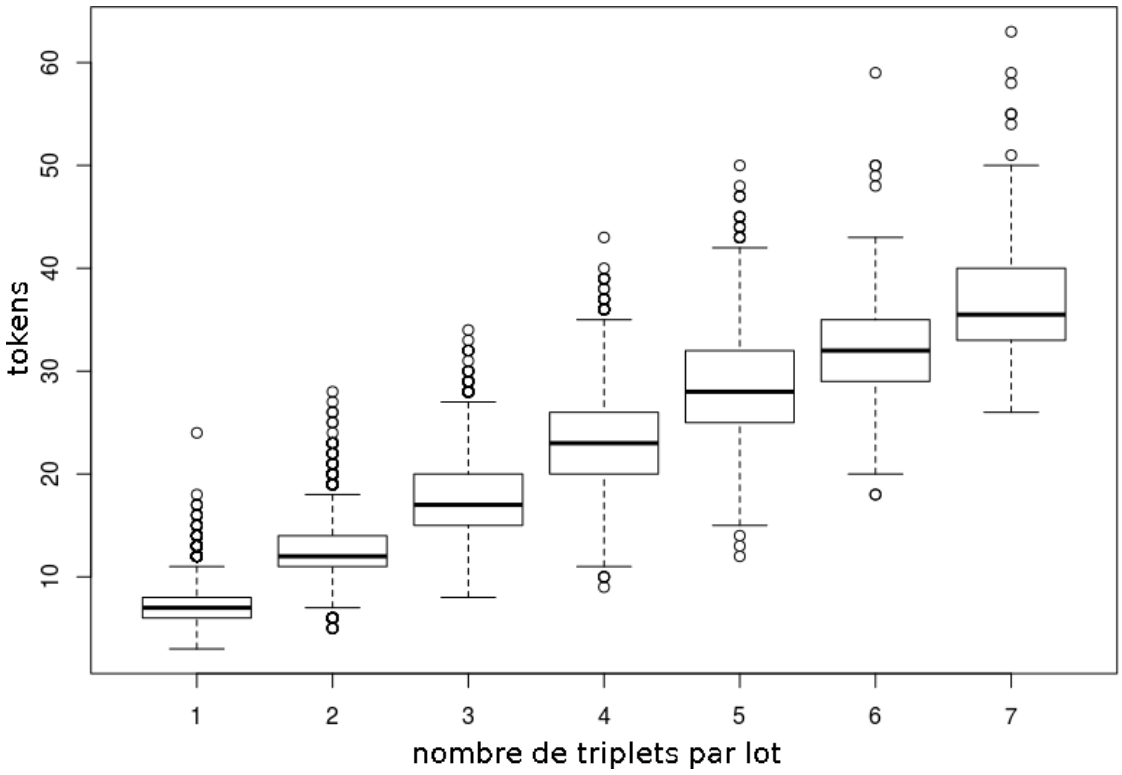


FIGURE D. 3 – WebNLG : Taille des textes par lot de triplets

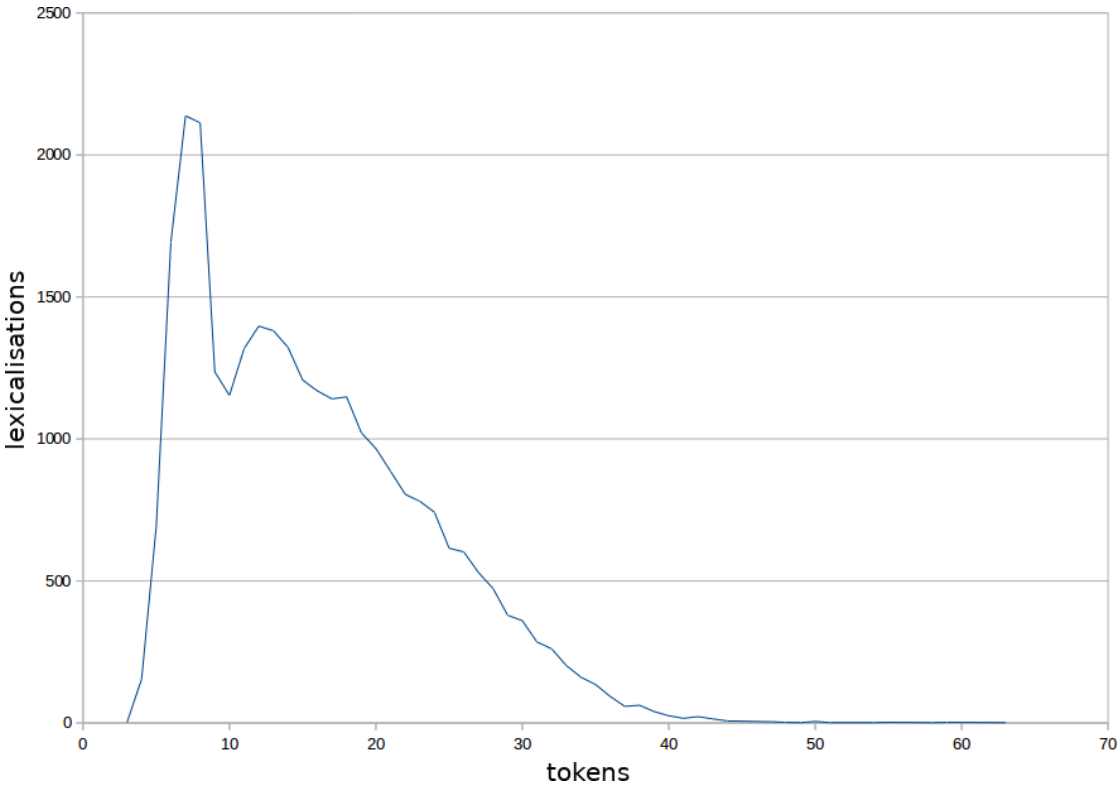


FIGURE D. 4 – WebNLG : Distributions des lexicalisations par tokens

D. 2 Identification des entités

Le triplet est décrit par les parties suivantes : [sujet] | [relation] | [objet]. Nous voulons trouver comment l'information se réalise, autrement dit quelle forme prend la relation pour un sujet et un objet donné. Le traitement se découpe ainsi :

- repérage des sujet et objet dans la réalisation → échecs stockés dans un fichier `entities_not_found.xml`, cf Figure D. 5.

- anonymisation de la phrase en vue d'une généralisation : remplacement de chaque concept sujet et objet par un identifiant. Les concepts sont numérotés par ordre de rencontre, et sont préfixé par "xlid", postfixés par "xrid".

- traitement des phrases originelles en analyse en dépendances et en constituants (Nivre et al., 2016). Les phrases rencontrant des problèmes majeurs d'analyse sont écartées. Ex : dans "Jens Härtel is in the 1. FC Magdeburg club." n'est pas repéré comme une entité. La phrase est considérée comme étant une succession de deux phrases. La segmentation de l'entité "1. FC Magdeburg club" permet de repérer le

souci. → échecs stockés dans un fichier `entities_not_correctly_parsed.xml`, cf Figure D. 6.

- les résultats finaux sont stockés dans du xml (`entities_parsed`, cf Figure D. 7)

```
<?xml version="1.0" encoding="utf-8"?>
<root>
 <file path="/home/colineem/Dropbox/benchmark_verified_manually_corrected/ \
       2triples/2triples_MeanOfTransportation_1020950_BMK_cleaned.xml">
  <entry category="MeanOfTransportation" eid="Id14" size="2">
   <tripleset>
    <triple>Plymouth_Plaza | manufacturer | Plymouth_(automobile)</triple>
    <triple>1955_Dodge | relatedMeanOfTransportation | Plymouth_Plaza</triple>
   </tripleset>
   <lex lid="3">
    <valeur>The 1955 Dodge and the Plymouth Plaza are both cars.</valeur>
    <missing>
     <element id="xlid14xrid" value="Plymouth_Plaza"/>
    </missing>
    <found>
     <element id="xlid15xrid" value="Plymouth_(automobile)"/>
     <element id="xlid0xrid" value="1955_Dodge"/>100
    </found>
    <valeur>The xlid0xrid and the xlid15xrid Plaza are both cars.</valeur>
   </lex>
  </entry>
   [...]
</root>
```

FIGURE D. 5 – Structure des données, des entités non trouvées dans des lexicalisations

```
<?xml version="1.0" encoding="utf-8"?>
<root>
 <file name="2triples_WrittenWork_983243_BMK_cleaned.xml">
  <eid name="Id37">
   <tripleset>
    <triple>ACM_Transactions_on_Information_Systems | LCCN_nb | 89646863</triple>
    <triple>ACM_Transactions_on_Information_Systems | abbr.| "ACM Trans. Inf. Syst."</triple>
   </tripleset>
   <tripleset_encoded>
    <triple>xlid765xrid | LCCN_number | xlid766xrid</triple>100
    <triple>xlid765xrid | abbr. | xlid767xrid</triple>
   </tripleset_encoded>
   <lex lid="1">
    <error cause="split on several sentences" key="xlid767xrid" position="[0, 9, 2, 2]"/>
    <encoded>xlid765xrid has the abbreviation of xlid767xrid and the LCCN number \
            xlid766xrid .</encoded>
    <analysed>ACM Transactions on Information Systems has the abbreviation of \
             ACM Trans. Inf. Syst and the LCCN number 89646863.</analysed>
   </lex>
   [...]
</root>
```

FIGURE D. 6 – Structure des données enregistrées comme non correctement analysées

(a) Données concernant les entités associées aux prédicats

```
<?xml version="1.0" encoding="utf-8"?>
<root>
 <file name="2triples_MeanOfTransportation_1020950_BMK_cleaned.xml">
  <eid name="Id1">
   <tripleset>
    <triple tid="0">1955_Dodge | bodyStyle | Convertible</triple>
    <triple tid="1">1955_Dodge | engine | 230 cubic inches</triple>
   </tripleset>
   <tripleset_encoded>
    <triple tid="0">xlid0xrid | bodyStyle | xlid1xrid</triple>
    <triple tid="1">xlid0xrid | engine | xlid2xrid</triple>
   </tripleset_encoded>
   <lexicalisation coref="analyzed" lid="1">
    <encoded>
     The xlid0xrid is a xlid1xrid with a xlid2xrid engine .
    </encoded>
    <analysed>
     The 1955 Dodge is a Convertible with a 230 cubic inches engine .
    </analysed>
    <sentences nombre="1">
     <sentence sid="0">
      <triplets>
       <triplet idObject="xlid1xrid" idSubject="xlid0xrid" partial="full" tid="0">
        <solutions>
         <sujet caracteristique="equal" indexDeb="2" indexFin="3"/>
         <objet caracteristique="equal" indexDeb="6" indexFin="6"/>
        </solutions>
       </triplet>
       <triplet idObject="xlid2xrid" idSubject="xlid0xrid" partial="full" tid="1">
        <solutions>
         <sujet caracteristique="equal" indexDeb="2" indexFin="3"/>
         <objet caracteristique="equal" indexDeb="9" indexFin="11"/>
        </solutions>
       </triplet>
      </triplets>
      <ROOT idx="0">
       <embeddedTriples>
        <triple id="xlid0xrid">
         <item idx="2" ner="DATE" pos="CD"/>
         <item idx="3" ner="ORGANIZATION" pos="NNP"/>
        </triple>
        <triple id="xlid1xrid">
         <item idx="6" ner="O" pos="JJ"/>
        </triple>
        <triple id="xlid2xrid"><item idx="9" ner="NUMBER" pos="CD"/>
         <item idx="10" ner="O" pos="JJ"/>
         <item idx="11" ner="O" pos="NNS"/>
        </triple>
       </embeddedTriples>
```

FIGURE D. 7 – Structures des analyses qui ont été correctement réalisées (a)

```
      <rootItems>
       <item idx="1" lemma="the" ner="O" pos="DT">The</item>
       <item idx="2" lemma="1955" ner="DATE" pos="CD">1955</item>
       <item idx="3" lemma="Dodge" ner="ORGANIZATION" pos="NNP">Dodge</item>
       <item idx="4" lemma="be" ner="O" pos="VBZ">is</item>
       <item idx="5" lemma="a" ner="O" pos="DT">a</item>
       <item idx="6" lemma="convertible" ner="O" pos="JJ">Convertible</item>
       <item idx="7" lemma="with" ner="O" pos="IN">with</item>
       <item idx="8" lemma="a" ner="O" pos="DT">a</item>
       <item idx="9" lemma="23\subfloat{0" ner="NUMBER" pos="CD">230</item>
       <item idx="10" lemma="cubic" ner="O" pos="JJ">cubic</item>
       <item idx="11" lemma="inch" ner="O" pos="NNS">inches</item>
       <item idx="12" lemma="engine" ner="O" pos="NN">engine</item>
       <item idx="13" lemma="." ner="O" pos="OTHERS">.</item>
      </rootItems>
     </ROOT>
     <graphe>
digraph { 6 [pos=JJ, word=Convertible]; 3 [pos=NNP, word=Dodge]; 6 -> 3 [key=nsubj];
4 [pos=VBZ, word=is]; 6 -> 4 [key=cop]; 5 [pos=DT, word=a]; 6 -> 5 [key=det];
12 [pos=NN, word=engine]; 6 -> 12 [key="nmod:with"]; 13 [pos=OTHERS, word="."];
6 -> 13 [key=punct]; 0 -> 6 [key=ROOT]; 1 [pos=DT, word=The]; 3 -> 1 [key=det];
2 [pos=CD, word=1955]; 3 -> 2 [key=nummod]; 7 [pos=IN, word=with]; 12 -> 7 [key=case];
8 [pos=DT, word=a]; 12 -> 8 [key=det]; 9 [pos=CD, word=230]; 12 -> 9 [key=nummod];
10 [pos=JJ, word=cubic]; 12 -> 10 [key=amod]; 11 [pos=NNS, word=inches];
12 -> 11 [key=compound]; }
     </graphe>
     <basic_graphe>
digraph { 6 [pos=JJ, word=Convertible]; 3 [pos=NNP, word=Dodge]; 6 -> 3 [key=nsubj];
4 [pos=VBZ, word=is]; 6 -> 4 [key=cop]; 5 [pos=DT, word=a]; 6 -> 5 [key=det];
12 [pos=NN, word=engine]; 6 -> 12 [key=nmod]; 13 [pos=OTHERS, word="."];
6 -> 13 [key=punct]; 0 -> 6 [key=ROOT]; 1 [pos=DT, word=The]; 3 -> 1 [key=det];
2 [pos=CD, word=1955]; 3 -> 2 [key=nummod]; 7 [pos=IN, word=with]; 12 -> 7 [key=case];
8 [pos=DT, word=a]; 12 -> 8 [key=det]; 9 [pos=CD, word=230]; 12 -> 9 [key=nummod];
10 [pos=JJ, word=cubic]; 12 -> 10 [key=amod]; 11 [pos=NNS, word=inches];
12 -> 11 [key=compound]; }
     </basic_graphe>
     <grapheOnEncodedString>
digraph { 5 [pos=NN, word=xlid1xrid]; 2 [pos=NN, word=xlid0xrid]; 5 -> 2 [key=nsubj];
3 [pos=VBZ, word=is]; 5 -> 3 [key=cop]; 4 [pos=DT, word=a]; 5 -> 4 [key=det];
9 [pos=NN, word=engine]; 5 -> 9 [key="nmod:with"]; 10 [pos=OTHERS, word="."];
5 -> 10 [key=punct]; 0 -> 5 [key=ROOT]; 1 [pos=DT, word=The]; 2 -> 1 [key=det];
6 [pos=IN, word=with]; 9 -> 6 [key=case]; 7 [pos=DT, word=a]; 9 -> 7 [key=det];
8 [pos=JJ, word=xlid2xrid]; 9 -> 8 [key=amod]; }
     </grapheOnEncodedString>
    </sentence>
   </sentences>
  </lexicalisation>
  [...]
</root>
```

(b) Données structurelles des phrases analysées

FIGURE D. 7 – Structures des analyses qui ont été correctement réalisées (b)

Afin de pouvoir travailler en déconnecté et de limiter l'usage de l'analyseur en dépendances, les retours de corenlp sont stockés dans un dictionnaire python, qui à chaque phrase parsée fait correspondre le tableau *json* contenant les réponses du serveur. Une clé est ajoutée en cas de réponse allégée. CoreNLP défaille parfois dans le traitement des coréférences (corefs). L'analyse lui est demandée une première fois avec coréférences (*corefs*). Si elle échoue, l'analyse est redemandée sans *corefs*. La clé ' `corefs_error`' est conséquemment ajoutée au dictionnaire fourni pour la phrase par pycorenlp, et est alors positionnée à True (que l'on sache ultérieurement qu'il y a eu un échec et que des coréférences ne se sont pas vues analysées. L'API de coreNLP détaille la structure *json* employée (https ://stanfordnlp.github.io/CoreNLP/corenlp-server.html#api-documentation). pycorenlp [80] n'est qu'un *wrapper* (fonction en appelant une autre) transférant dans un objet dictionnaire python l'ensemble des données issues de l'analyse. Les retours de corenlp sont stockés dans un fichier "stanford.json", la structure du dictionnaire associe à charge entrée (chaque lexicalisation parsée) le *json* renvoyé par le serveur. Pour python, cela devient un dictionnaire dont les clés sont `corefs`, `sentences`, et `corefs_error` (ajouté et dont la valeur est "True" si l'analyse des co-références n'a pas eu lieu).

Les données de stanford sont couplées aux lexicalisations. Un fichier entities_parsed.xml et output.xml sont produits (cf D. 7, le second identique au premier sans utilisation de BeautifulSoup donc sans indentation ni passage à la ligne, il est plus léger et plus facile à charger, mais guère affichable).

Les données (33 529 textes) sont dûment étiquetées. A niveau des *part of speech*, aux étiquettes de dialogue stanford (qui pour utilise celles définies pour le Penn Treebank [81] a été ajouté "*OTHERS*" car les éléments non taggués sont renvoyés tels quels (symbole renvoyé plutôt qu'un tag pré-défini). La redondance est minimalisée mais pensée pour favoriser l'exploitabilité. Un graphe est construit, il sera exploité pour la modélisation des contraintes.

D. 3 Enrichissement des données : les contraintes

Nous demandons donc au parseurs d'appliquer sur un texte différentes annotations en fonction de ce qu'il trouve. De base, nous demandons *tokenisation*, nature des *tokens* (ici *part-of-speech* : nature grammaticale, découpe en phrases, repérage des entités nommées, analyse en dépendances et des coréférences. Le parseurs nous renvoie poliment un *json* (traité pour créer un graphe (D. 2)) fort facile à traiter en *python*, d'autant avec la librairie `pygraphviz` qui permet de gérer facilement des graphes, et de les enregistrer dans du texte. Ces graphes sont le socle sur lequel les règles ont la possibilité de se réaliser.

Nous avons défini ainsi tout un petit ensemble de règles qui vient s'appliquer à un texte en une simple commande. À la commande d'application des règles, on passe un

80. pycorenlp : https://github.com/smilli/py-corenlp, consulté le 16 février 2020
81. *Penn Treebank Tag set* : https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html, fourni plus bas (F)

graphe en format textuel, elle l'instancie et le parcourt. Elle renvoie au final la liste des règles réalisées, du moins leur étiquette abstraite (*relative subject* quand la règle exemplifiant le processus vaut 1, cf (D. 3)).

```
<?xml version="1.0" encoding="utf-8"?>
<root>
 <exemple index="0" sentences="1" \
  source="2triples_MeanOfTransportation_1020950_BMK_cleaned.xml" theme="MeanOfTransportation" \
        triplets="2">
  <encoded>The xlid0xrid is a xlid1xrid with a xlid2xrid engine .</encoded>
  <origin>The 1955 Dodge is a Convertible with a 230 cubic inches engine .</origin>
```

(a) Données générales

FIGURE D. 8 – Fichier de travail construit (a)

```
  <relations>
   <relation identification="full" index="0" reference="bodyStyle" tokens="2" tokens_object="1" \
           tokens_subject="1">
    <subject id="xlid0xrid" id_calc="0" reference="1955 Dodge" ref_calc="0">
     <token identification="equal" index="2"/>
    </subject>
    <object id="xlid1xrid" id_calc="1" reference="Convertible" ref_calc="Convertible">
     <token identification="equal" index="5"/>
    </object>
   </relation>
   <relation identification="full" index="1" reference="engine" tokens="2" tokens_object="1" \
           tokens_subject="1">
    <subject id="xlid0xrid" id_calc="0" reference="1955 Dodge" ref_calc="0">
     <token identification="equal" index="2"/>
    </subject>
    <object id="xlid2xrid" id_calc="2" reference="230 cubic inches" ref_calc="230 cubic inches">
     <token identification="equal" index="8"/>
    </object>
   </relation>
  </relations>
  <constraints count="1" uniq="1">
   <contrainte name="existential">
    <socle id="0">
     <node index_node_rule="0" rule="existential"><token index="5"/></node>
     <node index_node_rule="1" pos="V.*" rule="existential"><token index="3"/></node>
    </socle>
   </contrainte>
  </constraints>
```

(b) Résultats de l'analyse

FIGURE D. 8 – Fichier de travail construit (b)

```
 <tokens begin="1" end="10">
  <token index="1" lemma="the" output_id="355" pos="DT" word_id="355">The</token>
  <token id_so="xlid0xrid" index="2" output_id="3427" pos="ENT" reference="1955 Dodge" \
        word_id="3427">
   1955 Dodge
  </token>
  <token index="3" lemma="be" output_id="571" pos="VBZ" word_id="571">is</token>
  <token index="4" lemma="a" output_id="5305" pos="DT" word_id="5305">a</token>
  <token id_so="xlid1xrid" index="5" output_id="4450" pos="ENT" \
    reference="Convertible" word_id="4450">
   Convertible
  </token>
  <token index="6" lemma="with" output_id="4831" pos="IN" word_id="4831">with</token>
  <token index="7" lemma="a" output_id="5305" pos="DT" word_id="5305">a</token>
  <token id_so="xlid2xrid" index="8" output_id="4791" pos="ENT" reference="230 cubic inches" \
        word_id="4791">230 cubic inches</token>
  <token index="9" lemma="engine" output_id="2932" pos="NN" word_id="2932">engine</token>
  <token index="10" lemma="." output_id="291" pos="OTHERS" word_id="291">.</token>
 </tokens>
</exemple>
 [...]
</root>
```

(c) Texte *tokenisé*

FIGURE D. 8 – Fichier de travail construit (c)

D. 4 Analyse des contraintes, le système de règles

D. 4.1 Graphe issu du parsing de graphe du stanford parser

Le paquet pygraphviz permet de créer et naviguer aisément dans des graphes, et de gérer des graphes orientés.

Le programme traitant les données sémantiques travaillent avec des graphes porteurs des informations de tagging (pos [part of speech] et valeur (le mot taggué)).

D. 4.2 Fonctionnement de la détection de structures

Tout repose sur la notion de système auquel des règles s'appliquent.

Soit s le système de gestion des graphes. (D. 1)

Soit D un ensemble de phrases annotées par le stanford parser, soit G un ensemble de graphes orientés défini par $G(V, E)$ où V est un ensemble de nœuds et E un ensemble de flèches.
La fonction :
$e(v_1, v_2)||\{v_1, v_2\} \subset V$ définit un lien de v_1 vers v_2
Un attribut d'un nœud v ou d'une flèche e lié par la fonction :

$p(x, nomAttribut) = valeurAttribut | x \in \{v, e\}$
Une phrase annotée est composée de tokens, l'ensemble des tokens d'un phrase d appartenant à D est noté T^d.
Elle offre une analyse des dépendances et des catégories grammaticales pour laquelle nous définissons une fonction a acceptant une valeur telle que :
$a(t_1) = (t_2, dep) | \{t_1, t_2\} \subset T, dep \in DEP$, DEP étant l'ensemble des dépendances telles que décrites dans l'annexe 1.
et une fonction a acceptant deux valeurs telle que :
$a(t, annot) = val | t \in T, annot \in NTAG$, $NTAG = \{$'*idx*', '*word*', '*lemma*'; '*ner*', '*pos*'$\}$ et *val* la valeur fournie par l'analyseur pour t_1 et l'annotation *annot* donnée.

Dans la phrase *C'était des belles intuitions qui valaient expérience* :

- a('belles') = {'intuition', 'amod'} : le gouverneur syntaxique de *belle* est *intuition*,
- a('belles', '*idx*') = 4 : l'index de *risquée* est 3.
- a('belles', '*word*') $=' belles'$: le mot de *belles* est *belles*,
- a('belles', '*lemma*') $=' belle'$: le lemme de *belles* est *belle*,
- a('belles', '*ner*') $=' \_'$: aucune entité nommée n'a été trouvée pour le mot,
- a('belles', '*pos*') $=' ADJ'$: la nature grammaticale de *belles* est adjectif.

pos et *word* seront recherchés sous forme d'expression régulière. L'usage d'assertions négatives est facilité pour *pos* et uniquement *pos*. Si `'-'` préfixe la valeur donnée à *pos*, il sera vérifié que le nœud n'est pas taggué par ce qui suit le préfixe.

$$\begin{array}{lll} \forall d \in D, & g_d \in G & \\ & \forall t \in T_d, & v_t \in g_d | v_t \text{ est un nœud identifiant } t \\ & & v_2, lien = a(t), e_i = e(v_t, v_2), p(e_i) = lien \\ & & \forall attrib \in NTAG, p(v_t, attrib) = a(t, attrib) \end{array} \tag{D. 2}$$

La création de règles se fait à partir de trois fonctions de base, portant sur des nœuds du graphe, les liens qui potentiellement les lient. Nous avons conçu un objet `system` : un système de règles, un espace où définir des règles.

Nos règles sont de simples successions de définitions exploitant le graphe permettant de générer le fichier de travail final dont un extrait est présenté figure D. 8.

Soit R un ensemble des règles liant G à des fonctions spécifique.
$r(g)$ est la fonction prototypale de R ; elle est définie dans R pour G par la fonction $R(G,$'rule name'$)$, comme dans l'exemple ci-dessous. Elle s'applique à $g \in G$, avec ''rule name'' valorisé par un nom de règle. La valeur par défaut de r est 0.

$$\begin{array}{l} rs = R(G, \text{'relative subject'}) \\ \forall g \in G,\ rs(g) = 1 \text{ si et seulement si } \end{array} \begin{array}{l} \exists v_1 \in g | p(v_1, \text{'}pos\text{'}) = \text{'V.*'} \\ \exists v_2 \in g \end{array} \Big| \begin{array}{l} p(v_2, v_1) = \text{'acl :relcl'} \\ p(v_1, v_2) = \text{'nsubj.*'} \end{array}$$

(D. 3)

`node()` est un alias de la fonction p permettant la définition d'un archétype de nœud. Par exemple, `node('pos', 'V.*')` permet de définir un nœud tel v_1 montré dans (D. 3)

`link()` est un alias de la fonction p permettant la définition d'un lien entre archétypes de nœuds. Par exemple, `link(`v_1, v_2`, 'nsubj.*')` permet de définir un lien de type *nsubj** entre v_1 et v_2 .
Soit ge_1 le graphe de la phrase *C'était des belles intuitions qui valaient expérience* :
$rs(ge_1)$ vaut 1.
Soit ge_2 le graphe de la phrase *C'était des belles intuitions, elles valaient expérience* :
$rs(ge_2)$ vaut 0.

Une règle est dotée, ainsi qu'illustré, d'un nom, elle fait partie parmi d'autres du système de gestion (D. 3) :

Soit R un ensemble de règles, $\forall r \in R$, $r \in s$,
Soit $nom(r)$ un la fonction qui à une règle r associe un nom. (D. 4)
$nom(rs) =$ 'relative subject'

La gestion des règles offre d'autres possibilités fondamentales que la contrainte sur propriété.

Soit $position(n)$ la fonction qui à un nœud n_1 associe une position relative à un nœud n_2 voisin de n_1 : g (gauche) ou d (droite).

$$position(n_1, n_2) = p | p \in \{g, d\} \quad \text{(D. 5)}$$

Nous avons aussi prévu dans le système la circonscription des recherches à un ensemble de nœuds. $\neg explore(N)$ est la fonction qui à une règle r associe un ensemble N aux archétypes de nœuds gérés par r, tel que N contient des séquences dans lesquels la règle ne pourra être vérifiée.
$\neg explore(N) | N = \{\{e_0, \cdots, e_m\}, \cdots, \{f_0, \cdots, f_o\}\}$ où $\forall M \in N$, M est un ensemble d'instances de nœuds dont les propriétés et connexions, les uns par rapport aux autres, ne se verront pas exploitées.

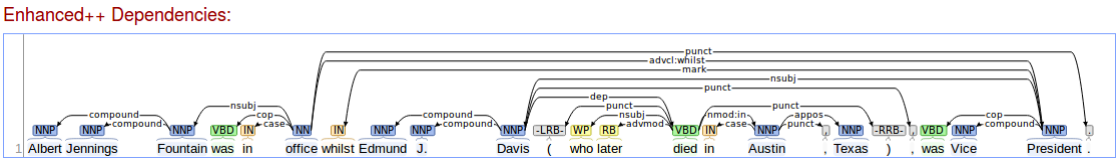
Cette vision vise le non traitement des composants internes, propriété et valeur de propriété pour i, prédicat donné, par la règle. Les propriétés et leurs valeurs sont verbalisés en état dans les phrases et sont parfois complexes (groupes verbaux, groupes nominaux structurés bien au-delà d'une simple adjectivation).

Un système contient un nombre libre de règles. Un graphe tel que celui illustré figure D. 10, en tant qu'objet graphe *pygraphviz*, est passé sous forme textuelle au système, qui l'instancie et y valide - ou non - les règles.

Le premier nœud (*nœud_a*) est le nœud d'où part la relation. Le second nœud (*nœud_b*) est celui où termine la relation (ex : *verbe* → *sujet*, voir section F. 2).

La définition de lien est un peu plus fine que celle des nœuds. La nature du lien attendue dispose d'une gestion plus complexe que celle du *pos* ou du *word* d'un nœud.

Si `nature_lien`, tel que `nature_lien` est le troisième paramètre de la fonction `link` est :




```
# soit une rule rule appartenant à  un système z dont la typologie est la suivante:
# soit x un nœud à  pos=WP (wh-pronoun) --> who est retenu pour le groupe
x = node(pos='WP')
# soit y un verbe --> was, died et was sont retenus pour le groupe
y = node(pos='V.*')
# soit un lien de y à  x de nature "nsubj.*"
--> les liens nsubj et nsubjpass seront retenus
l = link(y, x, 'nsubj.*')
# pour la phrase montrée en exemple, la règle se réalisera au sein de z.
```

FIGURE D. 9 – Exemple illustré de recherche de relations

- vide (=") ou égale à '*' : on demande juste à un lien d'être,
- égale à un lien tel que codifié en section F. 2 : on demande à ce lien précis d'être,
- se termine par '.*' : on demande au lien entre nœud_a et nœud_b de répondre à l'expression régulière de fait fournie en lieu et place de *nature_lien*.

D. 4.3 Application d'une règle

L'application d'une règle au sein d'un système de règle se réalise sut un graphe tel que l'illustre la Figure D. 10 (le nom *strict digraph* n'a pas d'importance). Le graphe est passé sous forme textuelle au système.

Fonction appelée :

```
def apply_rules(self, graphe, forbidden_relations):
    graph = pygraphviz.AGraph(graphe.replace("\n", "").replace("\t", "")
            .strip())
    rules_applied=[]
    [...]
    return rules_applied
```

Appel fonction

```
constraints = self.system.apply_rules(graphe.text, \
                internal_relation_triple_so_excluded)
```

Les relations à ne pas prendre en compte (par exemple entre liens reliant les éléments d'une entité nommée) sont un argument possible pour l'appel de fonction. Un tableau vide peut être fourni en absence de contrainte.

```
internal_relation_triple_so_excluded = [[]]
```

Pour exclure les relations de la réalisation d'une règle dans la phrase " *Alan Bean was born in Wheeler, Texas on "1932-03-15".* " (Figure D. 10) entre *Wheeler* (index 6), , (index 7) et *Texas* (index 8), mais aussi entre *Alan* et *Bean* (respectivement index 1 et 2) :

```
internal_relation_triple_so_excluded = [[6, 7, 8], [1, 2]]
```

Les liens de 6 à 7, de 7 à 8, de 7 à 6, de 7 à 8, de 8 à 6 et de 8 à 7, ainsi que les liens de 1 à 2 et de 2 à 1, ne seront pas pris en compte.

Concrètement, le système parcourt le graphe, enregistre un ensemble de nœuds correspondant à chaque type de nœud décrit.

Si on lui dit *soit x un noeud pos-taggué V.** au sein de la règle *relative subject* :

```
x = rule_relativeSubject.add_node(pos='V.*')
```

Le système va construire une liste des nœuds correspondants à la règle, et ainsi de suite pour chacun des nœuds imposés.

Ensuite, le système vérifie l'existence d'un lien répondant au critère de la règle. Si le lien de nœud_a a nœud_b existe, nœud_a est conservé, sinon, nœud_a est rejeté. Une fois tous les nœuds$a$ ayant bien un lien avec un nœud de l'ensemble potentiel b sélectionné, les nœuds b qui se sont pas reliés à un nœud a tel que la règle l'impose sont eux aussi éliminés.

Si un ensemble de nœuds est vide, la règle échoue.

Une fois les nœuds vérifiés, sont ensuite appliquées les règles des liens interdits. Pour ce stade, on ne défalque pas les nœud b, on se contente d'éliminer les nœuds a ayant un lien interdit vers un b.

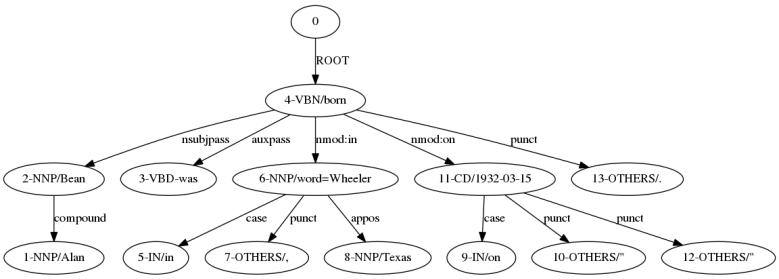
A nouveau, si un ensemble de nœuds est vide, la règle échoue. Sinon :

```
if nothing_forbidden and 0 not in [len(tab) for tab in nodes_candidats]:
    rules_applied.append(rule.name)
```

Le système d'évaluation des règles s'exerce dans un parcours de graphe tel que celui de la figure D. 10.

```
<item idx="1" lemma="Alan" ner="PERSON" pos="NNP">Alan</item>
<item idx="2" lemma="Bean" ner="PERSON" pos="NNP">Bean</item>
<item idx="3" lemma="be" ner="O" pos="VBD">was</item>
<item idx="4" lemma="bear" ner="O" pos="VBN">born</item>
<item idx="5" lemma="in" ner="O" pos="IN">in</item>
<item idx="6" lemma="Wheeler" ner="LOCATION" pos="NNP">
     Wheeler</item>
<item idx="7" lemma="," ner="O" pos="OTHERS">,</item>
<item idx="8" lemma="Texas" ner="LOCATION" pos="NNP">
     Texas</item>
<item idx="9" lemma="on" ner="O" pos="IN">on</item>
<item idx="10" lemma="‘‘" ner="O" pos="OTHERS">"</item>
<item idx="11" lemma="1932-03-15" ner="DATE" pos="CD">
     1932-03-15</item>
<item idx="12" lemma="’’" ner="O" pos="OTHERS">"</item>
<item idx="13" lemma="." ner="O" pos="OTHERS">.</item>
```

(a) Tokens : *Alan Bean was born in Wheeler, Texas on "1932-03-15".*



(b) Représentation graphique

```
strict digraph {
  4 [pos=VBN, word=born];
  2 [pos=NNP, word=Bean];
  4 -> 2 [key=nsubjpass];
  3 [pos=VBD, word=was];
  4 -> 3 [key=auxpass];
  6 [pos=NNP, word=Wheeler];
  4 -> 6 [key="nmod:in"];
  11 [pos=CD, word="1932-03-15"];
  4 -> 11 [key="nmod:on"];
  13 [pos=OTHERS, word="."];
  4 -> 13 [key=punct];
  0 -> 4 [key=ROOT];
  1 [pos=NNP, word=Alan];
  2 -> 1 [key=compound];
  5 [pos=IN, word=in];
  6 -> 5 [key=case];
  7 [pos=OTHERS, word=","];
  6 -> 7 [key=punct];
  8 [pos=NNP, word=Texas];
  6 -> 8 [key=appos];
  9 [pos=IN, word=on];
  11 -> 9 [key=case];
  10 [pos=OTHERS,word="\""];
  11 -> 10 [key=punct];
  12 [pos=OTHERS, word="\""];
  11 -> 12 [key=punct];}
```

(c) Linéarisation

FIGURE D. 10 – (c) : linéarisation du graphe (b) représentant l'organisation des *tokens* (a)

D. 5 Règles prises en charge

Nous travaillons sur un corpus anglais, le nom des règles est en anglais - approximatif peut-être. Les compétences linguistiques mises en œuvre dans ce projet sont en construction. Il s'agit pour la partie 1 de disposer de leviers pertinents.

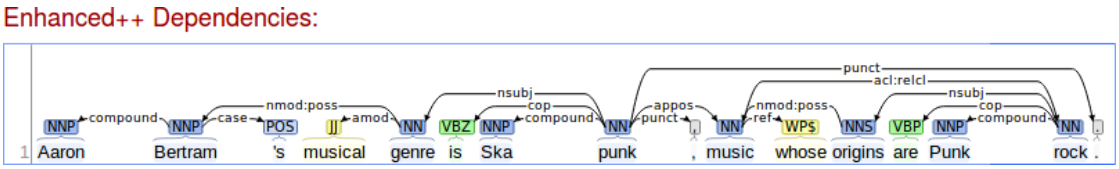
D. 5.1 Les règles en détail

Sachant un système self.system, nous créons les règles. Leur dénomination fine est liée à un historique de tests. Il peut y avoir une règle ˜5 sans règle ˜4, merci de ne pas prendre garde à ce genre de détail.

```
def make_systeme(self):
self.system=rules.System()
# rappel, la definition d'un lien entre deux noeuds est:
# add link(departfleche, arrivefleche, etiquettefleche)
```

D. 5.2 Apposition

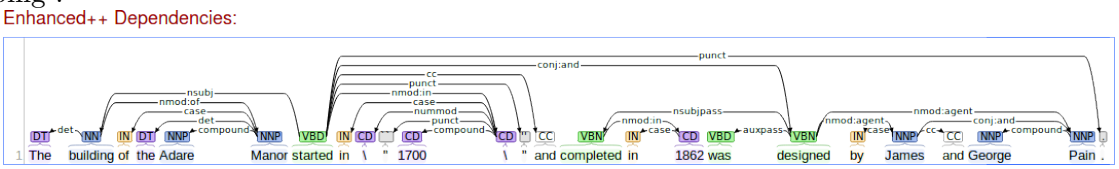
Nous utilisons le lien ‘appos’ pour déterminer si une relation se réalise dans le cadre d’une apposition.




```
rule_appos = rules.System.Rule(name="apposition")
# soit deux noeuds quelconques
x = rule_appos.add_node()
y = rule_appos.add_node()
# il existe un lien csubj.* unissant deux noeuds
rule_appos.add_link( x, y, 'appos')
# ajout regle au système
self.system.add_rule(rule_appos)
```

D. 5.3 Clausal subject (Groupe verbal sujet)

Nous nous contentons d’exploiter le tag “csubj.*”, qui permet de détecter les expressions verbales sujets d’un verbe, que ce verbe soit ou non au passif.
Le résultat n’est pas très brillant. Ce modèle est exclu des travaux sur les paraphrases webnlg .




```
rule_cs = rules.System.Rule(name="clausal_subject_1")
# soit verbe_maitre qui aura un sujet
verbe_maitre =rule_cs.add_node('VB.*', inside_allow=False)
# le verbe de la clause
verbe_clause =rule_cs.add_node('VB.*', inside_allow=False)
# nous sommes dans des phrases simples
rule_cs.set_position(verbe_clause, verbe_maitre)

# le verbe maître sera dépourvu de sujet
x = rule_cs.add_node()
rule_cs.forbid_link(verbe_maitre, x, ".subj.*")
# et il n'est pas le verbe être
rule_cs.forbid_link(x, verbe_maitre, "cop")
rule_cs.add_link(verbe_clause, verbe_maitre, "ccomp")
# cette règle vise à éviter l'intégration des relatives type
# The ground of A.S. Gubbio 1910 is located in Italy, where the people who live there
# are called Italians.
# sinon elles correspondraient
```

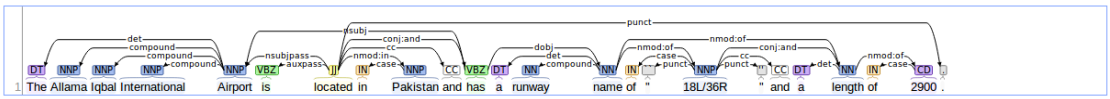
```
rule_cs.forbid_link(rule_cs.add_node(), verbe_clause, "acl:relcl")

# ajout de la règle au système
self.system.add_rule(rule_cs)

# nous pourrions aussi avoir le verbe être. c'est détectable avec clausal subject,
# plutôt avec des what kjiso wants is ksdj

rule_cs = rules.System.Rule(name="clausal_subject_3")
# soit verbe_maitre qui aura un sujet
verbe_maitre =rule_cs.add_node('VB.*', inside_allow=False)
#le verbe de la clause
verbe_clause =rule_cs.add_node('VB.*', inside_allow=False)
# nous sommes dans des phrases simples
rule_cs.set_position(verbe_clause, verbe_maitre)

# le verbe maître sera dépourvu de sujet
x = rule_cs.add_node()
# et il est le verbe être
rule_cs.add_link(x, verbe_maitre, "cop")

rule_cs.add_link(x, verbe_clause, "csubj*")

# ajout de la règle au système
self.system.add_rule(rule_cs)

# nous pourrions aussi avoir le verbe être. c'est détectable avec un ccomp quand pré-fixé

rule_cs = rules.System.Rule(name="clausal_subject_2")
# soit verbe_maitre qui aura un sujet
verbe_maitre =rule_cs.add_node('VB.*', inside_allow=False)
# le verbe de la clause
verbe_clause =rule_cs.add_node('VB.*', inside_allow=False)
# nous sommes dans des phrases simples
rule_cs.set_position(verbe_clause, verbe_maitre)

# le verbe maître sera dépourvu e sujet
x = rule_cs.add_node()
# et il est le verbe être
rule_cs.add_link(x, verbe_maitre, "cop")
rule_cs.forbid_link(x, rule_cs.add_node(), "nsubj.*")

rule_cs.add_link(verbe_clause, x, "ccomp")

# ajout de la règle au système
self.system.add_rule(rule_cs)
```

D. 5.4 Coordinated clauses (coordination verbale)

Cinq règles nous permettent de "récolter" les clauses coordonnées simples (type "sujet + verbe [coordination] verbe").

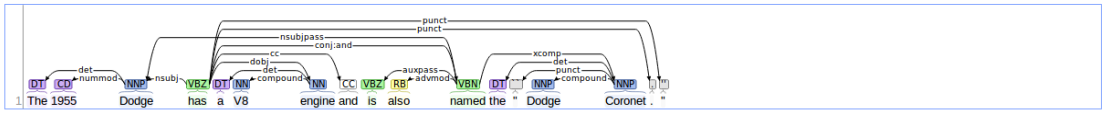
clause 1 *The Allama Iqbal International Airport is located in Pakistan and has a runway name of Ï8L/36Rând a length of 2900.*

Enhanced++ Dependencies:



clause 2 *Aaron Bertram performs Ska punk music and plays for the Suburban Legends band.*

Enhanced++ Dependencies:



clause 5 *Alison O'Donnell is a musician for the United Bible Studies band and was previously member of the Flibbertigibbet band.*

Enhanced++ Dependencies:


```
rule_coord = rules.System.Rule(name="coordinated_clauses_1")
#soit x et y deux verbes
v1 = rule_coord.add_node(pos='V.*', inside_allow=False)
v2 = rule_coord.add_node(pos='V.*', inside_allow=False)
#soit s1 et s2 deux sujets
s1 = rule_coord.add_node()
s2 = rule_coord.add_node()
rule_coord.add_link(v1, s1, "nsubj.*")
rule_coord.forbid_link(v2, s2, "nsubj.*")
x = rule_coord.add_node()
# on exclut les cop [x -> verbe être] aboutissant à ce que être n'ait pas de sujet.
rule_coord.forbid_link(x, s2, "cop")
x2 = rule_coord.add_node()

#soit c une coordination
c = rule_coord.add_node(pos="CC", inside_allow=False)
rule_coord.set_position(v1, c)
rule_coord.set_position(s1, c)
rule_coord.set_position(s2, c)#s2 aussi est après cc
rule_coord.set_position(c, v2)
rule_coord.set_position(c, x2)
rule_coord.forbid_link(s1, x2, "acl:relcl")
#j'ajoute la règle
self.system.add_rule(rule_coord)
```

Coordinated clauses 2

```
rule_coord = rules.System.Rule(name="coordinated_clauses_2")
#soit x et y deux verbes
```

```
v1 = rule_coord.add_node(pos='V.*', inside_allow=False)
v2 = rule_coord.add_node(pos='V.*', inside_allow=False)
#soit s1 le sujet commun
s1 = rule_coord.add_node()
s2 = rule_coord.add_node()
rule_coord.add_link(v1, s1, "nsubj.*")
rule_coord.add_link(v2, s1, "nsubj.*")

#soit c une coordination
c = rule_coord.add_node(pos="CC", inside_allow=False)
rule_coord.set_position(v1, c)
rule_coord.set_position(s1, c)
rule_coord.set_position(c, v2)
#j'ajoute la règle
self.system.add_rule(rule_coord)
```

Coordinated clauses 3

```
#on s'occupe des "is"
rule_coord = rules.System.Rule(name="coordinated_clauses_3")
#soit x et y deux verbes
v1 = rule_coord.add_node(pos='V.*', inside_allow=False)
v2 = rule_coord.add_node(pos='V.*', inside_allow=False)
#soit s1 le sujet commun
s1 = rule_coord.add_node()
#soit x ce qui est
x = rule_coord.add_node()
rule_coord.add_link(v1, s1, "nsubj.*")
#on relie x à v2 par un lien cop
rule_coord.add_link(x, v2, "cop")
#est relié à v1 dans le cadre de la coordination verbale
rule_coord.add_link(v1, x, "conj.*")
#et ce qui est cop n'est pas sujet
#pas sujet
nonsuj= rule_coord.add_node()
rule_coord.forbid_link(x, nonsuj, ".subj.*")

#soit c une coordination
c = rule_coord.add_node(pos="CC", inside_allow=False)
rule_coord.set_position(v1, c)
rule_coord.set_position(s1, c)
rule_coord.set_position(c, v2)
#j'ajoute la règle
self.system.add_rule(rule_coord)
```

Coordinated clauses 4

```
#on s'occupe des "is"
rule_coord = rules.System.Rule(name="coordinated_clauses_4")
#soit x et y deux verbes
v1 = rule_coord.add_node(pos='V.*', inside_allow=False)
v2 = rule_coord.add_node(pos='V.*', inside_allow=False)
```

```
#soit s1 le sujet commun
s1 = rule_coord.add_node()
#soit x ce qui est
x1 = rule_coord.add_node()
rule_coord.add_link(x1, s1, "nsubj.*")
rule_coord.add_link(v2, s1, "nsubj.*")

#on relie x à v1 par un lien cop
rule_coord.add_link(x1, v1, "cop")
#est relié à v2 dans le cadre de la coordination verbale
rule_coord.add_link(v2, x1, "conj.*")

#soit c une coordination
c = rule_coord.add_node(pos="CC", inside_allow=False)
rule_coord.set_position(x1, c)
rule_coord.set_position(v1, c)
rule_coord.set_position(s1, c)
rule_coord.set_position(c, v2)
#j'ajoute la règle
self.system.add_rule(rule_coord)
```

Coordinated clauses 5

```
#on s'occupe des "is"
rule_coord = rules.System.Rule(name="coordinated_clauses_5")
#soit x et y deux verbes
v1 = rule_coord.add_node(pos='V.*', inside_allow=False)
v2 = rule_coord.add_node(pos='V.*', inside_allow=False)
#soit s1 le sujet
s1 = rule_coord.add_node()
#soit x1 et x2 ce qui est
x1 = rule_coord.add_node()
x2 = rule_coord.add_node()
#on relie x à v1 par un lien cop
rule_coord.add_link(x1, v1, "cop")
#et x1 est relié au sujet par nsubj
rule_coord.add_link(x1, s1, "nsubj.*")
#il n'y a qu'un seul sujet référencé dans ce cas.

#on relie x2 à v2 par un lien cop
rule_coord.add_link(x2, v2, "cop")

#et un lien conj.* lie x1 et x2
rule_coord.add_link(x1, x2, "conj.*")

#par ailleurs nous avons une coordination au milieu

#soit c une coordination
c = rule_coord.add_node(pos="CC", inside_allow=False)
rule_coord.set_position(x1, c)
rule_coord.set_position(v1, c)
rule_coord.set_position(s1, c)
rule_coord.set_position(c, v2)
```

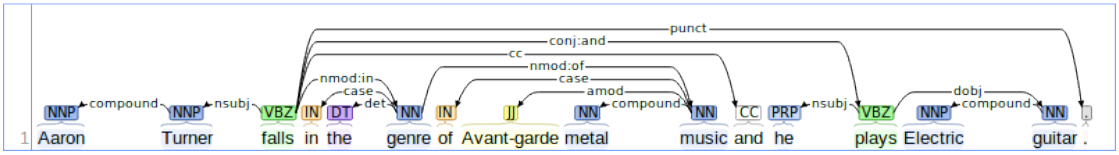
```
rule_coord.set_position(c, x2)
#j'ajoute la règle
self.system.add_rule(rule_coord)
```

D. 5.5 Coordinated full clauses (coordination)

Les *coordinated full clauses* disposent de trois règles de détection, suffixées _1, _2 et _3.
Nous avons eu de gros problèmes de parsing pour les clauses pleinement coordonnées (type "sujet + verbe [coordination] sujet + verbe").
Nous avons choisi la méthode brute, définissant deux verbes *v1* et *v2* chacun équipé de sujets *s1* et *s2*, le couple {*v1*, *s1*} étant sans ordre spécifié à gauche d'une conjontion de coordination, et {*v2*, *s2*} à droite.
Nous interdisons une relation acl :relacl entre le verbe de la partie droite et son sujet, pour éliminer une relative à droite de la coordination de deux noms communs.

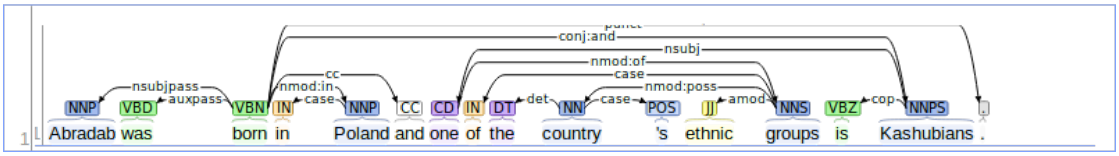
full clause 1

Enhanced++ Dependencies:



full clause 2 Une seconde règle permet de détecter les conjonctions où le verbe est *être*.

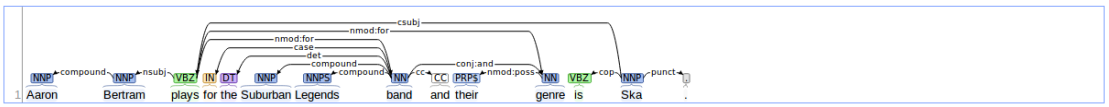
Enhanced++ Dependencies:



full clause 3

Aaron Bertram plays for the Suburban Legends band and their genre is Ska.

Enhanced++ Dependencies:




```
rule_coord = rules.System.Rule(name="coordinated_full_clauses_1")
#soit x et y deux verbes
```

```
v1 = rule_coord.add_node(pos='')
v2 = rule_coord.add_node(pos='V.*', inside_allow=False)
#soit s1 et s2 deux sujets
s1 = rule_coord.add_node()
s2 = rule_coord.add_node()
rule_coord.add_link(v1, s1, "nsubj.*")
rule_coord.add_link(v2, s2, "nsubj.*")

# il existe un nouud quelconque k
k=rule_coord.add_node()
# ce noeud est à droite du sujet 2
rule_coord.set_position(s2, k)
# il n'a pas de lien ref avec le sujet de v2
# si c'était le cas, nous aurions une relative sujet
rule_coord.forbid_link(s2, k, "ref")
rule_coord.forbid_link(s2, v2, "acl:relcl")

# soit c une coordination
c = rule_coord.add_node(pos="CC", inside_allow=False)
rule_coord.set_position(v1, c)
rule_coord.set_position(s1, c)
rule_coord.set_position(c, v2)
rule_coord.set_position(c, s2)
x = rule_coord.add_node()
rule_coord.forbid_link(x, s2, "cop") # on exclut les cop [x -> verbe être]

# j'ajoute la règle
self.system.add_rule(rule_coord)
```

Coordinated full clauses 2

```
#on s'occupe des "is"
rule_coord = rules.System.Rule(name="coordinated_full_clauses_2")
#soit x et y deux verbes
v1 = rule_coord.add_node(pos='')
v2 = rule_coord.add_node(pos='V.*', inside_allow=False)
#soit s1 le sujet du verbe 1
s1 = rule_coord.add_node()
#soit s2 le sujet du verbe 2
#soit x ce qui est
x = rule_coord.add_node()
rule_coord.add_link(v1, s1, "nsubj.*")
#on relie x à v2 par un lien cop
rule_coord.add_link(x, v2, "cop")
#est relié à v1 dans le cadre de la coordination verbale
rule_coord.add_link(v1, x, "conj.*")
#et ce qui est cop est sujet
sujet= rule_coord.add_node()
rule_coord.add_link(x, sujet, ".subj.*")


#soit c une coordination
c = rule_coord.add_node(pos="CC", inside_allow=False)
```

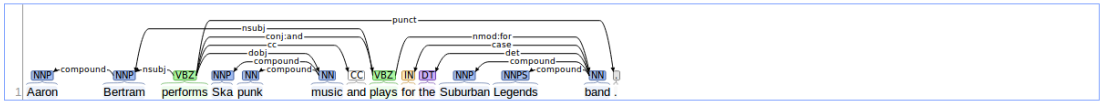
XXX

```
rule_coord.set_position(v1, c)
rule_coord.set_position(s1, c)
rule_coord.set_position(c, sujet)
rule_coord.set_position(c, x)
rule_coord.set_position(c, v2)
#j'ajoute la règle
self.system.add_rule(rule_coord)
```

D. 5.6 Direct object/Transitive clause (cod)

Aaron Bertram performs Ska punk music and plays for the Suburban Legends band.

Enhanced++ Dependencies:

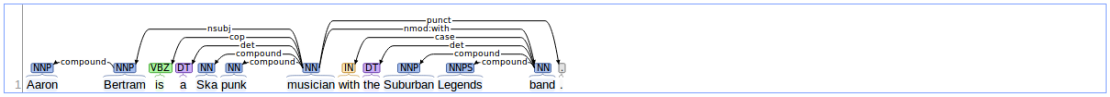



```
objectDirect = rules.System.Rule(name="direct_object")
#soit x et y deux verbes
x = objectDirect.add_node(pos='V.*', inside_allow=False)
y = objectDirect.add_node(pos='')

objectDirect.add_link(x, y, 'dobj')

#j'ajoute la règle
self.system.add_rule(objectDirect)
```

D. 5.7 Existential/predicative clause (existentielle)

Aaron Bertram is a Ska punk musician with the Suburban Legends band.

Enhanced++ Dependencies:

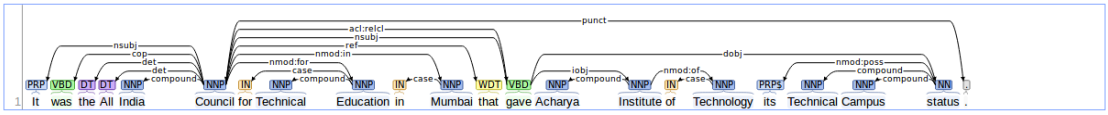



```
existencielle = rules.System.Rule(name="existential")
#soit x et y deux verbes
y = existencielle.add_node(pos='')
z = existencielle.add_node(pos='V.*', inside_allow=False)

existencielle.add_link(y, z, 'cop')

#j'ajoute la règle
self.system.add_rule(existencielle)
```

D. 5.8 Indirect object (COI)

Il y en a peu... : la structure est sous-représentée dans le corpus. Voici un exemple type :
It was the All India Council for Technical Education in Mumbai that gave Acharya Institute of Technology its Technical Campus status.

Enhanced++ Dependencies:

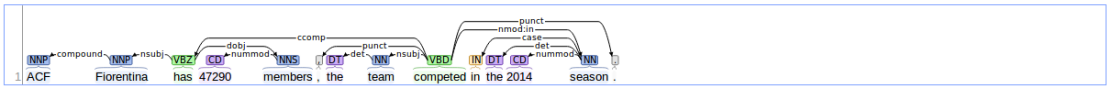



```
objectInDirect = rules.System.Rule(name="indirect_object")
#soit x et y deux verbes
x = objectInDirect.add_node(pos='V.*', inside_allow=False)
y = objectInDirect.add_node(pos='')

objectInDirect.add_link(x, y, 'iobj')

#j'ajoute la règle
self.system.add_rule(objectInDirect)
```

D. 5.9 Juxtaposition

ACF Fiorentina has 47290 members, the team competed in the 2014 season.

Enhanced++ Dependencies:



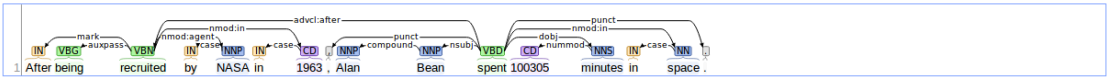

```
# coordonnées
rule_juxtaClauses = rules.System.Rule(name="juxtaposition")
#soit x et y deux verbes
verbe1 = rule_juxtaClauses.add_node(pos='V.*', word=".*(?!:ing)$", inside_allow=False)
verbe2 = rule_juxtaClauses.add_node(pos='V.*', word=".*(?!:ing)$", inside_allow=False)
# une virgule
virgule = rule_juxtaClauses.add_node(word=',')
# un point final
point = rule_juxtaClauses.add_node(word='.')
#un wrb potentiel
wrbpos = rule_juxtaClauses.add_node(inside_allow=False)

# on interdit un lien typé relative clause sur le dernier verbe avec un object entre les
# deux verbes.
rule_juxtaClauses.forbid_link(verbe2, wrbpos, "acl:relcl")
rule_juxtaClauses.set_position(verbe1, wrbpos)
rule_juxtaClauses.set_position(wrbpos, verbe2)

#deux sujets
sujet1= rule_juxtaClauses.add_node(pos="^(?!W).*$")
```

```
sujet2= rule_juxtaClauses.add_node(pos="^(?!W).*$")

rule_juxtaClauses.add_link(verbe1, sujet1, '.subj.*')
rule_juxtaClauses.add_link(verbe2, sujet2, '.subj.*')

#verbe1 est relié à deux puncts --> clôture virgule et clôture point final
rule_juxtaClauses.add_link(verbe1, virgule, "punct")
rule_juxtaClauses.add_link(verbe1, point, "punct")

rule_juxtaClauses.set_position(verbe1, virgule)
rule_juxtaClauses.set_position(sujet1, virgule)
rule_juxtaClauses.set_position(virgule, verbe2)
rule_juxtaClauses.set_position(virgule, sujet2)
rule_juxtaClauses.set_position(sujet1, sujet2)
rule_juxtaClauses.set_position(verbe1, verbe2)

self.system.add_rule(rule_juxtaClauses)
```

D. 5.10 Participle clause (clause participiale)

After being recruited by NASA in 1963, Alan Bean spent 100305 minutes in space.

Enhanced++ Dependencies:

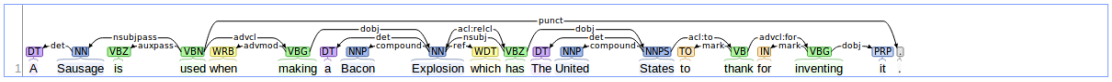



```
ptc = rules.System.Rule(name="participle clause")
# soit x, y et z
x = ptc.add_node(pos='V.*')
y = ptc.add_node(pos='V.*')
z = ptc.add_node(pos='VBG')
ptc.add_link(y, z, 'auxpass')
ptc.add_link(x, y, 'advcl.*')
self.system.add_rule(ptc)
```

Participle clause bis 1

A Sausage is used when making a Bacon Explosion which has The United States to thank for inventing it.

Enhanced++ Dependencies:

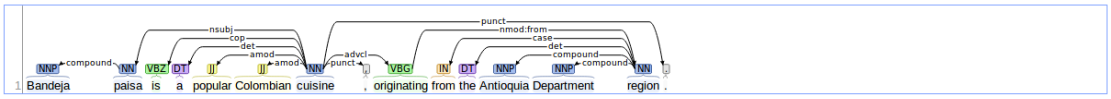



```
ptc = rules.System.Rule(name="participle clause_bis_1")
#soit x, y et z
y = ptc.add_node(pos='V.*', inside_allow=False)
z = ptc.add_node(pos='VBG')

ptc.add_link(y, z, 'advcl')
self.system.add_rule(ptc)
```

Participle clause bis 2

Bionico is a Dessert, requiring Granola as one of its ingredients.

Enhanced++ Dependencies:

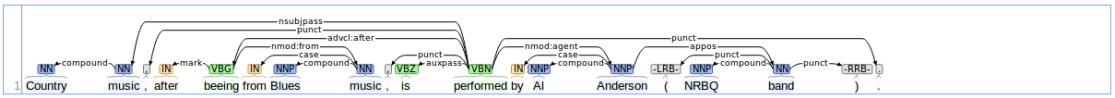



```
ptc = rules.System.Rule(name="participle clause_bis_2")
#soit x, y et z
y = ptc.add_node(pos='V.*', inside_allow=False)
z = ptc.add_node(pos='VBG')
x = ptc.add_node()

ptc.add_link(x, y, 'cop')
ptc.add_link(x, z, 'advcl')
self.system.add_rule(ptc)
```

Participle clause 3

Country music, originating from Blues music, is performed by Al Anderson (NRBQ band).

Enhanced++ Dependencies:



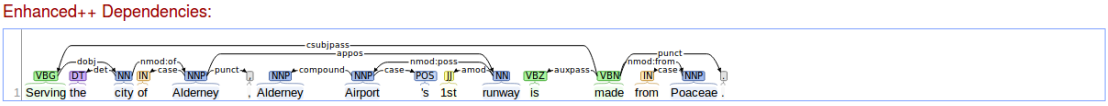

```
ptc = rules.System.Rule(name="participle clause_3")
#soit x, y
x = ptc.add_node()
y = ptc.add_node(pos='VBG')

ptc.add_link(x, y, 'acl.*')
self.system.add_rule(ptc)
```

D. 5.11 Participle clause subject

Serving the city of Alderney, Alderney Airport's 1st runway is made from Poaceae.

Du fait d'un fort taux d'erreur du parseurs, cette analyse n'a pas été conservée pour les travaux.

Enhanced++ Dependencies:

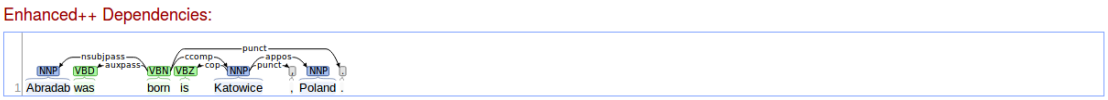


```
ptc = rules.System.Rule(name="participle clause_subject_2_1")
#soit x, y et z
y = ptc.add_node(pos='V.*')
z = ptc.add_node(pos='VBG')

ptc.add_link(y, z, 'csubj.*')
self.system.add_rule(ptc)

ptc = rules.System.Rule(name="participle clause_subject_2_2")
#soit x, y et z
y = ptc.add_node(pos='V.*')
z = ptc.add_node(pos='VBN')
ptc.add_link(y, z, 'csubj.*')
self.system.add_rule(ptc)
```

D. 5.12 Passive voice (voix passive)

La voie passive est simplement détectée par une liaison '.subjpass.*', et permet donc de référencer les voies passives dans des structures subordonnées et simples.

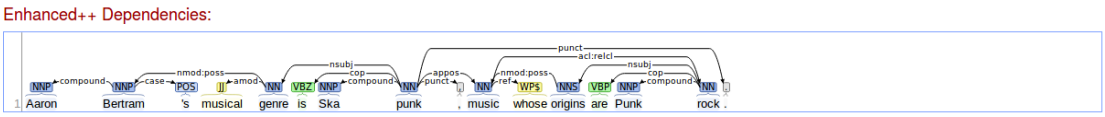
Abradab was born is Katowice, Poland.




```
rule_passive = rules.System.Rule(name="passive_voice")
#soit deux noeuds quelconques
x = rule_passive.add_node()
y = rule_passive.add_node()
#il existe un lien .subjpass unissant deux noeuds
rule_passive.add_link( x, y, '.subjpass.*')
#ajout de la règle au système
self.system.add_rule(rule_passive)
```

D. 5.13 Possessive case (possessif)

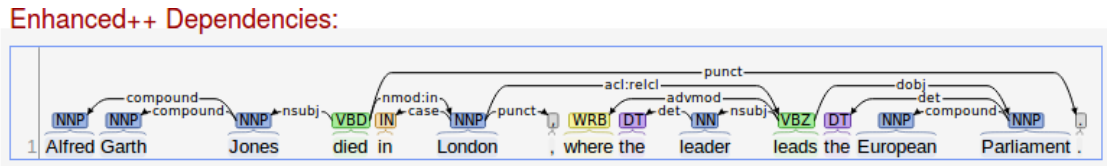
Aaron Bertram's musical genre is Ska punk, music whose origins are Punk rock.




```
case_rule = rules.System.Rule(name="possessif")
#soit deux noeuds quelconques
x = case_rule.add_node()
y = case_rule.add_node()
#il existe un lien csubj.* unissant deux noeuds
case_rule.add_link( x, y, 'nmod:poss')
#ajout de la règle au système
self.system.add_rule(case_rule)
```

D. 5.14 Relative adverb (relative adverbiale)

Pour la présence d'une relative adverbiale, nous nous reposons exclusivement sur la présence d'adverbe modifiant x, sachant que x est dans une relation clausale (`acl:relcl`) avec un nœud z.




```
#relative adverbs
rule_relativeAdverbs = rules.System.Rule(name="relative_adverb")
#soit x un noeud pos_taggué *
x = rule_relativeAdverbs.add_node()
#soit y un noeud quelconque
y = rule_relativeAdverbs.add_node()
#soit l1 le lien unissant faisant x modifié par y, de type advmod
l1 = rule_relativeAdverbs.add_link(x, y, 'advmod')

#soit z un noeud quelconque
#z un noeud qui sera dans une relation de relcl avec ce que l'adverbe modifie
z = rule_relativeAdverbs.add_node()
# soit l2 le lien unissant z à x, de nature 'acl:relcl'
l2 = rule_relativeAdverbs.add_link(z, x, 'acl:relcl')

#sachant que z n'est pas le sujet de x
l3 = rule_relativeAdverbs.forbid_link(x, z, 'nsubj.*')

self.system.add_rule(rule_relativeAdverbs)
```

D. 5.15 Relative object (relative objet)

Ahmet Ertegun is a Rock and roll music performer, whose origins came from Country music.

Pour détecter une relative objet, nous recherchons les nœuds x qui ont un sujet z.

```
rule_relativeObject.add_link(x, z, 'nsubj.*')
#mcela peut être un verbe ou un nom dans le cas des existencielles.
```

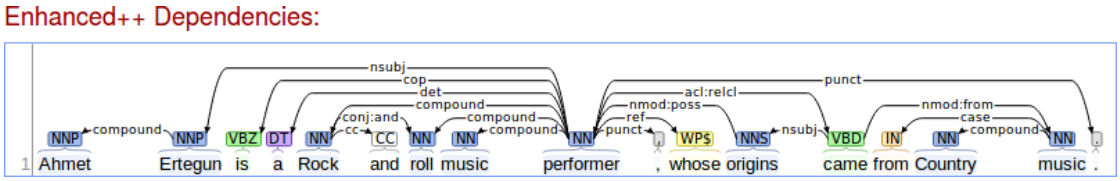
z ne doit pas être un 'W.*'. Il a donc été défini par (cf assertions négatives, D. 4.2) :

```
z = rule_relativeObject.add_node(pos="-W.*")
```

y est lui un nœud quelconque lié à x par une relation`acl:relcl` :

```
y = rule_relativeObject.add_node()
rule_relativeObject.add_link(x, y, 'acl:relcl')
```

La règle complète est un peu plus fine, et permet de détecter les structures relatives objet.




```
#relative object
rule_relativeObject = rules.System.Rule(name="relative_object")
#soit x un noeud pos_taggue V.*
x = rule_relativeObject.add_node()
#soit y un noeud quelconque vers lequel x sera lié dans le cadre d'une relative clause
y = rule_relativeObject.add_node()
#soit z un noeud quelconque dont x sera le sujet, mais qui ne sera pas y
z = rule_relativeObject.add_node(pos="-W.*")
#soit l1 le lien unissant y à x, de nature 'acl:relcl', ce lien n'unissant pas x et z
l1 = rule_relativeObject.add_link(x, y, 'acl:relcl')
l1 = rule_relativeObject.forbid_link(x, z, 'acl:relcl')
#soit l2 le lien unissant x à y, de nature 'dobj' : x a pour objet y
l2 = rule_relativeObject.add_link(x, z, 'nsubj.*')
l1 = rule_relativeObject.forbid_link(y, x, 'nsubj.*')

self.system.add_rule(rule_relativeObject)
```

D. 5.16 Relative subject (relative sujet)

Diverses possibilités se profilaient pour trouver les relatives sujet. Nous utilisons le stanford parser, et savons qu'il est définir sur des bases statistiques des clauses. Deux phrases-type nous ont questionnés.

Alfred Garth Jones died in London which is lead by the European Parliament. et *Aaron Bertram, who plays Ska punk, is an artist with the band Kids Imagine Nation.*

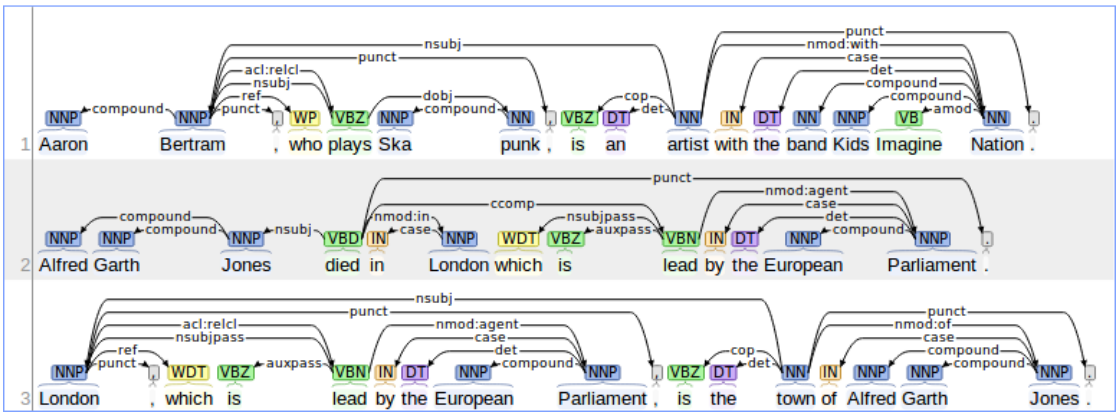


FIGURE D. 11 – Détection des `acl:relcl` (*relative clause modifier*) par le stanford parser

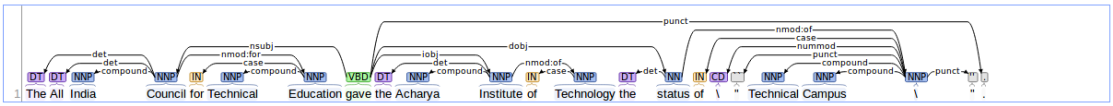
Il nous apparaît que l'absence du tag *relative clause modifier* n'est pas suffisante pour déclarer l'absence de relative sujet.
étant donné que nous n'avons que des phrases affirmatives, nous avons pris le pari que tout mot taggué 'W.*' (c.f Annexes F) sujet d'un verbe témoignait d'une relative sujet. Nous avons associé à cette règle la suivante : tout 'WP' non sujet d'un verbe mais lié à quelque chose en tant que réf qui est lui sujet d'un verbe et objet pour ce verbe d'une relation "acl :relcl" témoigne d'une relation relative objet. Nous avons mis une contrainte de position dans le cadre de la description, mais celle-ci pourrait être retirée. Cela réduit la liste des nœuds aux relations testées, donc l'intérêt du retrait de la contrainte n'est pas évident.

```
rule_relativeSubject = self.system.create_rule("relative_subject")
x = rule_relativeSubject.add_node(pos="WP", inside_allow=False)
y = rule_relativeSubject.add_node(pos="V.*", inside_allow=False)
z = rule_relativeSubject.add_node()
l1 = rule_relativeSubject.forbid_link(y, x, 'nsubj.*', "")
l1 = rule_relativeSubject.add_link(z, x, 'nsubj.*')
l1 = rule_relativeSubject.add_link(x, z, 'acl:relcl')
l2 = rule_relativeSubject.add_link(x, y, 'ref')
rule_relativeSubject.set_position(x, y)
rule_relativeSubject.set_position(y, z)

rule_relativeSubject = enhanced_system.create_rule("relative_subject_2")
x = rule_relativeSubject.add_node(pos="W.*", inside_allow=False)
y = rule_relativeSubject.add_node(pos="")
z = rule_relativeSubject.add_node(pos="")

rule_relativeSubject.add_link(z, y, "nsubj.*")
rule_relativeSubject.add_link(y, z, "acl:relcl")
rule_relativeSubject.add_link(y, x, "ref")

rule_relativeSubject.set_position(y, x)
rule_relativeSubject.set_position(x, z)
self.system.add_rule(rule_relativeSubject)
```

D. 5.17 Twice objects/Ditransitive clause (COD et COI pour un même verbe)

Le corpus en offre très peu...
The All India Council for Technical Education gave the Acharya Institute of Technology the status of T̈echnical Campus.

Enhanced++ Dependencies:




```
twiceRule = rules.System.Rule(name="twice_objects")
#soit x et y deux verbes
```

```
x = twiceRule.add_node(pos='V.*', inside_allow=False)
y = twiceRule.add_node(pos='')
z = twiceRule.add_node(pos='')

twiceRule.add_link(x, y, 'iobj')
twiceRule.add_link(x, z, 'dobj')

#j'ajoute la règle
self.system.add_rule(twiceRule)
```

D. 6 Statistiques corpus

D. 6.1 Répartition des contraintes selon modèle

Corpus	Base	$T2T_{syn}$	TX_{syn}	$D2T_{syn}$	TR_{syn}	$D2T_{5best}$	$T2T_{5best}$	ALL_{syn}
# textes en entrée	32,503	18,904	22,386	4,099	9,968	4,106	26,621	3,266
# textes en sortie	0	8,703	6,758	26,403	8,063	27,144	26,621	1,997
Constraintes utilisées								
Juxtaposition	299	13	100	294	136	NA	NA	35
VP Coordination	4,085	287	894	3,837	857	NA	NA	186
Sent. Coordination	2,890	162	622	2,751	669	NA	NA	105
Predicative Clause	25,429	4,023	1,609	18,352	1,304	NA	NA	480
Passive Voice	14,523	1,745	1,509	11,786	1,527	NA	NA	296
Possessive Cases	5,199	232	704	4,549	887	NA	NA	138
Apposition	4,669	401	1,206	4,390	1,178	NA	NA	223
Adverb Relative	3,166	159	494	2,991	491	NA	NA	179
Object Relative	1,433	48	259	1,394	385	NA	NA	118
Subject Relative	6,712	351	1,097	6,279	1,543	NA	NA	216
Transitive Clause	11,889	1,708	1,408	9,938	1,281	NA	NA	451
Intransitive Clause	9	0	0	9	0	NA	NA	0
Ditransitive Clause	8	0	0	8	0	NA	NA	0

D. 6.2 Exemples de sorties

TASK : $T2T_{5best}$

M **: { (American submarine NR-1 , builder , General Dynamics Electric Boat),**
(General Dynamics Electric Boat , locationCity , Groton , Connecticut) }

1. General Dynamics Electric Boat is located in Groton, Connecticut and built the American submarine NR-1.

2. The American submarine NR-1 was built by General Dynamics Electric Boat in Groton, Connecticut.

3. General Dynamics Electric Boat, located in Groton, Connecticut, is the builder of the American submarine NR-1.

4. The American submarine NR-1 was built on the General Dynamics Electric Boat, the location of which, is in Groton, Connecticut.

5. The American submarine NR-1 was built by General Dynamics Electric Boat, the location of which, is in Groton, Connecticut.

M **:** { **(AC Hotel Bella Sky Copenhagen , developer , Bella Center), (AC Hotel Bella Sky Copenhagen , numberOfRooms , 814)** }

1. The AC Hotel Bella Sky Copenhagen has 814 rooms and was designed by the Bella Center.

2. The AC Hotel Bella Sky Copenhagen has 814 rooms and was developed by the Bella Center.

3. The AC Hotel Bella Sky Copenhagen has 814 rooms and was given the "Technical Campus ".

4. The AC Hotel Bella Sky Copenhagen has 814 rooms and was built in the Year Bella Center.

M **:** { **(Adare Manor , location , Adare), (Adare Manor , location , County Limerick)** }

1. Adare Manor is located in Adare, County Limerick.

2. Adare Manor is in County Limerick, Adare.

3. Adare Manor is in Adare, County Limerick.

4. Adare Manor is found in Adare, County Limerick.

5. Adare Manor is located in County Limerick, Adare.

M **:** { **(1634 : The Baltic War , author , David Weber), (1634 : The Bavarian Crisis , precededBy , 1634 : The Baltic War)** }

1. '1634 : The Bavarian Crisis' was preceded by '1634 : The Baltic War' which was written by David Weber. 1634 : The Bavarian Crisis was preceded By 1634 : The Baltic War written by David Weber.

M **:** { **(1634 : The Ram Rebellion , author , Eric Flint), (Eric Flint , influencedBy , Robert A. Heinlein)** }

1. Eric Flint, author of 1634 : The Ram Rebellion, was influenced by Robert A. Heinlein.

2. Eric Flint, influenced by Robert A. Heinlein, is the author of 1634 : The Ram Rebellion.

3. Eric Flint was influenced by Robert A. Heinlein and wrote 1634 : The Ram Rebellion.

4. Eric Flint is the author of 1634 : The Ram Rebellion, which was influenced by Robert A. Heinlein.

5. Eric Flint is the author of 1634 : The Ram Rebellion which was influenced by Robert A. Heinlein.

TASK : D2T$_{5best}$

M **: { (Aaron Bertram , associatedBand , associatedMusicalArtist , Kids Imagine Nation), (Aaron Bertram , associatedBand , associatedMusicalArtist , Suburban Legends)** }

1. Aaron Bertram is associated with the musical artists Suburban Legends and Kids Imagine Nation.

2. Aaron Bertram is associated with Kids Imagine Nation and Suburban Legends.

M **: { (1634 : The Ram Rebellion , language , English language), (English language , spokenIn , Great Britain)** }

1. English language is spoken in Great Britain and is the language of Great Britain.

2. English language is the language of both Great Britain and Great Britain.

3. English language is spoken in Great Britain and is the language spoken in Great Britain.

4. English language is spoken in Great Britain and is the language of the Great Britain.

5. English language is spoken in Great Britain and is the language used in Great Britain.

M **: { (66391 1999 KW4 , apoapsis , 162164091.8388 kilometres), (66391 1999 KW4 , epoch , 2004-07-14)** }

1. 66391 1999 KW4 has an apoapsis of 162164091.8388 kilometres and an epoch 2004-07-14.

2. 66391 1999 KW4 has an epoch 2004-07-14 and its apoapsis is 162164091.8388 kilometres.

3. The epoch of 66391 1999 KW4 is 2004-07-14 and it has an apoapsis of 162164091.8388 kilometres.

4. 66391 1999 KW4 has an epoch 2004-07-14 and an apoapsis of 162164091.8388 kilometres.

5. 66391 1999 KW4 has an apoapsis of 162164091.8388 kilometres and its epoch is 2004-07-14.

M **:** { **(James Craig Watson , deathPlace , Madison , Wisconsin), (101 Helena , discoverer , James Craig Watson)** }

1. James Craig Watson, who died in Madison, Wisconsin, discovered 101 Helena.
2. 101 Helena was discovered by James Craig Watson, who died in Madison, Wisconsin.
3. 101 Helena was discovered by James Craig Watson who died in Madison, Wisconsin.
4. James Craig Watson, who discovered 101 Helena, later died in Madison, Wisconsin.
5. James Craig Watson, who died in Madison, Wisconsin, was the discoverer of 101 Helena.

M **:** { **(Alan Bean , birthPlace , Wheeler , Texas), (Alan Bean , occupation , Test pilot)** }

1. Alan Bean was born in Wheeler, Texas and served as a Test pilot.
2. Alan Bean, who was born in Wheeler, Texas, served as a Test pilot.
3. Alan Bean was born in Wheeler, Texas and is a Test pilot.
4. Alan Bean, a Test pilot, was born in Wheeler, Texas.
5. Alan Bean was born in the Wheeler, Texas and served as a Test pilot.

TASK : TR$_{syn}$

Mg **: { (floorCountSujet1 , floorCount , 23), (floorCountSujet1 , location , Birmingham)** }

t : 103_Colmore_Row/completionDate/1976
k : transitive
T_{in} : 103 Colmore Row, located in Birmingham, with 23 floors, was completed in 1976.
T_{out} : 103 Colmore Row, completed in 1976, has 23 floors.

t : 103_Colmore_Row/completionDate/1976
k : apposition
T_{in} : The building of 103 Colmore Row completed in 1976 is located in Birmingham and has 23 floors.
T_{out} : 103 Colmore Row, Birmingham, has a floor count of 23.

Mg **: { (Abel Caballero , birthPlace , Galicia (Spain)),**
(Abel Caballero , birthPlace , Spain) }

t : Abel_Caballero/party/Spanish_Socialist_Workers'_Party
k : apposition
T_{in} : Abel Caballero was born in Galicia (Spain) in Spain and is a member of the Spanish Socialist Workers' Party.
T_{out} : Abel Caballero was born in Galicia (Spain), Spain.

t : Abel_Caballero/party/Spanish_Socialist_Workers'_Party
k : passive voice
T_{in} : Abel Caballero, born in Galicia (Spain) (Spain), is a member of the Spanish Socialist Workers' Party.
T_{out} : Abel Caballero was born in Galicia (Spain), Spain.

t : Abel_Caballero/party/Spanish_Socialist_Workers'_Party
k : relative subject
T_{in} : Abel Caballero was born in Galicia (Spain), Spain and was a member of the Spanish Socialist Workers' Party Party there.
T_{out} : Abel Caballero was born in Galicia (Spain) which is in the Spain.

Mg **: { (Alan Shepard , birthPlace , New Hampshire),**
(Alan Shepard , deathPlace , California), (Alan Shepard , nationality , United States) }

t : Alan_Shepard/dateOfRetirement/"1974-08-01"
k : passive voice
T_{in} : Alan Shepard (born in New Hampshire of the United States) retired on "1974-08-01" and died in California.
T_{out} : Alan Shepard was born in New Hampshire and died in California.

Mg **:** { **(Bananaman , broadcastedBy , BBC), (Bananaman , firstAired , "1983-10-03"), (Bananaman , starring , Bill Oddie)** }

t : Bananaman/creator/Steve_Bright
k : relative subject
T_{in} : Bananaman, created by Steve Bright and starring Bill Oddie, was first broadcast by the BBC on the "1983-10-03".
T_{out} : Bananaman, which starred Bill Oddie, was broadcasted by the BBC.

Mg **:** { **(Aaron Boogaard , birthPlace , birthPlaceObjet1), (birthPlaceObjet1 , ethnicGroup , Asian Canadians), (birthPlaceObjet1 , leaderName , Elizabeth II)** }

t : Canada/anthem/O_Canada
k : apposition
T_{in} : O Canada is the national anthem of Canada, where Aaron Boogaard was born. Elizabeth II is the leader of Canada, where one of the ethnic groups, is Asian Canadians.
T_{out} : Aaron Boogaard was born in Canada, the country where Asian Canadians are an ethnic group.

TASK : TX$_{syn}$

Mg **:** { **(Anaheim , California , UTCOffset , "-8"), (Anaheim , California , areaCode , 657, 714), (Anaheim , California , areaTotal , 131.6 square kilometres)** }

t : Anaheim_,_California/UTCOffset/"-8"
k : juxtaposition
T_{in} : Anaheim, California has a total area of 131.6 square kilometres and the area codes are 657, 714.

T_{out} : Anaheim, California covers an area of 131.6 square kilometres, has the area codes 657, 714 and has a UTC offset of "-8".

t : Anaheim_,_California/UTCOffset/"-8"
k : predicative
T_{in} : Anaheim, California has a total area of 131.6 square kilometres and the area code 657, 714.
T_{out} : Anaheim, California has the area code : 657, 714 and a total area of 131.6 square kilometres. The UTC offset is "-8".

Mg **:** { **(Weymouth Sands , author , John Cowper Powys), (John Cowper Powys , birthPlace , Shirley , Derbyshire), (Weymouth Sands , followedBy , Maiden Castle (novel)), (A Glastonbury Romance , followedBy , Weymouth Sands)** }

t : Weymouth_Sands/followedBy/Maiden_Castle_(novel)
k : predicative
T_{in} : Weymouth Sands was written by Shirley, Derbyshire native John Cowper Powys.
T_{out} : John Cowper Powys was born in Shirley, Derbyshire and is the author of Weymouth Sands. The book is a sequel to A Glastonbury Romance and is followed by Maiden Castle (novel).

Mg **:** { **(Rolando Maran , club , Carrarese Calcio), (AC Chievo Verona , manager , Rolando Maran), (Rolando Maran , placeOfBirth , Italy)** }

t : Rolando_Maran/placeOfBirth/Italy
k : transitive
T_{in} : Rolando Maran has been the manager of AC Chievo Verona and is part of the club Carrarese Calcio.
T_{out} : Rolando Maran (born in Italy), has managed AC Chievo Verona and played for Carrarese Calcio.

t : Rolando_Maran/placeOfBirth/Italy
k : passive voice
T_{in} : Rolando Maran, who played for Carrarese Calcio has been the manager of AC Chievo Verona.
T_{out} : AC Chievo Verona is managed by Rolando Maran, who was born in Italy and is in the Carrarese Calcio club.

t : Rolando_Maran/placeOfBirth/Italy
k : predicative
T_{in} : AC Chievo Verona is managed by Rolando Maran who worked at Carrarese Calcio.
T_{out} : AC Chievo Verona is managed by Rolando Maran, who was born in Italy and is in the Carrarese Calcio club.

t : Rolando_Maran/placeOfBirth/Italy
k : coordinated clauses
T_{in} : Rolando Maran plays at Carrarese Calcio. He also manages AC Chievo Verona.
T_{out} : Rolando Maran (born in Italy), has managed AC Chievo Verona and played for Carrarese Calcio.

Mg **: { (Serie B , champions , Carpi FC 1909), (AC Cesena , ground , Cesena), (AC Cesena , league , Serie B)** }

t : Serie_B/champions/Carpi_FC_1909
k : transitive
T_{in} : AC Cesena, in the Serie B league, ground is in Cesena.
T_{out} : AC Cesena is in the Serie B league (with Carpi FC 1909 as champions) and has a ground in Cesena.

t : Serie_B/champions/Carpi_FC_1909
k : relative subject
T_{in} : AC Cesena play in Serie B and their ground is in Cesena.
T_{out} : Cesena is the ground of AC Cesena who play in the Serie B which Carpi FC 1909 have been champions of.

t : Serie_B/champions/Carpi_FC_1909
k : predicative
T_{in} : AC Cesena play at Cesena in Serie B.
T_{out} : With grounds at the Cesena, AC Cesena play in Serie B. Carpi FC 1909 are previous champions of that league.

t : Serie_B/champions/Carpi_FC_1909
k : coordinated full clauses
T_{in} : AC Cesena play in a league called the Serie B and their home ground is the Cesena.
T_{out} : AC Cesena is in the Serie B (previous champions : Carpi FC 1909) and their home ground is The Cesena.

t : Serie_B/champions/Carpi_FC_1909
k : relative subject

T_{in} : AC Cesena play in a league called the Serie B and their home ground is the Cesena.
T_{out} : AC Cesena who play in the Serie B, won by Carpi FC 1909, have their home ground at Cesena.

t : Serie_B/champions/Carpi_FC_1909
k : relative adverb
T_{in} : AC Cesena ground is in Cesena and the club plays in Serie B.
T_{out} : AC Cesena 's ground is in Cesena and they played in Serie B where the champions were Carpi FC 1909.

t : Serie_B/champions/Carpi_FC_1909
k : juxtaposition
T_{in} : AC Cesena 's home ground is the Cesena and the league they play in is Serie B.
T_{out} : Carpi FC 1909 are former champions of Serie B. AC Cesena also play in the same league, their ground is in Cesena.

t : Serie_B/champions/Carpi_FC_1909
k : coordinated full clauses
T_{in} : The Cesena, is the home ground of AC Cesena which is in the Serie B.
T_{out} : The ground for AC Cesena is the Cesena and they compete in the Serie B. Previous champions of the superleague are Carpi FC 1909.

t : Serie_B/champions/Carpi_FC_1909
k : relative object
T_{in} : AC Cesena compete in the Serie B and ground is in Cesena.
T_{out} : The home ground of AC Cesena is Cesena, they are in the Serie B league, of which Carpi FC 1909 have been champions.

t : AC_Cesena/ground/Cesena
k : possessive
T_{in} : AC Cesena are in the Serie B league, the previous champions of which are Carpi FC 1909.
T_{out} : AC Cesena play at their ground Cesena in the Serie B league. The champions of this league are Carpi FC 1909.

t : AC_Cesena/ground/Cesena
k : relative subject
T_{in} : AC Cesena play in the Serie B, which Carpi FC 1909 were once the champions of.
T_{out} : AC Cesena who 's grounds are in Cesena, play in the Serie B, the previous champions of which were Carpi FC 1909.

t : AC_Cesena/ground/Cesena
k : passive voice

T_{in} : AC Cesena play in Serie B where Carpi FC 1909 have been champions.
T_{out} : The AC Cesena team is based in Cesena and play in the Serie B where Carpi FC 1909 were previous champions.

TASK : T2T$_{syn}$

Mg **:** { **(Bandeja paisa , ingredient , Avocado), (Avocado , order , Laurales)** }
k : apposition
T_{in} : Avocado are classified in the order Laurales and are an ingredient in Bandeja paisa.
T_{out} : The Avocado serves as an ingredient in Bandeja paisa, a traditional dish from the order of Laurales.

Mg **:** { **(Bananaman , broadcastedBy , BBC), (BBC , locationCity , Broadcasting House)** }
k : relative subject
T_{in} : Bananaman the TV series was shown on the BBC, the BBC is headquartered in Broadcasting House.
T_{out} : The BBC broadcasted Bananaman which stars in Broadcasting House.

k : apposition
T_{in} : Bananaman is broadcast by the BBC, which is based in Broadcasting House.
T_{out} : Located in Broadcasting House, the BBC, broadcast Bananaman.

Mg **:** { **(Italy national under-16 football team , coach , Daniele Zoratto)** }
k : predicative
T_{in} : The coach of the Italy national under-16 football team is Daniele Zoratto.
T_{out} : Daniele Zoratto is coach for the Italy national under-16 football team.

Mg **:** { **(AIP Advances , editor , A.T. Charlie Johnson)** }
k : passive voice
T_{in} : The editors of AIP Advances are A.T. Charlie Johnson.
T_{out} : AIP Advances was edited by A.T. Charlie Johnson.

Mg **:** { **(Bionico , country , Mexico), (Bionico , course , Dessert), (Dessert , dishVariation , Cookie), (Mexico , language , Spanish language), (Mexico , leaderName , Silvano Aureoles Conejo)** }

k : relative adverb

L

T_{in} : Bionico, a Dessert, can be found in Mexico. The name of the leader of Mexico is Silvano Aureoles Conejo and the language spoken there is Spanish language. Another confectionery Dessert is Cookie.
T_{out} : Cookie is a nice confectionery Dessert, as is Bionico. Bionico comes from Mexico, where Silvano Aureoles Conejo is a leader.

Mg **: { (Illinois , country , United States), (Chicago , isPartOf , Illinois), (Chicago , leaderName , Rahm Emanuel), (300 North LaSalle , location , Chicago)** }

k : relative adverb
T_{in} : 300 North LaSalle is located in Chicago, Illinois, United States. The city leader is Rahm Emanuel.
T_{out} : Rahm Emanuel is the leader of Chicago, United States where the city is 300 North LaSalle. Illinois is located in the country.

TASK : D2T$_{syn}$

Mg **:** { **(Airey Neave , activeYearsEndDate , 1979-03-30), (Airey Neave , battles , Battle of France)** }

k : coordinated full clauses
T_{out} : Airey Neave fought in the Battle of France and he ended his career 1979-03-30.

Mg **:** { **(Abilene Regional Airport , cityServed , Abilene , Texas), (Abilene , Texas , country , United States)** }

k : transitive
T_{out} : Abilene Regional Airport serves the city of Abilene, Texas which is in United States.

Mg **:** { **(James Craig Watson , almaMater , University of Michigan), (James Craig Watson , deathCause , Peritonitis), (103 Hera , discoverer , James Craig Watson)** }

k : apposition
T_{out} : 103 Hera, was discovered by James Craig Watson, who studied at the University of Michigan and died from Peritonitis.

Mg **:** { **(AIDS (journal) , country , United Kingdom), (Lippincott Williams & Wilkins , parentCompany , Wolters Kluwer), (AIDS (journal) , publisher , Lippincott Williams & Wilkins)** }

k : relative object
T_{out} : AIDS (journal) is published in the United Kingdom by Lippincott Williams & Wilkins, whose parent company is Wolters Kluwer.

k : apposition
T_{out} : AIDS (journal) is published in the United Kingdom by Lippincott Williams & Wilkins, the parent company of which, is Wolters Kluwer.

Mg **:** { **(AWH Engineering College , country , India), (Kerala , has to its northwest , Mahé, India), (AWH Engineering College , state , Kerala)** }

k : predicative
T_{out} : AWH Engineering College is located in Kerala, India. To the west of Kerala is Mahe, India.

Mg **:** { **(Abarth 1000 GT Coupé , bodyStyle , "Two door coupé"), (Abarth 1000 GT Coupé , engine , Straight-four engine), (Abarth 1000 GT Coupé , wheelbase , 2160.0 millimetres)** }

k : transitive
T_{out} : The Abarth 1000 GT Coupe has a Straight-four engine, a "Two door coupe" and a wheelbase of 2160.0 millimetres.

Mg **:** { **(Andrew White (musician) , genre , Alternative rock), (Polydor Records , location , London), (Alternative rock , musicFusionGenre , Nu metal), (Andrew White (musician) , recordLabel , Polydor Records), (Alternative rock , stylisticOrigin , Punk rock)** }

k : transitive
T_{out} : Andrew White (musician) is an Alternative rock musician signed to Polydor Records in London. Nu metal music is a fusion genre of Alternative rock which has its stylistic origins in Punk rock. Nu metal is a fusion of Alternative rock.

TASK : all$_{syn}$

Mg **: { (Aaron Bertram , associatedBand , associatedMusicalArtist , Suburban Legends), (Aaron Bertram , genre , Ska punk), (Ska punk , stylisticOrigin , Ska)** }

k : predicative
T_{in} : Ska punk originates from Ska and is favored by Aaron Bertram, musician. Aaron Bertram plays for the Suburban Legends band.
T_{out} : Aaron Bertram performs for Suburban Legends in the genre of Ska punk. Ska punk originated from Ska.

k : predicative
T_{in} : Ska punk (Origin : Ska) is the genre of Aaron Bertram, who is a member of Suburban Legends band.
T_{out} : Ska punk, originated from Ska and is the musical genre of Aaron Bertram.

t : Ska_punk/stylisticOrigin/Ska
k : transitive
T_{in} : Aaron Bertram is a Ska punk musician with the Suburban Legends band.
T_{out} : Aaron Bertram plays Ska punk music for the Suburban Legends Band. Ska punk 's origin is Ska.

t : Ska_punk/stylisticOrigin/Ska
k : apposition
T_{in} : Aaron Bertram, who plays Ska punk, is an artist with the band Suburban Legends.
T_{out} : Ska punk (Origin : Ska) is the genre of Aaron Bertram, who is a member of Suburban Legends band.

t : Ska_punk/stylisticOrigin/Ska
k : predicative
T_{in} : Aaron Bertram performs Ska punk music and played with Suburban Legends.
T_{out} : Aaron Bertram plays Ska punk music for the Suburban Legends Band. Ska punk 's origin is Ska.

t : Aaron_Bertram/associatedBand/associatedMusicalArtist/Suburban_Legends
k : apposition
T_{in} : The musical genre of Aaron Bertram is Ska punk which has its origins in Ska.
T_{out} : Aaron Bertram, an artist with the band Suburban Legends, performs Ska punk music, which has its origins in Ska.

t : Aaron_Bertram/associatedBand/associatedMusicalArtist/Suburban_Legends
k : relative subject
T_{in} : Aaron Bertram plays Ska punk music, the stylistic origins of which, come from Ska.

T_{out} : Aaron Bertram, who plays Ska punk, which emerged from Ska, plays for the Suburban Legends band.

t : Aaron_Bertram/associatedBand/associatedMusicalArtist/Suburban_Legends
k : predicative
T_{in} : Aaron Bertram of NRBQ plays Ska punk which originated from Ska music.
T_{out} : Ska punk (Origin : Ska) is the genre of Aaron Bertram, who is a member of Suburban Legends band.

t : Aaron_Bertram/associatedBand/associatedMusicalArtist/Suburban_Legends
k : coordinated clauses
T_{in} : Aaron Bertram is a member of the band NRBQ that performs Ska punk, which has its stylistic origins in the Ska.
T_{out} : Aaron Bertram plays for the Suburban Legends band and performs Ska punk music which has its origins in Ska.

t : Aaron_Bertram/associatedBand/associatedMusicalArtist/Suburban_Legends
k : possessive
T_{in} : Aaron Bertram performs Ska punk music, a style that originated from Ska.
T_{out} : Aaron Bertram, an artist with the band Suburban Legends, performs Ska punk music, which has its origins in Ska.

Mg **:** { **(Aaron Turner , associatedBand , associatedMusicalArtist , Sumac (band)), (Aaron Turner , genre , Black metal), (Black metal , musicFusionGenre , Death metal)** }

k : passive voice
T_{in} : Aaron Turner who is associated with the group Sumac (band) plays Black metal music. Death metal is a musical fusion of Black metal.
T_{out} : Black metal music Aaron Turner performs with Sumac (band). Death metal is fused with Black metal.

k : predicative
T_{in} : Aaron Turner is an artist for Sumac (band) and Black metal is his music genre. Death metal is a musical fusion of Black metal.
T_{out} : Aaron Turner performed with Sumac (band) and is a Black metal musician. Death metal is a musical fusion of Black metal.

k : predicative
T_{in} : Death metal is part of the fusion genre, partly coming from Black metal which Aaron Turner played once with Sumac (band).

T_{out} : Aaron Turner performs for Sumac (band) in the genre of Black metal. Death metal is a musical fusion of Black metal.

t : Black_metal/musicFusionGenre/Death_metal
k : juxtaposition
T_{in} : Sumac (band) 'Aaron Turner is a Black metal musician.
T_{out} : Aaron Turner performed with Sumac (band), his musical style being Black metal ; a musical fusion of Black metal is Death metal.

t : Black_metal/musicFusionGenre/Death_metal
k : coordinated full clauses
T_{in} : Aaron Turner plays for the Sumac (band) band and his genre is Black metal.
T_{out} : Aaron Turner (an associate of the musician Sumac (band)), is a Black metal artiste and Death metal is a fusion genre.

t : Black_metal/musicFusionGenre/Death_metal
k : possessive
T_{in} : Aaron Turner is a musician in Sumac (band) and uses the musical genre Black metal.
T_{out} : Aaron Turner performed for Sumac (band) and his music genre is Black metal. Death metal is a musical fusion of Black metal.

t : Black_metal/musicFusionGenre/Death_metal
k : predicative
T_{in} : Aaron Turner performs Black metal music and played with Sumac (band).
T_{out} : Associated with the group Sumac (band), Aaron Turner, is a Black metal musician. Death metal is a musical fusion of Black metal.

t : Black_metal/musicFusionGenre/Death_metal
k : transitive
T_{in} : Aaron Turner plays with the band Sumac (band) and his genre is Black metal.
T_{out} : Death metal is a musical fusion of Black metal which Aaron Turner performed for Sumac (band).

t : Black_metal/musicFusionGenre/Death_metal
k : relative subject
T_{in} : Aaron Turner plays with the band Sumac (band) and performs Black metal.
T_{out} : Aaron Turner plays with Sumac (band) and performs Black metal music which helped create Death metal music.

t : Black_metal/musicFusionGenre/Death_metal
k : relative subject
T_{in} : Aaron Turner plays with the band Sumac (band) and their genre is Black metal.

T_{out} : Aaron Turner is a Black metal musician who played with Sumac (band). Death metal is a musical fusion of Black metal.

t : Black_metal/musicFusionGenre/Death_metal
k : relative object
T_{in} : Aaron Turner performs Black metal music and is a singer associated with Sumac (band).
T_{out} : Aaron Turner plays with the band Sumac (band), in the Black metal genre, a musical fusion of which is Death metal.

t : Aaron_Turner/associatedBand/associatedMusicalArtist/Sumac_(band)
k : relative object
T_{in} : Aaron Turner is a performer in Black metal music. A musical fusion of Black metal is called Death metal.
T_{out} : Aaron Turner is a Black metal musician, whose one fusion is the Death metal and he is a member of the Sumac (band).

t : Aaron_Turner/associatedBand/associatedMusicalArtist/Sumac_(band)
k : coordinated full clauses
T_{in} : Aaron Turner is a performer in Black metal music. A musical fusion of Black metal is called Death metal.
T_{out} : The band Sumac (band) includes the member Aaron Turner and they play Black metal. Death metal is a musical fusion of rock.

t : Aaron_Turner/associatedBand/associatedMusicalArtist/Sumac_(band)
k : passive voice
T_{in} : Death metal is a fusion of Black metal music which is the genre of Aaron Turner..
T_{out} : Death metal music is a fusion genre of Black metal which is the musical genre of Aaron Turner who is associated with the musical artist Sumac (band).

t : Aaron_Turner/associatedBand/associatedMusicalArtist/Sumac_(band)
k : coordinated clauses
T_{in} : Aaron Turner is an exponent of Black metal, which when part of fusion genre, is what Death metal music is.
T_{out} : Aaron Turner performs Black metal music and played with Sumac (band). Death metal is a musical fusion of Black metal.

k : predicative
T_{out} : Aaron Turner, a member of the band Sumac (band), performs Black metal. Death metal is a musical fusion of Black metal.

k : apposition
T_{out} : Aaron Turner, a member of the band Sumac (band), performs Black metal. Death

metal is a musical fusion of Black metal.

t : Aaron_Turner/instrument/instrumentObjet1
k : apposition
T_{in} : Aaron Turner is a Black metal musician and plays instrumentObjet1 for the Sumac (band). Death metal is a musical fusion of Black metal.
T_{out} : Aaron Turner, a musician in Sumac (band), performs Black metal music. Death metal is a musical fusion of Black metal.
$T_{initial}$: Aaron Turner, a musician in Mamiffer, performs Black metal music. Death metal is a musical fusion of Black metal.

$TM_{initial}$:associatedBandassociatedMusicalArtistSujet1/associatedBand/
associatedMusicalArtist/associatedBandassociatedMusicalArtistObjet1
associatedBandassociatedMusicalArtistSujet1/genre/genreObjet1
associatedBandassociatedMusicalArtistSujet1/instrument/instrumentObjet1
genreObjet1/musicFusionGenre/musicFusionGenreObjet1

t : Aaron_Turner/instrument/instrumentObjet1
k : predicative
T_{in} : Aaron Turner plays instrumentObjet1 and performed Black metal music for Sumac (band). A fusion of Black metal music is called Death metal.
T_{out} : Aaron Turner is a Black metal musician with the Sumac (band) band.
$T_{initial}$: Aaron Turner is a Black metal musician with the House of Low Culture band.

$TM_{initial}$:associatedBandassociatedMusicalArtistSujet1/associatedBand/
associatedMusicalArtist/associatedBandassociatedMusicalArtistObjet1
associatedBandassociatedMusicalArtistSujet1/genre/genreObjet1
associatedBandassociatedMusicalArtistSujet1/instrument/instrumentObjet1
genreObjet1/musicFusionGenre/musicFusionGenreObjet1

t : Aaron_Turner/instrument/instrumentObjet1
k : possessive
T_{in} : instrumentObjet1 player Aaron Turner is a Black metal musician who performed with Sumac (band). Death metal is a fusion of Black metal music.
T_{out} : Sumac (band) 'Aaron Turner is a Black metal musician.
$T_{initial}$: Sumac (band) 'Aaron Turner is a Black metal musician.

$TM_{initial}$:associatedBandassociatedMusicalArtistSujet1/associatedBand/
associatedMusicalArtist/associatedBandassociatedMusicalArtistObjet1
associatedBandassociatedMusicalArtistSujet1/genre/genreObjet1
associatedBandassociatedMusicalArtistSujet1/instrument/instrumentObjet1
genreObjet1/musicFusionGenre/musicFusionGenreObjet1

Mg **: { (Al Anderson (NRBQ band) , associatedBand , associatedMusicalArtist , NRBQ), (Al Anderson (NRBQ band) , associatedBand , associatedMusicalArtist , The Wildweeds), (NRBQ , genre , Country music)** }

k : coordinated clauses
T_{in} : Al Anderson (NRBQ band) is associated with The Wildweeds and NRBQ. Burns is an exponent of Country music.
T_{out} : Al Anderson (NRBQ band) plays Country music music and performed for the The Wildweeds. Al Anderson (NRBQ band) is also associated with NRBQ.

t : NRBQ/genre/Country_music
k : coordinated clauses
T_{in} : Al Anderson (NRBQ band) is an artist for NRBQ and formerly played for The Wildweeds.
T_{out} : Al Anderson (NRBQ band) is part of NRBQ and once was part of The Wildweeds. NRBQ is a Country music band.

t : NRBQ/genre/Country_music
k : relative subject
T_{in} : Al Anderson (NRBQ band) is associated with musical artist NRBQ and The Wildweeds.
T_{out} : Country music artist NRBQ, is associated with Al Anderson (NRBQ band), who is associated with artist The Wildweeds.

k : apposition
T_{out} : Al Anderson (NRBQ band), a member of the band NRBQ, performs Country music. He also played once with The Wildweeds.

t : Country_music/instrument/instrumentObjet1
k : passive voice
T_{in} : instrumentObjet1 is an instrument for Country music, the genre of the band, NRBQ. Al Anderson (NRBQ band) from the NRBQ band was a member of The Wildweeds.
T_{out} : Country music artist Al Anderson (NRBQ band), who is part of the The Wildweeds, is associated with musical artist NRBQ.
$T_{initial}$: Country music artist Al Anderson (NRBQ band), who is part of the The Wildweeds, is associated with musical artist NRBQ.

$TM_{initial}$:associatedBandassociatedMusicalArtistSujet1/associatedBand/
associatedMusicalArtist/associatedBandassociatedMusicalArtistObjet1
associatedBandassociatedMusicalArtistSujet1/associatedBand/
associatedMusicalArtist/associatedBandassociatedMusicalArtistObjet2
associatedBandassociatedMusicalArtistObjet1/genre/genreObjet1
genreObjet1/instrument/instrumentObjet1

Mg **:** { **(Antioch , California , isPartOf , California), (Antioch , California , isPartOf , Contra Costa County , California), (California , language , Spanish language)** }

k : relative adverb
T_{in} : Antioch, California, is part of Contra Costa County, California. Spanish language is spoken in California.
T_{out} : Antioch, California is part of Contra Costa County, California and California.

t : California/language/Spanish_language
k : passive voice
T_{in} : Antioch, California is part of Contra Costa County, California but most of it is located in California.
T_{out} : Antioch, California is part of California and a city in the Contra Costa County, California. Antioch, California is also home to the Spanish language.

t : California/language/Spanish_language
k : passive voice
T_{in} : Antioch, California is part of California, Contra Costa County, California.
T_{out} : Antioch, California, is part of Contra Costa County, California. Spanish language is spoken in California.

t : Antioch_,_California/isPartOf/Contra_Costa_County_,_California
k : relative adverb
T_{in} : Spanish language is the language spoken in California.
T_{out} : Spanish language is one of the languages of California where the city of Antioch, California is located in Contra Costa County, California.

t : Antioch_,_California/isPartOf/Contra_Costa_County_,_California
k : predicative
T_{in} : The book California is written in Spanish language.
T_{out} : Antioch, California, is part of Contra Costa County, California. Spanish language is spoken in California.

t : Antioch_,_California/leaderTitle/leaderTitleObjet1
k : relative adverb
T_{in} : Antioch, California, is part of Contra Costa County, California and is led by the leaderTitleObjet1. A language spoken in California is Spanish language.
T_{out} : Antioch, California is part of California in the Contra Costa County, California, where Spanish language is the language spoken.
$T_{initial}$: Antioch, California is part of California in the Contra Costa County, California, where Chinese language is the language spoken.

$TM_{initial}$:isPartOfSujet1/isPartOf/isPartOfObjet1
isPartOfSujet1/isPartOf/isPartOfObjet2
isPartOfObjet1/language/languageObjet1
isPartOfSujet1/leaderTitle/leaderTitleObjet1

Mg **: { (Arròs negre , country , Spain), (Catalonia , leaderName , Parliament of Catalonia), (Arròs negre , region , Catalonia)** }

k : predicative
T_{in} : Parliament of Catalonia is a leader in Catalonia. Arros negre is also from Spain.
T_{out} : Parliament of Catalonia is the leader of Catalonia. Arros negre, from the Catalonia region, is a traditional dish from Spain.

k : predicative
T_{in} : Parliament of Catalonia is a leader in Catalonia where the dish Arros negre originates from. It is also found in Spain.
T_{out} : Arros negre is from the Catalonia region in Spain, the leaders of Catalonia are the Parliament of Catalonia.

k : predicative
T_{in} : Parliament of Catalonia is the leader of Catalonia. Arros negre, from the Catalonia region, is a traditional dish from Spain.
T_{out} : Parliament of Catalonia is a leader in Catalonia. Arros negre is also from Spain.

t : Catalonia/leaderName/Parliament_of_Catalonia
k : relative subject
T_{in} : Arros negre is found in the region of Spain, Catalonia.
T_{out} : Arros negre comes from the region of Catalonia in Spain which is lead by the Parliament of Catalonia.

t : Catalonia/leaderName/Parliament_of_Catalonia
k : transitive
T_{in} : Arros negre is a food found in the Catalonia, Spain.
T_{out} : Catalonia (leader Parliament of Catalonia) and Spain have the dish Arros negre.

t : Catalonia/leaderName/Parliament_of_Catalonia
k : relative adverb
T_{in} : Arros negre is a food found in Spain that 's from the Catalonia region.
T_{out} : Parliament of Catalonia is the leader of Catalonia (in Spain) where Arros negre is from.

t : Catalonia/leaderName/Parliament_of_Catalonia
k : apposition

T_{in} : Arros negre is typical Spain from the Catalonia.
T_{out} : Arros negre comes from the region of the Catalonia, Spain, The leader of the community of Valencia is Parliament of Catalonia.

t : Arros_negre/country/Spain
k : apposition
T_{in} : Arros negre is a dish of Parliament of Catalonia led, Catalonia.
T_{out} : Parliament of Catalonia was the leader of Catalonia, Spain, the location of Arros negre.

k : relative subject
T_{out} : Parliament of Catalonia is a leader in Catalonia, which is where Arros negre comes from. It is also from Spain.

t : languageSujet1/language/languageObjet1
k : predicative
T_{in} : languageSujet1 has Parliament of Catalonia as a leader, languageObjet1 and Arros negre, (from Spain), as a traditional dish.
T_{out} : Parliament of Catalonia is the leader of languageSujet1, where one can find Arros negre.
$T_{initial}$: Jusuf Kalla is the leader of Indonesia, where one can find Bakso.

$TM_{initial}$:countrySujet1/country/countryObjet1
languageSujet1/language/languageObjet1
languageSujet1/leaderName/leaderNameObjet1
countrySujet1/region/languageSujet1

t : Spain/ethnicGroup/ethnicGroupObjet1
k : relative subject
T_{in} : The ethnicGroupObjet1 are an ethnic group in Spain, where the dish Arros negre is found. The dish originates in Catalonia, where Parliament of Catalonia is the leader.
T_{out} : Parliament of Catalonia is the leader of Catalonia which is also the location of Arros negre.
$T_{initial}$: Tony Tan is the leader of Singapore which is also the location of Ayam penyet.

$TM_{initial}$:countrySujet1/country/countryObjet1
countryObjet1/ethnicGroup/ethnicGroupObjet1
leaderNameSujet1/leaderName/leaderNameObjet1
countrySujet1/region/leaderNameSujet1

t : Spain/ethnicGroup/ethnicGroupObjet1
k : apposition
T_{in} : ethnicGroupObjet1 is an ethnic group of Spain, where the dish Arros negre can be found. This dish is from the Catalonia region, where Parliament of Catalonia is a leader.

T_{out} : Arros negre is a cuisine found in Parliament of Catalonia led, Spain.
$T_{initial}$: Ayam penyet is a cuisine found in Tony Tan led, Java.

$TM_{initial}$:countrySujet1/country/countryObjet1
countryObjet1/ethnicGroup/ethnicGroupObjet1
leaderNameSujet1/leaderName/leaderNameObjet1
countrySujet1/region/leaderNameSujet1

t : languageSujet1/language/languageObjet1
k : transitive
T_{in} : The languageObjet1 is spoken in languageSujet1, which is where Parliament of Catalonia is the leader. It is also where the dish Arros negre (which is also popular in Spain) comes from.
T_{out} : Arros negre is a traditional dish from Spain where they speak languageObjet1 and the leader is Parliament of Catalonia.
$T_{initial}$: Beef kway teow is a traditional dish from Indonesia where they speak English language and the leader is Tony Tan.

$TM_{initial}$:countrySujet1/country/countryObjet1
languageSujet1/language/languageObjet1
languageSujet1/leaderName/leaderNameObjet1
countrySujet1/region/languageSujet1

t : Spain/ethnicGroup/ethnicGroupObjet1
k : transitive
T_{in} : The ethnicGroupObjet1 are an ethnic group in Spain, where the dish Arros negre is found. The dish originates in Catalonia, where Parliament of Catalonia is the leader.
T_{out} : Parliament of Catalonia is the leader of Catalonia, where one can find Arros negre.
$T_{initial}$: Tony Tan is the leader of Singapore, where one can find Ayam penyet.

$TM_{initial}$:countrySujet1/country/countryObjet1
countryObjet1/ethnicGroup/ethnicGroupObjet1
leaderNameSujet1/leaderName/leaderNameObjet1
countrySujet1/region/leaderNameSujet1

t : Spain/ethnicGroup/ethnicGroupObjet1
k : passive voice
T_{in} : Arros negre originates from the Catalonia region of Spain where Parliament of Catalonia is the leader. The country 's main ethnic group are the ethnicGroupObjet1.
T_{out} : Parliament of Catalonia is the leader of Catalonia (in Spain) where Arros negre is from.
$T_{initial}$: Susana Diaz is the leader of Andalusia (in Spain) where Ajoblanco is from.

$TM_{initial}$:countrySujet1/country/countryObjet1
countryObjet1/ethnicGroup/ethnicGroupObjet1
leaderNameSujet1/leaderName/leaderNameObjet1
countrySujet1/region/leaderNameSujet1

t : Spain/ethnicGroup/ethnicGroupObjet1
k : relative adverb
T_{in} : Arros negre is originates from Catalonia and Parliament of Catalonia is the country 's leader. It can also be found in Spain where ethnicGroupObjet1 people are one of the ethnic groups.
T_{out} : Arros negre comes from the region of the Catalonia, in Spain, where Parliament of Catalonia is the leader.
$T_{initial}$: Ayam penyet comes from the region of the Singapore, in Java, where Tony Tan is the leader.

$TM_{initial}$:countrySujet1/country/countryObjet1
countryObjet1/ethnicGroup/ethnicGroupObjet1
leaderNameSujet1/leaderName/leaderNameObjet1
countrySujet1/region/leaderNameSujet1

Mg **:** { **(South Africa , ethnicGroup , Asian South Africans), (South Africa , leaderName , Cyril Ramaphosa), (11 Diagonal Street , location , South Africa)** }

k : relative adverb
T_{in} : South Africa (led by Cyril Ramaphosa) is the home of the ethnic group Asian South Africans and the location of 11 Diagonal Street.
T_{out} : 11 Diagonal Street is located in South Africa where the Asian South Africans are an ethnic group.

k : relative adverb
T_{in} : South Africa is the location of 11 Diagonal Street and is home to the ethnic group of Asian South Africans. Cyril Ramaphosa is a leader of the country.
T_{out} : Cyril Ramaphosa is one leader of South Africa, where Asian South Africans live.

t : South_Africa/leaderName/Cyril_Ramaphosa
k : predicative
T_{in} : There is an ethnic group of Asian South Africans and the address 11 Diagonal Street is also located in South Africa.
T_{out} : Cyril Ramaphosa is one leader of South Africa, where Asian South Africans reside at 11 Diagonal Street.

t : South_Africa/ethnicGroup/Asian_South_Africans

k : passive voice
T_{in} : Cyril Ramaphosa is a leader in the South Africa and 11 Diagonal Street is also located in the South Africa.
T_{out} : The address, 11 Diagonal Street is located in South Africa and Cyril Ramaphosa is a leader in Asian South Africans.

t : South_Africa/ethnicGroup/Asian_South_Africans
k : apposition
T_{in} : 11 Diagonal Street is located in the South Africa, where Queen Cyril Ramaphosa reigns.
T_{out} : The address, 11 Diagonal Street is located in South Africa and Cyril Ramaphosa is a leader in Asian South Africans.

t : South_Africa/ethnicGroup/Asian_South_Africans
k : relative adverb
T_{in} : 11 Diagonal Street is located in the South Africa, the leader of which was Cyril Ramaphosa.
T_{out} : Cyril Ramaphosa is one leader of South Africa, where Asian South Africans reside at 11 Diagonal Street.

k : relative object
T_{out} : 11 Diagonal Street is located in South Africa, the country in which Asian South Africans are one of the ethnic groups and Cyril Ramaphosa is a leader.

t : capitalSujet1/capital/capitalObjet1
k : relative adverb
T_{in} : Cyril Ramaphosa is one of the leaders of capitalSujet1, which capital is capitalObjet1. The address, 11 Diagonal Street is located in that country and one of the the ethnic groups is Asian South Africans.
T_{out} : 11 Diagonal Street is located in the capitalSujet1, where Queen Cyril Ramaphosa reigns.
$T_{initial}$: 11 Diagonal Street is located in the South Africa, where Queen Cyril Ramaphosa reigns.

$TM_{initial}$:capitalSujet1/capital/capitalObjet1
capitalSujet1/ethnicGroup/ethnicGroupObjet1
capitalSujet1/leaderName/leaderNameObjet1
locationSujet1/location/capitalSujet1

t : capitalSujet1/capital/capitalObjet1
k : apposition
T_{in} : 11 Diagonal Street is located in capitalSujet1, capitalObjet1 and the leader is Cyril Ramaphosa of the Asian South Africans.
T_{out} : Cyril Ramaphosa is the leader of capitalSujet1, the city where 11 Diagonal Street

is located.
$T_{initial}$: Jacob Zuma is the leader of South Africa, the city where 11 Diagonal Street is located.

$TM_{initial}$:capitalSujet1/capital/capitalObjet1
capitalSujet1/ethnicGroup/ethnicGroupObjet1
capitalSujet1/leaderName/leaderNameObjet1
locationSujet1/location/capitalSujet1

Mg **: { (250 Delaware Avenue , floorArea , 30843.8 square metres), (250 Delaware Avenue , floorCount , 12), (250 Delaware Avenue , location , Buffalo , New York)** }

k : predicative
T_{in} : 250 Delaware Avenue, Buffalo, New York, has 12 floors and a floor area of 30843.8 square metres.
T_{out} : 250 Delaware Avenue is a location in Buffalo, New York, it has 12 floors and 30843.8 square metres space.

t : 250_Delaware_Avenue/floorArea/30843.8_square_metres
k : relative subject
T_{in} : 250 Delaware Avenue actually on Buffalo, New York has 12 floors.
T_{out} : 250 Delaware Avenue in the Buffalo, New York has 12 floors that cover 30843.8 square metres.

t : 250_Delaware_Avenue/floorArea/30843.8_square_metres
k : transitive
T_{in} : There are 12 floors of 250 Delaware Avenue in Buffalo, New York.
T_{out} : 250 Delaware Avenue in the Buffalo, New York has 12 floors with a total area of 30843.8 square metres.

t : 250_Delaware_Avenue/floorArea/30843.8_square_metres
k : apposition
T_{in} : There are 12 floors at 250 Delaware Avenue in Buffalo, New York.
T_{out} : There are 12 floors at 250 Delaware Avenue, Buffalo, New York, with a total floor area of 30843.8 square metres.

k : transitive
T_{out} : 250 Delaware Avenue in Buffalo, New York has 12 floors with a total area of 30843.8 square metres.

t : costSujet1/cost/costObjet1
k : apposition

T_{in} : costSujet1 in Buffalo, New York cost costObjet1 and has 12 floors with a total area of 30843.8 square metres.
T_{out} : costSujet1, Buffalo, New York, has 12 floors and a floor area of 30843.8 square metres.
$T_{initial}$: 250 Delaware Avenue, Buffalo, New York, has 12 floors and a floor area of 30843.8 square metres.

$TM_{initial}$:costSujet1/cost/costObjet1
costSujet1/floorArea/floorAreaObjet1
costSujet1/floorCount/floorCountObjet1
costSujet1/location/locationObjet1

t : completionDateSujet1/completionDate/completionDateObjet1
k : apposition
T_{in} : completionDateSujet1, with 12 floors covering 30843.8 square metres, is located in Buffalo, New York and was completed in completionDateObjet1.
T_{out} : completionDateSujet1, Buffalo, New York, has 12 floors and was completed in completionDateObjet1.
$T_{initial}$: 200 Public Square, Cleveland, has 45 floors and was completed in 1985.

$TM_{initial}$:completionDateSujet1/completionDate/completionDateObjet1
completionDateSujet1/floorArea/floorAreaObjet1
completionDateSujet1/floorCount/floorCountObjet1
completionDateSujet1/location/locationObjet1

Mg **:** { **(Asilomar Conference Grounds , architecture , "Arts and Crafts Movement and American craftsman Bungalows"), (Asilomar Conference Grounds , location , Pacific Grove , California), (Asilomar Conference Grounds , yearOfConstruction , 1913)** }

k : predicative
T_{in} : The Asilomar Conference Grounds constructed in 1913 at Pacific Grove, California is based on the architecture style of "Arts and Crafts Movement and American craftsman Bungalows".
T_{out} : The Asilomar Conference Grounds is one of the "Arts and Crafts Movement and American craftsman Bungalows" constructed in 1913 in Pacific Grove, California.

t : Asilomar_Conference_Grounds/architecture/"Arts_and_Crafts_Movement_and_-American_craftsman_Bungalows"
k : coordinated clauses
T_{in} : Asilomar Conference Grounds was built in 1913 and is located in Pacific Grove, California.
T_{out} : The Asilomar Conference Grounds, Pacific Grove, California, were constructed in

1913 and designed based on the architecture style of "Arts and Crafts Movement and American craftsman Bungalows".

t : Asilomar_Conference_Grounds/architecture/"Arts_and_Crafts_Movement_and_-American_craftsman_Bungalows"
k : predicative
T_{in} : Asilomar Conference Grounds located at Pacific Grove, California. and constructed in 1913.
T_{out} : The Asilomar Conference Grounds is one of the "Arts and Crafts Movement and American craftsman Bungalows" constructed in 1913 and is located in Pacific Grove, California.

k : predicative
T_{out} : The Asilomar Conference Grounds, Pacific Grove, California, were constructed in 1913 and designed based on the architecture style of "Arts and Crafts Movement and American craftsman Bungalows".

t : ReferenceNumber/in/the
k : coordinated clauses
T_{in} : National Register of Historic PlacesSujet1 ReferenceNumber in the National Register of Historic Places ReferenceNumber in the National Register of Historic PlacesObjet1 The ReferenceNumber in the National Register of Historic PlacesSujet1 built in 1913 is located at Pacific Grove, California and was built in the architectural style of the "Arts and Crafts Movement and American craftsman Bungalows". ReferenceNumber in the National Register of Historic PlacesObjet1 is the reference number in the National Register of Historic Places.
T_{out} : The ReferenceNumber in the National Register of Historic PlacesSujet1 is located on Pacific Grove, California and was built in 1913.
$T_{initial}$: The Asilomar Conference Grounds is located on Pacific Grove, California and was built in 1913.

$TM_{initial}$:ReferenceNumber in the National Register of Historic PlacesSujet1/
ReferenceNumber in the National Register of Historic Places/
ReferenceNumber in the National Register of Historic PlacesObjet1
ReferenceNumber in the National Register of Historic PlacesSujet1/architecture/
architectureObjet1
ReferenceNumber in the National Register of Historic PlacesSujet1/location/
locationObjet1
ReferenceNumber in the National Register of Historic PlacesSujet1/yearOfConstruction/
yearOfConstructionObjet1

Mg **:** { **(ALCO RS-3 , builder , Montreal Locomotive Works), (ALCO RS-3 , engine , V12 engine), (ALCO RS-3 , length , 17068.8 millimetres)** }

k : passive voice
T_{in} : The ALCO RS-3, built by the Montreal Locomotive Works, is 17068.8 millimetres long with a V12 engine.
T_{out} : The ALCO RS-3, 17068.8 millimetres long with a V12 engine, is built by Montreal Locomotive Works.

k : predicative
T_{in} : The ALCO RS-3, 17068.8 millimetres long with a V12 engine, is built by Montreal Locomotive Works.
T_{out} : The ALCO RS-3, built by Montreal Locomotive Works, is 17068.8 millimetres long with a V12 engine.

t : ALCO_RS-3/length/17068.8_millimetres
k : apposition
T_{in} : The ALCO RS-3 has a V12 engine and is built by the Montreal Locomotive Works.
T_{out} : The ALCO RS-3, built by Montreal Locomotive Works, has a V12 engine and is 17068.8 millimetres in length.

k : transitive
T_{out} : The ALCO RS-3 was built by the Montreal Locomotive Works. It has a V12 engine and is 17068.8 millimetres in length.

k : predicative
T_{out} : The ALCO RS-3 was built by the Montreal Locomotive Works. It has a V12 engine and is 17068.8 millimetres in length.

t : ALCO_RS-3/cylinderCount/cylinderCountObjet1
k : apposition
T_{in} : The ALCO RS-3 was built by the Montreal Locomotive Works and has cylinderCountObjet1 cylinders, a V12 engine and a length of 17068.8 millimetres.
T_{out} : The ALCO RS-3 has a V12 engine, a length of 17068.8 millimetres and was built by the Montreal Locomotive Works.
$T_{initial}$: The ALCO RS-3 has a Four-stroke engine, a length of 17068.8 millimetres and was built by the Montreal Locomotive Works.

$TM_{initial}$:builderSujet1/builder/builderObjet1
builderSujet1/cylinderCount/cylinderCountObjet1
builderSujet1/engine/engineObjet1
builderSujet1/length/lengthObjet1

t : ALCO_RS-3/cylinderCount/cylinderCountObjet1
k : predicative
T_{in} : The ALCO RS-3, built by Montreal Locomotive Works, has a cylinder count of cylinderCountObjet1, a V12 engine and a length of 17068.8 millimetres.
T_{out} : The ALCO RS-3 was built by the Montreal Locomotive Works. It is 17068.8 millimetres long and has a V12 engine.
$T_{initial}$: The ALCO RS-3 was built by the Montreal Locomotive Works. It is 17068.8 millimetres long and has a V12 engine.

$TM_{initial}$:builderSujet1/builder/builderObjet1
builderSujet1/cylinderCount/cylinderCountObjet1
builderSujet1/engine/engineObjet1
builderSujet1/length/lengthObjet1

t : buildDateSujet1/buildDate/buildDateObjet1
k : predicative
T_{in} : The buildDateSujet1 has a V12 engine and a length of 17068.8 millimetres. It was produced between buildDateObjet1 by the Montreal Locomotive Works.
T_{out} : The buildDateSujet1 was built by the Montreal Locomotive Works. It is 17068.8 millimetres long and has a V12 engine.
$T_{initial}$: The ALCO RS-3 was built by the American Locomotive Company. It is 17068.8 millimetres long and has a V12 engine.

$TM_{initial}$:buildDateSujet1/buildDate/buildDateObjet1
buildDateSujet1/builder/builderObjet1
buildDateSujet1/engine/engineObjet1
buildDateSujet1/length/lengthObjet1

t : ALCO_RS-3/powerType/powerTypeObjet1
k : relative subject
T_{in} : The ALCO RS-3 has a V12 engine and a powerTypeObjet1. It is 17068.8 millimetres long and was built by the Montreal Locomotive Works.
T_{out} : The Montreal Locomotive Works built the ALCO RS-3 which is 17068.8 millimetres long and has a V12 engine.
$T_{initial}$: The American Locomotive Company built the ALCO RS-3 which is 17068.8 millimetres long and has a Four-stroke engine.

$TM_{initial}$:builderSujet1/builder/builderObjet1
builderSujet1/engine/engineObjet1
builderSujet1/length/lengthObjet1
builderSujet1/powerType/powerTypeObjet1

t : ALCO_RS-3/totalProduction/totalProductionObjet1
k : relative subject

T_{in} : The ALCO RS-3, built by the Montreal Locomotive Works has a V12 engine and is 17068.8 millimetres long. Production time is totalProductionObjet1.
T_{out} : The ALCO RS-3, which is 17068.8 millimetres long, was manufactured by the Montreal Locomotive Works.
$T_{initial}$: The ALCO RS-3, which is 17068.8 millimetres long, was manufactured by the American Locomotive Company.

$TM_{initial}$:builderSujet1/builder/builderObjet1
builderSujet1/engine/engineObjet1
builderSujet1/length/lengthObjet1
builderSujet1/totalProduction/totalProductionObjet1

t : ALCO_RS-3/cylinderCount/cylinderCountObjet1
k : passive voice
T_{in} : The ALCO RS-3, built by Montreal Locomotive Works, has a cylinder count of cylinderCountObjet1, a V12 engine and a length of 17068.8 millimetres.
T_{out} : The ALCO RS-3 has a V12 engine, a length of 17068.8 millimetres and was built by the Montreal Locomotive Works.
$T_{initial}$: The ALCO RS-3 has a V12 engine, a length of 17068.8 millimetres and was built by the Montreal Locomotive Works.

$TM_{initial}$:builderSujet1/builder/builderObjet1
builderSujet1/cylinderCount/cylinderCountObjet1
builderSujet1/engine/engineObjet1
builderSujet1/length/lengthObjet1

Annexe E

Représentation automatique de sens

L'analyse a été réalisée avec trois parseurs :

- Grew, version 0.48.0, http://Grew.fr/,
- Talismane, version 5.1.2, http://redac.univ-tlse2.fr/applications/Talismane.html,
- Stanford, version 2018-02-27, https://nlp.Stanford.edu/software/lex-parser.shtml.

E. 1 Réalisation de surface, exemples

Notre référence est un échantillon de 50 phrases, jamais vues pendant l'entraînement. Pour une entrée x :

- x (a) est une référence de sortie, la représentation de sens donnée en entrée en dérive,
- x (b) est la phrase prédite (relexicalisée).

Pour cet extrait du corpus de test, nous avons un nombre maximum de 28 # *tokens* par entrée (17 mots maximum pour les références correspondantes). Dans le corpus de test, ce nombre par phrase a un maximum de 100 *tokens* (69 mots pour les références). La distribution est :

	# tokens					
	]04-08]	]08-12]	]12-16]	]16-20]	]20-24]	]24-100]
échantillon de 50 phrases	9	21	7	0	12	1
corpus de test global	309	12705	14851	9741	8759	2661

Dans le corpus de test complet (49026 phrases),

- Dans 34.76% des cas, les phrases générées sont exactement les mêmes que la phrase de référence,
- Dans 34.07 % des cas nous ne pouvons re-lexicaliser tous les mots.
- 94.8% des entités attendues et de leurs modificateurs sont obtenues,

- 87 % du vocabulaire fonctionnel attendu.

Dans notre échantillon de 50 phrases nous avons :

- 34% des phrases générées sont exactement les mêmes que la phrase de référence,
- 58% des phrases générées sont déterminées correctes (vérification humaine),
- 78% n'ont pas de problème syntaxique (vérification humaine),
- Tous les verbes (temps et auxiliaires) sont correctement ajustés, et nous ne détectons qu'une seule erreur pour l'accord entre un nom et son déterminant (mais son lié à une erreur d'analyse en amont).

Pour les erreurs que nous avons, sur ces 50 phrases prédites :

- 1 erreur de punctuation, : 35(b)
- 2 adverbes mal positionnés, 43(b), 44(b)
- 1 confusion entre un quantificateur et un modifieur, 31(b)
- 6 mauvais positionnement concernant un couple modifieur/nom, 30(b), 32(b), 37(b), 38(b), 41(b), 42(b)
- 5 phrases dépourvues de sens réel, 33(b), 34(b), 36(b), 39(b), 40(b)
- 1 accord entre un nom et son déterminant : le mot avait été demandé sans la propriété appropriée. 45(b)
- 1 problème élision 45(b)

Génération identique à la référence

01(a) Malheureusement les répétitions ne vont pas .
01(b) Malheureusement les répétitions ne vont pas .

02(a) Ils se sont enfuis sur les débris du monde .
02(b) Ils se sont enfuis sur les débris du monde .

03(a) Il fallait de l' indulgence pour les Roussillonnais .
03(b) Il fallait de l' indulgence pour les Roussillonnais .

04(a) Vous répondrez de ma vie devant Dieu .
04(b) Vous répondrez de ma vie devant Dieu .

05(a) Cette douceur de la prisonnière aiguisait cent poignards contre Élisabeth .
05(b) Cette douceur de la prisonnière aiguisait cent poignards contre Élisabeth .

06(a) Au milieu de la réprobation générale , Anicet fils ne perd pas le sentiment de sa dignité .
06(b) Au milieu de la réprobation générale , Anicet fils ne perd pas le sentiment de sa dignité .

07(a) Vous ne cherchez pas un éclat , n' est -ce pas ?
07(b) Vous ne cherchez pas un éclat , n' est -ce pas ?

08(a) La ligne de centaures aux ventres rouges avalait l' espace .
08(b) La ligne de centaures aux ventres rouges avalait l' espace .

09(a) Il aimait à sentir la terre sous ses pieds .
09(b) Il aimait à sentir la terre sous ses pieds .

10(a) Je vais tremper ce tison dans l' eau .
10(b) Je vais tremper ce tison dans l' eau .

11(a) La France renaît au repos , à l' espérance .
11(b) La France renaît au repos , à l' espérance .

12(a) Un sentiment de joie délirante s' empara de lui !
12(b) Un sentiment de joie délirante s' empara de lui !

13(a) Depuis longtemps l' empereur français pressait les Américains de prendre les armes .
13(b) Depuis longtemps l' empereur français pressait les Américains de prendre les armes .

14(a) Il avait quitté la main de Fantine .
14(b) Il avait quitté la main de Fantine .

15(a) Ces paysans étaient incapables de cela .
15(b) Ces paysans étaient incapables de cela .

16(a) Il ne se serait pas jeté dans la bataille !
16(b) Il ne se serait pas jeté dans la bataille !

17(a) Il ne répondit que par un léger signe affirmatif .
17(b) Il ne répondit que par un léger signe affirmatif .

Génération différant modestement
18(a) Les propos de serviteurs inconsidérés ont donné une consistance fâcheuse à cette accusation .
18(b) Les propos des inconsidérés serviteurs ont donné une consistance fâcheuse à cette accusation .

19(a) Cet homme , c' était le comte .
19(b) Cet homme était le comte .

20(a) Une vaine ambition vous a poussé jusques au bord du précipice .

20(b) jusques au bord du précipice , une ambition vaine vous a poussé .

21(a) Que se passa -t-il alors entre les deux amants ?
21(b) Alors se passa Que entre les deux amants ?

22(a) Par elle , la cour se transformait en un temple des Muses .
22(b) En un temple des Muses , la cour se transformait par elle .

23(a) Nous n' avions même pas besoin de nous cacher .
23(b) Nous n' avions pas même besoin de nous cacher .

24(a) Sans doute ces animaux sont dressés à revenir au logis .
24(b) Ces animaux sont sans doute dressés à revenir au logis .

25(a) Le mot est de création récente .
25(b) Le mot est de récente création .

26(a) Une femme était là , toute seule .
26(b) Une femme était là la seule .

27(a) Ainsi qu' il l' avait prévu , l' arrivée subite de Léocadie avait surexcité le sentiment naissant de la jeune fille .
27(b) il l' avait prévu , l' arrivée subite de Léocadie avait surexcité le sentiment naissant de la fille jeune .

28(a) Le sacristain était un gros gars qui avait peut-être servi la messe aux chartreux dans son enfance , et qui désormais était dépositaire des clefs du couvent .
28(b) Le sacristain était un gars gros qui avait peut-être servi la messe aux chartreux dans son enfance , et un gars était dépositaire des clefs du couvent .

29(a) Il regardait froidement le peuple , qui le lui rendait .
29(b) Il regardait froidement le peuple qui lui le rendait .

Génération considérée mauvaise
30(a) Après les déclarations et les intimités du fiacre , ma liaison avec Franz prit une tournure particulière .
30(b) Après les déclarations et les intimités du fiacre , ma liaison avec Franz prit une particulière tournure .

31(a) Le porteur s' est dérobé , et la malle a pris un bain d' un quart d' heure .
31(b) Le porteur s' est dérobé , et la malle a pris un bain du quart de heure .

32(a) Les élections des collèges d' arrondissement eurent lieu le 17 novembre , et celles

des collèges de département le 24 .
32(b) Les élections des collèges d' arrondissement eurent lieu le novembre 17 et celles des collèges de département le 24 .

33(a) Le Gouvernement était mis en demeure de répondre ; il l' a fait par le décret que vous connaissez .
33(b) vous connaissez qu' il l' a fait par le décret , le Gouvernement était mis en demeure .

34(a) Mon coeur est si plein que je crois par moments qu' il va se briser .
34(b) je crois par moments que mon coeur est plein , il va se briser .

35(a) Tu ouvres l' armoire !
35(b) Tu ouvres , l' armoire !

36(a) Caroline , c' est un infâme , il vous mentait dans vos promenades au bois , la nuit , dans ses lettres , toujours et partout .
36(b) Dans vos promenades au bois , Caroline est toujours et partout un infâme dont il vous mentait la nuit dans ses lettres .

37(a) La même chose se produit actuellement pour l' enseignement moderne .
37(b) La chose même se produit actuellement pour l' enseignement moderne .

38(a) Le roi Pépin , vainqueur des Lombards , accorde au pape Étienne III le gouvernement temporel de Rome .
38(b) Le roi Pépin , vainqueur des Lombards , accorde au Étienne pape III le gouvernement temporel de Rome .

39(a) Un éclat de lumière , semblable aux plus perçants éclairs , sortit de ses yeux .
39(b) Un éclat de lumière plus perçants sortit de ses yeux semblable aux éclairs .

40(a) Seule , une vieille femme pouvait le dire .
40(b) Une Seule femme vieille pouvait le dire .

41(a) Tous ces gens pensent à la mort , comme moi .
41(b) Ces Tous gens pensent à la mort comme moi .

42(a) Cela fait un grand remue-ménage dans le monde musical de Paris .
42(b) Cela fait un grand remue-ménage dans le musical monde de Paris .

43(a) Je comprends aujourd' hui que mon âme était toujours bien malade .
43(b) Mon âme était toujours bien malade , Je aujourd' hui comprends .

44(a) Le soulèvement , la séparation même d' une province ne font pas tout le sort d' un empire .
44(b) Le tout soulèvement , la séparation d' une province ne font pas le sort même d' un empire .

45(a) Il paraît qu' on a refusé de le recevoir aux prisons .
45(b) Il paraît que on a refusé de le recevoir au prisons .

Génération indiscutablement mauvaise
46(a) Ces rudes débuts ne découragèrent personne ; de suite on se mit au travail .
46(b) on se mit au travail ; ces rudes débuts ne personne découragèrent pas .

47(a) Le capitaine m' a expliqué que le mot hippopotame signifie " cheval de fleuve " .
47(b) Le capitaine a expliqué que le mot signifie cheval de fleuve , le capitaine hippopotame signifie cheval de cheval .

48(a) Un député surtout étonna l' assemblée , le Bourguignon Philippe Pot , docile courtisan de Charles le Téméraire , puis de Louis XI .
48(b) Un député étonna surtout le docile assemblée le Bourguignon courtisan de Charles , puis de Louis le Bourguignon docile XI .

49(a) Vers 1160 , on ouvrit dans la salle capitulaire de Vézelay , bâtie depuis dix ans , trois arcades donnant sur le cloître .
49(b) on ouvrit , dans la capitulaire salle , des arcades trois , donnant sur le cloître , des dix ans .

50(a) On avait encore de lui deux tragédies mauvaises dans toutes les règles , un éloge des crocodiles , et quelques opéras .
50(b) On avait encore de lui un éloge des crocodiles mauvaises dans les règles toutes , et quelques deux tragédies un éloge des crocodiles .

E. 2 Statistiques : association classe LVF à un verbe

Remarque : trouver la classe LVF n'est pas toujours possible, si la même signature est partagée entre plusieurs classes. Une vraie classe LVF est appelée “pure’, une classe LVF créée après la désambiguïsation est appelée “classe Z”.

À propos des phrases après traitements.

- 2 623/2 835 livres utilisés,
- 1 727 606 phrases analysées,
- 553 000 phrases après étiquetage (= données expérimentales),
- 80 078 *tokens*,

- Fréquence des *tokens* : min 1, max 651 475, médiane 2.0, moyenne 99.82.
- *Tokens* par phrase : min 3, max 69, médiane 13.0, moyenne 14.45.
- 1.65 verbes par phrase.
- 514 LVF classes détectées ou formées, avec 237 classes pures :
  - 3,094 lemmes distincts pour les classes pures,
  - 1,983 lemmes distincts pour les classes Z,
  - 3,542 lemmes distincts pour l'ensemble.
  - 1,559 lemmes verbaux sont dans une classe pure mais pas dans une classe Z,
  - 448 lemmes verbaux sont dans une classe Z mais pas dans une classe pure,
  - 1 535 lemmes verbaux sont à la fois dans une classe Z et dans une classe pure.
- Phrases par classe : min 1, max 145 186, médiane 48.0, moyenne 1 777.59.
- Phrases par classe pure : min 1, max 59 856, médiane 38, moyenne 1 144.92.
- 358 classes pour les verbes conjugués.
- Phrases par classes sur formes conjuguées : min 1, max 142 066, médiane 78.5, moyenne 2 219.0.
- 514 classes présentes dans tous les modes verbaux (participe passé, participe présent, forme conjuguée, infinitif).
- Phrases par classe réalisée dans tous les modes (un verbe peut n'être que dans un seul mode, chaque classe est réalisée dans tous les modes) :
  min 8, max 145,186, médiane 1 266, moyenne 5 609.52 avec, en terme de nombre de phrases contenant au moins un verbe :
  - conjugué : min 1, max 142 066, médiane 1 016, moyenne 4 896.87
  - au participe passé : min 1, max 969, médiane 22, moyenne 81.1
  - au participe présent : min 1, max 972, médiane 20, moyenne 66.44
  - à l'infinitif : min 1, max 6 693, médiane 106, moyenne 565.11
- Pour les 799 lemmes distincts, en terme de phrases : min 4, max 149 576, médiane 251, moyenne 928.29.
- 799 verbes présents dans tous les modes (participe passé, participe présent, conj, infinitif),
- 659 047 phrases (498 292 phrases distinctes) pour les verbes déclinés dans tous les modes (participe passé, participe présent, conjugué, infinitif),
- Phrases par classes de verbe de tous les modes (dont chaque verbe et chaque classe sont réalisés dans tous les modes) :
  min 7, max 144 686, médiane 1 302, moyenne 5 937.36 avec, en terme de nombre de phrases contenant au moins un verbe :
  - conjugué : min 1, max 141 569, médiane 1 077, moyenne 5 224.12
  - au participe passé : min 1, max 533, médiane 16, moyenne 64.63
  - au participe présent : min 1, max 881, médiane 17, moyenne 71.02
  - à l'infinitif : min 1, max 5575, médiane 104, moyenne 577.59

Évènements verbaux

TABLE E. 1 – Distribution des verbes en terme de verbes par phrases

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**13**
1	309 761	161 067	57 130	18 171	5 097	1 429	387	109	28	14	3	1

553 198 phrases préservées.
1 174 408 phrases perdues.
1.65 verbes par phrase.
3 371 verbes conservés (contre 3 542 dans le corpus complet).

E. 3 Données avec et sans perte intégrale

Pour chaque table, *disp.* signifie dispersion, *vpp* verbe au participe passé, *vpa* verbe au participe présent, *inf.* infinitif.

E. 3.1 3 371 verbes sans perte intégrale

TABLE E. 2 – Données sans perte intégrale (3371 verbes)

Verbe	compte	vpp	vpr	inf.	autres	losses		frames		
						direct	collatéral	init.	final	disp.
être	149 677	6	91	3 018	146 562	57 482	95 074	4	4	4
avoir	50 736	8	274	1 565	48 889	14 866	37 383	6	9	9
faire	29 567	243	246	3 282	25 796	13 624	23 023	23	18	18
pouvoir	14 874	0	6	0	14 868	2 210	18 596	4	4	4
voir	12 403	19	347	2 471	9 566	4 773	11 925	17	10	10
prendre	11 233	165	151	1 617	9 300	5 271	10 817	29	17	17
donner	11 063	95	140	1 148	9 680	4 808	9 172	22	14	14
aller	10 140	0	37	5	10 098	2 133	8 075	15	8	8
trouver	9 496	60	68	768	8 600	5 582	10 747	15	11	11
passer	8 816	55	154	447	8 160	3 180	6 754	33	14	14
dire	8 559	177	196	1 334	6 852	10 176	14 120	15	9	9
vouloir	8 150	38	9	21	8 082	3 299	8 387	7	4	4
mettre	7 369	28	59	1 171	6 111	4 920	6 200	23	12	12
savoir	7 186	0	33	336	6 817	2 306	10 313	6	4.00	4
venir	7 144	124	19	148	6 853	4 404	12 408	16	8	8
rester	6 231	82	7	512	5 630	3 246	5 166	12	4	4
parler	5 876	8	108	222	5 538	2 281	4 498	17	11	11
porter	5 464	17	232	609	4 606	2 593	4 958	25	9	9
rendre	5 350	130	29	634	4 557	2 649	4 615	16	12	12
falloir	5 180	0	0	1	5 179	841	4 597	4	3	3
croire	4 763	0	26	676	4 061	1 532	6 488	8	6	6
connaître	4 731	90	50	475	4 116	2 436	3 883	8	7	7
paraître	4 730	7	3	230	4 490	2 279	4 960	6	4	4
entendre	4 654	104	100	739	3 711	1 790	3 827	13	9	9
laisser	4 450	41	177	562	3 670	4 435	4 647	15	8	8
demander	4 248	23	29	371	3 825	1 567	4 642	13	5	5
devenir	4 231	269	20	419	3 523	4 484	5 329	1	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
regarder	4 128	3	202	416	3 507	3 263	3 750	8	5	5
tenir	4 071	27	156	420	3 468	3 043	5 400	30	14	14
aimer	4 057	29	19	328	3 681	2 500	3 676	5	5	5
attendre	4 009	32	449	338	3 190	2 105	3 437	8	5	5
recevoir	3 948	62	39	582	3 265	2 451	3 600	8	6	6
suivre	3 827	31	65	591	3 140	2 589	3 493	13	6	6
commencer	3 815	83	28	126	3 578	971	2 598	11	8	8
sembler	3 699	0	0	16	3 683	1 610	5 169	3	4	4
perdre	3 604	174	45	339	3 046	1 579	2 824	13	8	8
tomber	3 455	76	28	1	3 350	1 397	2 907	18	9	9
ouvrir	3 412	47	37	270	3 058	1 394	3 220	18	11	11
jeter	3 382	93	99	260	2 930	1 643	3 494	18	15	15
arrêter	3 205	94	14	312	2 785	1 519	3 236	15	14	14
arriver	2 818	73	62	574	2 109	9 168	3 875	9	3	3
montrer	2 672	1	44	203	2 424	1 814	2 825	10	4	4
appeler	2 667	74	20	238	2 335	2 029	2 901	17	9	9
chercher	2 651	12	85	708	1 846	1 351	2 065	10	6	6
sortir	2 632	9	23	90	2 510	1 628	2 973	22	8	8
manquer	2 595	27	9	109	2 450	741	1 648	15	11	11
répondre	2 595	0	8	411	2 176	2 422	1 981	10	5	5
partir	2 555	7	37	452	2 059	1 404	1 842	18	2	2
servir	2 490	28	14	219	2 229	1 684	2 260	17	9	9
mourir	2 471	7	4	418	2 042	970	1 236	5	4	4
penser	2 324	1	22	178	2 123	756	3 122	6	6	6
reprendre	2 310	6	34	362	1 908	845	1 873	18	9	9
comprendre	2 305	7	47	369	1 882	1 538	2 400	8	5	5
pousser	2 263	14	121	199	1 929	1 003	1 993	20	9	9
présenter	2 260	16	23	155	2 066	1 278	2 240	21	8	8
garder	2 221	16	39	317	1 849	1 098	1 598	14	6	6
frapper	2 204	119	65	179	1 841	1 088	1 881	19	15	15
quitter	2 190	2	115	471	1 602	775	2 022	6	3	3
reconnaître	2 174	36	68	395	1 675	1 246	2 279	11	8	8
occuper	2 157	49	0	206	1 902	1 522	2 103	12	7	7
sentir	2 104	3	33	130	1 938	2 060	3 008	6	7	7
revenir	2 100	28	57	360	1 655	2 997	3 256	13	4	4
écrire	2 089	44	28	178	1 839	1 094	2 199	12	6	6
lever	2 087	24	53	0	2 010	704	2 482	20	10	10
tirer	2 016	42	50	368	1 556	1 037	1 965	26	13	13
continuer	2 016	5	26	162	1 823	554	1 511	9	8	8
former	1 976	88	65	211	1 612	1 315	2 132	15	10	10
jouer	1 924	27	63	224	1 610	761	1 352	20	10	10
changer	1 909	27	9	184	1 689	810	1 288	16	8	8
rappeler	1 825	4	22	130	1 669	583	1 861	13	8	8
vivre	1 787	5	26	69	1 687	903	1 690	14	7	7
rencontrer	1 787	16	16	200	1 555	1 021	1 942	12	6	6
descendre	1 759	15	46	59	1 639	768	1 966	21	10	10
finir	1 725	3	6	153	1 563	523	1 204	19	7	7
retrouver	1 670	16	14	250	1 390	763	1 269	11	6	6

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
conduire	1 666	9	20	317	1 320	1 419	1 620	14	8	8
demeurer	1 662	20	3	0	1 639	614	1 413	8	6	6
envoyer	1 635	10	7	184	1 434	1 601	1 553	10	5	5
disparaître	1 620	6	2	82	1 530	731	1 176	7	2	2
élever	1 619	22	20	166	1 411	866	1 626	18	8	8
cacher	1 616	166	21	327	1 102	1 101	1 465	12	7	7
oublier	1 564	49	36	202	1 277	635	1 507	10	10	10
agir	1 538	0	28	253	1 257	1 133	1 110	4	4	4
apercevoir	1 529	6	94	199	1 230	861	1 855	4	4	4
remplir	1 488	20	10	219	1 239	1 185	1 391	8	3	3
tourner	1 486	80	12	98	1 296	896	1 472	21	14	14
apporter	1 459	27	31	124	1 277	1 003	1 358	3	4	4
produire	1 436	31	6	128	1 271	1 082	1 333	9	6	6
battre	1 424	27	40	175	1 182	726	1 169	15	8	8
toucher	1 418	17	99	127	1 175	659	1 210	15	8	8
compter	1 415	6	4	168	1 237	429	1 021	17	7	7
tuer	1 415	80	10	291	1 034	742	1 167	7	7	7
traverser	1 401	9	70	142	1 180	494	1 603	6	7	7
remettre	1 393	7	13	159	1 214	707	1 267	25	15	15
gagner	1 377	31	24	263	1 059	604	1 120	16	12	12
placer	1 373	41	17	171	1 144	1 441	1 703	16	5	5
lire	1 360	8	47	231	1 074	651	1 351	9	6	6
approcher	1 360	0	23	88	1 249	577	1 368	12	7	7
apprendre	1 335	10	54	162	1 109	926	2 054	5	5	5
courir	1 332	3	8	70	1 251	727	1 293	15	7	7
annoncer	1 329	17	33	89	1 190	715	1 297	6	5	5
emporter	1 318	46	51	56	1 165	803	1 165	15	10	10
établir	1 318	61	6	302	949	1 047	1 182	11	5	5
charger	1 315	33	8	42	1 232	1 118	1 364	19	11	11
retourner	1 303	11	31	60	1 201	657	1 471	20	12	12
suffire	1 296	0	0	53	1 243	324	839	7	4	4
rentrer	1 295	6	41	30	1 218	919	1 256	15	8	8
naître	1 295	206	4	77	1 008	755	1 012	8	5	5
préparer	1 294	73	18	189	1 014	738	1 103	15	8	8
avancer	1 293	5	27	53	1 208	663	1 433	22	12	12
abandonner	1 287	78	43	180	986	762	1 406	11	8	8
écouter	1 271	7	33	244	987	375	1 136	6	5	5
conserver	1 259	32	29	178	1 020	848	1 201	8	8	8
apparaître	1 237	4	4	57	1 172	504	941	4	5	5
décider	1 228	11	3	69	1 145	416	1 065	10	8	8
amener	1 227	13	14	179	1 021	677	1 036	10	9	9
expliquer	1 221	3	12	237	969	589	869	9	7	7
exister	1 220	1	1	88	1 130	1 152	1 005	5	3	3
posséder	1 212	3	6	77	1 126	431	1 180	5	4	4
étendre	1 207	116	24	63	1 004	807	1 267	16	9	9
valoir	1 204	0	2	33	1 169	583	1 069	6	3	3
accompagner	1 204	5	22	140	1 037	859	1 129	8	6	6
retirer	1 200	62	11	93	1 034	862	1 456	16	9	9

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
fermer	1 199	8	27	124	1 040	551	1 032	11	5	5
achever	1 198	42	50	127	979	386	912	6	6	6
causer	1 193	2	41	68	1 082	770	1 209	10	9	9
régner	1 192	0	8	101	1 083	358	694	6	4	4
défendre	1 190	20	24	392	754	655	869	10	7	7
représenter	1 186	18	49	78	1 041	521	1 163	16	9	9
éclater	1 147	0	2	8	1 137	307	643	11	7	7
réunir	1 146	137	21	106	882	794	1 177	8	8	8
saisir	1 138	17	17	170	934	649	1 381	14	8	8
travailler	1 133	31	30	24	1 048	454	881	15	9	9
prononcer	1 122	90	58	220	754	573	920	8	6	6
sauver	1 120	12	4	354	750	691	767	9	6	6
éprouver	1 118	13	4	126	975	842	1 292	7	4	4
tromper	1 115	26	5	205	879	597	882	7	5	5
cesser	1 114	1	3	63	1 047	190	830	3	2	2
poser	1 113	148	26	88	851	656	1 214	18	8	8
distinguer	1 094	63	1	219	811	685	782	7	6	6
atteindre	1 082	12	11	234	825	768	1 036	12	5	5
payer	1 082	32	12	187	851	519	889	18	11	11
monter	1 064	11	0	80	973	2 534	1 752	32	5	5
souffrir	1 058	2	9	13	1 034	468	1 057	8	6	6
couvrir	1 056	4	11	88	953	815	1 083	19	4	4
songer	1 051	1	30	84	936	379	965	3	3	3
fixer	1 049	216	19	133	681	797	926	19	14	14
terminer	1 034	127	15	85	807	686	961	6	4	4
sonner	1 029	2	11	44	972	248	466	9	5	5
permettre	1 025	7	13	23	982	1 085	1 211	5	5	5
remarquer	1 019	5	11	136	867	566	1 212	6	4	4
rire	1 000	0	0	547	453	246	365	4	5	5
composer	1 000	29	8	56	907	923	1 252	6	6	6
épouser	994	4	12	194	784	367	559	4	4	4
dormir	980	0	4	220	756	328	602	5	2	2
chanter	977	17	40	68	852	375	799	8	6	6
oser	974	2	0	5	967	143	895	6	5	5
employer	965	23	9	135	798	664	890	7	6	6
pleurer	963	3	28	37	895	376	783	11	7	7
ignorer	961	26	35	48	852	359	1 161	4	2	2
ressembler	959	0	1	21	937	792	862	3	3	3
relever	957	43	21	168	725	626	1 200	25	10	10
brûler	947	34	2	63	848	439	773	14	13	13
essayer	945	3	5	96	841	320	774	7	6	6
échapper	942	9	2	202	729	861	987	6	3	3
nommer	936	226	8	95	607	2 104	1 451	10	5	5
attacher	931	55	10	92	774	992	1 280	13	8	8
découvrir	930	21	16	187	706	497	1 006	10	7	7
manger	930	14	23	102	791	477	787	13	8	8
refuser	927	5	16	122	784	808	971	9	3	3
assurer	913	44	4	127	738	450	1 437	16	10	10

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
engager	898	38	2	98	760	480	794	10	7	7
poursuivre	882	18	30	127	707	603	850	6	4	4
agiter	876	37	29	22	788	545	827	10	4	4
mener	865	9	8	120	728	625	804	14	8	8
réussir	856	9	3	16	828	301	713	6	5	5
accepter	843	11	21	186	625	783	961	5	1	1
écouler	840	30	1	27	782	235	489	7	4	4
répéter	840	27	18	58	737	395	905	8	6	6
espérer	837	6	33	174	624	240	895	4	2	2
partager	836	24	14	120	678	468	622	13	6	6
éclairer	833	39	20	102	672	509	675	11	10	10
durer	829	0	0	89	740	1 087	445	2	1	1
traiter	819	12	0	122	685	454	710	10	7	7
dresser	816	71	8	50	687	402	738	15	9	9
sourire	813	0	23	0	790	345	721	4	1	1
exprimer	806	14	19	154	619	468	775	8	7	7
briser	805	68	10	116	611	526	819	13	9	9
éloigner	803	39	6	146	612	701	888	9	4	4
commander	799	4	0	51	744	496	836	11	4	4
craindre	799	1	46	147	605	365	1 228	7	9	9
briller	794	0	0	40	754	215	496	4	2	2
soutenir	790	25	12	248	505	791	1 216	12	8	8
retenir	782	33	36	279	434	651	870	17	8	8
augmenter	782	10	4	97	671	265	592	4	5	5
empêcher	782	1	6	112	663	336	617	5	2	2
secouer	775	7	36	41	691	229	489	10	5	5
soulever	773	24	35	98	616	408	776	12	8	8
obtenir	772	31	2	308	431	1 100	1 157	4	4	4
coucher	771	20	0	57	694	501	741	16	8	8
répandre	770	29	11	80	650	465	719	13	8	8
deviner	768	2	14	182	570	435	761	5	5	5
embrasser	766	0	5	131	630	406	633	7	3	3
entourer	761	1	12	11	737	1 074	1 203	8	5	5
considérer	755	23	14	117	601	531	942	4	2	2
habiter	750	4	0	56	690	316	710	4	6	6
attirer	749	1	0	107	641	632	711	6	3	3
mêler	743	18	11	57	657	920	797	16	6	6
inspirer	742	21	5	49	667	526	900	7	6	6
enlever	732	38	8	154	532	519	924	12	6	6
tendre	728	0	20	33	675	700	681	11	3	3
subir	723	3	7	127	586	303	650	4	2	2
remonter	717	5	39	56	617	282	725	16	11	11
marquer	717	48	8	40	621	303	564	20	11	11
fournir	713	17	1	122	573	629	607	9	3	3
étonner	709	83	0	49	577	671	1 012	5	6	6
résister	709	0	0	255	454	268	405	3	2	2
lancer	706	14	6	64	622	465	678	27	8	8
trembler	705	1	26	8	670	194	474	8	8	8

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
accorder	705	8	9	43	645	627	771	8	3	3
indiquer	704	53	19	84	548	805	894	6	4	4
presser	704	39	0	54	611	325	568	14	7	7
exercer	702	34	12	106	550	398	662	10	10	10
dominer	700	3	31	36	630	278	638	7	5	5
crier	693	0	47	38	608	279	573	11	6	6
entraîner	692	10	24	85	573	712	683	10	7	7
ramener	686	13	12	132	529	417	598	13	8	8
accomplir	681	50	7	147	477	389	580	4	4	4
reposer	679	10	4	43	622	337	594	17	7	7
interrompre	669	39	1	64	565	302	491	4	5	5
exécuter	667	50	6	143	468	497	667	6	5	5
disposer	666	86	9	35	536	345	739	10	9	9
couler	665	8	5	26	626	228	475	13	8	8
hésiter	664	0	4	111	549	151	459	4	4	4
résoudre	662	24	0	85	553	283	490	10	7	7
repousser	661	30	9	85	537	312	707	12	9	9
contenir	656	39	34	138	445	719	1 022	5	4	4
mériter	654	14	2	47	591	421	828	5	4	4
ordonner	653	19	2	24	608	189	604	8	5	5
détruire	650	32	5	176	437	380	661	7	6	6
réveiller	648	13	6	83	546	324	521	5	4	4
imaginer	642	6	5	171	460	228	745	5	5	5
prier	640	1	6	60	573	245	639	5	2	2
rouler	639	23	0	23	593	214	492	25	15	15
prouver	639	3	2	104	530	387	1 012	4	4	4
livrer	639	65	9	183	382	1 208	858	7	3	3
examiner	636	1	31	167	437	239	625	2	1	1
rejoindre	632	1	2	242	387	263	409	6	5	5
observer	631	15	20	132	464	422	693	7	6	6
concevoir	631	21	3	129	478	299	702	4	4	4
choisir	630	51	10	129	440	899	820	3	3	3
allumer	630	69	4	61	496	275	611	9	6	6
marcher	629	0	50	10	569	1 565	884	15	6	6
recommencer	627	3	1	85	538	195	312	6	5	5
revoir	625	1	1	283	340	232	304	7	7	7
rapporter	622	12	12	54	544	567	666	13	5	5
baisser	619	32	27	19	541	260	603	9	7	7
convenir	618	43	0	17	558	288	872	8	6	6
accuser	618	11	8	41	558	347	735	6	7	7
juger	618	15	17	138	448	951	962	6	5	5
vendre	617	19	8	161	429	450	610	6	4	4
précipiter	617	21	16	9	571	364	712	12	9	9
retentir	609	0	4	25	580	219	342	3	1	1
appartenir	607	0	0	48	559	1 363	1 235	4	1	1
rompre	598	22	8	136	432	270	519	12	6	6
rapprocher	597	16	6	95	480	388	585	16	4	4
hausser	593	2	30	15	546	92	222	7	4	4

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
serrer	592	16	18	42	516	283	511	16	12	12
jouir	589	1	4	51	533	294	538	3	4	4
sauter	588	3	16	37	532	200	649	20	8	8
créer	586	27	4	96	459	421	546	6	8	8
arracher	584	17	6	128	433	534	689	14	8	8
intéresser	581	8	0	76	497	271	448	9	9	9
désirer	580	8	0	46	526	170	701	2	3	3
devoir	579	3	1	0	575	15 053	4 673	6	3	3
admirer	576	10	11	134	421	342	571	4	5	5
révéler	570	6	7	61	496	368	468	8	3	3
visiter	568	2	14	259	293	237	440	6	2	2
couper	567	29	7	77	454	312	517	25	11	11
étouffer	565	54	25	71	415	215	342	9	9	9
promener	562	3	26	25	508	548	645	9	7	7
condamner	558	12	3	60	483	470	735	9	4	4
accueillir	557	6	0	22	529	588	526	4	3	3
troubler	552	11	0	91	450	274	356	9	5	5
obéir	551	1	1	63	486	264	501	5	4	4
éveiller	550	9	6	72	463	309	470	7	8	8
parcourir	548	7	19	112	410	242	618	4	4	4
acheter	541	18	7	141	375	371	497	5	5	5
respirer	537	1	10	34	492	183	436	5	7	7
accourir	531	8	1	20	502	144	439	1	1	1
imposer	529	15	2	55	457	475	624	15	9	9
élancer	529	9	6	39	475	276	638	8	5	5
attaquer	526	12	1	182	331	639	604	12	3	3
préférer	524	6	9	22	487	143	410	2	3	3
parvenir	523	22	0	58	443	653	1 052	5	5	5
supporter	523	4	5	204	310	250	360	7	4	4
profiter	518	0	1	50	467	293	422	4	3	3
constituer	517	33	6	50	428	248	488	8	6	6
tenter	516	9	0	137	370	373	419	4	3	3
glisser	515	5	19	27	464	265	544	16	10	10
importer	512	3	0	1	508	375	657	7	3	3
transporter	511	13	3	102	393	332	464	9	4	4
développer	510	27	4	79	400	232	520	6	6	6
joindre	509	7	30	63	409	240	522	11	7	7
diriger	507	5	0	110	392	1 538	989	8	2	2
chasser	505	26	5	107	367	271	471	14	7	7
plaindre	501	0	6	151	344	372	592	6	3	3
dépasser	499	1	10	31	457	177	440	8	6	6
éviter	498	2	10	243	243	139	301	8	6	6
trahir	498	4	6	72	416	216	375	10	4	4
rêver	497	8	16	27	446	228	501	7	7	7
emmener	494	2	18	99	375	243	384	4	4	4
écarter	493	35	15	92	351	254	526	13	7	7
exciter	492	4	6	86	396	337	491	9	5	5
réclamer	490	0	15	85	390	275	479	9	4	4

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
interroger	489	1	19	121	348	253	410	4	3	3
renverser	482	87	9	69	317	289	589	13	8	8
enfoncer	477	56	13	15	393	337	541	11	7	7
traîner	476	6	29	26	415	272	565	17	9	9
témoigner	475	1	3	35	436	269	559	7	3	3
contempler	472	4	21	128	319	155	380	4	3	3
consentir	472	3	2	57	410	182	514	4	3	3
forcer	470	25	2	70	373	135	436	14	9	9
boire	469	0	0	203	266	143	239	4	2	2
reculer	468	6	7	14	441	171	424	9	7	7
menacer	463	24	16	20	403	629	586	4	3	3
prolonger	461	78	8	75	300	349	398	5	6	6
prévoir	461	23	16	149	273	311	426	7	7	7
exposer	461	21	6	95	339	457	639	8	3	3
déposer	459	4	8	57	390	335	576	10	6	6
réserver	459	25	1	12	421	397	465	12	8	8
envahir	457	1	1	36	419	206	378	3	3	3
saluer	457	2	22	88	345	176	524	6	5	5
surprendre	457	40	0	96	321	737	686	5	5	5
fuir	455	3	12	59	381	192	400	10	5	5
diviser	454	7	5	35	407	280	371	6	6	6
franchir	453	17	10	116	310	179	494	4	4	4
recueillir	449	28	6	119	296	324	488	5	3	3
étudier	447	14	15	123	295	200	491	5	4	4
appliquer	445	31	9	89	316	444	441	7	5	5
signer	444	28	5	67	344	411	406	6	4	4
marier	436	10	5	213	208	251	243	9	4	4
peser	436	1	0	14	421	160	278	10	5	5
réduire	436	27	3	68	338	305	445	11	9	9
remuer	434	15	16	53	350	187	318	9	8	8
effacer	433	15	2	60	356	243	317	13	7	7
blesser	432	34	0	41	357	484	497	6	5	5
respecter	431	17	3	62	349	248	429	3	3	3
renouveler	430	17	5	75	333	267	336	9	6	6
adorer	430	25	4	28	373	190	437	3	2	2
éteindre	424	6	4	52	362	229	377	6	6	6
rassurer	423	19	1	103	300	188	323	2	2	2
désigner	422	38	16	44	324	407	464	6	4	4
acquérir	422	13	3	61	345	317	469	8	7	7
abattre	421	52	4	64	301	324	428	11	3	3
pencher	421	77	0	18	326	425	585	6	4	4
envelopper	420	42	9	16	353	365	603	17	7	7
périr	420	0	0	48	372	177	295	2	3	3
dévorer	417	5	6	44	362	264	347	8	5	5
peindre	416	49	0	109	258	387	395	12	8	8
douter	415	0	1	124	290	463	455	4	3	3
combattre	415	1	0	142	272	204	347	7	4	4
céder	415	6	31	96	282	466	450	9	4	4

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
constater	413	8	9	133	263	156	466	3	5	5
jurer	412	0	4	6	402	90	459	10	5	5
opérer	408	21	4	87	296	214	275	8	7	7
détourner	408	16	13	56	323	204	383	11	3	3
fonder	407	8	0	79	320	463	440	6	3	3
percer	407	37	14	78	278	243	303	12	8	8
regretter	405	4	3	58	340	221	482	3	3	3
consulter	404	9	0	135	260	168	266	4	4	4
lier	404	39	2	31	332	379	453	13	7	7
animer	402	23	1	37	341	258	369	10	7	7
confondre	400	12	8	64	316	285	338	9	7	7
reparaître	400	1	3	46	350	205	294	3	3	3
construire	399	72	3	63	261	288	483	4	5	5
affecter	398	11	14	17	356	179	389	11	4	4
renfermer	398	21	9	22	346	469	542	3	3	3
réfléchir	397	28	4	31	334	193	377	10	8	8
incliner	397	45	12	2	338	299	514	8	7	7
rejeter	396	12	11	42	331	237	443	13	8	8
déterminer	394	22	0	91	281	208	314	8	4	4
figurer	394	3	4	12	375	247	577	6	4	4
emparer	393	0	0	21	372	568	497	3	1	1
déchirer	393	16	4	30	343	208	416	13	9	9
nourrir	392	23	5	73	291	254	396	13	8	8
maintenir	390	8	0	157	225	316	379	11	6	6
bâtir	390	77	3	31	279	263	380	9	9	9
asseoir	383	118	2	198	65	2 773	1 366	5	3	3
commettre	383	16	1	78	288	443	505	7	5	5
organiser	383	27	4	66	286	193	265	10	6	6
installer	382	10	0	26	346	288	354	14	6	6
redoubler	380	14	10	8	348	92	260	6	7	7
retomber	380	0	4	33	343	231	320	8	4	4
renoncer	379	0	0	97	282	226	330	3	2	2
entretenir	378	8	6	98	266	334	464	9	8	8
lutter	377	0	3	175	199	161	184	5	3	3
vaincre	376	35	0	109	232	256	404	5	4	4
admettre	375	16	8	92	259	730	509	6	3	3
ébranler	374	17	7	40	310	148	286	5	7	7
aider	372	0	5	93	274	471	351	5	2	2
pendre	371	19	0	17	335	229	383	8	6	6
enfermer	371	75	5	52	239	465	515	11	5	5
souffler	369	3	12	26	328	117	274	13	9	9
réaliser	367	19	2	96	250	205	237	7	8	8
amuser	366	4	2	51	309	246	294	6	5	5
grandir	365	0	6	7	352	108	291	9	6	6
veiller	362	0	1	49	312	96	273	5	4	4
introduire	361	2	10	98	251	581	422	6	2	2
écraser	360	34	9	34	283	240	357	16	9	9
célébrer	360	11	3	70	276	174	228	4	4	4

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
dessiner	358	16	5	27	310	178	317	7	6	6
consacrer	356	19	5	56	276	495	366	7	4	4
user	355	8	1	42	304	180	334	8	6	6
protéger	354	9	7	107	231	362	325	8	5	5
dîner	354	0	2	0	352	154	225	2	1	1
opposer	352	52	1	43	256	722	559	11	3	3
inviter	348	6	2	31	309	134	268	6	4	4
arranger	348	14	6	77	251	228	306	15	7	7
multiplier	346	13	6	61	266	114	268	7	6	6
renvoyer	343	7	4	69	263	316	349	10	5	5
déployer	341	17	8	30	286	172	315	8	7	7
diminuer	341	8	7	37	289	146	282	5	6	6
souhaiter	341	11	4	30	296	228	348	3	2	2
succéder	340	0	0	14	326	669	377	2	1	1
calmer	337	2	0	114	221	96	219	7	4	4
manifester	336	4	0	31	301	145	265	11	5	5
ranger	336	21	1	63	251	269	374	18	5	5
détacher	334	5	4	21	304	234	480	12	6	6
précéder	332	26	20	4	282	499	588	6	6	6
haïr	331	4	0	43	284	106	249	3	4	4
accroître	330	0	6	53	271	128	231	2	3	3
goûter	329	2	5	67	255	146	247	7	4	4
imiter	328	4	15	90	219	130	245	4	3	3
dégager	326	21	6	65	234	255	332	15	8	8
tressaillir	324	0	3	11	310	54	309	1	1	1
ôter	322	2	7	41	272	185	398	8	5	5
ennuyer	320	16	0	37	267	175	202	4	5	5
assister	320	5	0	211	104	880	309	3	2	2
lâcher	319	3	3	44	269	126	249	11	5	5
aborder	315	1	11	55	248	146	295	7	4	4
séparer	314	25	4	8	277	1 605	803	12	6	6
venger	311	1	2	150	158	148	178	6	2	2
refermer	311	18	7	22	264	208	335	6	6	6
rougir	309	2	1	7	299	139	349	4	7	7
bouger	308	0	0	2	306	83	175	9	5	5
transformer	307	29	8	51	219	225	280	7	4	4
suspendre	305	57	4	30	214	288	388	8	4	4
rétablir	305	4	5	108	188	142	181	7	4	4
épargner	304	2	0	33	269	172	245	11	5	5
décrire	303	9	9	113	172	228	301	4	2	2
conclure	303	28	0	31	244	138	366	10	6	6
louer	302	7	6	74	215	179	291	7	4	4
voler	301	1	0	34	266	379	328	11	3	3
endormir	301	26	1	31	243	290	345	6	6	6
tracer	300	56	7	42	195	269	318	5	5	5
régler	300	11	6	80	203	111	196	12	7	7
gouverner	299	2	1	50	246	173	282	6	3	3
adopter	299	25	2	47	225	516	479	4	3	3

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
procéder	299	0	4	69	226	161	166	3	3	3
redouter	299	18	11	35	235	321	314	2	2	2
provoquer	299	2	2	81	214	263	362	6	3	3
allonger	296	39	9	10	238	124	298	18	11	11
dissiper	295	2	0	55	238	116	233	8	5	5
inquiéter	295	0	2	52	241	577	318	6	3	3
embarrasser	292	65	0	31	196	311	276	8	6	6
taire	292	0	0	10	282	185	242	6	3	3
flotter	290	0	0	6	284	69	201	7	4	4
creuser	289	45	7	34	203	204	353	9	5	5
ressentir	288	6	0	41	241	371	431	4	3	3
effrayer	288	45	1	53	189	518	387	3	3	3
disputer	288	4	5	42	237	203	256	9	7	7
modifier	287	14	3	59	211	166	246	2	3	3
loger	284	18	0	35	231	101	214	11	8	8
prévenir	284	10	8	133	133	361	325	6	2	2
avertir	283	7	1	51	224	197	364	6	4	4
confier	283	0	0	51	232	564	560	4	2	2
guérir	283	3	0	61	219	119	167	9	6	6
dater	282	0	1	4	277	223	186	5	4	4
rassembler	281	19	10	55	197	203	363	9	6	6
situer	281	30	0	4	247	563	376	6	3	3
plaire	278	0	2	136	140	1 151	514	6	3	3
apprécier	277	10	4	102	161	168	227	5	4	4
épuiser	276	21	1	26	228	201	280	9	5	5
promettre	276	11	3	10	252	1 184	581	10	4	4
enfuir	275	1	1	43	230	182	321	2	2	2
inventer	274	5	3	36	230	161	293	5	5	5
soupçonner	274	0	2	54	218	144	318	4	5	5
confirmer	272	6	1	41	224	211	318	8	6	6
surveiller	271	8	9	89	165	126	193	3	3	3
subsister	269	0	2	31	236	65	142	3	3	3
supprimer	269	10	11	47	201	153	235	8	5	5
enseigner	268	5	0	31	232	132	250	4	2	2
danser	268	0	7	14	247	111	201	9	3	3
estimer	267	16	3	19	229	195	415	6	5	5
citer	267	24	1	146	96	712	233	4	4	4
flatter	266	7	2	50	207	180	320	6	3	3
justifier	265	6	2	75	182	185	226	9	6	6
entrevoir	265	16	1	57	191	83	216	3	5	5
aboutir	265	0	4	44	217	161	200	6	3	3
pénétrer	264	24	0	56	184	1 018	610	8	3	3
triompher	263	0	9	34	220	106	200	4	5	5
compléter	262	0	6	60	196	100	197	7	2	2
conquérir	261	23	2	78	158	197	261	4	3	3
rechercher	260	15	4	58	183	139	251	7	4	4
adresser	259	1	0	24	234	1 256	1 095	5	3	3
mesurer	259	15	5	74	165	106	198	10	9	9

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
applaudir	258	6	2	10	240	131	252	7	5	5
discuter	258	3	8	106	141	226	232	7	5	5
différer	255	1	1	18	235	103	216	4	2	2
dissimuler	255	21	6	78	150	202	219	10	3	3
détester	254	11	2	6	235	82	195	3	4	4
protester	253	0	0	47	206	70	207	6	1	1
ramasser	252	17	15	47	173	130	324	11	8	8
soumettre	252	8	0	41	203	929	587	3	3	3
mépriser	251	5	1	38	207	98	230	4	2	2
apaiser	250	16	3	59	172	94	186	5	4	4
proposer	249	20	5	51	173	1 082	541	6	4	4
satisfaire	248	0	0	145	103	144	140	8	3	3
évanouir	248	14	0	1	233	193	219	5	5	5
gâter	247	29	0	29	189	129	174	5	4	4
déclarer	246	12	0	5	229	1 495	811	6	2	2
frémir	246	0	0	9	237	78	165	2	2	2
favoriser	246	7	2	53	184	174	195	3	1	1
insister	245	0	1	80	164	90	180	2	1	1
affirmer	245	0	6	48	191	209	448	4	1	1
baiser	245	0	9	0	236	124	256	8	4	4
redresser	244	2	3	8	231	78	283	7	6	6
accabler	244	14	1	22	207	356	295	3	3	3
croiser	241	0	8	9	224	104	252	9	4	4
étaler	241	4	9	23	205	160	226	14	6	6
attribuer	240	1	3	59	177	489	453	3	4	4
risquer	240	0	1	73	166	115	162	5	2	2
réfugier	239	10	0	44	185	170	265	2	2	2
distribuer	238	14	8	23	193	180	244	9	8	8
émouvoir	237	54	0	25	158	571	404	2	3	3
publier	236	16	0	35	185	548	331	2	1	1
obliger	235	9	3	14	209	641	397	5	5	5
réparer	235	5	3	119	108	85	188	4	4	4
regagner	235	0	6	65	164	119	201	4	4	4
pratiquer	235	26	4	28	177	211	281	6	7	7
voyager	234	0	7	59	168	85	191	3	1	1
instruire	233	15	4	86	128	146	172	8	7	7
irriter	232	44	3	27	158	217	275	4	3	3
traduire	232	4	3	49	176	236	258	6	4	4
échouer	231	12	1	22	196	101	181	8	4	4
planter	231	49	4	34	144	234	304	16	9	9
combler	230	4	2	29	195	168	196	6	3	3
honorer	229	7	1	44	177	155	220	8	5	5
reproduire	227	6	4	51	166	194	245	8	7	7
appuyer	226	125	26	52	23	1 457	819	7	3	3
ruiner	226	27	1	40	158	165	252	8	6	6
cultiver	226	23	4	37	162	126	194	10	6	6
nier	225	3	1	74	147	130	177	3	1	1
repartir	225	1	2	31	191	115	193	4	5	5

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
froncer	223	2	11	1	209	37	88	3	3	3
pâlir	222	4	4	3	211	79	235	5	3	3
fâcher	220	2	1	41	176	103	218	6	3	3
approuver	218	1	1	17	199	270	272	2	1	1
murmurer	216	1	11	12	192	91	190	6	5	5
frissonner	214	0	5	11	198	55	138	3	2	2
redevenir	214	16	0	13	185	193	214	1	1	1
consoler	213	0	0	113	100	246	196	3	3	3
attester	212	0	5	9	198	77	167	6	4	4
pardonner	212	1	1	62	148	534	290	9	2	2
charmer	212	6	0	29	177	119	182	3	2	2
piquer	209	11	3	21	174	115	186	31	13	13
attendrir	209	57	2	23	127	136	177	4	5	5
punir	207	8	1	47	151	264	249	4	3	3
débarquer	207	15	2	35	155	112	213	8	7	7
plonger	206	1	0	9	196	600	395	10	3	3
prêter	206	2	1	10	193	833	530	5	4	4
compromettre	205	7	0	79	119	158	200	6	2	2
sacrifier	205	1	0	52	152	172	195	13	3	3
envoler	205	9	3	23	170	83	144	4	2	2
fumer	204	0	3	18	183	62	173	6	3	3
délivrer	204	4	1	54	145	170	222	7	5	5
conseiller	204	0	2	10	192	170	186	3	2	2
déborder	203	4	16	8	175	77	181	12	8	8
désespérer	203	13	0	18	172	90	194	7	10	10
égarer	203	18	1	36	148	130	193	6	6	6
imprimer	201	30	1	33	137	212	223	9	3	3
abaisser	201	16	7	18	160	131	201	11	9	9
présider	200	0	2	15	183	93	170	5	3	3
armer	200	30	0	23	147	442	394	13	6	6
surmonter	200	42	9	36	113	137	247	3	4	4
abriter	200	22	13	30	135	121	176	7	5	5
défier	199	2	8	37	152	128	140	6	3	3
contracter	198	44	1	14	139	128	192	7	6	6
abuser	198	2	2	20	174	136	180	4	2	2
essuyer	197	5	13	32	147	194	245	7	5	5
noyer	197	40	4	6	147	181	230	10	10	10
plier	196	32	5	36	123	89	213	11	11	11
gonfler	195	13	2	3	177	117	174	13	8	8
concerner	195	1	38	1	155	86	244	1	1	1
encourager	194	4	0	36	154	173	218	4	2	2
supposer	194	8	6	67	113	684	521	4	2	2
disperser	193	19	1	15	158	185	238	10	8	8
circuler	193	0	5	3	185	45	181	4	3	3
assembler	193	26	0	15	152	97	213	6	4	4
caresser	191	4	6	32	149	98	196	3	2	2
dérouler	190	7	3	4	176	105	167	6	5	5
étinceler	190	0	2	6	182	34	97	3	3	3

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
fondre	190	6	0	29	155	101	169	17	8	8
autoriser	189	8	4	14	163	128	195	8	5	5
hâter	188	2	6	52	128	432	245	3	1	1
aspirer	188	1	0	22	165	75	172	7	3	3
négliger	188	5	2	22	159	79	189	4	3	3
doubler	184	4	1	27	152	143	191	15	6	6
succomber	184	0	4	27	153	132	177	3	2	2
border	184	3	6	4	171	203	258	7	2	2
assassiner	183	13	1	42	127	151	187	1	1	1
altérer	183	17	1	24	141	136	165	7	5	5
gronder	183	0	3	18	162	72	136	5	2	2
voter	183	2	0	31	150	114	112	3	2	2
balancer	182	1	12	11	158	81	179	21	5	5
mouiller	182	34	0	7	141	89	143	14	12	12
environner	180	0	0	2	178	121	193	6	3	3
absorber	180	16	0	25	139	366	269	4	6	6
coûter	180	0	0	4	176	688	287	4	1	1
mentir	180	1	0	53	126	133	105	5	4	4
gêner	180	3	0	33	144	213	147	4	2	2
évoquer	179	11	8	29	131	107	129	6	2	2
abonder	178	0	0	0	178	41	84	2	1	1
déjeuner	177	0	2	66	109	99	129	2	2	2
communiquer	177	0	1	36	140	405	283	12	4	4
fatiguer	177	13	2	22	140	182	191	13	10	10
glacer	176	12	2	6	156	66	111	14	12	12
couronner	176	0	2	20	154	153	169	7	5	5
crever	175	5	3	13	154	67	114	11	9	9
invoquer	175	1	11	38	125	108	154	3	2	2
dérober	175	3	0	17	155	94	183	11	6	6
calculer	175	15	5	26	129	99	181	7	6	6
utiliser	174	13	21	47	93	84	109	4	2	2
empoisonner	174	20	2	27	125	106	152	11	6	6
ménager	172	19	5	19	129	133	168	8	5	5
tordre	171	1	10	6	154	79	156	9	8	8
enterrer	171	8	0	20	143	139	126	8	6	6
guetter	170	0	20	22	128	47	125	4	2	2
exaspérer	170	9	5	26	130	87	111	5	5	5
luire	170	1	1	25	143	55	94	4	2	2
élire	170	11	1	10	148	147	179	1	1	1
comporter	169	0	3	6	160	74	187	3	3	3
enchanter	168	39	0	8	121	246	192	4	4	4
débuter	168	0	0	8	160	34	95	6	2	2
retarder	167	3	2	41	121	83	135	6	3	3
résonner	166	0	0	16	150	72	83	2	2	2
éclaircir	166	1	0	48	117	60	80	4	4	4
parer	165	15	2	47	101	238	199	11	2	2
heurter	165	2	9	13	141	267	218	8	8	8
rayonner	165	0	2	0	163	60	96	8	3	3

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
siffler	164	0	12	7	145	42	109	10	6	6
signifier	164	2	2	13	147	316	249	4	2	2
séduire	164	2	0	26	136	117	189	2	1	1
expirer	164	6	2	1	155	55	143	3	3	3
résumer	163	3	4	24	132	79	127	7	5	5
préoccuper	162	7	0	27	128	188	127	3	3	3
caractériser	161	9	2	18	132	86	128	1	1	1
résigner	160	27	1	29	103	176	182	3	3	3
lasser	160	2	0	27	131	100	119	5	4	4
tâcher	160	0	0	1	159	13	137	1	1	1
filer	160	1	9	12	138	63	146	14	6	6
encombrer	160	6	0	5	149	136	168	6	4	4
entamer	160	3	1	56	100	92	96	6	4	4
varier	156	0	1	3	152	50	109	6	4	4
courber	156	19	6	6	125	120	219	8	7	7
élargir	156	18	8	14	116	89	156	9	7	7
dénoncer	156	4	2	28	122	135	158	5	4	4
solliciter	155	2	9	36	108	122	152	8	5	5
jaillir	155	0	2	18	135	45	94	4	5	5
consommer	155	49	0	19	87	79	176	4	3	3
adoucir	154	3	3	40	108	55	123	6	5	5
réprimer	154	7	1	71	75	61	97	4	3	3
rouvrir	154	0	2	13	139	36	136	7	5	5
reporter	154	0	3	22	129	82	117	8	3	3
concentrer	154	10	0	31	113	94	135	9	4	4
exhaler	153	1	14	11	127	91	130	4	3	3
remplacer	153	2	0	40	111	952	391	3	1	1
vider	152	0	7	32	113	65	132	10	3	3
dédaigner	152	6	6	14	126	136	153	2	1	1
refaire	151	3	0	56	92	84	96	9	4	4
persister	151	0	1	4	146	97	146	4	4	4
tourmenter	151	9	0	27	115	100	148	7	6	6
déranger	151	0	0	34	117	77	127	10	4	4
ranimer	150	3	1	42	104	62	95	7	5	5
prodiguer	150	13	3	11	123	107	190	5	4	4
enflammer	148	43	0	10	95	110	169	4	6	6
fabriquer	148	13	0	46	89	82	157	8	6	6
enrichir	146	5	1	33	107	123	135	9	3	3
envisager	145	14	4	27	100	62	105	2	3	3
exagérer	145	14	8	18	105	65	113	7	6	6
dépenser	145	5	0	14	126	69	119	4	3	3
égaler	144	0	3	19	122	75	94	5	3	3
exalter	144	7	5	12	120	105	165	7	5	5
contenter	143	0	1	44	98	516	291	3	4	4
soupirer	143	1	5	7	130	51	138	4	2	2
excuser	143	0	3	44	96	63	130	5	2	2
frotter	142	0	4	9	129	75	160	13	4	4
rattacher	142	1	0	17	124	244	158	7	2	2

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
effectuer	141	4	1	26	110	72	118	2	3	3
garantir	140	0	0	39	101	76	97	10	4	4
errer	140	0	0	30	110	26	99	4	1	1
expédier	140	6	0	21	113	102	145	5	4	4
croître	140	0	0	12	128	37	123	3	2	2
gémir	139	0	13	0	126	54	134	6	5	5
résulter	139	0	0	1	138	502	300	1	1	1
exploiter	139	7	2	38	92	76	95	4	2	2
anéantir	138	7	3	38	90	112	122	6	4	4
contribuer	138	0	0	9	129	501	202	3	1	1
repasser	138	1	13	11	113	112	172	16	5	5
inonder	137	0	1	4	132	92	136	10	1	1
décourager	137	33	0	19	85	122	131	4	2	2
rédiger	137	9	3	29	96	104	176	2	3	3
associer	136	2	4	37	93	172	140	9	5	5
bouleverser	135	1	0	6	128	47	104	4	3	3
résider	134	0	0	17	117	98	93	2	1	1
affaiblir	133	6	2	24	101	79	127	4	6	6
survenir	133	26	2	6	99	180	329	2	1	1
longer	133	0	22	8	103	47	163	2	1	1
refléter	133	3	7	7	116	75	111	4	3	3
démontrer	132	5	1	37	89	140	259	4	2	2
entasser	132	39	1	2	90	89	158	6	3	3
hasarder	132	2	0	33	97	55	104	5	2	2
destiner	132	8	0	1	123	632	442	4	3	3
renaître	132	0	1	36	95	52	59	4	3	3
remporter	131	14	0	19	98	108	136	4	2	2
démentir	131	0	0	17	114	103	141	4	3	3
hurler	131	0	12	4	115	50	116	9	4	4
remercier	130	0	0	1	129	519	383	1	1	1
rallier	130	0	1	43	86	76	139	9	4	4
fouiller	130	3	9	18	100	113	147	10	5	5
crisper	129	34	4	0	91	69	97	9	8	8
correspondre	129	0	0	17	112	100	117	6	3	3
échanger	128	27	2	8	91	834	265	3	1	1
embarquer	127	13	1	51	62	334	145	11	6	6
chauffer	127	10	3	23	91	85	115	11	8	8
accrocher	126	6	0	17	103	165	189	20	8	8
égayer	126	1	1	27	97	88	104	4	4	4
projeter	126	37	4	7	78	135	177	7	5	5
définir	126	14	1	52	59	73	94	5	3	3
semer	126	19	2	23	82	111	131	9	6	6
pleuvoir	125	0	0	22	103	138	86	3	1	1
fréquenter	125	2	5	17	101	90	187	6	5	5
épanouir	125	5	1	1	118	61	124	6	6	6
effleurer	125	0	1	17	107	86	84	6	2	2
professer	125	0	1	6	118	57	133	4	3	3
procurer	124	0	0	34	90	301	290	4	4	4

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
répliquer	124	0	1	0	123	21	125	6	3	3
revêtir	124	4	1	28	91	168	171	8	3	3
ruisseler	124	0	0	3	121	33	69	4	3	3
bondir	124	0	2	7	115	37	140	6	4	4
unir	124	0	4	32	88	317	162	9	2	2
enivrer	123	8	0	11	104	112	116	4	5	5
corriger	123	1	1	47	74	78	90	8	4	4
tremper	123	27	3	21	72	147	170	9	5	5
fleurir	122	0	0	14	108	35	81	7	4	4
fendre	122	4	1	17	100	61	109	11	5	5
planer	122	0	8	5	109	36	83	8	5	5
gravir	122	0	8	23	91	51	184	2	2	2
rôder	121	0	2	20	99	32	66	2	1	1
guider	121	1	2	23	95	141	122	7	3	3
pressentir	121	3	7	24	87	39	116	3	5	5
dépouiller	121	6	2	24	89	165	201	12	4	4
trancher	120	22	0	23	75	72	125	10	7	7
fortifier	120	8	0	40	72	102	131	6	4	4
épouvanter	120	35	0	13	72	198	161	3	4	4
baigner	120	7	3	14	96	93	108	10	8	8
hanter	119	4	0	4	111	67	79	2	2	2
surpasser	119	0	0	17	102	46	138	3	3	3
saigner	119	0	0	8	111	34	65	9	3	3
noter	119	3	6	33	77	93	113	9	5	5
bénir	119	13	4	19	83	137	141	2	2	2
emprunter	119	37	0	11	71	277	186	7	4	4
faciliter	119	0	0	42	77	44	69	1	1	1
vieillir	118	1	8	3	106	51	104	8	3	3
inscrire	118	1	2	24	91	92	108	11	6	6
barrer	118	0	4	10	104	61	100	8	3	3
recouvrer	117	0	0	32	85	45	75	3	3	3
percevoir	117	5	0	15	97	79	110	5	4	4
sécher	117	9	2	18	88	44	111	9	8	8
craquer	116	0	1	10	105	25	66	9	3	3
daigner	116	0	0	0	116	11	99	2	1	1
exclure	116	0	1	17	98	86	108	6	4	4
raisonner	115	5	1	6	103	47	98	7	5	5
vérifier	115	1	0	50	64	48	78	4	3	3
assiéger	115	8	2	21	84	107	127	5	1	1
recourir	115	0	1	71	43	38	41	3	3	3
soulager	114	6	4	29	75	91	129	4	2	2
interdire	114	1	2	15	96	139	151	4	3	3
défiler	113	0	7	7	99	52	83	9	5	5
débattre	113	2	2	17	92	113	128	7	2	2
contraindre	113	8	0	19	86	70	135	6	4	4
méditer	113	0	4	8	101	51	148	4	3	3
cueillir	113	2	5	38	68	40	97	6	3	3
éblouir	112	7	1	13	91	88	126	2	2	2

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
laver	112	4	1	46	61	65	86	11	7	7
découper	112	15	2	13	82	66	139	6	4	4
accumuler	112	22	5	8	77	68	124	4	3	3
assombrir	111	1	2	7	101	43	61	4	4	4
déboucher	110	0	5	5	100	48	127	7	7	7
révolter	110	10	1	25	74	246	136	5	4	4
blâmer	109	0	0	29	80	83	119	2	2	2
camper	109	1	0	1	107	54	110	10	6	6
ensuivre	109	0	0	2	107	21	81	2	1	1
siéger	108	0	7	13	88	51	67	3	1	1
avaler	108	1	4	23	80	50	89	4	5	5
tailler	108	37	0	2	69	98	161	10	8	8
flamber	108	0	0	2	106	27	50	10	4	4
confesser	107	0	1	27	79	74	153	6	3	3
intervenir	107	2	1	48	56	140	138	5	3	3
claquer	106	1	1	6	98	19	39	11	6	6
casser	106	0	0	38	68	57	66	16	6	6
réjouir	106	1	2	22	81	279	185	2	2	2
resplendir	105	0	3	2	100	30	58	2	2	2
récompenser	105	3	0	16	86	120	113	3	2	2
épier	105	0	5	13	87	47	122	3	3	3
devancer	105	0	4	6	95	50	101	4	1	1
engendrer	105	0	0	11	94	74	98	3	1	1
graver	104	14	0	16	74	102	116	8	5	5
replier	104	13	5	2	84	38	145	10	8	8
pincer	104	11	1	8	84	25	61	8	7	7
flairer	104	0	16	6	82	33	70	3	5	5
habiller	104	3	1	49	51	196	150	12	2	2
sculpter	103	53	1	6	43	87	185	6	6	6
nouer	103	28	2	12	61	99	126	12	8	8
acheminer	103	0	1	10	92	68	85	5	3	3
racheter	103	3	1	34	65	55	75	7	3	3
affliger	103	10	0	14	79	148	144	4	3	3
passionner	103	6	0	14	83	65	76	5	4	4
concourir	102	0	2	24	76	45	66	2	2	2
relire	102	1	6	24	71	61	111	2	3	3
braver	102	1	5	41	55	41	75	2	2	2
assigner	102	4	1	11	86	109	99	3	4	4
écrouler	102	12	0	2	88	91	117	5	3	3
refroidir	102	10	0	15	77	36	76	8	7	7
contester	101	5	0	28	68	68	104	6	3	3
transmettre	101	8	0	5	88	228	169	6	2	2
impliquer	101	0	1	3	97	56	91	3	1	1
fonctionner	101	0	1	19	81	37	60	5	2	2
soigner	101	6	0	39	56	73	76	10	5	5
conjurer	101	2	1	32	66	81	61	5	3	3
ronger	101	4	0	5	92	99	153	7	3	3
ralentir	100	1	1	12	86	38	97	9	4	4

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
décréter	100	0	2	11	87	49	116	2	3	3
seconder	99	1	1	24	73	82	104	2	2	2
voiler	99	2	4	7	86	60	84	7	3	3
ressusciter	99	5	1	21	72	38	64	5	5	5
amasser	98	13	0	7	78	62	99	5	3	3
contrarier	98	1	0	21	76	51	78	5	3	3
distraire	98	1	0	61	36	38	39	7	4	4
déconcerter	98	12	0	12	74	58	76	4	3	3
accoucher	98	0	0	14	84	32	53	3	2	2
séjourner	98	0	3	17	78	44	79	2	1	1
souper	97	0	0	34	63	40	70	2	1	1
paralyser	97	3	4	9	81	95	80	3	1	1
apprêter	97	1	0	5	91	52	67	9	4	4
prévaloir	97	0	0	19	78	57	46	3	1	1
grimper	97	1	1	5	90	45	157	6	4	4
maudire	96	7	13	13	63	100	107	3	3	3
raser	95	22	8	11	54	77	126	11	7	7
orner	95	0	0	0	95	609	380	5	2	2
accentuer	95	16	9	5	65	66	93	5	4	4
étrangler	95	6	3	22	64	45	82	6	5	5
dompter	95	7	3	32	53	53	64	7	3	3
coller	95	19	2	4	70	95	128	24	10	10
brouiller	95	0	0	21	74	53	57	10	3	3
ravager	95	4	3	9	79	78	119	3	3	3
déplacer	94	7	2	15	70	60	93	15	5	5
échauffer	94	6	1	13	74	39	104	7	4	4
reprocher	94	0	0	11	83	366	355	2	2	2
enfler	94	8	1	6	79	33	82	8	8	8
offenser	94	4	2	20	68	81	118	5	4	4
débarrasser	94	3	0	6	85	185	159	6	3	3
instituer	94	6	0	8	80	58	112	5	7	7
féliciter	93	3	3	29	58	94	157	3	4	4
masquer	93	19	2	16	56	72	113	10	9	9
abolir	93	3	2	13	75	47	85	3	3	3
recouvrir	92	1	0	5	86	188	177	7	3	3
participer	92	0	3	28	61	57	92	4	3	3
défaire	92	0	3	12	77	92	148	9	2	2
grouper	92	20	1	2	69	105	138	7	9	9
fouetter	92	1	1	10	80	67	97	8	4	4
tinter	91	1	1	13	76	16	34	3	2	2
massacrer	91	3	1	12	75	79	134	6	2	2
assaillir	91	0	0	19	72	105	108	4	2	2
améliorer	91	5	0	26	60	55	73	4	4	4
aggraver	91	1	0	17	73	58	81	2	3	3
consumer	91	5	0	0	86	53	110	7	5	5
hériter	91	1	0	2	88	70	82	3	2	2
gratter	90	1	2	10	77	27	56	17	8	8
fléchir	90	0	1	12	77	29	73	6	3	3

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
resserrer	90	9	1	7	73	42	92	4	5	5
fouler	89	4	5	9	71	80	97	6	8	8
secourir	89	0	1	65	23	50	47	2	1	1
attrister	89	25	1	6	57	147	100	2	2	2
affluer	89	0	0	0	89	13	48	2	1	1
aviser	89	8	2	4	75	196	162	8	6	6
souiller	88	10	1	7	70	76	110	3	4	4
grossir	88	0	3	15	70	38	52	7	4	4
étreindre	87	0	4	4	79	43	88	4	4	4
relier	87	8	1	11	67	117	141	7	5	5
priver	87	1	0	14	72	138	123	6	4	4
désarmer	86	2	1	13	70	65	110	11	4	4
hisser	86	1	5	14	66	34	82	8	5	5
improviser	86	19	2	8	57	51	87	5	6	6
susciter	86	3	1	12	70	92	87	4	5	5
bourdonner	86	0	2	3	81	14	30	3	3	3
contraster	86	1	0	0	85	52	75	5	1	1
copier	86	8	4	7	67	44	85	5	4	4
redescendre	85	1	3	4	77	62	86	7	7	7
questionner	85	0	0	26	59	45	65	3	2	2
convoquer	85	2	1	8	74	124	83	3	3	3
comparer	85	0	0	34	51	358	162	4	2	2
décliner	84	0	5	8	71	19	52	11	3	3
prédire	84	0	1	11	72	72	90	2	2	2
fêter	84	1	0	25	58	28	45	3	1	1
explorer	84	3	5	34	42	55	73	3	2	2
dénouer	84	15	4	8	57	46	85	2	3	3
vibrer	83	0	0	2	81	22	40	5	4	4
prescrire	83	3	1	6	73	86	112	4	4	4
égorger	83	4	2	20	57	66	99	2	1	1
grincer	83	0	0	3	80	14	64	1	1	1
plaisanter	83	0	5	1	77	37	84	4	4	4
affaisser	83	9	0	0	74	81	102	5	3	3
émettre	82	5	4	13	60	67	71	8	3	3
ruer	82	0	5	0	77	34	95	5	4	4
torturer	82	4	0	15	63	55	74	6	3	3
étourdir	82	5	1	16	60	60	85	7	5	5
plisser	82	8	0	2	72	28	44	8	6	6
taper	82	0	5	3	74	27	47	22	7	7
affranchir	81	2	2	24	53	87	92	7	3	3
déchaîner	81	5	2	11	63	67	63	6	4	4
violer	81	1	3	20	57	41	93	5	4	4
tâter	81	0	2	22	57	51	68	6	2	2
mordre	81	1	2	14	64	232	187	12	1	1
sommeiller	81	0	2	9	70	27	58	3	3	3
combiner	81	10	12	11	48	82	110	7	6	6
enchaîner	81	6	0	7	68	70	100	8	5	5
arroser	81	1	5	12	63	91	105	9	3	3

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
rivaliser	80	0	0	22	58	33	42	4	1	1
coiffer	80	7	0	6	67	126	173	12	6	6
accommoder	80	3	1	15	61	73	86	10	7	7
prêcher	80	4	2	16	58	221	125	4	4	4
nettoyer	80	3	2	24	51	28	70	9	3	3
illustrer	80	3	2	6	69	151	109	5	3	3
ronfler	79	0	0	8	71	16	46	3	2	2
exiger	79	0	0	1	78	1 007	392	3	3	3
corrompre	79	1	1	12	65	44	89	6	4	4
choquer	79	2	0	9	68	132	109	4	4	4
nager	79	0	7	0	72	28	75	6	5	5
piller	79	4	3	17	55	98	160	8	5	5
consterner	79	14	0	2	63	63	64	1	1	1
peupler	79	4	2	10	63	100	82	9	2	2
agrandir	79	0	1	14	64	39	100	4	3	3
tarir	78	0	0	7	71	20	41	5	4	4
humilier	78	2	1	20	55	41	67	3	2	2
manier	78	3	6	20	49	42	85	9	6	6
obscurcir	78	3	2	12	61	33	71	5	4	4
mouvoir	78	0	0	7	71	35	73	8	4	4
esquisser	78	3	2	15	58	21	47	6	4	4
froisser	77	6	3	10	58	58	91	6	6	6
classer	77	13	2	12	50	51	56	10	7	7
interpréter	77	4	1	14	58	45	66	6	5	5
déplorer	77	1	2	12	62	39	78	2	1	1
analyser	77	1	1	34	41	31	60	6	3	3
ressortir	77	0	0	10	67	51	104	11	3	3
envier	77	7	0	4	66	64	108	2	3	3
ériger	76	4	0	9	63	59	70	6	3	3
exiler	76	20	1	4	51	70	95	5	3	3
impatienter	76	2	0	13	61	21	45	2	2	2
débiter	75	4	0	8	63	36	90	10	5	5
réchauffer	75	1	2	32	40	18	34	6	6	6
aboyer	75	0	4	0	71	18	71	3	5	5
abîmer	74	11	1	13	49	55	65	7	7	7
formuler	74	11	0	25	38	54	67	4	4	4
attraper	74	1	1	18	54	24	59	12	4	4
blanchir	74	8	0	10	56	31	85	11	7	7
négocier	74	0	0	45	29	81	42	7	3	3
empourprer	74	5	2	2	65	26	37	2	2	2
compliquer	74	0	1	9	64	40	46	6	4	4
démolir	74	6	2	21	45	45	70	6	3	3
dissoudre	73	4	1	19	49	37	73	8	6	6
affronter	73	1	0	37	35	33	41	9	3	3
isoler	73	24	0	4	45	264	190	8	3	3
reconduire	73	0	2	23	48	120	98	6	2	2
embaumer	73	32	0	1	40	41	77	4	6	6
clouer	72	14	2	3	53	49	44	5	5	5

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
raconter	72	2	0	36	34	2 279	591	4	2	2
bercer	72	0	1	10	61	64	87	7	4	4
escorter	72	4	3	7	58	109	126	2	2	2
aligner	71	9	2	2	58	52	64	14	6	6
édifier	71	3	1	12	55	38	50	6	3	3
évacuer	71	0	0	21	50	42	94	8	3	3
renforcer	71	6	0	20	45	54	69	8	7	7
dessécher	71	26	0	1	44	66	94	8	7	7
reconquérir	71	5	1	26	39	33	45	4	3	3
insulter	71	2	2	15	52	90	79	4	2	2
chanceler	70	0	0	4	66	13	60	3	2	2
couver	70	1	1	2	66	27	49	6	4	4
intimider	70	2	0	21	47	40	61	1	1	1
propager	69	2	1	15	51	57	64	4	3	3
proférer	69	0	9	22	38	31	47	1	1	1
encadrer	69	5	0	3	61	76	116	9	6	6
méconnaître	69	3	1	32	33	56	73	2	1	1
acharner	68	8	0	0	60	46	59	5	3	3
déserter	68	3	2	0	63	32	44	4	5	5
mûrir	68	3	2	4	59	27	77	7	4	4
rayer	67	28	4	4	31	57	71	7	6	6
ravir	67	4	0	17	46	172	93	3	1	1
abréger	67	2	4	23	38	32	34	4	4	4
flétrir	67	0	2	13	52	27	57	7	5	5
ajourner	67	1	2	7	57	42	65	2	2	2
refouler	66	9	5	9	43	51	77	7	3	3
dériver	66	1	0	2	63	30	54	7	3	3
proclamer	66	9	0	0	57	396	164	4	1	1
stationner	66	2	2	0	62	22	46	4	3	3
alimenter	66	3	0	13	50	50	74	8	3	3
nuire	66	0	0	27	39	82	62	3	1	1
trotter	65	0	2	7	56	18	31	5	4	4
tacher	65	21	3	2	39	47	69	6	7	7
rabattre	65	1	1	15	48	40	58	12	4	4
côtoyer	65	0	13	8	44	19	58	5	3	3
hocher	65	0	2	0	63	24	45	1	1	1
manoeuvrer	65	1	4	9	51	27	44	6	5	5
expier	65	0	0	11	54	33	47	1	1	1
embellir	64	0	1	7	56	32	69	6	4	4
indigner	64	19	0	8	37	196	104	2	2	2
abdiquer	64	0	0	9	55	20	45	2	2	2
relâcher	64	9	0	5	50	46	91	6	4	4
incendier	64	22	6	3	33	37	71	4	4	4
perfectionner	64	4	0	15	45	38	71	6	3	3
balayer	64	7	4	1	52	58	76	7	4	4
concilier	63	0	0	56	7	75	39	6	2	2
scintiller	63	0	1	0	62	17	47	1	1	1
perpétuer	63	0	0	17	46	43	67	2	3	3

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
flanquer	63	1	2	11	49	103	87	10	5	5
rattraper	63	0	1	22	40	26	41	10	5	5
divertir	63	0	1	26	36	62	53	3	1	1
déceler	62	0	0	3	59	21	40	6	4	4
rafraîchir	62	1	0	9	52	22	43	12	6	6
sillonner	62	6	0	2	54	26	83	3	3	3
éparpiller	62	11	1	2	48	48	66	7	8	8
impressionner	62	1	0	14	47	34	41	4	3	3
enlacer	62	11	0	1	50	40	81	5	5	5
rallumer	62	1	0	14	47	26	58	2	3	3
maîtriser	62	1	1	41	19	29	25	4	2	2
dégoûter	62	4	0	6	52	92	60	4	5	5
naviguer	61	0	6	19	36	21	58	5	2	2
inaugurer	61	5	0	16	40	42	53	4	3	3
défaillir	61	0	1	20	40	14	26	3	2	2
qualifier	61	0	1	17	43	36	47	7	2	2
arrondir	61	3	3	2	53	39	54	10	5	5
engloutir	61	3	0	15	43	66	77	5	5	5
languir	61	0	1	2	58	21	48	7	5	5
ressaisir	61	0	1	23	37	21	34	3	2	2
enfanter	61	1	0	12	48	40	48	4	4	4
répartir	61	6	0	1	54	49	62	13	6	6
assommer	60	6	0	17	37	42	53	3	1	1
sangloter	60	0	1	0	59	16	52	5	4	4
faiblir	60	0	1	13	46	11	26	2	1	1
tolérer	60	1	1	17	41	70	71	3	1	1
limiter	60	2	2	9	47	33	62	11	6	6
désoler	59	0	0	4	55	83	77	4	2	2
feindre	59	0	2	4	53	19	104	3	2	2
épancher	59	3	0	17	39	27	35	3	3	3
initier	59	1	0	24	34	65	45	4	3	3
déguiser	59	3	1	27	28	42	62	5	3	3
émerger	59	1	8	2	48	21	31	4	3	3
ébaucher	59	6	1	7	45	30	52	2	3	3
ensevelir	58	21	1	15	21	187	143	5	5	5
atténuer	58	4	1	19	34	30	48	4	3	3
appesantir	58	4	1	11	42	31	43	5	3	3
palpiter	58	0	0	1	57	20	46	2	2	2
voltiger	58	0	0	7	51	18	42	2	2	2
émigrer	58	3	0	7	48	16	83	3	1	1
ensanglanter	58	32	0	0	26	74	70	4	3	3
brandir	58	0	15	3	40	31	83	3	2	2
afficher	57	7	2	10	38	86	75	8	4	4
décharger	57	1	1	12	43	41	69	13	5	5
balbutier	57	1	5	5	46	24	68	4	4	4
intriguer	57	7	0	7	43	40	57	3	2	2
détendre	57	0	0	4	53	12	62	5	4	4
immoler	57	0	1	15	41	42	48	4	3	3

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
renier	57	0	0	4	53	26	46	3	2	2
articuler	56	9	4	22	21	21	41	6	4	4
dégénérer	56	1	0	3	52	53	46	3	2	2
baptiser	56	3	0	10	43	52	52	6	2	2
évader	56	0	0	28	28	17	37	2	1	1
tempérer	55	13	0	10	32	57	65	5	6	6
restaurer	55	13	0	13	29	54	47	7	6	6
contredire	55	1	0	20	34	30	44	3	4	4
recruter	55	7	1	15	32	38	36	3	2	2
administrer	55	4	0	12	39	75	75	4	4	4
convaincre	55	0	0	1	54	313	259	2	1	1
bouillonner	55	0	0	3	52	19	25	6	2	2
vouer	55	0	0	4	51	181	113	4	2	2
persécuter	55	0	0	6	49	52	76	2	2	2
dénoter	54	0	1	1	52	30	58	2	2	2
sonder	54	0	5	12	37	28	51	3	2	2
foudroyer	54	12	4	3	35	63	73	5	3	3
empoigner	54	0	2	3	49	17	65	7	5	5
regorger	54	0	0	0	54	15	25	5	3	3
régir	54	0	2	3	49	36	60	2	1	1
délibérer	54	0	0	0	54	17	48	5	2	2
éclipser	54	1	0	5	48	37	31	4	4	4
pétiller	53	0	0	1	52	16	47	3	2	2
acclamer	53	3	1	5	44	19	47	2	1	1
habituer	53	1	0	20	32	181	104	3	3	3
tousser	53	0	0	7	46	8	48	2	2	2
hérisser	53	10	2	1	40	129	126	9	4	4
motiver	53	3	0	12	38	41	48	7	3	3
aveugler	53	2	0	2	49	76	78	5	2	2
sévir	53	0	0	14	39	18	41	5	2	2
accoutumer	52	7	0	11	34	330	164	3	2	2
concerter	52	1	0	17	34	18	44	3	2	2
suffoquer	52	4	0	4	44	39	38	3	4	4
rajeunir	52	1	0	2	49	36	41	9	4	4
griser	51	2	0	4	45	53	53	6	4	4
réitérer	51	12	0	3	36	50	62	2	1	1
haleter	51	0	3	3	45	7	36	3	3	3
culbuter	51	3	1	5	42	29	92	8	5	5
rebuter	51	2	1	9	39	35	53	3	2	2
affermir	51	0	0	14	37	27	53	4	3	3
complaire	51	0	0	6	45	34	43	4	1	1
chérir	51	0	1	8	42	15	64	2	2	2
plaider	51	0	0	14	37	111	67	5	1	1
viser	51	1	9	4	37	110	93	6	4	4
cracher	51	0	5	2	44	25	34	9	3	3
engraisser	51	0	0	5	46	26	44	7	4	4
vomir	51	0	6	4	41	28	39	4	2	2
cheminer	50	0	1	0	49	21	62	5	2	2

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
reconstruire	50	5	1	12	32	28	44	4	5	5
frôler	50	0	2	2	46	21	45	4	2	2
décomposer	50	10	2	3	35	48	45	7	6	6
desserrer	50	1	0	10	39	16	27	3	3	3
reparler	50	0	0	0	50	9	29	7	2	2
accélérer	50	0	1	10	39	16	49	3	2	2
savourer	50	1	0	10	39	18	38	2	1	1
railler	50	2	1	3	44	68	61	2	2	2
emprisonner	50	13	0	6	31	106	101	5	2	2
sceller	50	22	0	5	23	48	80	7	7	7
entonner	49	0	1	5	43	21	38	3	2	2
consigner	49	4	2	11	32	40	52	8	4	4
galoper	49	0	0	6	43	15	28	7	2	2
terrasser	49	8	3	10	28	35	54	4	3	3
tonner	49	0	1	6	42	25	39	4	2	2
pécher	49	0	0	9	40	21	29	4	2	2
desservir	49	3	3	8	35	39	48	6	2	2
comprimer	49	9	0	16	24	32	52	4	4	4
exaucer	49	0	0	4	45	22	42	2	2	2
cerner	49	9	0	1	39	33	61	6	5	5
régaler	49	0	0	9	40	20	34	6	3	3
contourner	49	5	5	7	32	17	70	7	5	5
captiver	49	2	1	14	32	14	37	2	2	2
comparaître	49	0	0	12	37	9	29	1	1	1
prospérer	49	0	0	5	44	14	32	2	1	1
décevoir	48	12	0	0	36	45	48	2	1	1
préserver	48	0	1	4	43	134	70	3	2	2
commenter	48	1	2	8	37	51	54	3	1	1
enregistrer	48	3	0	14	31	36	47	10	6	6
subjuguer	48	4	0	10	34	36	57	1	1	1
attabler	48	7	0	5	36	15	31	1	1	1
accoster	48	2	1	3	42	20	35	4	4	4
navrer	48	8	0	0	40	35	42	1	1	1
pêcher	48	1	0	18	29	16	17	4	2	2
redire	47	0	0	13	34	37	38	7	2	2
émerveiller	47	6	0	0	41	109	60	3	3	3
piétiner	47	1	3	1	42	24	51	6	5	5
ternir	47	0	1	7	39	20	30	5	5	5
grelotter	47	0	2	2	43	15	46	4	3	3
aiguiser	47	2	1	9	35	22	38	4	4	4
exterminer	47	0	0	25	22	25	32	1	1	1
feuilleter	47	0	0	18	29	14	22	4	3	3
maltraiter	47	3	0	11	33	49	60	2	1	1
bâiller	47	0	0	5	42	14	27	2	2	2
extraire	47	3	0	24	20	31	43	6	4	4
raccommoder	47	0	0	18	29	19	30	4	2	2
reléguer	47	5	1	0	41	31	54	5	3	3
arborer	47	1	4	4	38	34	30	6	4	4

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
reconstituer	46	4	1	22	19	27	29	6	5	5
discerner	46	0	0	19	27	39	35	4	3	3
garnir	46	0	0	0	46	316	177	5	1	1
faillir	46	1	0	0	45	16	79	5	5	5
amonceler	46	5	0	1	40	26	41	5	3	3
dilater	46	7	1	2	36	31	51	4	5	5
perler	46	6	0	1	39	22	18	6	5	5
rembrunir	46	0	0	0	46	7	19	5	2	2
agréer	46	0	0	20	26	26	45	5	3	3
critiquer	46	1	2	17	26	24	42	4	4	4
défigurer	45	4	1	5	35	43	64	3	2	2
retrancher	45	12	0	1	32	131	113	6	3	3
bavarder	45	0	4	9	32	12	21	3	2	2
sombrer	45	0	0	1	44	17	24	5	2	2
flamboyer	45	0	0	1	44	8	23	2	2	2
épaissir	45	1	0	3	41	21	39	9	5	5
dorer	45	2	1	4	38	19	50	7	5	5
pulluler	45	0	0	1	44	9	23	4	1	1
blottir	45	1	0	9	35	26	50	3	1	1
intercepter	44	1	3	6	34	33	66	2	3	3
investir	44	4	0	4	36	92	58	7	2	2
fusiller	44	7	0	6	31	54	60	7	4	4
clore	44	0	0	20	24	39	29	4	2	2
empirer	44	0	1	4	39	10	28	2	3	3
emplir	44	0	0	0	44	538	213	2	1	1
délaisser	44	8	0	3	33	53	62	3	1	1
vénérer	44	5	0	3	36	39	53	2	1	1
simplifier	44	1	1	6	36	15	37	4	4	4
rasseoir	44	0	1	9	34	31	41	5	1	1
retrousser	44	15	3	1	25	33	67	5	3	3
raidir	43	2	1	1	39	14	53	8	5	5
déchiffrer	43	0	1	24	18	15	26	4	2	2
endurer	43	2	0	14	27	36	66	2	1	1
broyer	43	8	6	1	28	22	55	4	3	3
adhérer	43	0	0	9	34	9	37	4	2	2
revivre	43	0	1	13	29	12	29	5	3	3
fourrer	43	1	1	16	25	43	45	9	2	2
crouler	43	0	0	9	34	10	32	4	1	1
bousculer	43	0	4	1	38	24	67	6	5	5
oppresser	42	2	0	1	39	24	44	2	2	2
réagir	42	0	1	15	26	28	24	6	4	4
affoler	42	4	0	0	38	33	46	6	5	5
dévaster	42	5	1	5	31	67	79	2	3	3
inspecter	42	0	3	11	28	11	36	3	1	1
lécher	42	0	1	6	35	25	35	8	5	5
renouer	42	0	0	15	27	15	18	5	4	4
agoniser	42	0	0	3	39	5	26	2	1	1
aventurer	42	2	0	1	39	39	27	7	4	4

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
chevaucher	42	1	3	0	38	21	57	8	3	3
rehausser	42	2	1	3	36	38	46	4	3	3
déjouer	42	0	1	12	29	19	32	2	3	3
ramper	41	0	0	7	34	18	41	6	1	1
suer	41	0	2	1	38	21	34	6	4	4
digérer	41	1	1	15	24	11	13	3	2	2
extasier	41	3	1	2	35	23	32	1	1	1
excéder	41	7	0	2	32	27	46	5	3	3
attarder	40	0	0	0	40	31	48	4	3	3
chuchoter	40	1	0	1	38	11	49	5	3	3
décorer	40	0	0	0	40	270	116	5	1	1
approprier	40	5	3	17	15	46	28	3	3	3
pointer	40	1	2	1	36	18	35	15	7	7
avorter	40	10	0	3	27	18	33	4	5	5
reluire	40	0	2	7	31	16	29	3	4	4
tranquilliser	40	2	0	7	31	17	34	2	2	2
restituer	40	0	0	9	31	37	56	8	3	3
démêler	40	0	0	21	19	24	26	5	4	4
calomnier	40	4	0	1	35	27	46	1	1	1
déplier	40	6	3	1	30	17	48	5	4	4
dégrader	39	4	0	5	30	31	56	9	5	5
importuner	39	0	0	9	30	23	22	1	1	1
persuader	39	1	0	6	32	191	208	4	1	1
escalader	39	1	3	13	22	22	69	2	1	1
offrir	39	0	0	39	0	3 518	1 010	6	1	1
agacer	39	1	1	5	32	39	28	4	3	3
façonner	39	7	0	8	24	26	31	6	3	3
engourdir	39	7	0	5	27	45	37	4	3	3
poindre	39	0	0	37	2	23	6	2	1	1
broncher	39	0	0	7	32	10	19	3	2	2
poster	39	10	0	1	28	31	51	4	3	3
approfondir	39	4	1	24	10	15	34	5	5	5
répudier	39	0	1	9	29	25	24	2	1	1
esquiver	39	1	1	15	22	15	45	2	3	3
déplaire	39	0	0	28	11	345	158	5	3	3
allier	39	0	0	10	29	40	24	7	2	2
révoquer	39	0	0	12	27	25	41	3	2	2
râler	39	0	1	1	37	15	22	3	2	2
convoiter	38	1	0	6	31	25	57	2	2	2
interposer	38	1	0	12	25	31	22	5	2	2
envenimer	38	3	2	7	26	27	39	2	2	2
reformer	38	1	0	14	23	16	26	2	2	2
sourciller	38	0	0	13	25	2	16	2	1	1
décrocher	38	1	1	5	31	8	37	12	6	6
alarmer	38	5	0	4	29	100	64	2	2	2
scruter	38	0	1	7	30	16	14	4	4	4
acquiescer	38	0	0	4	34	4	14	2	2	2
sucer	38	0	3	7	28	22	32	8	3	3

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
discontinuer	38	0	1	15	22	8	13	1	1	1
réformer	38	1	0	24	13	15	29	4	3	3
persévérer	38	0	2	13	23	22	19	4	3	3
tâtonner	38	0	15	1	22	13	30	2	1	1
énumérer	38	0	2	15	21	14	46	1	1	1
immobiliser	37	4	0	3	30	25	23	6	6	6
écarquiller	37	10	2	2	23	21	22	2	3	3
vaciller	37	0	0	0	37	10	28	4	3	3
coudre	37	7	1	12	17	58	56	6	3	3
émousser	37	4	0	2	31	29	35	6	4	4
détailler	37	2	3	8	24	16	30	5	4	4
enrager	37	14	0	1	22	82	62	3	2	2
assoupir	37	0	0	3	34	27	43	4	3	3
entraver	37	1	2	14	20	40	47	4	2	2
bouder	37	0	4	7	26	16	26	3	4	4
avoisiner	37	1	5	0	31	19	48	3	2	2
embraser	37	4	0	11	22	27	33	6	3	3
décerner	37	3	0	2	32	43	45	3	4	4
mugir	37	0	2	0	35	13	31	4	3	3
tourbillonner	37	0	1	2	34	3	20	2	2	2
économiser	36	1	1	5	29	8	17	5	3	3
opprimer	36	1	0	8	27	32	61	1	1	1
suppléer	36	0	0	32	4	122	30	5	1	1
mâcher	36	3	7	5	21	16	26	5	4	4
cribler	36	0	3	2	31	52	43	7	3	3
aviver	36	1	1	3	31	12	20	4	4	4
forger	36	7	0	2	27	22	40	9	6	6
paître	36	0	0	4	32	6	32	2	2	2
trouer	36	1	3	1	31	22	34	7	4	4
terrifier	36	9	0	4	23	50	63	1	1	1
orienter	35	1	0	13	21	21	21	7	1	1
assouvir	35	0	0	18	17	13	18	4	4	4
activer	35	0	5	8	22	20	14	6	3	3
affamer	35	24	0	6	5	57	55	3	4	4
récolter	35	2	0	13	20	14	33	3	2	2
masser	35	4	1	3	27	35	28	6	4	4
amollir	35	1	0	3	31	12	18	4	4	4
stupéfier	35	3	0	3	29	12	19	1	1	1
bouillir	35	0	0	4	31	8	14	4	4	4
éventer	35	0	2	1	32	12	26	8	5	5
étager	35	6	2	0	27	13	24	5	4	4
simuler	35	1	3	4	27	24	35	2	1	1
écorcher	34	0	0	5	29	24	38	7	2	2
ratifier	34	0	1	10	23	42	46	2	1	1
osciller	34	0	1	3	30	5	35	4	2	2
flâner	34	0	1	7	26	11	29	2	1	1
proportionner	34	6	1	3	24	53	46	4	3	3
trôner	34	0	0	1	33	11	28	2	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
annuler	34	1	0	5	28	20	44	2	3	3
percher	34	19	0	0	15	26	49	8	5	5
absoudre	34	0	0	9	25	16	14	4	2	2
déclamer	34	1	3	2	28	20	49	4	5	5
consolider	34	0	0	12	22	22	39	2	3	3
confisquer	34	4	0	7	23	56	60	3	2	2
dévisager	34	0	1	3	30	12	23	1	1	1
alourdir	34	0	0	1	33	32	47	5	4	4
dédier	34	0	0	1	33	68	50	2	2	2
effondrer	34	0	0	4	30	25	37	6	2	2
toiser	33	1	0	3	29	8	31	2	1	1
cuire	33	2	0	17	14	12	24	9	6	6
échoir	33	0	0	6	27	15	23	2	2	2
émaner	33	0	0	0	33	57	77	1	1	1
découler	33	0	0	0	33	42	40	1	1	1
obséder	33	0	0	3	30	22	30	4	2	2
rectifier	33	0	0	12	21	24	23	7	1	1
démasquer	33	1	1	5	26	24	37	7	2	2
empiler	33	8	0	2	23	20	23	7	3	3
boucher	33	1	2	0	30	24	49	4	4	4
proscrire	33	3	2	6	22	44	51	5	2	2
rugir	32	0	5	5	22	24	31	5	1	1
alterner	32	0	0	1	31	25	30	6	2	2
bloquer	32	5	0	4	23	15	39	14	7	7
draper	32	8	0	1	23	73	51	7	3	3
refluer	32	0	0	1	31	11	25	2	2	2
amortir	32	0	2	6	24	13	28	6	4	4
confectionner	32	2	0	12	18	23	37	3	4	4
labourer	32	1	3	2	26	32	43	5	4	4
voûter	32	22	0	0	10	81	89	4	5	5
raviver	32	0	0	2	30	11	34	3	4	4
trébucher	32	0	2	0	30	18	30	3	3	3
évoluer	32	0	0	0	32	7	23	7	3	3
avilir	32	1	1	6	24	28	36	4	5	5
munir	31	1	0	3	27	189	176	7	2	2
épandre	31	3	2	1	25	17	18	2	3	3
ameuter	31	2	0	9	20	12	24	3	4	4
baser	31	0	0	4	27	52	36	5	2	2
aigrir	31	0	1	5	25	11	35	6	4	4
rasséréner	31	3	0	3	25	14	21	2	2	2
ployer	31	4	2	0	25	26	49	4	5	5
tresser	31	11	1	6	13	28	44	4	3	3
délier	31	1	0	2	28	24	37	8	3	3
abjurer	31	0	1	6	24	10	33	2	2	2
assimiler	31	0	2	10	19	52	26	5	4	4
stimuler	31	0	0	9	22	19	24	4	3	3
superposer	31	22	1	0	8	19	40	6	3	3
écumer	31	0	2	1	28	15	31	6	5	5

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
sympathiser	31	0	0	3	28	14	10	2	1	1
déménager	30	1	0	4	25	14	19	6	4	4
modérer	30	0	0	19	11	15	17	7	3	3
rebrousser	30	1	0	6	23	5	30	4	3	3
congédier	30	0	1	2	27	58	58	2	1	1
pétrir	30	3	1	4	22	26	40	6	4	4
injurier	30	0	3	4	23	26	38	3	2	2
dégringoler	30	0	0	1	29	10	28	6	4	4
embrouiller	30	2	1	3	24	17	24	6	2	2
surexciter	30	2	0	6	22	41	25	3	3	3
avouer	30	6	0	16	8	996	297	7	2	2
endosser	30	0	0	5	25	14	46	3	1	1
tutoyer	30	0	1	3	26	22	16	3	2	2
darder	30	3	6	0	21	30	29	4	2	2
imprégner	30	0	0	1	29	73	59	5	2	2
crépiter	30	0	2	0	28	6	13	1	1	1
tressauter	30	0	0	0	30	3	15	2	1	1
pétrifier	30	13	0	1	16	53	36	7	5	5
grouiller	30	0	0	1	29	16	18	6	1	1
désavouer	30	2	1	9	18	23	43	4	2	2
dérouter	30	2	0	10	18	24	17	5	2	2
voguer	30	0	2	7	21	14	27	1	1	1
enfiler	29	5	1	2	21	28	56	10	5	5
joncher	29	0	0	0	29	72	46	3	2	2
accaparer	29	0	1	4	24	15	21	4	3	3
déchoir	29	6	0	2	21	37	27	6	4	4
cogner	29	0	0	4	25	17	25	13	5	5
chagriner	29	0	0	3	26	11	14	4	2	2
désapprouver	29	0	0	2	27	18	31	2	1	1
germer	29	0	0	7	22	5	27	2	2	2
déformer	29	11	0	1	17	28	47	8	6	6
coïncider	29	0	0	2	27	22	19	3	2	2
arpenter	29	0	3	5	21	9	29	1	1	1
enhardir	29	0	0	2	27	51	38	4	2	2
ruminer	29	0	2	3	24	5	17	4	2	2
onduler	28	0	1	1	26	11	33	5	2	2
cligner	28	0	0	2	26	11	22	4	1	1
présager	28	0	1	1	26	12	31	4	2	2
aplatir	28	3	1	4	20	18	32	10	2	2
dépérir	28	0	0	1	27	14	27	3	2	2
cicatriser	28	0	0	8	20	14	18	4	3	3
tisser	28	1	2	2	23	16	23	7	3	3
fasciner	28	3	0	1	24	45	35	4	2	2
modeler	28	4	1	7	16	17	27	8	5	5
rétrécir	28	0	5	0	23	12	44	3	3	3
gérer	27	0	0	7	20	21	18	4	4	4
murer	27	5	0	3	19	25	46	9	2	2
bannir	27	1	0	6	20	91	50	3	2	2

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
ajuster	27	0	1	1	25	18	30	9	3	3
communier	27	0	0	0	27	24	24	3	1	1
défoncer	27	5	0	1	21	31	26	8	4	4
bander	27	4	0	4	19	11	23	5	5	5
chômer	27	0	0	2	25	3	14	2	1	1
substituer	27	1	0	7	19	130	75	3	2	2
éperdre	27	23	0	0	4	52	61	3	4	4
infliger	27	8	1	6	12	112	83	3	4	4
mutiler	27	8	1	4	14	26	46	6	6	6
détraquer	27	2	0	4	21	9	14	7	6	6
fausser	27	1	0	7	19	21	23	3	4	4
maigrir	27	0	0	0	27	6	13	5	2	2
transcrire	27	0	2	11	14	19	20	6	2	2
vaquer	27	0	0	12	15	12	11	2	3	3
contrôler	27	1	0	17	9	21	17	5	3	3
embêter	27	0	1	2	24	11	12	2	2	2
alléger	27	1	0	11	15	17	15	10	4	4
réconcilier	27	2	0	0	25	87	40	4	1	1
grimacer	27	0	1	1	25	13	21	3	2	2
purifier	27	0	0	7	20	26	37	4	3	3
moucher	27	1	0	12	14	8	10	10	4	4
exécrer	27	1	0	1	25	12	28	4	3	3
rejaillir	27	0	0	2	25	18	22	2	1	1
opiner	26	0	0	1	25	1	20	3	1	1
dénaturer	26	3	0	7	16	24	31	3	4	4
panser	26	0	0	5	21	11	20	4	4	4
fermenter	26	1	0	1	24	10	29	2	2	2
symboliser	26	0	3	2	21	11	20	2	2	2
démonter	26	3	1	4	18	24	28	8	3	3
meubler	26	8	0	0	18	76	50	7	2	2
obstruer	26	2	1	2	21	20	42	2	3	3
coudoyer	26	0	0	1	25	13	26	3	4	4
éclore	26	1	0	21	4	28	19	4	1	1
indisposer	26	1	0	8	17	11	39	2	2	2
jaunir	26	3	2	1	20	12	15	3	4	4
héler	26	0	0	6	20	5	19	2	2	2
salir	26	0	2	2	22	15	17	6	3	3
assourdir	26	3	0	0	23	21	22	6	3	3
traquer	25	8	0	4	13	22	30	4	3	3
broder	25	5	1	4	15	59	41	5	1	1
appréhender	25	1	0	6	18	16	21	4	2	2
délirer	25	0	0	1	24	4	14	2	1	1
étirer	25	0	1	0	24	14	17	6	5	5
noircir	25	0	3	1	21	27	36	8	5	5
vociférer	25	0	5	5	15	8	19	4	2	2
usurper	25	4	0	4	17	17	42	1	1	1
transpirer	25	0	1	0	24	22	28	4	3	3
régénérer	25	4	0	11	10	10	11	5	3	3

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
tapisser	25	1	0	1	23	32	35	6	2	2
apprivoiser	25	2	0	8	15	16	24	3	2	2
châtier	25	0	0	5	20	18	21	5	2	2
parfumer	25	15	0	0	10	123	84	8	3	3
sanctifier	25	1	0	3	21	19	26	3	3	3
expulser	25	5	0	9	11	44	30	3	3	3
mirer	25	1	3	6	15	12	19	6	3	3
meurtrir	25	2	2	1	20	22	31	6	4	4
rebâtir	25	0	0	7	18	27	34	2	2	2
ajouter	25	0	0	25	0	1 932	560	4	1	1
jaser	25	0	0	4	21	8	9	3	1	1
recoucher	25	1	0	4	20	11	18	2	3	3
pondre	25	0	1	4	20	9	22	5	4	4
enquérir	25	0	0	1	24	38	36	1	1	1
bivouaquer	24	0	1	6	17	8	11	1	1	1
assumer	24	0	0	10	14	11	26	1	1	1
frayer	24	0	1	2	21	48	30	5	2	2
faner	24	1	0	1	22	17	25	5	5	5
profiler	24	3	0	1	20	14	27	7	4	4
influencer	24	0	0	8	16	31	27	2	2	2
peiner	24	1	0	1	22	8	27	4	2	2
chatouiller	24	0	0	2	22	8	25	6	3	3
rassasier	24	1	0	7	16	37	32	3	2	2
préméditer	24	8	2	0	14	15	20	1	1	1
rebondir	24	4	4	0	16	8	28	4	4	4
incarner	24	9	0	3	12	49	36	5	2	2
transfigurer	24	2	0	3	19	72	29	4	1	1
appareiller	23	3	0	3	17	7	21	7	4	4
tricoter	23	0	2	0	21	9	15	7	2	2
tramer	23	0	0	2	21	35	34	4	2	2
réconforter	23	1	0	2	20	22	30	2	2	2
récuser	23	0	0	2	21	4	12	3	2	2
décimer	23	2	0	1	20	28	52	1	1	1
figer	23	1	0	0	22	26	23	5	4	4
bourrer	23	0	1	0	22	35	35	8	2	2
omettre	23	0	0	3	20	40	33	2	1	1
agenouiller	23	21	2	0	0	301	187	2	2	2
surnager	23	0	1	2	20	5	14	2	3	3
estomper	23	3	0	0	20	13	30	6	5	5
élaborer	23	0	0	1	22	22	47	4	4	4
fraterniser	23	0	0	5	18	11	7	2	1	1
fraîchir	23	0	1	0	22	2	8	1	1	1
emballer	23	1	0	1	21	15	17	13	4	4
éluder	23	1	0	8	14	9	30	1	1	1
mentionner	23	5	0	2	16	120	55	3	1	1
friser	23	10	0	3	10	21	31	4	2	2
végéter	23	0	0	4	19	14	26	3	2	2
enserrer	23	0	0	1	22	22	40	6	2	2

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
bégayer	23	2	1	0	20	6	18	4	5	5
accouder	23	22	1	0	0	96	64	2	2	2
mélanger	23	6	1	2	14	26	40	8	3	3
incruster	23	7	0	1	15	44	33	7	4	4
braquer	23	6	2	1	14	18	31	9	5	5
scandaliser	23	5	0	5	13	65	44	2	1	1
sanctionner	22	0	0	7	15	19	35	2	2	2
offusquer	22	0	0	4	18	22	21	2	1	1
réorganiser	22	1	0	9	12	14	15	2	2	2
rôtir	22	2	2	5	13	16	22	8	6	6
cantonner	22	8	0	2	12	11	22	7	5	5
contrefaire	22	2	1	5	14	15	26	3	4	4
abreuver	22	0	0	3	19	43	35	6	2	2
libérer	22	1	1	6	14	22	27	12	4	4
décharner	22	18	0	0	4	27	29	2	3	3
effaroucher	22	2	1	7	12	46	29	3	1	1
enjamber	22	0	4	4	14	10	28	4	3	3
pérorer	22	0	0	2	20	2	15	1	1	1
progresser	22	0	0	6	16	15	14	5	1	1
trépigner	22	0	2	1	19	11	19	1	1	1
égratigner	22	1	2	2	17	11	23	3	2	2
duper	22	3	0	6	13	10	10	1	1	1
humer	22	0	4	3	15	10	29	2	2	2
décroître	22	0	0	7	15	7	11	1	1	1
déferler	22	0	0	0	22	11	15	4	1	1
affectionner	22	1	0	0	21	9	42	2	1	1
primer	22	0	0	3	19	10	14	4	2	2
personnifier	22	3	0	2	17	26	22	3	2	2
exhiber	22	0	1	6	15	18	20	5	2	2
rater	22	1	0	6	15	10	9	9	6	6
entrouvrir	22	0	3	3	16	6	22	2	2	2
désemplir	22	0	0	0	22	4	8	3	3	3
griller	22	2	0	6	14	15	15	14	7	7
incommoder	22	0	0	4	18	5	27	2	2	2
rajuster	21	2	2	3	14	8	27	7	3	3
emmêler	21	9	0	1	11	9	13	5	2	2
ennoblir	21	1	0	2	18	15	27	4	4	4
promulguer	21	4	0	2	15	18	20	1	1	1
glorifier	21	0	0	9	12	49	36	3	3	3
suggérer	21	0	0	0	21	151	72	3	2	2
disséminer	21	11	0	0	10	28	43	4	3	3
fomenter	21	1	0	6	14	16	17	2	2	2
sauvegarder	21	0	0	9	12	10	7	3	3	3
immortaliser	21	1	0	9	11	19	18	2	3	3
évaporer	21	2	1	0	18	18	24	6	5	5
aplanir	21	0	1	3	17	10	21	3	4	4
généraliser	21	1	1	4	15	13	22	6	4	4
déverser	21	0	0	0	21	15	28	7	3	3

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
foisonner	21	0	0	0	21	8	7	6	2	2
valser	21	0	4	3	14	6	19	3	3	3
distiller	21	0	1	2	18	10	17	6	5	5
tirailler	21	1	2	4	14	27	20	6	3	3
tortiller	21	2	3	2	14	6	24	8	5	5
débrouiller	21	0	0	10	11	8	15	5	2	2
ricaner	21	0	0	0	21	2	13	3	3	3
suinter	21	0	0	0	21	9	10	3	2	2
pourrir	21	0	0	1	20	12	17	7	4	4
sommer	21	0	0	2	19	13	21	4	1	1
décontenancer	21	10	0	1	10	19	17	2	2	2
amoindrir	21	0	1	3	17	10	27	5	5	5
recourber	20	14	0	0	6	30	44	2	3	3
ballotter	20	3	1	1	15	16	15	5	6	6
éterniser	20	1	0	5	14	10	17	5	2	2
damner	20	1	0	5	14	17	20	2	2	2
appauvrir	20	1	1	3	15	10	19	4	3	3
détenir	20	1	0	3	16	64	52	2	1	1
souffleter	20	1	1	4	14	8	9	1	1	1
harmoniser	20	0	0	3	17	17	13	3	2	2
survivre	20	1	0	19	0	336	97	4	2	2
cacheter	20	13	0	0	7	14	32	3	4	4
décéder	20	8	0	0	12	13	19	1	1	1
ourdir	20	3	0	2	15	13	22	5	5	5
déblayer	20	0	1	5	14	19	22	4	2	2
profaner	20	1	0	6	13	17	25	3	4	4
souscrire	20	2	0	4	14	41	32	2	3	3
renchérir	20	0	0	8	12	11	16	4	2	2
dénicher	20	0	0	2	18	13	18	4	5	5
advenir	20	1	1	7	11	69	85	2	2	2
moissonner	20	0	0	1	19	8	18	4	3	3
gazouiller	20	0	0	0	20	7	20	4	2	2
extirper	20	1	1	11	7	7	8	4	2	2
affiner	20	6	0	1	13	17	22	2	2	2
accroupir	20	19	1	0	0	91	61	2	2	2
outrer	20	5	0	0	15	29	30	5	4	4
entrecouper	20	7	1	0	12	73	35	4	2	2
atterrir	20	0	0	6	14	4	12	2	2	2
attiser	20	0	1	2	17	7	16	2	3	3
disserter	19	0	1	1	17	11	7	3	2	2
haranguer	19	0	0	4	15	9	19	1	1	1
enfiévrer	19	7	0	0	12	13	20	4	5	5
rabaisser	19	1	0	8	10	11	15	6	3	3
évaluer	19	0	0	11	8	22	22	2	2	2
barricader	19	0	0	2	17	11	27	3	2	2
dévier	19	0	0	2	17	27	26	5	1	1
départir	19	0	0	2	17	43	27	3	1	1
décamper	19	0	0	2	17	3	20	1	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
épurer	19	1	0	4	14	10	21	4	4	4
transiger	19	0	0	8	11	10	11	3	1	1
outrager	19	0	0	3	16	26	29	5	1	1
débander	19	4	0	0	15	11	44	6	6	6
aménager	19	4	1	2	12	18	21	3	4	4
narguer	19	0	0	8	11	6	8	2	2	2
neiger	19	0	0	3	16	4	5	1	1	1
cabrer	19	0	0	0	19	11	25	6	1	1
enfouir	19	19	0	0	0	107	55	3	2	2
prohiber	19	4	1	2	12	10	15	1	1	1
dévouer	19	15	4	0	0	305	141	3	3	3
blêmir	19	0	0	0	19	5	15	3	3	3
ciseler	19	12	0	2	5	13	35	3	3	3
égrener	19	0	3	1	15	33	16	5	4	4
interpeller	19	1	0	5	13	29	33	4	1	1
surplomber	19	0	5	0	14	17	26	1	1	1
brouter	19	0	0	1	18	6	27	3	2	2
démoraliser	19	4	1	2	12	11	9	2	2	2
enrhumer	19	3	0	1	15	7	15	2	2	2
miner	19	0	1	2	16	16	24	8	5	5
transférer	18	2	1	4	11	87	36	6	3	3
brosser	18	3	1	6	8	10	27	9	3	3
capturer	18	2	0	3	13	15	45	3	2	2
escamoter	18	1	0	8	9	10	12	5	3	3
badiner	18	0	2	4	12	5	14	3	2	2
fredonner	18	0	3	2	13	11	31	2	2	2
déterrer	18	1	0	9	8	12	25	4	3	3
légitimer	18	0	0	14	4	4	9	7	3	3
éplucher	18	1	1	4	12	11	12	4	2	2
énerver	18	1	0	2	15	18	12	3	2	2
dénuer	18	17	0	0	1	87	67	5	3	3
relayer	18	0	0	2	16	18	12	3	1	1
river	18	7	0	1	10	19	18	5	3	3
marteler	18	0	1	1	16	11	16	5	3	3
éventrer	18	6	0	3	9	13	27	3	3	3
essouffler	18	5	0	2	11	27	61	5	5	5
circonscrire	18	0	2	1	15	10	18	6	3	3
révérer	18	3	0	2	13	13	23	1	1	1
rétracter	18	0	0	1	17	3	22	6	4	4
inhumer	18	0	0	3	15	13	18	1	1	1
brusquer	18	0	0	10	8	6	6	2	2	2
pâmer	18	3	0	3	12	34	35	2	2	2
assujettir	18	0	0	6	12	56	40	4	2	2
ganter	18	18	0	0	0	32	45	4	3	3
comploter	18	0	0	2	16	10	14	3	2	2
aliéner	18	0	0	6	12	21	22	4	2	2
accréditer	18	10	0	1	7	49	19	4	2	2
jucher	18	5	0	0	13	24	19	7	4	4

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
décupler	18	2	0	1	15	17	27	2	2	2
guillotiner	18	1	0	1	16	11	14	1	1	1
boutonner	18	5	1	1	11	11	28	4	5	5
décacheter	17	1	1	4	11	13	22	4	2	2
tasser	17	4	0	1	12	15	25	11	5	5
neutraliser	17	0	0	8	9	7	17	4	3	3
mécontenter	17	0	0	5	12	2	16	1	1	1
attenter	17	0	0	7	10	18	9	1	1	1
taquiner	17	0	1	6	10	10	14	4	2	2
déballer	17	1	0	2	14	3	7	6	2	2
puer	17	0	0	0	17	7	7	3	2	2
barbouiller	17	1	0	0	16	27	30	9	6	6
dégarnir	17	1	0	3	13	12	13	5	2	2
discourir	17	0	0	9	8	11	8	2	1	1
atteler	17	0	0	3	14	204	132	5	2	2
concorder	17	0	0	3	14	8	13	2	1	1
trinquer	17	0	0	3	14	10	8	4	1	1
courroucer	17	15	0	0	2	36	28	2	1	1
jalouser	17	1	0	1	15	15	22	2	2	2
raffoler	17	0	0	0	17	5	7	1	1	1
ébrécher	17	10	1	3	3	11	11	4	3	3
disloquer	17	1	0	2	14	14	20	4	3	3
filtrer	17	0	2	2	13	6	16	9	4	4
chantonner	17	0	2	1	14	7	18	2	2	2
idolâtrer	17	0	1	0	16	6	9	2	2	2
nicher	17	3	0	0	14	9	19	9	5	5
charrier	17	0	3	3	11	5	19	5	3	3
saturer	17	0	0	0	17	22	16	8	3	3
voleter	17	0	1	3	13	14	23	2	2	2
rendormir	17	0	0	3	14	22	18	2	2	2
peigner	17	3	0	4	10	25	19	4	4	4
grogner	16	0	1	3	12	14	28	3	1	1
assaisonner	16	2	1	2	11	21	27	6	4	4
régulariser	16	1	0	6	9	9	19	2	3	3
récriminer	16	0	0	8	8	3	2	2	1	1
enrouer	16	8	0	0	8	9	12	2	3	3
abrutir	16	0	1	1	14	13	20	3	2	2
scier	16	1	1	5	9	17	30	7	3	3
cingler	16	0	0	0	16	19	17	7	2	2
exténuer	16	7	0	0	9	26	39	3	2	2
attrouper	16	0	0	3	13	7	12	2	2	2
officier	16	0	0	0	16	3	14	2	1	1
ramer	16	0	0	0	16	12	14	6	2	2
patauger	16	0	2	2	12	5	12	2	2	2
idéaliser	16	6	1	1	8	12	14	2	3	3
endommager	16	3	0	1	12	24	25	2	3	3
éprendre	16	1	0	0	15	91	51	1	1	1
pervertir	16	1	0	0	15	13	22	4	3	3

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
enrayer	16	1	0	11	4	5	9	4	4	4
redonner	16	0	0	2	14	18	15	5	2	2
exulter	16	0	0	0	16	2	13	1	1	1
boucler	16	0	0	3	13	10	21	10	4	4
subordonner	16	0	0	0	16	34	32	4	1	1
préluder	16	0	1	2	13	9	20	2	2	2
arroger	16	0	0	3	13	6	16	1	1	1
subvenir	16	0	0	14	2	8	9	1	1	1
saccager	16	0	1	0	15	12	31	2	2	2
sursauter	16	0	1	1	14	0	13	1	1	1
tournoyer	16	0	2	0	14	9	17	3	2	2
japper	16	0	2	0	14	0	7	1	1	1
parsemer	16	0	0	0	16	71	37	4	2	2
faucher	16	1	0	5	10	16	12	9	3	3
incriminer	16	4	0	3	9	6	9	1	1	1
capter	16	0	0	7	9	8	4	5	1	1
démanger	16	0	0	0	16	0	6	3	2	2
chicaner	16	0	0	4	12	4	3	6	3	3
déboutonner	16	4	2	1	9	4	13	4	2	2
supputer	16	0	0	2	14	4	17	2	2	2
acculer	16	0	0	7	9	26	14	6	3	3
élucider	16	0	0	12	4	11	6	2	2	2
miroiter	16	0	1	0	15	12	25	1	1	1
tracasser	16	0	0	2	14	8	11	2	1	1
bombarder	16	3	0	4	9	12	14	6	2	2
chavirer	16	0	0	2	14	2	9	4	4	4
geindre	15	0	3	0	12	5	16	4	4	4
électriser	15	1	0	2	12	16	29	3	3	3
suspecter	15	0	0	6	9	11	19	3	2	2
féconder	15	2	0	0	13	15	20	4	2	2
enfourcher	15	0	0	1	14	6	21	3	3	3
retremper	15	0	0	9	6	23	19	4	1	1
branler	15	0	0	0	15	3	9	6	1	1
suicider	15	0	0	7	8	10	14	2	2	2
fureter	15	0	4	0	11	9	21	5	4	4
dévaler	15	0	2	0	13	5	17	6	3	3
durcir	15	5	0	0	10	18	27	7	6	6
polir	15	1	0	0	14	10	13	6	3	3
mollir	15	0	0	0	15	5	12	6	1	1
pacifier	15	1	1	8	5	15	13	3	2	2
diversifier	15	1	1	3	10	6	4	2	2	2
poignarder	15	0	0	3	12	11	19	1	1	1
condenser	15	0	0	4	11	11	31	3	2	2
relancer	15	0	0	6	9	5	5	5	3	3
usiter	15	15	0	0	0	44	30	2	2	2
endolorir	15	9	0	1	5	12	17	2	3	3
méfier	15	0	0	0	15	57	28	1	1	1
léguer	15	0	0	0	15	103	34	3	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
implanter	15	1	0	2	12	8	13	6	3	3
vaguer	15	0	0	1	14	3	11	4	1	1
renfrogner	15	13	0	0	2	8	12	2	2	2
tenailler	15	0	0	2	13	6	8	3	2	2
roucouler	15	0	0	0	15	3	9	4	3	3
infecter	15	2	0	0	13	13	17	3	2	2
nuancer	15	4	0	1	10	15	17	6	3	3
cramponner	15	3	0	3	9	83	38	4	2	2
supplanter	15	0	0	5	10	10	12	2	2	2
réapparaître	15	0	1	0	14	5	15	2	1	1
rider	15	0	0	0	15	7	4	5	3	3
ensorceler	15	2	0	1	12	8	14	2	1	1
bafouer	14	0	0	3	11	12	16	1	1	1
colporter	14	1	0	2	11	9	20	3	3	3
tancer	14	0	0	3	11	4	8	1	1	1
divorcer	14	1	0	2	11	14	9	2	1	1
cahoter	14	5	0	0	9	5	6	3	2	2
atterrer	14	6	0	0	8	61	47	1	1	1
diviniser	14	0	0	1	13	7	10	4	1	1
travestir	14	6	0	2	6	14	20	3	3	3
folâtrer	14	0	1	4	9	4	8	1	1	1
conter	14	1	1	0	12	413	134	2	3	3
renifler	14	0	2	1	11	7	12	6	6	6
caser	14	1	0	8	5	10	10	9	3	3
fuser	14	0	0	0	14	8	7	6	1	1
roser	14	8	0	0	6	14	28	2	3	3
bleuir	14	0	1	0	13	8	16	2	2	2
déparer	14	1	0	0	13	11	15	4	2	2
lorgner	14	0	1	0	13	15	15	2	1	1
espionner	14	0	1	2	11	10	11	3	4	4
submerger	14	1	0	2	11	40	23	4	2	2
sautiller	14	0	0	1	13	3	24	3	2	2
gesticuler	14	0	0	1	13	7	30	2	2	2
entrer	14	0	14	0	0	5 781	1 923	6	3	3
adapter	14	0	0	7	7	55	58	6	1	1
stopper	14	0	0	0	14	2	10	5	3	3
désaltérer	14	0	0	6	8	11	6	2	3	3
buter	14	1	0	0	13	1	3	9	3	3
*assavoir	14	0	0	14	0	7	0	0	1	1
réprimander	14	0	0	1	13	10	20	1	1	1
encaisser	14	5	1	0	8	32	29	7	3	3
révolutionner	14	0	0	2	12	5	10	3	2	2
regrouper	14	1	3	0	10	4	5	6	3	3
emboîter	14	0	0	0	14	13	8	4	2	2
enguirlander	14	10	2	1	1	8	18	3	3	3
réprouver	13	0	1	1	11	21	29	2	1	1
foncer	13	0	0	0	13	3	12	10	3	3
égaliser	13	0	0	5	8	6	8	5	2	2

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
statuer	13	0	1	9	3	9	8	3	1	1
collaborer	13	0	0	1	12	8	10	4	1	1
éclabousser	13	1	1	1	10	25	21	7	4	4
gâcher	13	0	0	3	10	4	11	5	3	3
émietter	13	0	0	1	12	12	18	6	2	2
verdir	13	0	0	0	13	5	16	3	3	3
pavoiser	13	7	0	0	6	27	26	4	3	3
accoupler	13	2	0	1	10	30	23	6	2	2
carrer	13	4	0	0	9	12	15	5	2	2
épauler	13	1	0	1	11	2	15	6	4	4
ébruiter	13	0	0	5	8	5	10	2	3	3
brider	13	0	0	4	9	3	9	7	3	3
visser	13	3	0	0	10	11	20	10	4	4
courtiser	13	0	1	2	10	18	21	1	1	1
niveler	13	2	0	2	9	10	13	4	3	3
convier	13	0	0	0	13	72	22	2	1	1
aiguillonner	13	0	0	3	10	5	5	2	1	1
trembloter	13	0	0	0	13	3	9	4	1	1
raviser	13	0	0	2	11	1	23	1	1	1
ahurir	13	7	0	0	6	15	24	1	1	1
abhorrer	13	0	0	0	13	6	30	1	1	1
épeler	13	0	2	4	7	3	9	2	3	3
désorganiser	13	0	0	5	8	2	13	4	5	5
entrebâiller	13	4	1	0	8	11	24	2	2	2
intégrer	13	0	2	3	8	11	8	9	1	1
commercialiser	13	1	0	2	10	4	2	2	2	2
aguerrir	13	4	0	2	7	8	9	2	2	2
bêler	13	0	0	0	13	3	4	3	3	3
patienter	13	0	0	5	8	4	4	1	1	1
mander	13	2	0	0	11	244	99	2	2	2
gifler	13	0	0	3	10	2	2	3	2	2
foutre	13	0	0	1	12	11	7	14	3	3
déroger	13	0	1	3	9	14	15	3	2	2
enchevêtrer	13	1	0	0	12	20	27	4	2	2
éliminer	13	1	0	6	6	9	15	7	2	2
moduler	12	3	0	0	9	11	8	3	3	3
cliqueter	12	0	1	0	11	4	4	1	1	1
déraciner	12	2	0	6	4	16	18	4	3	3
censurer	12	2	0	3	7	5	9	3	1	1
dépeindre	12	0	0	6	6	72	21	3	1	1
mouler	12	1	0	3	8	14	15	10	4	4
rouiller	12	5	0	0	7	12	15	8	6	6
mendier	12	0	0	0	12	20	12	2	1	1
beugler	12	0	0	0	12	2	12	5	3	3
rapetisser	12	0	0	0	12	18	27	5	3	3
dépayser	12	2	0	4	6	12	16	3	4	4
infester	12	0	0	0	12	23	38	4	2	2
réimprimer	12	4	0	0	8	30	9	2	3	3

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
cimenter	12	1	0	2	9	18	16	6	2	2
conspirer	12	0	0	0	12	103	39	3	1	1
démettre	12	0	0	1	11	16	25	6	2	2
raccourcir	12	0	0	4	8	7	11	7	4	4
défricher	12	1	0	5	6	11	16	4	2	2
prostituer	12	0	2	2	8	6	17	5	4	4
démarrer	12	0	1	0	11	7	8	13	5	5
fleurer	12	0	0	0	12	8	7	4	1	1
annexer	12	2	0	5	5	27	24	6	2	2
remanier	12	4	0	0	8	16	16	3	3	3
bomber	12	1	0	0	11	10	16	7	2	2
récapituler	12	0	1	4	7	3	11	3	2	2
accéder	12	0	0	11	1	54	13	4	2	2
harceler	12	1	0	1	10	24	29	2	1	1
affaler	12	0	0	1	11	12	9	3	1	1
décapiter	12	2	0	4	6	21	17	4	2	2
grommeler	12	0	7	2	3	13	15	3	1	1
surmener	12	3	0	0	9	8	6	3	3	3
enjouer	12	12	0	0	0	24	16	2	2	2
enchérir	12	0	1	2	9	3	7	3	2	2
expérimenter	12	0	0	6	6	4	6	5	2	2
dégeler	12	0	0	2	10	4	6	8	4	4
piocher	12	0	1	2	9	3	7	6	4	4
exhausser	12	0	1	2	9	13	16	1	1	1
pâtir	12	0	0	1	11	4	5	2	1	1
râper	12	7	0	1	4	14	18	6	4	4
ragaillardir	12	0	0	0	12	8	14	2	2	2
garrotter	12	5	0	2	5	19	26	4	1	1
tondre	12	1	0	1	10	11	9	6	4	4
hébéter	12	6	1	0	5	24	28	2	2	2
exporter	12	2	0	0	10	17	18	2	2	2
adonner	12	5	0	6	1	74	57	2	2	2
étancher	12	1	0	6	5	2	15	4	3	3
dédire	12	0	0	7	5	6	2	2	1	1
brailler	12	1	0	1	10	6	9	2	2	2
croquer	12	1	0	3	8	10	19	8	3	3
dandiner	12	0	2	0	10	11	13	3	1	1
effarer	12	10	0	0	2	46	25	2	2	2
réhabiliter	12	0	0	0	12	28	15	7	3	3
abstenir	12	0	0	0	12	97	46	2	1	1
frétiller	11	0	1	1	9	4	11	5	2	2
lustrer	11	9	0	0	2	22	14	4	4	4
poudrer	11	9	0	0	2	42	38	7	3	3
débaucher	11	4	0	3	4	22	20	5	4	4
écarteler	11	1	1	1	8	13	13	2	1	1
cambrer	11	0	1	1	9	7	15	4	4	4
régenter	11	0	0	4	7	6	10	1	1	1
réputer	11	4	0	0	7	40	22	2	2	2

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
refleurir	11	0	0	3	8	2	4	1	1	1
lisser	11	0	1	1	9	6	21	4	3	3
leurrer	11	0	0	1	10	7	6	4	2	2
entortiller	11	2	0	0	9	9	14	10	3	3
calciner	11	6	2	0	3	16	23	2	3	3
congestionner	11	1	0	1	9	9	8	4	4	4
interjeter	11	0	0	0	11	2	7	1	1	1
moquer	11	0	1	5	5	382	202	2	3	3
chiffonner	11	0	0	2	9	1	11	4	2	2
louvoyer	11	0	0	1	10	0	9	3	2	2
fracasser	11	1	0	0	10	17	22	2	2	2
brocher	11	3	1	0	7	7	10	4	3	3
racler	11	0	2	0	9	10	12	7	3	3
maçonner	11	7	0	0	4	4	16	4	3	3
dégorger	11	0	1	1	9	7	11	6	3	3
entailler	11	5	0	0	6	7	11	4	4	4
galvaniser	11	1	1	1	8	5	7	5	3	3
tanner	11	7	0	1	3	9	11	4	2	2
centraliser	11	3	1	1	6	4	13	7	4	4
adjoindre	11	0	0	2	9	12	23	3	3	3
estropier	11	0	0	1	10	7	7	3	4	4
gorger	11	0	0	3	8	19	16	5	2	2
entacher	11	3	0	0	8	12	20	4	3	3
remiser	11	0	0	1	10	5	9	9	3	3
pourchasser	11	0	1	1	9	5	15	3	3	3
localiser	11	2	0	0	9	6	8	4	3	3
agrafer	11	7	0	1	3	10	12	8	4	4
gourmander	11	0	0	0	11	5	8	1	1	1
violenter	11	0	1	0	10	7	13	2	1	1
babiller	11	0	1	1	9	3	13	5	4	4
dépister	11	1	0	6	4	4	9	6	3	3
émanciper	11	3	0	3	5	23	20	5	1	1
caqueter	11	0	0	3	8	4	6	3	1	1
dégriser	11	3	0	1	7	15	13	4	2	2
priser	11	1	0	3	7	10	25	3	2	2
chiffrer	11	2	1	6	2	4	13	6	4	4
pulvériser	11	2	1	1	7	17	12	5	2	2
quêter	11	0	1	2	8	15	13	3	3	3
spécialiser	11	5	0	1	5	1	6	4	4	4
raffermir	11	0	1	1	9	5	20	4	2	2
éditer	11	2	0	1	8	8	13	3	3	3
identifier	11	0	0	4	7	25	12	4	2	2
proroger	11	0	0	2	9	4	2	1	1	1
détaler	11	0	0	0	11	4	9	1	1	1
canonner	11	1	2	5	3	6	12	3	2	2
huer	11	0	0	0	11	5	11	3	2	2
dénier	11	0	0	0	11	6	9	2	1	1
bêcher	11	1	1	3	6	1	7	4	3	3

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
vêtir	11	10	0	1	0	621	292	3	2	2
tripler	11	0	0	4	7	5	5	2	2	2
amaigrir	11	4	0	0	7	17	29	4	4	4
obstiner	11	1	0	0	10	137	59	5	3	3
convulser	10	3	0	0	7	17	12	3	3	3
encenser	10	0	0	1	9	9	15	4	2	2
roussir	10	2	0	0	8	12	17	5	4	4
vicier	10	2	0	0	8	6	12	3	2	2
certifier	10	0	0	3	7	9	25	3	2	2
curer	10	2	0	2	6	14	19	5	5	5
verser	10	6	0	4	0	611	205	10	4	4
divaguer	10	0	1	4	5	4	7	2	1	1
lapider	10	1	0	3	6	7	11	1	1	1
boiter	10	0	1	0	9	11	11	3	1	1
réintégrer	10	1	2	2	5	35	13	3	2	2
annihiler	10	0	1	1	8	10	8	2	2	2
voisiner	10	0	0	2	8	3	10	4	1	1
dessiller	10	0	0	2	8	4	7	3	2	2
parquer	10	3	1	1	5	8	24	8	4	4
bossuer	10	3	0	0	7	6	19	2	3	3
parader	10	0	0	1	9	5	9	3	1	1
hiverner	10	0	0	2	8	2	6	4	2	2
remorquer	10	2	0	4	4	6	9	3	3	3
psalmodier	10	0	1	0	9	4	16	2	2	2
équilibrer	10	1	0	4	5	3	2	8	3	3
désenchanter	10	2	0	3	5	4	12	1	1	1
injecter	10	1	1	0	8	29	17	4	2	2
falsifier	10	1	0	2	7	7	6	4	3	3
apitoyer	10	3	0	2	5	20	13	2	2	2
héberger	10	0	0	2	8	16	11	4	2	2
trafiquer	10	0	0	3	7	20	9	4	2	2
éponger	10	0	1	1	8	15	9	7	2	2
raffiner	10	1	1	0	8	6	11	5	4	4
fourvoyer	10	2	0	3	5	2	5	4	3	3
parodier	10	0	1	1	8	5	11	5	1	1
étayer	10	1	0	5	4	10	19	4	2	2
dédommager	10	0	0	0	10	58	33	3	2	2
induire	10	0	0	5	5	24	9	3	1	1
mâchonner	10	0	4	2	4	3	14	3	3	3
grignoter	10	0	2	3	5	2	16	9	4	4
désobéir	10	0	0	2	8	11	10	2	1	1
chausser	10	3	0	0	7	56	46	8	1	1
détériorer	10	2	0	2	6	9	12	6	6	6
désunir	10	1	0	3	6	6	8	8	4	4
tapoter	10	0	2	2	6	6	12	6	3	3
gambader	10	0	1	0	9	6	11	1	1	1
trottiner	10	0	2	0	8	4	17	2	2	2
rogner	10	0	0	2	8	8	20	6	2	2

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
argumenter	10	0	0	0	10	1	4	3	2	2
ridiculiser	10	0	1	4	5	5	8	4	2	2
déguster	10	0	1	3	6	3	15	5	2	2
pouffer	10	0	0	0	10	1	4	1	1	1
désappointer	10	1	0	1	8	12	20	1	1	1
ébattre	10	0	0	1	9	6	9	1	1	1
apposer	10	1	1	6	2	38	18	5	2	2
transpercer	10	3	0	0	7	12	7	6	5	5
caler	10	3	0	2	5	3	13	13	5	5
raccrocher	10	0	0	3	7	12	12	10	3	3
corroborer	10	0	0	3	7	8	7	2	2	2
maculer	10	5	0	0	5	9	17	2	3	3
embusquer	10	2	0	2	6	9	21	2	2	2
cadencer	10	7	0	0	3	13	10	4	4	4
tambouriner	10	0	0	2	8	5	7	5	2	2
converger	10	0	2	3	5	9	17	4	3	3
clapoter	9	0	0	0	9	8	9	2	2	2
enthousiasmer	9	1	0	2	6	44	18	3	2	2
flageoler	9	0	0	0	9	0	5	3	2	2
actionner	9	0	0	2	7	11	17	3	2	2
rouer	9	0	0	1	8	5	12	6	3	3
timbrer	9	6	0	0	3	4	9	5	4	4
occulter	9	0	0	0	9	5	28	4	1	1
hennir	9	0	0	0	9	8	11	3	2	2
disséquer	9	0	0	6	3	7	18	4	2	2
transparaître	9	0	0	0	9	4	6	1	1	1
purger	9	1	0	3	5	8	16	3	2	2
exhumer	9	0	0	3	6	17	11	3	2	2
centupler	9	0	1	0	8	6	9	2	2	2
caracoler	9	0	1	0	8	4	13	2	1	1
trimer	9	0	0	0	9	2	0	1	1	1
réinstaller	9	2	0	2	5	9	8	2	3	3
pavaner	9	0	0	1	8	1	10	1	1	1
incomber	9	0	0	0	9	10	22	1	1	1
canoniser	9	0	0	0	9	9	7	2	2	2
compatir	9	0	0	6	3	6	7	2	1	1
saupoudrer	9	0	0	0	9	27	14	5	2	2
aliter	9	0	0	0	9	9	10	4	1	1
hypnotiser	9	1	0	1	7	17	8	3	2	2
angoisser	9	3	0	1	5	4	6	3	2	2
prédominer	9	0	0	1	8	13	12	2	1	1
ronronner	9	0	2	0	7	1	7	3	2	2
télégraphier	9	1	1	0	7	7	7	4	3	3
déchiqueter	9	2	2	1	4	7	14	3	2	2
crayonner	9	0	0	2	7	11	9	6	1	1
vexer	9	4	0	2	3	33	16	2	1	1
détremper	9	2	0	0	7	21	12	3	1	1
ferrer	9	2	0	1	6	6	12	6	4	4

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
enfreindre	9	0	0	4	5	6	10	2	3	3
désarçonner	9	0	0	1	8	4	7	3	2	2
éternuer	9	0	0	4	5	4	16	1	1	1
infléchir	9	1	1	0	7	6	5	4	2	2
effiler	9	2	0	0	7	9	15	7	4	4
recomposer	9	0	0	5	4	6	7	3	2	2
dorloter	9	0	0	4	5	3	11	1	1	1
gaspiller	9	0	0	0	9	9	16	3	3	3
griffer	9	0	0	0	9	1	7	7	3	3
dépraver	9	0	0	1	8	7	15	4	3	3
prélasser	9	0	0	0	9	2	8	1	1	1
délabrer	9	3	0	0	6	6	17	3	2	2
jalonner	9	0	0	0	9	8	9	4	2	2
perpétrer	9	2	0	2	5	2	3	2	2	2
alanguir	9	1	0	0	8	7	14	3	3	3
endoctriner	9	1	0	3	5	2	9	1	1	1
glapir	9	0	0	0	9	8	10	4	2	2
bouffer	9	1	0	0	8	4	1	6	3	3
téter	9	0	0	1	8	3	2	4	4	4
déifier	9	0	0	0	9	2	10	2	2	2
déloger	9	0	0	3	6	6	10	3	2	2
naturaliser	9	1	0	2	6	6	14	6	2	2
décoiffer	9	1	0	0	8	9	6	5	1	1
diffuser	9	4	0	4	1	15	23	8	4	4
muer	9	0	0	3	6	13	8	4	1	1
miauler	9	0	1	0	8	1	4	4	2	2
sermonner	9	0	1	3	5	4	12	1	1	1
entremêler	9	1	0	0	8	90	41	6	1	1
hacher	9	0	0	1	8	18	15	4	2	2
calquer	9	0	0	2	7	12	8	5	3	3
amincir	9	2	0	0	7	14	17	6	4	4
décommander	9	1	0	4	4	2	2	3	1	1
ébrouer	9	0	2	1	6	4	10	4	1	1
empester	9	4	0	1	4	35	18	4	1	1
palper	9	0	0	0	9	5	11	3	2	2
somnoler	9	0	0	0	9	3	5	2	2	2
rigoler	9	0	0	0	9	4	6	4	1	1
liquider	9	1	0	3	5	5	9	7	4	4
lacer	9	2	0	1	6	5	13	4	3	3
bredouiller	9	0	1	0	8	2	8	3	2	2
siffloter	9	0	2	2	5	2	12	2	2	2
terroriser	9	0	0	1	8	5	6	1	1	1
chamarrer	9	9	0	0	0	11	8	2	2	2
sourdre	9	0	0	2	7	3	4	2	1	1
empâter	9	3	0	0	6	12	12	6	4	4
inculquer	8	1	0	3	4	8	10	3	3	3
amorcer	8	1	0	1	6	6	9	5	2	2
malmener	8	0	1	1	6	12	12	2	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
verrouiller	8	1	0	1	6	6	8	5	3	3
ulcérer	8	5	0	0	3	13	19	3	3	3
déduire	8	2	0	1	5	50	37	3	2	2
brutaliser	8	0	0	1	7	1	5	1	1	1
haler	8	0	1	2	5	6	16	5	3	3
charbonner	8	1	0	0	7	7	7	6	2	2
convertir	8	3	0	5	0	224	87	4	3	3
amender	8	0	0	5	3	11	17	5	2	2
scander	8	2	0	1	5	8	7	4	2	2
choyer	8	1	0	0	7	17	22	2	2	2
repêcher	8	1	0	2	5	2	9	6	2	2
guigner	8	0	1	0	7	4	3	2	1	1
clôturer	8	0	0	1	7	6	4	3	3	3
graviter	8	0	1	0	7	5	5	2	1	1
garer	8	0	0	5	3	6	8	6	2	2
cadrer	8	0	0	1	7	5	7	4	1	1
rosser	8	0	0	3	5	12	11	2	2	2
surabonder	8	0	0	0	8	7	11	2	1	1
taxer	8	0	0	0	8	31	15	3	1	1
machiner	8	2	0	3	3	7	10	2	2	2
saper	8	0	1	5	2	4	11	6	3	3
carguer	8	3	0	3	2	3	8	2	3	3
civiliser	8	0	0	2	6	7	9	3	2	2
destituer	8	0	0	0	8	57	25	1	1	1
rechigner	8	1	5	1	1	17	14	3	2	2
bronzer	8	0	0	0	8	14	11	5	2	2
tyranniser	8	0	0	1	7	10	10	2	2	2
écheveler	8	5	0	0	3	14	10	3	2	2
mystifier	8	2	0	2	4	6	4	1	1	1
loucher	8	0	0	2	6	7	16	2	1	1
doter	8	0	0	0	8	67	34	4	1	1
baragouiner	8	0	1	1	6	2	5	3	2	2
talonner	8	0	2	1	5	5	10	5	2	2
unifier	8	1	0	2	5	3	4	4	2	2
dégrossir	8	2	0	2	4	7	4	7	3	3
apostropher	8	0	0	0	8	10	12	2	1	1
pester	8	0	2	0	6	8	8	2	2	2
glaner	8	0	0	3	5	7	7	3	2	2
outrepasser	8	0	0	1	7	2	4	2	2	2
morfondre	8	0	0	2	6	3	12	1	1	1
sacrer	8	0	1	2	5	7	12	3	2	2
surbaisser	8	8	0	0	0	3	19	2	1	1
replonger	8	0	1	0	7	47	13	6	2	2
accoter	8	3	0	0	5	14	16	6	1	1
fêler	8	1	0	0	7	6	9	5	3	3
équarrir	8	5	0	1	2	3	10	3	2	2
agglomérer	8	1	0	0	7	10	9	4	4	4
déraisonner	8	0	0	2	6	3	6	1	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
surchauffer	8	7	0	0	1	7	11	4	3	3
ouater	8	7	0	0	1	9	9	4	3	3
discipliner	8	1	0	2	5	14	8	6	4	4
ancrer	8	4	0	0	4	10	8	8	3	3
rudoyer	8	2	0	0	6	10	9	1	1	1
humaniser	8	0	0	3	5	3	11	4	2	2
rengorger	8	0	0	0	8	2	10	2	2	2
inclure	8	0	0	0	8	12	8	3	1	1
tapir	8	0	0	1	7	6	12	3	1	1
compenser	8	0	0	1	7	45	29	2	1	1
rétrograder	8	0	1	1	6	10	9	4	2	2
flageller	8	0	0	0	8	8	8	3	1	1
asservir	8	4	0	3	1	36	22	4	2	2
fulminer	8	0	0	1	7	17	3	3	1	1
poétiser	8	0	0	2	6	3	8	2	3	3
tarder	8	0	0	2	6	727	271	3	1	1
moisir	8	7	0	0	1	9	17	5	2	2
recéler	8	0	0	0	8	5	7	1	1	1
dégonfler	8	0	0	1	7	1	8	8	3	3
souligner	8	1	0	0	7	57	22	6	2	2
mortifier	8	0	0	3	5	3	20	5	2	2
dénuder	8	3	0	0	5	13	13	4	3	3
détonner	8	0	0	0	8	6	6	3	2	2
embaucher	7	0	0	1	6	5	5	4	2	2
efflanquer	7	6	0	0	1	6	11	3	4	4
meugler	7	0	0	1	6	0	3	1	1	1
étoiler	7	7	0	0	0	29	11	5	2	2
bouffir	7	5	0	0	2	17	16	6	5	5
édicter	7	0	0	2	5	7	8	2	1	1
carillonner	7	0	0	0	7	1	3	4	2	2
anoblir	7	1	0	0	6	9	9	2	2	2
recoudre	7	0	0	1	6	3	7	3	1	1
vagabonder	7	0	0	0	7	8	8	3	1	1
brasser	7	0	1	0	6	10	12	7	3	3
dérégler	7	4	0	0	3	5	0	5	5	5
incarcérer	7	1	0	0	6	8	12	3	2	2
amalgamer	7	2	0	0	5	11	4	2	1	1
gueuler	7	0	0	0	7	0	2	8	4	4
empaqueter	7	4	0	1	2	8	12	2	2	2
tamiser	7	0	0	0	7	12	13	4	2	2
gercer	7	3	0	0	4	7	5	4	2	2
enlaidir	7	0	0	2	5	8	7	7	4	4
regimber	7	0	0	3	4	7	6	4	1	1
interloquer	7	3	0	0	4	11	12	1	1	1
déchausser	7	1	0	0	6	2	11	7	2	2
désobliger	7	0	0	3	4	0	6	1	1	1
arquer	7	1	0	0	6	9	12	6	3	3
barder	7	6	0	0	1	5	12	7	3	3

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
enclaver	7	7	0	0	0	6	10	2	2	2
éperonner	7	0	1	0	6	8	20	2	1	1
coloniser	7	2	0	2	3	5	7	5	2	2
embrocher	7	0	0	1	6	6	6	5	2	2
soupeser	7	0	0	0	7	4	9	2	2	2
écorner	7	3	1	0	3	6	6	7	3	3
enchâsser	7	7	0	0	0	32	24	2	2	2
martyriser	7	2	0	0	5	12	15	3	1	1
ravaler	7	0	0	1	6	6	10	10	2	2
évaser	7	0	2	1	4	7	17	6	2	2
moyenner	7	0	7	0	0	45	26	4	1	1
recommander	7	0	0	7	0	607	150	8	1	1
rengainer	7	1	0	1	5	5	6	3	1	1
lutiner	7	0	0	2	5	3	7	1	1	1
insérer	7	0	0	1	6	49	20	5	1	1
fulgurer	7	0	0	0	7	3	1	1	1	1
tamponner	7	0	0	0	7	4	15	6	2	2
obliquer	7	0	2	1	4	1	13	2	2	2
présumer	7	7	0	0	0	123	44	2	2	2
échafauder	7	3	0	2	2	5	3	7	4	4
becqueter	7	0	1	2	4	8	4	5	4	4
affubler	7	0	0	0	7	28	26	4	2	2
daller	7	7	0	0	0	11	17	2	2	2
dodeliner	7	0	0	0	7	4	5	2	1	1
attiédir	7	1	0	1	5	6	8	2	2	2
croasser	7	0	2	1	4	4	8	1	1	1
dételer	7	2	0	0	5	23	13	4	1	1
piloter	7	0	0	2	5	0	1	5	1	1
picorer	7	0	0	0	7	5	7	7	2	2
commercer	7	0	0	5	2	2	9	2	1	1
panacher	7	4	0	0	3	2	3	3	2	2
jongler	7	0	0	0	7	2	2	3	1	1
crucifier	7	1	0	0	6	14	9	3	2	2
inoculer	7	0	0	0	7	8	6	4	2	2
pâturer	7	0	1	1	5	2	0	2	2	2
bâcler	7	0	0	2	5	2	8	3	3	3
enorgueillir	7	0	0	1	6	27	29	2	2	2
appendre	7	5	0	0	2	13	16	2	2	2
harasser	7	1	0	0	6	7	18	1	1	1
vivifier	7	0	0	1	6	12	12	2	2	2
renommer	7	7	0	0	0	56	27	3	1	1
radoter	7	0	0	1	6	7	10	3	2	2
saillir	7	0	0	1	6	4	5	4	1	1
dénigrer	7	0	0	2	5	3	9	1	1	1
étriller	7	0	0	2	5	4	8	4	1	1
affiler	7	3	0	1	3	7	8	2	3	3
fier	7	0	7	0	0	183	31	1	1	1
riposter	7	0	0	0	7	29	14	2	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
parfaire	7	0	0	7	0	1	0	0	1	1
fanatiser	7	3	0	0	4	4	11	2	2	2
octroyer	7	2	0	0	5	41	23	2	2	2
désabuser	7	0	0	1	6	8	8	4	2	2
biffer	7	0	0	2	5	5	10	5	2	2
solder	7	2	0	1	4	32	19	4	1	1
dépopulariser	7	3	0	1	3	0	1	2	2	2
découpler	7	2	0	0	5	11	6	5	2	2
ratisser	7	0	0	0	7	4	10	7	4	4
désagréger	7	0	0	0	7	3	17	4	1	1
repentir	7	0	0	0	7	62	73	1	1	1
tronquer	7	6	0	1	0	3	10	2	2	2
délayer	7	1	1	0	5	8	15	3	2	2
enfumer	7	3	0	3	1	10	19	5	1	1
affadir	7	1	0	1	5	7	4	2	3	3
déambuler	6	0	0	1	5	1	0	1	1	1
tanguer	6	0	0	0	6	3	6	2	1	1
moutonner	6	0	0	0	6	5	5	7	1	1
larder	6	0	0	1	5	3	10	3	2	2
grésiller	6	0	0	0	6	4	7	5	2	2
dépourvoir	6	6	0	0	0	95	74	3	1	1
objecter	6	0	0	6	0	89	13	3	2	2
lézarder	6	4	0	0	2	11	9	3	2	2
choir	6	0	0	5	1	3	2	3	1	1
infuser	6	1	0	1	4	4	5	5	3	3
croupir	6	1	0	0	5	5	11	5	1	1
réquisitionner	6	1	0	2	3	8	9	3	1	1
recroqueviller	6	0	0	0	6	5	7	3	2	2
écourter	6	3	0	1	2	10	8	2	2	2
retoucher	6	0	0	1	5	16	8	5	1	1
amarrer	6	1	0	2	3	35	27	3	2	2
revirer	6	1	1	2	2	3	2	1	1	1
distendre	6	0	0	0	6	3	14	5	2	2
intervertir	6	0	0	0	6	7	3	2	1	1
érailler	6	0	1	0	5	5	5	7	5	5
irradier	6	0	0	0	6	5	7	7	1	1
dénombrer	6	0	0	1	5	2	2	1	1	1
pivoter	6	0	1	0	5	7	6	4	1	1
singer	6	0	0	2	4	4	8	1	1	1
fracturer	6	1	0	3	2	4	10	3	1	1
recopier	6	0	0	1	5	1	6	2	1	1
solidifier	6	1	1	1	3	3	11	2	2	2
maugréer	6	0	1	1	4	12	6	2	1	1
enrôler	6	6	0	0	0	60	29	2	2	2
évider	6	3	0	0	3	2	7	2	1	1
préposer	6	6	0	0	0	26	14	3	1	1
ouïr	6	0	0	0	6	7	16	1	1	1
éduquer	6	2	0	1	3	3	5	5	4	4

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
épater	6	1	0	2	3	6	5	4	2	2
diffamer	6	1	0	1	4	2	6	1	1	1
désapprendre	6	0	0	1	5	5	6	2	1	1
friper	6	2	0	1	3	5	9	4	3	3
ficeler	6	0	0	2	4	7	7	6	2	2
corner	6	1	0	0	5	3	5	9	3	3
conformer	6	0	6	0	0	95	52	3	1	1
étoffer	6	2	0	3	1	7	3	6	1	1
démocratiser	6	0	0	4	2	0	2	4	2	2
assouplir	6	0	0	2	4	5	17	5	2	2
cristalliser	6	2	0	0	4	7	6	7	3	3
décocher	6	0	0	1	5	34	18	7	1	1
concéder	6	0	0	0	6	43	28	3	1	1
coexister	6	0	0	2	4	4	6	2	1	1
bosseler	6	4	0	0	2	5	8	3	2	2
enjoliver	6	0	0	0	6	3	8	3	2	2
moirer	6	2	0	0	4	9	11	3	2	2
fourrager	6	0	0	3	3	1	3	3	2	2
chatoyer	6	0	0	0	6	1	6	1	1	1
récupérer	6	1	0	2	3	6	2	8	2	2
limer	6	1	0	2	3	8	9	5	2	2
endurcir	6	0	0	0	6	6	12	7	3	3
réciter	6	1	0	2	3	161	50	3	1	1
truffer	6	6	0	0	0	9	9	4	1	1
stériliser	6	1	0	2	3	10	6	4	3	3
effriter	6	2	0	2	2	5	8	6	2	2
dialoguer	6	1	0	3	2	8	6	5	1	1
pirouetter	6	0	1	1	4	4	8	3	2	2
alléguer	6	2	1	2	1	93	35	3	1	1
intensifier	6	0	0	2	4	0	1	4	2	2
matérialiser	6	1	0	0	5	5	6	2	2	2
écoeurer	6	3	0	0	3	5	7	3	2	2
léser	6	0	0	1	5	2	13	1	1	1
moraliser	6	0	0	1	5	4	4	4	2	2
abroger	6	0	0	0	6	2	10	2	3	3
dévider	6	0	0	1	5	5	5	5	1	1
besogner	6	0	0	0	6	1	3	2	2	2
piailler	6	0	0	0	6	2	6	3	2	2
ravoir	6	0	0	6	0	1	2	0	2	2
valider	6	0	0	2	4	1	8	1	1	1
recenser	6	0	1	0	5	4	3	2	1	1
butiner	6	0	1	0	5	3	4	2	2	2
contaminer	6	0	0	1	5	3	5	3	3	3
décrier	6	2	0	0	4	8	14	1	1	1
baver	6	0	0	0	6	3	7	5	2	2
pomper	6	0	0	1	5	4	14	10	3	3
enraciner	6	3	0	0	3	22	21	5	2	2
sécréter	6	3	0	0	3	4	5	2	2	2

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
empaler	6	1	0	0	5	4	11	2	2	2
jubiler	6	0	0	0	6	1	4	1	1	1
esclaffer	6	0	1	1	4	1	4	1	1	1
philosopher	6	0	2	1	3	2	12	3	3	3
frustrer	6	0	0	5	1	8	14	2	1	1
happer	6	0	0	1	5	15	12	3	2	2
péter	6	0	0	0	6	2	1	8	4	4
pelotonner	6	0	1	1	4	9	12	4	1	1
remémorer	6	1	0	0	5	19	13	2	1	1
enduire	6	0	0	0	6	12	12	4	1	1
démembrer	6	0	1	0	5	6	10	5	2	2
emboucher	6	0	0	0	6	6	8	5	2	2
griffonner	6	0	0	0	6	10	20	2	1	1
poivrer	6	6	0	0	0	6	2	7	2	2
teinter	6	0	0	0	6	30	18	2	1	1
trépasser	6	0	0	1	5	16	7	1	1	1
tisonner	6	0	1	2	3	1	6	2	3	3
affûter	6	2	0	2	2	0	5	4	2	2
dévaliser	6	0	0	1	5	34	11	3	1	1
asphyxier	6	1	0	2	3	10	8	7	3	3
émoustiller	6	1	0	1	4	2	3	1	1	1
innocenter	6	0	0	3	3	6	4	2	1	1
soûler	6	0	1	1	4	2	6	5	4	4
solenniser	6	0	0	1	5	5	2	2	3	3
béer	6	0	1	0	5	9	10	2	2	2
pactiser	5	0	0	2	3	3	3	2	1	1
sarcler	5	0	0	0	5	0	6	2	1	1
débouter	5	0	0	0	5	3	1	3	1	1
domicilier	5	3	0	0	2	4	7	4	1	1
molester	5	1	0	0	4	12	5	1	1	1
atrophier	5	0	0	0	5	4	13	3	4	4
parachever	5	0	0	0	5	6	7	2	2	2
bayer	5	0	1	1	3	1	3	1	1	1
trémousser	5	0	0	0	5	2	9	1	1	1
maniérer	5	4	0	0	1	13	6	2	1	1
mimer	5	0	1	0	4	7	10	3	2	2
déprimer	5	0	0	0	5	4	8	7	3	3
emmancher	5	1	0	0	4	20	6	5	3	3
vendanger	5	0	0	2	3	0	2	2	2	2
corroder	5	1	0	1	3	2	4	2	2	2
taillader	5	1	0	0	4	3	9	2	2	2
vivoter	5	0	0	0	5	1	2	1	1	1
inventorier	5	1	0	1	3	2	9	1	1	1
ponctuer	5	0	0	0	5	10	14	4	2	2
basaner	5	5	0	0	0	16	8	2	2	2
implorer	5	0	0	0	5	168	83	2	1	1
transgresser	5	0	0	0	5	1	3	2	1	1
défriser	5	4	0	0	1	2	2	4	2	2

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
pleurnicher	5	0	0	0	5	3	3	2	1	1
minauder	5	0	0	0	5	4	4	1	1	1
mordiller	5	0	0	1	4	6	11	1	1	1
compulser	5	0	1	1	3	3	11	2	1	1
monopoliser	5	1	0	1	3	5	3	5	2	2
rimer	5	0	0	1	4	18	5	4	2	2
franger	5	4	1	0	0	22	11	3	1	1
réélire	5	0	1	0	4	18	12	1	1	1
préciter	5	5	0	0	0	6	10	2	1	1
radoucir	5	0	0	0	5	4	4	5	2	2
stigmatiser	5	0	0	1	4	6	10	3	1	1
augurer	5	0	0	0	5	22	7	1	1	1
éreinter	5	2	0	1	2	17	13	3	2	2
fourbir	5	1	1	0	3	2	9	3	2	2
rebattre	5	0	0	0	5	8	8	6	1	1
tacheter	5	5	0	0	0	16	12	2	2	2
embrumer	5	1	0	0	4	11	13	3	2	2
masturber	5	0	0	1	4	6	8	2	1	1
verdoyer	5	0	0	0	5	4	15	1	1	1
dessoûler	5	0	0	0	5	1	1	3	2	2
grever	5	0	0	0	5	9	5	3	2	2
préconcevoir	5	5	0	0	0	2	12	2	1	1
juxtaposer	5	5	0	0	0	21	5	2	1	1
rincer	5	1	0	2	2	5	9	6	2	2
énoncer	5	3	0	0	2	39	26	2	2	2
moudre	5	0	0	3	2	3	3	6	3	3
plomber	5	2	0	0	3	4	5	14	4	4
denteler	5	3	0	0	2	11	15	2	2	2
surélever	5	1	0	1	3	7	7	1	1	1
boursoufler	5	3	0	0	2	7	8	4	3	3
déclencher	5	0	0	0	5	1	3	3	3	3
caserner	5	4	0	1	0	6	9	3	2	2
rêvasser	5	0	0	0	5	6	9	1	1	1
brunir	5	1	0	0	4	19	18	6	3	3
papilloter	5	0	0	0	5	2	2	5	1	1
suborner	5	0	0	1	4	6	8	1	1	1
nacrer	5	5	0	0	0	3	3	4	1	1
diagnostiquer	5	0	0	2	3	3	5	2	2	2
piler	5	2	0	2	1	2	10	5	3	3
égosiller	5	0	0	0	5	2	5	1	1	1
militer	5	0	0	0	5	4	4	3	1	1
gangrener	5	2	0	0	3	5	6	4	4	4
assainir	5	0	0	1	4	4	8	2	3	3
remontrer	5	0	0	0	5	12	8	2	1	1
amplifier	5	0	0	0	5	8	11	4	2	2
convoyer	5	0	0	1	4	2	4	1	1	1
déporter	5	1	0	0	4	15	12	3	1	1
survoler	5	0	0	1	4	0	3	3	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
exempter	5	0	0	0	5	9	7	2	1	1
plaquer	5	1	1	0	3	23	14	11	2	2
bistrer	5	4	0	0	1	4	8	2	3	3
sucrer	5	1	0	2	2	3	6	8	2	2
coordonner	5	0	0	1	4	12	11	5	1	1
préfigurer	5	1	0	0	4	0	0	2	1	1
mousser	5	0	0	1	4	2	2	2	2	2
cheviller	5	5	0	0	0	4	2	2	2	2
espacer	5	5	0	0	0	49	32	2	1	1
refondre	5	0	0	3	2	5	6	5	1	1
dévisser	5	0	0	0	5	3	4	8	3	3
gicler	5	0	1	0	4	1	2	3	2	2
crisser	5	0	0	1	4	2	1	1	1	1
débrider	5	1	0	1	3	5	6	6	2	2
bramer	5	0	0	0	5	2	2	3	2	2
allécher	5	2	0	1	2	17	6	2	1	1
recréer	5	1	0	1	3	1	6	2	2	2
coasser	5	0	0	1	4	1	2	1	1	1
grimer	5	2	0	0	3	1	8	2	2	2
gober	5	0	0	3	2	5	5	6	2	2
fréter	5	4	0	0	1	10	3	2	2	2
borner	5	3	2	0	0	535	115	5	3	3
scalper	5	1	0	1	3	6	4	2	2	2
vagir	4	0	2	0	2	2	5	3	3	3
raviner	4	1	0	0	3	7	4	3	2	2
promouvoir	4	2	0	2	0	37	13	2	2	2
viriliser	4	0	0	0	4	1	2	4	2	2
câliner	4	0	0	0	4	3	6	1	1	1
lacérer	4	1	0	0	3	6	9	3	2	2
sabler	4	4	0	0	0	24	10	2	2	2
chansonner	4	0	1	0	3	1	5	1	1	1
tuméfier	4	2	0	0	2	6	8	1	1	1
acérer	4	4	0	0	0	18	11	4	2	2
agrémenter	4	0	0	0	4	21	11	4	1	1
carboniser	4	4	0	0	0	3	6	4	2	2
débusquer	4	0	0	0	4	3	4	4	2	2
encanailler	4	0	0	2	2	3	3	2	2	2
transposer	4	1	1	1	1	16	4	5	3	3
perforer	4	0	0	1	3	2	6	2	2	2
papillonner	4	0	1	1	2	2	3	1	1	1
mater	4	0	0	1	3	4	3	7	2	2
calfeutrer	4	3	0	0	1	1	8	3	2	2
émarger	4	0	0	1	3	2	0	5	1	1
écailler	4	2	0	1	1	9	6	5	2	2
transsuder	4	0	0	0	4	1	1	1	1	1
coaguler	4	2	0	1	1	0	7	5	3	3
enjôler	4	0	0	0	4	3	6	1	1	1
blondir	4	0	0	0	4	2	1	2	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
emmagasiner	4	2	0	0	2	18	20	4	2	2
velouter	4	3	0	0	1	10	12	2	1	1
rationner	4	0	0	0	4	4	3	4	1	1
ébouriffer	4	3	0	0	1	9	15	3	2	2
receler	4	0	0	0	4	0	9	5	3	3
aérer	4	0	0	0	4	10	8	9	2	2
atermoyer	4	0	0	0	4	0	1	1	1	1
endiguer	4	1	0	3	0	3	4	2	1	1
fouir	4	0	1	2	1	0	1	3	2	2
morceler	4	0	0	1	3	3	9	4	2	2
exploser	4	0	0	0	4	0	0	5	2	2
clarifier	4	0	0	2	2	3	3	2	1	1
reployer	4	0	1	0	3	2	1	2	2	2
renfler	4	0	0	1	3	8	14	3	2	2
engluer	4	1	0	0	3	5	4	5	2	2
tempêter	4	0	0	2	2	3	5	2	1	1
téléphoner	4	0	0	2	2	0	3	4	3	3
astiquer	4	1	1	1	1	4	5	5	2	2
érafler	4	0	1	0	3	4	5	5	2	2
créneler	4	4	0	0	0	18	8	3	2	2
accidenter	4	4	0	0	0	11	11	4	2	2
avérer	4	3	0	0	1	14	36	3	2	2
invectiver	4	0	0	3	1	9	11	2	1	1
cadenasser	4	1	0	1	2	6	5	4	2	2
désillusionner	4	0	0	0	4	2	6	2	2	2
strier	4	4	0	0	0	13	3	2	2	2
expatrier	4	0	0	0	4	6	2	6	2	2
oblitérer	4	0	0	0	4	2	6	3	3	3
pomponner	4	1	0	2	1	1	2	2	1	1
réexpédier	4	1	0	0	3	1	0	4	2	2
écharper	4	0	0	1	3	4	5	3	1	1
corser	4	0	0	0	4	3	2	6	1	1
entêter	4	0	0	0	4	62	28	4	1	1
circonvenir	4	0	0	1	3	8	7	3	2	2
numériser	4	1	0	0	3	6	6	1	1	1
fertiliser	4	0	0	1	3	4	16	2	2	2
informer	4	0	1	3	0	424	158	2	2	2
convoler	4	0	0	1	3	3	3	2	1	1
chiper	4	1	0	0	3	5	7	7	3	3
exorciser	4	0	0	1	3	2	3	3	1	1
catéchiser	4	0	0	2	2	4	3	1	1	1
démoder	4	3	0	0	1	11	4	6	3	3
émonder	4	1	0	1	2	4	3	2	1	1
remâcher	4	1	1	0	2	2	1	2	1	1
pétarader	4	0	0	1	3	1	2	1	1	1
reboutonner	4	0	1	0	3	0	0	2	2	2
prédestiner	4	2	0	0	2	12	8	2	1	1
magnétiser	4	0	0	1	3	4	4	3	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
abasourdir	4	1	0	0	3	15	18	2	1	1
peler	4	1	0	1	2	8	6	5	2	2
hypothéquer	4	2	1	1	0	7	5	3	2	2
trousser	4	3	0	0	1	7	7	6	3	3
rembarrer	4	0	0	1	3	0	1	1	1	1
gazer	4	0	0	0	4	0	2	6	1	1
empreindre	4	1	0	0	3	114	63	2	2	2
prophétiser	4	0	1	2	1	40	15	3	1	1
dogmatiser	4	0	0	1	3	0	3	1	1	1
serpenter	4	0	4	0	0	57	26	1	1	1
volatiliser	4	0	0	0	4	3	7	4	2	2
raturer	4	1	0	0	3	6	1	2	3	3
engouffrer	4	1	0	0	3	70	37	6	2	2
désemparer	4	1	0	0	3	7	9	2	1	1
piauler	4	0	0	0	4	0	0	1	1	1
déméríter	4	0	0	0	4	2	3	2	1	1
assortir	4	2	0	0	2	4	8	8	1	1
affilier	4	4	0	0	0	19	12	2	2	2
consteller	4	4	0	0	0	9	9	2	1	1
anticiper	4	2	0	0	2	9	4	2	2	2
transir	4	4	0	0	0	20	15	2	2	2
mouvementer	4	3	0	0	1	6	5	4	3	3
mobiliser	4	0	0	1	3	5	5	7	2	2
infirmer	4	0	0	2	2	4	6	3	3	3
étiqueter	4	0	0	1	3	15	14	2	1	1
étioler	4	0	0	0	4	0	12	5	2	2
jauger	4	0	0	2	2	3	2	3	2	2
engorger	4	2	0	0	2	3	6	2	2	2
vêler	4	0	0	1	3	1	1	2	1	1
coaliser	4	4	0	0	0	18	5	2	2	2
endiabler	4	4	0	0	0	2	8	3	3	3
ravauder	4	0	0	2	2	3	2	2	1	1
bisser	4	0	0	0	4	1	2	3	3	3
retraiter	4	0	1	0	3	3	7	4	3	3
mijoter	4	0	0	0	4	1	2	5	2	2
goguenarder	4	0	0	1	3	1	2	2	1	1
plafonner	4	1	0	0	3	3	2	4	2	2
débourser	4	0	0	2	2	2	1	2	2	2
récrire	4	0	0	1	3	2	1	3	1	1
nomadiser	4	0	0	0	4	1	4	3	1	1
englober	4	1	0	0	3	28	13	2	2	2
peinturlurer	4	4	0	0	0	5	2	2	2	2
embourber	4	2	0	2	0	13	17	7	1	1
manigancer	4	0	0	2	2	4	3	2	3	3
canaliser	4	0	0	1	3	0	2	3	2	2
soucier	4	0	0	1	3	232	67	2	3	3
patiner	4	0	0	2	2	3	4	5	2	2
asperger	4	0	0	0	4	11	10	4	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
générer	4	0	0	2	2	1	2	2	2	2
moderniser	4	0	0	1	3	4	4	4	2	2
tripoter	4	0	0	0	4	3	6	8	2	2
agripper	4	0	0	0	4	8	10	4	1	1
congeler	4	3	0	0	1	6	6	2	2	2
ratatiner	4	0	0	1	3	4	2	7	2	2
récréer	4	0	0	2	2	5	6	2	1	1
étrenner	4	0	1	0	3	0	3	3	1	1
bûcher	3	0	1	0	2	3	1	7	2	2
rafler	3	0	0	1	2	4	4	4	1	1
ahaner	3	0	0	1	2	0	0	1	1	1
écrouer	3	0	0	0	3	1	8	1	1	1
marqueter	3	3	0	0	0	4	6	3	2	2
épingler	3	1	0	0	2	11	3	5	3	3
évangéliser	3	0	0	1	2	6	2	1	1	1
imputer	3	0	0	0	3	48	37	2	1	1
introniser	3	0	0	1	2	2	2	3	2	2
ausculter	3	0	1	0	2	3	5	3	1	1
puiser	3	2	1	0	0	246	91	1	1	1
débloquer	3	0	0	0	3	4	7	8	2	2
invétérer	3	1	0	0	2	5	8	3	3	3
clairsemer	3	1	0	0	2	5	7	2	1	1
piéter	3	0	0	0	3	0	0	5	2	2
gîter	3	0	0	0	3	0	2	8	2	2
zébrer	3	3	0	0	0	16	8	2	1	1
embuer	3	3	0	0	0	3	3	3	2	2
recorder	3	0	1	2	0	1	1	4	1	1
empêtrer	3	2	1	0	0	20	11	4	1	1
enfieller	3	0	0	0	3	0	0	2	1	1
arguer	3	0	0	2	1	6	2	5	2	2
mitiger	3	1	0	0	2	6	1	4	2	2
fusionner	3	0	0	0	3	3	2	4	2	2
diverger	3	0	0	0	3	4	3	4	1	1
rebaptiser	3	0	0	0	3	5	1	2	1	1
caparaçonner	3	3	0	0	0	8	12	5	1	1
biner	3	0	0	0	3	1	3	3	2	2
piger	3	0	0	1	2	0	1	6	2	2
occasionner	3	0	0	0	3	106	51	1	1	1
revivifier	3	0	0	1	2	0	6	1	1	1
rougeoyer	3	0	0	0	3	3	4	1	1	1
monnayer	3	0	0	3	0	6	4	3	1	1
terrer	3	0	0	0	3	3	6	3	1	1
supplicier	3	0	0	0	3	5	6	2	1	1
déplisser	3	0	1	0	2	0	1	3	1	1
enrouler	3	3	0	0	0	78	57	4	3	3
rissoler	3	0	0	0	3	1	1	3	2	2
enluminer	3	3	0	0	0	18	15	3	1	1
épiloguer	3	0	0	0	3	2	2	2	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
distancer	3	0	0	0	3	5	9	1	1	1
graisser	3	0	0	2	1	3	5	5	1	1
truquer	3	0	0	1	2	1	3	5	2	2
charpenter	3	3	0	0	0	3	5	6	3	3
ressasser	3	0	0	0	3	6	3	3	1	1
découdre	3	0	0	2	1	6	15	5	1	1
acclimater	3	0	0	2	1	17	8	5	1	1
engoncer	3	3	0	0	0	8	5	2	1	1
déboucler	3	0	0	0	3	3	7	4	2	2
abêtir	3	1	0	0	2	3	3	2	2	2
costumer	3	3	0	0	0	20	11	2	2	2
bougonner	3	0	0	0	3	6	6	2	1	1
vulgariser	3	0	0	0	3	6	4	1	1	1
adjuger	3	0	0	0	3	15	4	2	1	1
agréger	3	0	0	0	3	3	4	4	2	2
racornir	3	0	0	0	3	3	1	6	2	2
ceinturer	3	1	0	0	2	1	6	5	2	2
dramatiser	3	0	0	2	1	3	2	3	1	1
réfréner	3	1	0	0	2	4	2	2	2	2
hâler	3	1	0	0	2	20	13	2	2	2
reboucher	3	2	0	0	1	1	3	2	2	2
épousseter	3	1	0	2	0	15	15	2	1	1
controverser	3	2	0	0	1	2	4	1	1	1
renfoncer	3	0	0	0	3	23	12	3	1	1
bedonner	3	1	0	0	2	0	0	1	1	1
boulonner	3	2	0	0	1	2	2	4	2	2
pédaler	3	0	0	1	2	0	1	4	2	2
gracier	3	0	0	0	3	4	11	1	1	1
violacer	3	0	0	0	3	5	1	3	2	2
saouler	3	0	0	1	2	3	2	2	3	3
iriser	3	0	0	0	3	10	5	3	2	2
piper	3	2	0	0	1	3	4	5	2	2
repiquer	3	0	0	0	3	6	6	5	1	1
instrumenter	3	0	0	0	3	1	3	5	1	1
anathématiser	3	0	0	0	3	1	6	1	1	1
échancrer	3	1	0	0	2	12	9	3	1	1
sabrer	3	0	0	1	2	18	14	8	2	2
péricliter	3	0	0	0	3	3	2	1	1	1
souder	3	2	1	0	0	39	16	5	2	2
ergoter	3	0	0	1	2	0	4	2	1	1
braconner	3	0	0	2	1	1	0	1	1	1
soudoyer	3	0	0	1	2	7	9	1	1	1
formaliser	3	0	0	2	1	5	3	4	2	2
débarbouiller	3	0	1	1	1	6	7	3	2	2
chuter	3	0	0	0	3	0	0	5	2	2
réédifier	3	0	0	1	2	4	4	2	2	2
raréfier	3	0	2	0	1	4	9	2	2	2
gazonner	3	3	0	0	0	5	4	3	3	3

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
jaboter	3	0	0	0	3	2	0	3	1	1
éborgner	3	0	0	2	1	2	3	4	1	1
fouailler	3	0	0	1	2	0	4	6	2	2
muser	3	0	0	0	3	0	2	2	1	1
vanter	3	0	1	2	0	323	140	2	1	1
maquiller	3	1	1	0	1	2	0	9	1	1
repenser	3	0	1	0	2	3	7	3	2	2
sangler	3	1	0	0	2	19	8	2	1	1
bouffonner	3	0	0	0	3	1	1	1	1	1
démâter	3	3	0	0	0	2	4	3	1	1
enferrer	3	1	0	0	2	4	3	5	1	1
aiguiller	3	0	0	1	2	2	2	5	1	1
libeller	3	1	0	0	2	3	1	2	3	3
dépolir	3	2	0	0	1	2	6	2	2	2
touer	3	0	0	1	2	0	2	2	2	2
sophistiquer	3	1	0	0	2	3	4	2	2	2
bouter	3	0	0	0	3	3	1	1	1	1
confire	3	1	0	0	2	7	4	3	3	3
calmir	3	0	0	0	3	0	0	1	1	1
nimber	3	3	0	0	0	11	2	4	1	1
briqueter	3	3	0	0	0	0	2	4	2	2
dévoyer	3	2	0	0	1	3	4	5	2	2
parapher	3	1	0	0	2	3	2	2	2	2
monologuer	3	0	2	0	1	3	2	1	1	1
drainer	3	0	0	0	3	1	3	5	2	2
rabrouer	3	0	0	1	2	3	6	1	1	1
remmener	3	0	0	0	3	1	3	1	1	1
toquer	3	1	0	0	2	10	1	4	2	2
entériner	3	0	0	0	3	3	3	3	1	1
crevasser	3	3	0	0	0	5	14	2	1	1
légaliser	3	1	0	1	1	4	5	1	1	1
manipuler	3	1	0	1	1	3	5	5	2	2
clabauder	3	0	0	1	2	2	3	3	1	1
saccader	3	1	0	0	2	8	8	2	2	2
préexister	3	0	0	0	3	1	1	1	1	1
compiler	3	0	0	2	1	3	4	3	2	2
mitrailler	3	0	1	0	2	7	6	5	2	2
photographier	3	1	0	1	1	5	8	3	2	2
fructifier	3	0	0	0	3	5	10	1	1	1
mastiquer	3	0	0	1	2	2	7	4	2	2
homologuer	3	1	0	0	2	2	0	2	2	2
affleurer	3	0	1	1	1	4	7	4	2	2
pagayer	3	0	1	0	2	0	1	1	1	1
rapiécer	3	1	0	0	2	5	14	2	2	2
quadriller	3	2	1	0	0	4	1	3	1	1
basculer	3	0	0	1	2	1	1	8	2	2
inciser	3	0	0	0	3	2	5	2	2	2
signaler	3	0	0	3	0	798	169	6	2	2

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
séculariser	3	1	0	2	0	3	8	3	2	2
aposter	3	1	0	0	2	7	7	2	2	2
notifier	3	0	0	1	2	13	12	2	1	1
trompeter	3	0	0	2	1	1	1	4	1	1
tituber	3	0	1	0	2	5	12	2	1	1
jouter	3	0	0	0	3	6	2	2	1	1
déjeter	3	2	0	0	1	4	3	2	2	2
raccorder	3	0	0	0	3	9	4	5	1	1
thésauriser	3	0	0	1	2	2	5	2	2	2
exacerber	3	0	0	0	3	1	3	3	2	2
élaguer	3	2	1	0	0	5	1	2	2	2
appertiser	3	0	0	0	3	0	0	2	1	1
pourlécher	3	0	0	1	2	1	3	4	1	1
dépiter	3	1	0	0	2	18	12	2	2	2
chier	3	0	0	0	3	0	2	3	1	1
torcher	3	0	0	0	3	4	3	7	1	1
gruger	3	1	0	0	2	0	2	4	3	3
forniquer	3	0	0	1	2	1	0	3	1	1
baller	2	0	0	0	2	2	1	1	1	1
harponner	2	0	0	1	1	2	2	3	1	1
stagner	2	0	0	0	2	1	0	3	1	1
procréer	2	0	1	1	0	5	4	2	1	1
délurer	2	2	0	0	0	2	2	1	1	1
contresigner	2	1	0	0	1	4	2	2	1	1
accoutrer	2	2	0	0	0	17	7	2	2	2
liquéfier	2	0	0	1	1	5	6	4	1	1
fronder	2	0	0	0	2	3	1	2	1	1
confronter	2	0	1	0	1	22	10	4	2	2
désencombrer	2	1	0	1	0	2	1	2	1	1
ambrer	2	1	0	0	1	3	1	4	2	2
bouillotter	2	0	0	1	1	0	0	1	1	1
endeuiller	2	2	0	0	0	2	2	3	2	2
complimenter	2	0	0	0	2	61	22	1	1	1
gigoter	2	0	0	0	2	1	2	2	1	1
barrir	2	0	0	0	2	0	0	1	1	1
déférer	2	1	1	0	0	37	24	3	1	1
laïciser	2	0	0	0	2	1	0	2	1	1
vanner	2	0	0	2	0	6	4	6	1	1
empocher	2	0	0	0	2	17	8	3	1	1
contrebalancer	2	0	0	0	2	5	3	3	1	1
bâfrer	2	0	0	0	2	0	0	2	1	1
piqueter	2	0	0	0	2	3	5	6	1	1
réviser	2	0	0	0	2	2	9	4	1	1
gauchir	2	0	0	0	2	3	2	4	2	2
démancher	2	1	0	0	1	0	1	8	2	2
bouchonner	2	0	0	0	2	0	3	6	1	1
renâcler	2	0	0	0	2	4	3	2	1	1
dérailler	2	0	0	0	2	1	5	7	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
déchanter	2	0	0	1	1	0	0	2	1	1
conspuer	2	0	0	0	2	6	12	1	1	1
vernisser	2	2	0	0	0	10	6	2	1	1
émacier	2	1	0	0	1	6	3	2	1	1
nasiller	2	0	1	0	1	1	2	5	2	2
encourir	2	2	0	0	0	60	18	1	1	1
dilapider	2	0	0	0	2	2	6	3	2	2
cuivrer	2	2	0	0	0	6	7	4	2	2
tester	2	0	0	1	1	12	7	5	2	2
oxygéner	2	2	0	0	0	4	5	5	2	2
inculper	2	0	0	0	2	10	5	1	1	1
méduser	2	1	0	0	1	3	2	1	1	1
gager	2	2	0	0	0	24	5	5	1	1
récidiver	2	0	0	1	1	0	0	2	1	1
vernir	2	1	0	0	1	3	5	8	2	2
cabrioler	2	0	0	1	1	0	3	1	1	1
diaprer	2	2	0	0	0	11	7	2	1	1
sertir	2	2	0	0	0	11	2	2	1	1
lamper	2	0	0	0	2	3	7	2	1	1
agencer	2	0	0	0	2	5	5	7	2	2
individualiser	2	0	0	1	1	2	1	4	1	1
imbiber	2	0	0	0	2	25	14	4	1	1
torréfier	2	2	0	0	0	1	0	2	1	1
diamanter	2	1	0	0	1	3	4	2	1	1
trimballer	2	0	0	1	1	0	0	3	1	1
cajoler	2	0	0	0	2	2	12	2	1	1
bourreler	2	1	0	0	1	6	3	1	1	1
siroter	2	0	1	0	1	2	4	2	1	1
réoccuper	2	0	0	0	2	3	5	1	1	1
compter	2	0	0	2	0	1	0	0	2	2
illimiter	2	2	0	0	0	14	11	2	1	1
ramollir	2	0	0	0	2	2	7	7	1	1
émailler	2	0	0	0	2	46	23	3	1	1
galvauder	2	0	0	0	2	0	1	1	1	1
gloser	2	0	0	0	2	5	5	4	1	1
coupler	2	0	0	0	2	4	0	4	1	1
scanner	2	1	0	0	1	4	3	2	2	2
zipper	2	2	0	0	0	0	0	2	1	1
imbriquer	2	2	0	0	0	3	4	5	1	1
bafouiller	2	0	0	0	2	0	4	3	2	2
anatomiser	2	0	0	0	2	0	0	1	1	1
retaper	2	1	0	0	1	1	2	6	2	2
ester	2	0	2	0	0	49	26	1	1	1
diaphanéiser	2	0	0	0	2	0	0	2	2	2
repriser	2	0	0	0	2	3	2	2	1	1
expurger	2	2	0	0	0	2	1	2	1	1
rengager	2	0	0	2	0	2	3	6	2	2
bâtonner	2	0	0	2	0	7	5	4	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
infiltrer	2	1	0	0	1	13	9	5	2	2
métamorphoser	2	2	0	0	0	53	20	3	1	1
rémunérer	2	0	0	0	2	7	8	3	1	1
dévoiler	2	2	0	0	0	163	52	5	1	1
biaiser	2	0	0	1	1	4	1	2	1	1
pisser	2	0	0	0	2	2	2	8	2	2
débouler	2	0	0	0	2	4	4	5	1	1
entraider	2	0	0	0	2	0	0	1	1	1
boxer	2	0	0	1	1	0	3	5	2	2
répugner	2	0	2	0	0	186	57	3	2	2
persifler	2	0	0	0	2	2	5	1	1	1
dégoiser	2	0	0	0	2	0	0	2	1	1
frire	2	0	0	0	2	1	2	3	2	2
systématiser	2	0	0	1	1	3	2	2	2	2
confiner	2	1	1	0	0	74	33	4	1	1
encastrer	2	2	0	0	0	16	8	2	2	2
regrimper	2	0	0	0	2	1	3	2	1	1
avarier	2	1	0	0	1	4	7	3	1	1
frelater	2	1	0	0	1	2	2	3	1	1
naufrager	2	2	0	0	0	4	6	2	1	1
rejuger	2	0	0	0	2	0	0	1	1	1
ciller	2	0	1	0	1	1	2	6	2	2
surajouter	2	2	0	0	0	4	4	2	2	2
brancher	2	1	0	0	1	0	0	15	2	2
tympaniser	2	0	0	0	2	1	2	1	1	1
envaser	2	0	0	1	1	0	1	2	1	1
busquer	2	1	0	0	1	2	4	3	1	1
révulser	2	0	0	0	2	0	1	4	2	2
paver	2	0	0	0	2	23	9	2	1	1
reluquer	2	0	0	0	2	2	3	3	1	1
innover	2	0	0	1	1	0	6	2	1	1
toiletter	2	1	0	0	1	0	0	3	1	1
braire	2	0	0	0	2	1	5	3	1	1
dessaisir	2	0	0	1	1	6	1	2	1	1
épeurer	2	2	0	0	0	3	0	2	2	2
étriquer	2	1	0	0	1	4	3	3	2	2
fauter	2	0	0	0	2	0	1	1	1	1
bruiner	2	0	0	0	2	0	1	1	1	1
dépoétiser	2	0	0	0	2	0	1	2	2	2
larguer	2	0	0	0	2	8	11	6	1	1
sélectionner	2	1	1	0	0	0	7	4	1	1
titrer	2	2	0	0	0	18	14	4	1	1
empresser	2	2	0	0	0	509	168	2	1	1
bleuter	2	2	0	0	0	0	1	2	1	1
guinder	2	1	0	0	1	14	7	5	2	2
conjuguer	2	0	0	0	2	2	1	3	1	1
picoter	2	0	0	0	2	1	3	6	2	2
triturer	2	0	0	0	2	2	3	3	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
apeurer	2	1	0	0	1	8	5	1	1	1
verbaliser	2	0	0	2	0	3	0	0	1	1
abstraire	2	1	0	1	0	2	1	2	2	2
savonner	2	1	0	0	1	5	1	4	2	2
déteindre	2	1	0	0	1	21	6	3	1	1
diluer	2	0	0	0	2	4	2	5	1	1
temporiser	2	0	0	1	1	1	4	1	1	1
rabâcher	2	0	0	0	2	8	2	5	1	1
gaver	2	0	0	0	2	8	3	4	1	1
sublimiser	2	0	0	0	2	0	1	2	1	1
zigzaguer	2	0	0	0	2	1	3	2	1	1
ingurgiter	2	0	1	0	1	2	6	1	1	1
hoqueter	2	0	0	0	2	2	1	3	1	1
peloter	2	0	0	0	2	3	0	6	2	2
fluer	2	0	0	0	2	0	3	2	1	1
admonester	2	0	0	0	2	2	3	1	1	1
désorienter	2	0	0	0	2	16	10	2	1	1
déguerpir	2	0	0	1	1	2	5	1	1	1
crêper	2	1	0	0	1	1	6	2	1	1
assécher	2	0	0	0	2	3	1	3	1	1
excommunier	2	2	0	0	0	37	25	1	1	1
constiper	2	1	0	0	1	1	0	2	2	2
vermillonner	2	1	0	0	1	1	3	3	2	2
informatiser	2	2	0	0	0	1	0	3	2	2
versifier	2	0	0	0	2	5	3	3	1	1
agglutiner	2	1	0	0	1	0	4	3	2	2
dépareiller	2	0	0	0	2	0	0	2	2	2
déganter	2	2	0	0	0	9	6	2	1	1
dicter	2	0	0	0	2	274	114	2	1	1
saucer	2	0	0	0	2	4	1	4	1	1
abouler	2	0	0	1	1	0	1	3	1	1
blaser	2	2	0	0	0	22	10	2	2	2
reperdre	2	0	0	0	2	3	1	2	2	2
putréfier	2	0	0	0	2	4	5	3	2	2
cautionner	2	0	0	2	0	1	0	0	1	1
coter	2	0	0	0	2	2	8	4	2	2
décalquer	2	1	0	0	1	1	3	2	2	2
déposséder	2	0	0	0	2	19	10	2	1	1
pocher	2	0	0	0	2	4	4	5	1	1
ficher	2	0	0	2	0	0	0	0	1	1
décolorer	2	2	0	0	0	51	15	2	1	1
forfaire	2	0	0	1	1	2	6	2	1	1
amputer	2	0	0	1	1	8	8	4	1	1
infatuer	2	2	0	0	0	6	3	1	1	1
disjoindre	2	0	0	0	2	4	5	4	1	1
paganiser	2	1	0	0	1	1	0	2	2	2
quadrupler	2	0	0	0	2	1	2	2	2	2
polariser	2	1	0	0	1	0	0	6	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
incurver	2	0	0	0	2	3	4	4	1	1
légiférer	2	0	0	0	2	2	1	2	1	1
occire	2	0	0	0	2	1	4	1	1	1
échelonner	2	2	0	0	0	40	23	4	2	2
fricoter	2	0	0	1	1	0	1	7	1	1
chipoter	2	0	0	1	1	2	1	5	1	1
pommader	2	2	0	0	0	3	10	3	1	1
crotter	2	2	0	0	0	14	3	4	1	1
moucheter	2	2	0	0	0	14	2	2	1	1
étouper	2	2	0	0	0	0	1	2	1	1
pointiller	2	2	0	0	0	3	4	2	1	1
dissocier	2	0	0	0	2	3	1	4	1	1
barboter	2	0	0	0	2	7	6	4	1	1
poudroyer	2	0	0	0	2	1	3	2	2	2
pouponner	2	0	0	0	2	0	0	1	1	1
cirer	2	2	0	0	0	15	7	2	2	2
mécaniser	2	2	0	0	0	3	2	2	1	1
surcharger	2	0	0	0	2	47	23	3	1	1
poisser	2	1	0	0	1	1	4	6	1	1
converser	2	0	2	0	0	39	12	2	2	2
remplumer	2	0	0	1	1	0	0	6	2	2
placarder	2	0	0	0	2	10	9	6	2	2
pommeler	2	2	0	0	0	4	4	3	1	1
prélever	2	2	0	0	0	33	11	2	1	1
roter	2	0	0	1	1	0	0	2	1	1
résorber	2	0	0	0	2	2	2	3	1	1
excaver	2	0	0	0	2	2	2	2	2	2
pioncer	2	0	0	1	1	0	0	1	1	1
spiritualiser	2	0	1	0	1	5	3	2	2	2
rhabiller	2	0	0	0	2	11	11	5	1	1
consister	2	0	2	0	0	798	154	1	1	1
ramager	1	0	0	0	1	0	5	2	1	1
suggestionner	1	0	1	0	0	4	4	1	1	1
alambiquer	1	0	0	0	1	2	2	2	1	1
encapuchonner	1	1	0	0	0	16	8	3	1	1
horripiler	1	0	0	0	1	4	2	4	1	1
berner	1	0	0	0	1	5	12	1	1	1
fanfaronner	1	0	0	1	0	0	0	0	1	1
cercler	1	0	0	0	1	21	13	1	1	1
redécouvrir	1	0	0	1	0	0	0	0	1	1
râbler	1	1	0	0	0	3	0	2	1	1
béatifier	1	0	0	0	1	3	6	2	1	1
voiturer	1	0	0	0	1	1	2	1	1	1
obérer	1	1	0	0	0	8	4	2	1	1
damasser	1	1	0	0	0	0	2	4	1	1
maquereller	1	0	0	0	1	0	0	1	1	1
gausser	1	0	0	0	1	4	2	1	1	1
défraîchir	1	1	0	0	0	2	3	2	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
plébisciter	1	0	0	1	0	0	0	0	1	1
lyncher	1	0	0	0	1	2	2	1	1	1
orchestrer	1	0	0	0	1	0	2	5	1	1
sabouler	1	0	0	0	1	0	0	3	1	1
émécher	1	1	0	0	0	1	2	2	1	1
férir	1	0	0	0	1	3	3	1	1	1
abâtardir	1	1	0	0	0	2	8	4	1	1
cravacher	1	0	0	0	1	3	4	3	1	1
indemniser	1	0	0	0	1	14	13	1	1	1
traînasser	1	0	0	0	1	3	1	2	1	1
permuter	1	0	0	1	0	1	2	6	1	1
nantir	1	1	0	0	0	3	5	6	1	1
hospitaliser	1	0	0	0	1	0	2	1	1	1
vitrer	1	1	0	0	0	23	14	1	1	1
préjuger	1	0	1	0	0	11	3	2	1	1
rauquer	1	0	0	0	1	2	0	1	1	1
masculiniser	1	0	0	0	1	0	0	4	1	1
malaxer	1	0	0	0	1	0	0	4	1	1
militariser	1	0	0	0	1	2	0	2	1	1
disproportionner	1	1	0	0	0	4	2	2	1	1
prétendre	1	0	1	0	0	1 190	306	3	1	1
fricasser	1	0	1	0	0	3	2	3	1	1
récrier	1	0	1	0	0	96	16	2	1	1
réassigner	1	0	0	1	0	1	1	1	1	1
dépriser	1	0	0	0	1	0	0	1	1	1
republier	1	0	0	0	1	0	0	1	1	1
chambrer	1	0	0	1	0	2	2	4	1	1
cinématographier	1	0	0	0	1	0	0	2	1	1
débrailler	1	1	0	0	0	4	1	3	1	1
imager	1	1	0	0	0	9	3	1	1	1
pondérer	1	0	0	0	1	3	2	5	1	1
croûter	1	0	0	1	0	0	0	0	1	1
enfourner	1	0	0	1	0	2	4	4	1	1
patouiller	1	0	0	0	1	0	0	2	1	1
désordonner	1	1	0	0	0	0	2	3	1	1
remailler	1	0	0	0	1	0	0	3	1	1
*conniver	1	0	0	1	0	0	0	0	1	1
cuver	1	0	0	0	1	5	4	1	1	1
buvoter	1	0	1	0	0	0	0	2	1	1
défeuiller	1	1	0	0	0	1	3	3	1	1
sodomiser	1	0	0	0	1	0	0	1	1	1
cohabiter	1	0	0	1	0	2	1	0	1	1
exsuder	1	0	0	0	1	0	0	2	1	1
arriérer	1	1	0	0	0	11	11	3	1	1
surfaire	1	0	0	0	1	6	6	1	1	1
domestiquer	1	1	0	0	0	3	6	4	1	1
gaffer	1	0	0	0	1	0	0	3	1	1
gouailler	1	0	0	0	1	1	1	2	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
réexaminer	1	0	0	1	0	0	0	0	1	1
nationaliser	1	1	0	0	0	2	1	1	1	1
traînailler	1	0	0	0	1	0	0	1	1	1
emperler	1	1	0	0	0	8	2	2	1	1
flirter	1	0	0	0	1	2	4	2	1	1
additionner	1	0	0	0	1	15	9	6	1	1
cendrer	1	1	0	0	0	7	4	5	1	1
bauger	1	0	0	1	0	1	3	1	1	1
transmuer	1	1	0	0	0	0	2	2	1	1
détromper	1	1	0	0	0	30	8	2	1	1
incorporer	1	1	0	0	0	39	19	6	1	1
transfuser	1	0	0	0	1	0	2	1	1	1
repopulariser	1	0	0	0	1	0	0	2	1	1
déclasser	1	1	0	0	0	4	2	5	1	1
racoler	1	0	0	0	1	4	4	2	1	1
badigeonner	1	0	0	0	1	17	8	4	1	1
déniveler	1	0	0	0	1	0	0	1	1	1
subtiliser	1	0	0	0	1	5	3	2	1	1
clocher	1	0	0	0	1	1	1	9	1	1
contrer	1	0	0	1	0	2	6	0	1	1
désintéresser	1	1	0	0	0	26	11	2	1	1
recycler	1	0	0	1	0	0	0	0	1	1
filmer	1	0	0	0	1	0	0	4	1	1
actualiser	1	0	0	0	1	3	2	2	1	1
commuter	1	1	0	0	0	0	0	4	1	1
aromatiser	1	1	0	0	0	1	5	2	1	1
sevrer	1	0	0	0	1	16	5	3	1	1
duveter	1	1	0	0	0	1	3	2	1	1
bassiner	1	0	0	0	1	2	3	4	1	1
plâtrer	1	0	0	0	1	6	3	6	1	1
*hâbler	1	0	0	1	0	0	0	0	1	1
recrépir	1	1	0	0	0	1	2	2	1	1
gribouiller	1	0	0	0	1	3	0	6	1	1
déshériter	1	0	0	0	1	15	4	4	1	1
déniaiser	1	0	0	0	1	1	3	4	1	1
rétrocéder	1	0	0	0	1	2	0	3	1	1
dégoter	1	0	0	1	0	2	1	2	1	1
liguer	1	1	0	0	0	47	21	1	1	1
étrésillonner	1	0	0	1	0	2	1	2	1	1
franciser	1	1	0	0	0	2	3	2	1	1
embrever	1	0	1	0	0	1	2	2	1	1
vilipender	1	0	0	0	1	2	2	1	1	1
entreprendre	1	0	1	0	0	544	146	2	1	1
toussoter	1	0	0	0	1	0	2	1	1	1
bécoter	1	0	0	0	1	0	0	2	1	1
rapproprier	1	0	0	1	0	0	0	0	1	1
dégingander	1	1	0	0	0	6	1	1	1	1
marivauder	1	0	0	0	1	2	0	1	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
familiariser	1	1	0	0	0	32	23	2	1	1
fesser	1	0	0	0	1	2	1	1	1	1
rognonner	1	0	0	0	1	0	0	2	1	1
givrer	1	0	0	0	1	0	2	9	1	1
casquer	1	1	0	0	0	4	8	3	1	1
intercaler	1	1	0	0	0	24	7	4	1	1
confédérer	1	1	0	0	0	4	5	2	1	1
soliloquer	1	0	0	0	1	0	0	1	1	1
galantiser	1	0	0	0	1	0	3	1	1	1
chorégraphier	1	0	0	0	1	0	0	2	1	1
vadrouiller	1	0	0	0	1	0	0	1	1	1
subroger	1	1	0	0	0	0	2	1	1	1
déséquilibrer	1	0	0	1	0	0	0	0	1	1
chamailler	1	0	1	0	0	10	4	1	1	1
intenter	1	0	0	0	1	8	4	2	1	1
conférer	1	0	0	1	0	129	44	4	1	1
jaspiner	1	0	0	0	1	0	2	2	1	1
réfracter	1	1	0	0	0	4	2	2	1	1
*braisiller	1	0	0	1	0	0	0	0	1	1
rutiler	1	0	0	0	1	3	1	1	1	1
excursionner	1	0	0	1	0	2	0	0	1	1
capitaliser	1	1	0	0	0	5	0	2	1	1
estampiller	1	1	0	0	0	1	1	1	1	1
raccompagner	1	0	0	0	1	3	5	1	1	1
essanger	1	0	0	0	1	0	0	2	1	1
trimbaler	1	0	0	0	1	1	1	3	1	1
épiler	1	1	0	0	0	9	3	2	1	1
décroiser	1	0	0	0	1	0	2	1	1	1
*loqueter	1	0	0	1	0	0	0	0	1	1
striduler	1	0	0	0	1	0	2	1	1	1
hiérarchiser	1	0	0	0	1	2	1	4	1	1
tronçonner	1	1	0	0	0	1	1	3	1	1
parasiter	1	0	0	0	1	2	3	4	1	1
cumuler	1	0	0	0	1	16	5	4	1	1
rançonner	1	1	0	0	0	30	10	1	1	1
appointer	1	0	0	0	1	3	4	3	1	1
bourgeonner	1	0	0	0	1	1	6	3	1	1
haver	1	0	0	0	1	2	1	2	1	1
patoiser	1	0	0	0	1	1	1	1	1	1
lambrisser	1	1	0	0	0	8	8	1	1	1
déchristianiser	1	0	0	0	1	0	0	2	1	1
parisianiser	1	0	0	0	1	0	0	2	1	1
acquitter	1	1	0	0	0	243	98	3	1	1
tabler	1	0	1	0	0	4	5	2	1	1
trier	1	1	0	0	0	14	7	3	1	1
outiller	1	1	0	0	0	5	3	5	1	1
intoxiquer	1	0	0	0	1	1	1	5	1	1
débobiner	1	0	0	0	1	0	0	1	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
démantibuler	1	0	0	0	1	2	2	4	1	1
amnistier	1	0	0	0	1	7	8	2	1	1
quintupler	1	0	0	0	1	0	2	2	1	1
fourcher	1	0	0	0	1	3	0	4	1	1
sensibiliser	1	1	0	0	0	4	0	4	1	1
matir	1	1	0	0	0	0	0	2	1	1
ventiler	1	0	0	0	1	0	2	4	1	1
baraquer	1	1	0	0	0	2	2	3	1	1
*chouanner	1	0	0	1	0	0	0	0	1	1
aiguilleter	1	1	0	0	0	5	7	4	1	1
enverguer	1	1	0	0	0	10	1	2	1	1
répertorier	1	0	0	0	1	4	1	2	1	1
pontifier	1	0	0	0	1	2	3	1	1	1
prôner	1	1	0	0	0	26	8	2	1	1
soutacher	1	1	0	0	0	4	2	1	1	1
ramifier	1	1	0	0	0	18	5	3	1	1
huiler	1	1	0	0	0	3	3	3	1	1
godailler	1	0	0	1	0	0	0	0	1	1
remanger	1	0	0	0	1	0	0	3	1	1
morguer	1	1	0	0	0	0	0	1	1	1
embéguiner	1	1	0	0	0	2	3	2	1	1
engueuler	1	0	0	0	1	3	1	2	1	1
insupporter	1	0	0	0	1	0	0	1	1	1
particulariser	1	0	0	0	1	1	4	5	1	1
turlupiner	1	0	0	0	1	2	1	2	1	1
aviner	1	1	0	0	0	4	2	4	1	1
vrombir	1	0	0	0	1	0	0	1	1	1
brasiller	1	0	0	0	1	1	3	1	1	1
pendiller	1	0	0	0	1	0	5	1	1	1
étrécir	1	0	0	0	1	1	1	3	1	1
dégrafer	1	0	0	0	1	12	16	4	1	1
engainer	1	1	0	0	0	2	0	3	1	1
crachoter	1	0	0	0	1	0	0	3	1	1
crosser	1	0	0	0	1	1	1	3	1	1
escrimer	1	0	1	0	0	11	5	1	1	1
spolier	1	0	0	0	1	8	2	1	1	1
redessiner	1	0	0	1	0	0	0	0	1	1
dépiauter	1	1	0	0	0	0	0	7	1	1
enclore	1	1	0	0	0	1	1	3	1	1
enténébrer	1	1	0	0	0	6	1	2	1	1
doper	1	0	0	1	0	0	0	0	1	1
empanacher	1	1	0	0	0	12	7	2	1	1
rembourrer	1	1	0	0	0	15	9	1	1	1
emparadiser	1	0	0	0	1	1	2	1	1	1
ourler	1	0	0	0	1	9	1	1	1	1
retravailler	1	0	0	0	1	0	2	2	1	1
déblatérer	1	0	1	0	0	10	3	2	1	1
abonner	1	0	0	0	1	12	4	3	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
réabonner	1	0	0	0	1	0	0	2	1	1
ankyloser	1	1	0	0	0	3	0	2	1	1
marronner	1	0	0	0	1	1	1	1	1	1
empiéter	1	0	0	0	1	19	5	3	1	1
écluser	1	0	0	0	1	0	0	4	1	1
polissonner	1	0	0	0	1	1	1	1	1	1
remarcher	1	0	0	1	0	0	0	0	1	1
emboutir	1	1	0	0	0	0	0	3	1	1
commémorer	1	0	0	0	1	1	1	1	1	1
styliser	1	1	0	0	0	1	2	2	1	1
encoder	1	0	0	1	0	0	0	0	1	1
brouetter	1	0	0	0	1	1	1	1	1	1
islamiser	1	0	0	0	1	0	3	2	1	1
grillager	1	1	0	0	0	7	4	1	1	1
cailler	1	0	0	0	1	1	2	8	1	1
brouillasser	1	0	0	0	1	0	0	1	1	1
cuisiner	1	0	0	0	1	4	3	6	1	1
banaliser	1	0	0	0	1	2	0	2	1	1
patenter	1	0	0	0	1	3	5	4	1	1
décaler	1	0	0	0	1	2	0	6	1	1
appâter	1	0	0	0	1	0	0	4	1	1
draguer	1	0	0	0	1	2	0	3	1	1
débouquer	1	0	0	0	1	1	2	1	1	1
intimer	1	0	0	0	1	15	2	2	1	1
reblanchir	1	1	0	0	0	1	1	5	1	1
décoller	1	0	0	0	1	16	11	9	1	1
dégraisser	1	1	0	0	0	1	2	6	1	1
répercuter	1	1	0	0	0	50	18	5	1	1
clampser	1	0	0	0	1	0	0	1	1	1
désengourdir	1	0	0	0	1	0	0	2	1	1
cataloguer	1	1	0	0	0	8	8	4	1	1
embouteiller	1	1	0	0	0	0	0	5	1	1
ameublir	1	1	0	0	0	1	1	2	1	1
déshabituer	1	0	0	1	0	5	4	2	1	1
inféoder	1	0	0	0	1	10	3	3	1	1
astreindre	1	0	1	0	0	18	10	2	1	1
épanneler	1	1	0	0	0	5	0	2	1	1
ébraser	1	0	1	0	0	0	0	2	1	1
innerver	1	0	0	0	1	0	0	1	1	1
couder	1	0	0	0	1	2	0	2	1	1
élonger	1	0	0	0	1	2	4	2	1	1
claironner	1	0	0	0	1	1	1	3	1	1
musarder	1	0	0	0	1	0	0	1	1	1
culotter	1	1	0	0	0	3	3	7	1	1
immerger	1	1	0	0	0	14	4	4	1	1
dédoubler	1	1	0	0	0	16	5	6	1	1
ballonner	1	0	0	0	1	4	4	3	1	1
élimer	1	0	0	0	1	3	2	2	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
recompter	1	0	0	0	1	2	2	2	1	1
farcir	1	0	0	0	1	3	4	8	1	1
déjeûner	1	0	0	1	0	2	0	0	1	1
*repeigner	1	0	0	1	0	0	0	0	1	1
retendre	1	0	0	0	1	4	1	4	1	1
efféminer	1	0	0	0	1	10	9	2	1	1
agneler	1	0	0	0	1	0	0	2	1	1
émoudre	1	1	0	0	0	0	0	4	1	1
époumoner	1	0	0	0	1	2	2	2	1	1
raboter	1	0	0	0	1	3	6	3	1	1
sous-entendre	1	0	0	0	1	2	0	2	1	1
empuantir	1	1	0	0	0	3	2	2	1	1
désaccoutumer	1	0	0	0	1	1	1	2	1	1
vieller	1	1	0	0	0	0	0	2	1	1
fourber	1	0	0	0	1	0	0	1	1	1
promettre	1	0	0	0	1	0	0	10	1	1
amouracher	1	0	0	0	1	5	6	1	1	1
rabougrir	1	0	0	0	1	1	4	5	1	1
tergiverser	1	0	0	0	1	0	4	1	1	1
glousser	1	0	0	0	1	3	8	2	1	1
pourprer	1	1	0	0	0	3	5	2	1	1
relater	1	0	1	0	0	37	20	1	1	1
claustrer	1	1	0	0	0	0	0	1	1	1
sublimer	1	1	0	0	0	1	1	2	1	1
entrelacer	1	1	0	0	0	59	33	2	1	1
centrer	1	0	1	0	0	1	2	3	1	1
criailler	1	0	0	1	0	0	3	3	1	1
*digresser	1	0	0	1	0	0	0	0	1	1
échauder	1	0	0	0	1	6	3	5	1	1
harnacher	1	1	0	0	0	22	11	5	1	1
déjuger	1	0	0	0	1	0	0	2	1	1
recauser	1	0	0	0	1	3	0	6	1	1
tiercer	1	0	0	0	1	0	2	2	1	1
recoiffer	1	0	0	0	1	2	6	2	1	1
maquignonner	1	0	0	0	1	0	3	3	1	1
canoter	1	0	0	0	1	0	0	2	1	1
défleurir	1	0	0	0	1	3	2	3	1	1
uriner	1	0	0	0	1	0	2	2	1	1
désister	1	0	0	0	1	11	4	1	1	1
retracer	1	0	0	0	1	110	50	2	1	1
rallonger	1	0	0	0	1	1	2	5	1	1
engraver	1	0	0	0	1	7	4	4	1	1
revisiter	1	0	0	0	1	2	0	1	1	1
odorer	1	0	1	0	0	0	3	3	1	1
conditionner	1	0	0	0	1	6	9	6	1	1
momifier	1	1	0	0	0	3	4	4	1	1
attitrer	1	1	0	0	0	6	2	1	1	1
empailler	1	1	0	0	0	7	5	4	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
ardoiser	1	1	0	0	0	2	4	2	1	1
terrir	1	0	0	0	1	0	0	1	1	1
endetter	1	1	0	0	0	22	6	2	1	1
cléricaliser	1	0	0	0	1	0	0	2	1	1
écornifler	1	0	0	0	1	0	0	1	1	1
transplanter	1	0	0	0	1	26	16	3	1	1
calligraphier	1	1	0	0	0	0	3	2	1	1
circonstancier	1	1	0	0	0	4	5	2	1	1
impétrer	1	0	0	0	1	0	0	1	1	1
crépir	1	1	0	0	0	3	1	2	1	1
replâtrer	1	0	0	1	0	1	1	0	1	1
minotauriser	1	1	0	0	0	0	0	1	1	1
transborder	1	1	0	0	0	6	4	1	1	1
mâchurer	1	0	0	0	1	0	0	6	1	1
musser	1	0	0	0	1	0	0	4	1	1
dédorer	1	1	0	0	0	1	3	2	1	1
crailler	1	0	0	0	1	0	0	1	1	1
retailler	1	1	0	0	0	0	0	2	1	1
démarquer	1	0	0	0	1	2	1	7	1	1
aduler	1	0	0	0	1	4	8	1	1	1
licencier	1	1	0	0	0	26	14	1	1	1
avachir	1	0	0	0	1	1	4	5	1	1
carder	1	0	0	0	1	0	2	1	1	1
crâner	1	0	0	1	0	0	0	0	1	1
déhancher	1	0	0	0	1	2	4	1	1	1
envoiler	1	0	0	0	1	0	0	1	1	1
dévernir	1	1	0	0	0	1	1	2	1	1
ébahir	1	0	0	0	1	10	6	2	1	1
départager	1	0	0	0	1	1	1	4	1	1
allaiter	1	0	1	0	0	41	10	2	1	1
méprendre	1	0	1	0	0	67	21	1	1	1
extrader	1	0	0	0	1	0	0	1	1	1
stéréotyper	1	1	0	0	0	2	4	2	1	1
surnommer	1	1	0	0	0	92	26	2	1	1
ingérer	1	0	0	0	1	4	5	2	1	1
poncer	1	0	0	0	1	0	0	2	1	1
pailleter	1	1	0	0	0	4	6	3	1	1
grasseyer	1	0	0	1	0	0	0	0	1	1
silhouetter	1	0	0	0	1	2	0	2	1	1
gréer	1	1	0	0	0	8	7	1	1	1
débâcler	1	0	0	0	1	0	0	4	1	1
exciper	1	0	0	0	1	3	1	1	1	1
désarticuler	1	0	0	0	1	3	3	1	1	1
pasticher	1	0	0	0	1	1	1	2	1	1
magnifier	1	0	0	1	0	2	2	2	1	1
vocaliser	1	0	0	0	1	2	2	3	1	1
juter	1	0	0	0	1	0	0	5	1	1
rapatrier	1	0	0	1	0	3	4	2	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
larmoyer	1	0	0	0	1	6	3	3	1	1
autographier	1	1	0	0	0	1	1	2	1	1
suifer	1	0	0	1	0	0	0	0	1	1
goudronner	1	1	0	0	0	16	11	2	1	1
débaptiser	1	1	0	0	0	2	2	2	1	1
portraiturer	1	1	0	0	0	1	2	2	1	1
boycotter	1	0	0	0	1	0	0	1	1	1
éroder	1	1	0	0	0	0	0	3	1	1
extravaser	1	1	0	0	0	0	5	2	1	1
tabouer	1	1	0	0	0	1	1	2	1	1
annoter	1	1	0	0	0	8	4	2	1	1
trisser	1	0	0	0	1	0	0	6	1	1
helléniser	1	1	0	0	0	2	1	2	1	1
glouglouter	1	0	0	0	1	0	0	2	1	1
barioler	1	0	0	0	1	29	16	2	1	1
fédéraliser	1	1	0	0	0	0	0	2	1	1
déraper	1	0	0	0	1	0	0	6	1	1
parjurer	1	0	0	0	1	2	1	1	1	1
cloîtrer	1	1	0	0	0	18	9	3	1	1
authentifier	1	1	0	0	0	0	0	1	1	1
fidéliser	1	0	0	1	0	0	0	0	1	1
déléguer	1	0	0	0	1	16	4	3	1	1
concasser	1	0	0	0	1	1	3	2	1	1
scarifier	1	0	0	0	1	1	1	2	1	1
policer	1	1	0	0	0	2	2	2	1	1
affriander	1	0	0	0	1	2	3	2	1	1
marner	1	0	0	0	1	2	0	4	1	1
recroître	1	1	0	0	0	0	3	2	1	1
recontrer	1	0	0	0	1	0	0	1	1	1
pleuviner	1	0	0	0	1	0	0	1	1	1
relaver	1	0	0	0	1	0	0	4	1	1
restreindre	1	0	0	1	0	29	8	0	1	1
tarauder	1	1	0	0	0	2	2	5	1	1
dauber	1	0	0	0	1	1	1	4	1	1
refréner	1	0	0	1	0	6	4	2	1	1
satiner	1	1	0	0	0	2	2	4	1	1
apparenter	1	0	1	0	0	19	6	5	1	1
méconduire	1	0	0	0	1	0	0	1	1	1
cartonner	1	1	0	0	0	3	4	7	1	1
*réfectionner	1	0	0	1	0	0	0	0	1	1
enrégimenter	1	1	0	0	0	17	4	4	1	1
colloquer	1	0	0	0	1	1	2	5	1	1
natter	1	1	0	0	0	6	7	4	1	1
palmer	1	1	0	0	0	3	4	3	1	1
remployer	1	0	0	0	1	0	0	3	1	1
canneler	1	1	0	0	0	4	2	2	1	1
délester	1	0	0	0	1	4	3	3	1	1
exonérer	1	0	0	0	1	2	0	1	1	1

'Données sans perte intégrale' page suivante

TABLE E. 2 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
espagnoliser	1	0	0	0	1	0	0	2	1	1
sonnailler	1	0	0	0	1	0	0	1	1	1
dissuader	1	0	0	0	1	12	3	1	1	1
concréter	1	0	0	0	1	2	0	2	1	1
parcheminer	1	0	0	0	1	6	2	4	1	1
contrevenir	1	0	1	0	0	6	2	1	1	1
herser	1	0	0	0	1	2	3	1	1	1
aguicher	1	0	0	0	1	1	1	2	1	1
tiller	1	0	0	1	0	0	0	0	1	1
harper	1	0	0	0	1	1	2	3	1	1
affréter	1	0	0	1	0	3	0	0	1	1
gommer	1	0	0	0	1	5	3	7	1	1
cuirasser	1	1	0	0	0	17	8	5	1	1
blinder	1	1	0	0	0	2	2	6	1	1
désobstruer	1	0	0	0	1	0	0	2	1	1
tonifier	1	0	0	0	1	3	1	4	1	1
exproprier	1	0	0	0	1	3	6	3	1	1
corder	1	0	0	0	1	0	2	9	1	1
instaurer	1	0	0	0	1	1	1	3	1	1
empoussiérer	1	1	0	0	0	0	0	2	1	1
lester	1	0	0	0	1	29	18	4	1	1
émotionner	1	0	0	0	1	9	3	2	1	1
embobeliner	1	0	0	0	1	1	1	1	1	1
recirculer	1	0	0	0	1	0	0	1	1	1
réapprendre	1	0	0	1	0	1	1	3	1	1
*boîter	1	0	0	1	0	0	0	0	1	1
rechanter	1	0	0	0	1	0	0	2	1	1
suriner	1	0	0	0	1	0	0	1	1	1
universaliser	1	0	0	0	1	3	0	2	1	1
deviser	1	0	1	0	0	56	14	2	1	1
ébaudir	1	0	0	0	1	2	0	1	1	1
jargonner	1	0	0	0	1	1	1	3	1	1

E. 3.2 171 verbes avec perte complète

TABLE E. 3 – avec perte complète (171 verbes)

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
illuminer	261	2	3	12	244	106	155	8	6	6
surgir	187	1	3	33	150	68	119	4	5	5
préciser	83	0	3	29	51	33	50	6	3	3
geler	80	7	0	1	72	26	54	13	9	9
exceller	61	1	0	2	58	21	40	2	1	1
réfuter	40	1	0	19	20	19	21	3	3	3
fourmiller	30	0	0	1	29	5	25	6	3	3
dérider	29	0	0	7	22	8	21	4	2	2
absenter	27	0	0	19	8	11	16	2	1	1

'avec perte complète' page suivante

TABLE E. 3 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
ensoleiller	22	14	0	0	8	6	16	3	3	3
piaffer	20	0	0	1	19	6	14	2	1	1
bruisser	19	0	0	0	19	8	11	2	1	1
virer	18	0	0	9	9	2	16	9	3	3
gratifier	16	0	0	1	15	7	9	4	1	1
déprécier	16	0	1	10	5	6	10	4	2	2
grisonner	16	0	0	3	13	5	11	1	1	1
tricher	14	0	0	4	10	5	9	5	1	1
délimiter	14	2	0	1	11	5	9	3	4	4
populariser	14	0	1	5	8	6	8	4	2	2
ondoyer	14	0	2	0	12	6	8	2	2	2
clignoter	14	0	0	1	13	4	10	3	1	1
coopérer	13	0	0	7	6	6	7	4	2	2
ébouler	13	4	0	1	8	6	7	2	1	1
dégoutter	12	0	0	1	11	5	7	2	2	2
égoutter	12	0	0	0	12	3	9	3	1	1
encercler	12	2	1	0	9	5	7	3	2	2
blaguer	11	0	0	3	8	3	8	4	2	2
démanteler	10	3	0	0	7	3	7	3	2	2
reverdir	8	1	0	0	7	3	5	4	2	2
dégourdir	8	1	0	0	7	3	5	6	2	2
farder	8	0	0	1	7	5	3	4	2	2
jacasser	8	0	1	0	7	4	4	3	1	1
crocheter	7	0	0	6	1	2	5	6	1	1
venter	7	0	0	0	7	1	6	2	1	1
obtempérer	7	0	0	3	4	2	5	2	1	1
retraverser	7	0	0	1	6	2	5	1	1	1
abominer	6	0	0	0	6	2	4	1	1	1
forer	6	1	0	1	4	2	4	2	3	3
pressurer	6	3	0	0	3	0	6	4	3	3
fustiger	6	0	0	1	5	2	4	2	1	1
repérer	5	0	1	1	3	4	1	5	3	3
délacer	5	0	0	2	3	1	4	2	1	1
essaimer	5	0	0	0	5	2	3	4	1	1
assagir	5	1	0	0	4	2	3	2	3	3
tiédir	4	2	0	0	2	1	3	2	1	1
découcher	4	0	0	0	4	2	2	1	1	1
ferrailler	4	0	0	4	0	1	3	4	1	1
bruire	4	0	0	4	0	4	0	2	1	1
rééditer	4	1	0	2	1	2	2	3	2	2
cabaler	3	0	0	0	3	0	3	2	1	1
laper	3	0	0	0	3	0	3	2	2	2
flagorner	3	1	0	0	2	0	3	1	1	1
revoler	3	0	0	0	3	2	1	3	1	1
christianiser	3	2	0	0	1	0	3	2	1	1
sécuriser	3	3	0	0	0	2	1	4	2	2
chevroter	3	0	0	0	3	1	2	4	2	2
dégouliner	3	0	0	0	3	2	1	2	1	1

'avec perte complète' page suivante

TABLE E. 3 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
biseauter	3	2	0	0	1	2	1	3	3	3
refrapper	3	0	0	1	2	1	2	5	1	1
ânonner	3	0	0	0	3	2	1	4	2	2
latiniser	3	1	0	0	2	1	2	4	2	2
horrifier	3	3	0	0	0	2	1	1	1	1
suppurer	3	0	0	0	3	2	1	1	1	1
emménager	3	0	0	0	3	2	1	2	1	1
résilier	3	0	0	0	3	0	3	2	2	2
culminer	3	0	1	0	2	0	3	2	1	1
dérouiller	3	0	0	1	2	2	1	9	1	1
synthétiser	3	1	0	0	2	1	2	3	1	1
pourfendre	2	0	0	1	1	0	2	2	1	1
mâtiner	2	1	0	0	1	2	0	5	2	2
courbaturer	2	0	0	0	2	1	1	1	1	1
lover	2	0	0	0	2	1	1	4	2	2
accourcir	2	0	0	0	2	0	2	4	2	2
indifférer	2	0	0	0	2	1	1	1	1	1
éjaculer	2	0	1	0	1	0	2	2	1	1
véhiculer	2	0	0	0	2	1	1	2	1	1
doser	2	0	0	1	1	1	1	2	1	1
stranguler	2	2	0	0	0	2	0	1	1	1
caricaturer	2	0	0	0	2	2	0	4	1	1
recracher	2	0	0	0	2	0	2	4	2	2
replanter	2	0	0	1	1	1	1	2	1	1
marauder	2	0	0	1	1	0	2	2	2	2
automatiser	2	1	0	0	1	1	1	2	1	1
dactylographier	2	2	0	0	0	2	0	2	2	2
trépider	2	0	0	1	1	0	2	1	1	1
éructer	2	0	0	1	1	0	2	2	1	1
mariner	2	2	0	0	0	2	0	3	2	2
extravaguer	2	0	0	2	0	2	0	1	1	1
féminiser	2	0	0	0	2	1	1	6	1	1
pépier	2	0	0	0	2	0	2	1	1	1
zézayer	2	0	1	0	1	0	2	2	1	1
rosir	2	0	0	0	2	1	1	3	2	2
filouter	2	0	0	0	2	1	1	2	1	1
damer	2	0	0	2	0	2	0	0	1	1
encocher	2	0	0	0	2	0	2	2	2	2
oranger	2	2	0	0	0	2	0	2	2	2
titiller	2	0	0	0	2	2	0	3	1	1
coltiner	1	0	0	0	1	0	1	2	1	1
bicher	1	0	0	0	1	1	0	3	1	1
noliser	1	1	0	0	0	0	1	2	1	1
vriller	1	0	0	1	0	0	1	0	1	1
fluctuer	1	0	0	0	1	1	0	1	1	1
défriper	1	0	0	0	1	1	0	1	1	1
tapager	1	0	0	0	1	0	1	1	1	1
retordre	1	0	0	1	0	1	0	0	1	1

'avec perte complète' page suivante

TABLE E. 3 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
écrabouiller	1	0	0	0	1	1	0	4	1	1
recreuser	1	0	0	0	1	1	0	4	1	1
luncher	1	0	0	1	0	0	1	2	1	1
déraidir	1	0	0	1	0	0	1	2	1	1
transiter	1	0	1	0	0	0	1	2	1	1
parfiler	1	0	0	0	1	0	1	1	1	1
chiquer	1	0	0	1	0	0	1	0	1	1
détoner	1	0	0	0	1	0	1	1	1	1
engrosser	1	0	0	1	0	0	1	0	1	1
calotter	1	0	0	0	1	0	1	3	1	1
bluter	1	1	0	0	0	0	1	2	1	1
ramoner	1	0	0	1	0	0	1	5	1	1
volcaniser	1	1	0	0	0	0	1	2	1	1
flancher	1	0	0	0	1	0	1	6	1	1
rappliquer	1	0	0	1	0	1	0	2	1	1
arraisonner	1	0	0	1	0	1	0	0	1	1
ambuler	1	0	1	0	0	0	1	1	1	1
gondoler	1	0	0	0	1	1	0	4	1	1
amurer	1	0	0	0	1	0	1	2	1	1
ressauter	1	0	0	0	1	0	1	4	1	1
batifoler	1	0	0	0	1	1	0	2	1	1
panteler	1	0	1	0	0	0	1	2	1	1
apparier	1	1	0	0	0	0	1	4	1	1
volter	1	0	0	1	0	0	1	1	1	1
boulanger	1	1	0	0	0	0	1	3	1	1
impartir	1	0	0	0	1	0	1	2	1	1
bichonner	1	1	0	0	0	1	0	3	1	1
tintinnabuler	1	0	0	0	1	0	1	1	1	1
décomprimer	1	0	0	0	1	0	1	1	1	1
américaniser	1	0	0	0	1	0	1	2	1	1
pommer	1	1	0	0	0	1	0	1	1	1
désaffectionner	1	0	0	0	1	1	0	2	1	1
délaver	1	1	0	0	0	0	1	3	1	1
ébouillanter	1	0	0	0	1	1	0	3	1	1
détortiller	1	0	0	0	1	0	1	2	1	1
réentendre	1	0	0	0	1	0	1	4	1	1
inverser	1	0	0	0	1	0	1	4	1	1
estomaquer	1	0	0	0	1	0	1	1	1	1
mitonner	1	0	1	0	0	0	1	5	1	1
déplanter	1	1	0	0	0	0	1	4	1	1
cousiner	1	0	0	0	1	0	1	2	1	1
vallonner	1	0	0	0	1	1	0	2	1	1
clinquer	1	0	1	0	0	0	1	1	1	1
brocanter	1	0	0	0	1	0	1	1	1	1
regratter	1	0	0	1	0	0	1	0	1	1
carotter	1	0	0	0	1	1	0	3	1	1
industrialiser	1	1	0	0	0	1	0	4	1	1
cordonner	1	1	0	0	0	1	0	2	1	1

'avec perte complète' page suivante

TABLE E. 3 – *Suite de la page précédente*

Verbe	compte	vpp	vpr	inf.	autres	pertes		frames		
						direct	collatéral	init.	final	disp.
aristocratiser	1	0	0	0	1	0	1	2	1	1
pyramider	1	0	1	0	0	1	0	1	1	1
tiquer	1	1	0	0	0	1	0	2	1	1
visualiser	1	0	0	0	1	1	0	1	1	1
contacter	1	0	0	0	1	1	0	2	1	1
luter	1	0	0	0	1	0	1	2	1	1
esthétiser	1	0	0	0	1	1	0	1	1	1
cardinaliser	1	0	0	0	1	0	1	2	1	1
affrioler	1	0	0	0	1	1	0	1	1	1
incinérer	1	0	0	0	1	1	0	2	1	1
claver	1	1	0	0	0	0	1	2	1	1
banqueter	1	0	1	0	0	1	0	1	1	1
vaticiner	1	0	0	0	1	0	1	1	1	1
éberluer	1	0	0	0	1	1	0	1	1	1
calibrer	1	0	0	0	1	0	1	3	1	1
recacheter	1	1	0	0	0	1	0	2	1	1
bisquer	1	0	0	0	1	1	0	1	1	1
cautériser	1	0	0	0	1	1	0	5	1	1

E. 4 Frames

E. 4.1 Information globale

Informations générales : http://rali.iro.umontreal.ca/LVF+1/documents/PresentationLVF_JF_DLP_DL.pdf

Le LVF codifie les types traditionnels (transitif direct ou non, intransitif, pronominal), la nature du sujet (ex : humain, chose ...), objet et compléments, et pour le complément prépositionnel s'il y a, le rôle du complément (ex : modalité , instrumental ...).

E. 4.2 357 frames des verbes conjugués.

TABLE E. 4 – Frames des verbes conjugués (357 frames)

Frame	verbes	conj.	vpp	vpa	inf.
A90	être, avoir, rentrer, engraisser, transpirer, préexister	7879	0	0	0
AZ2	être, tomber, approcher, avancer, fuir, piquer	142112	11	93	3018
Z9a	être	13	0	0	0
A91	avoir, arrêter	164	0	0	0
A9Z	avoir	3346	0	0	0
T1106	avoir, reconnaître, payer, nommer, choisir, obéir, ...	2069	268	55	388
T1300	avoir, donner, passer, mettre, savoir, rendre, ...	58333	97	138	1304
T1Z06	avoir, trouver, aimer, suivre, commencer, chercher, ...	25652	183	509	4468
A41	faire, tomber, retomber, souffler, flotter, pleuvoir, ...	281	3	1	25
AZ0	faire, prendre, passer, rester, parler, rendre, ...	25236	14	30	904
AZ6	faire, rester, rendre, paraître, attendre, pousser, ...	20762	176	88	1836
P1Z00	faire, envoyer, payer, imaginer, figurer, fabriquer	755	1	5	129
P3000	faire, voir, prendre, trouver, dire, mettre, ...	17611	950	45	1041
P3006	faire, commencer, compter, gagner, conserver, saisir, ...	6979	623	37	234
T1500	faire, aller, laisser, manquer, compter, songer, ...	2107	0	7	5
TZ106	faire, former, posséder, constituer, bâtir, inquiéter, ...	11087	62	297	3679
TZ300	faire, trouver, dire, suivre, montrer, appeler, ...	44435	490	514	5097
A40	pouvoir, rester, falloir, paraître, demander, sembler, ...	5472	16	4	37
P4000	pouvoir, trouver, falloir, agir, imposer	978	3	0	0
TZ500	pouvoir, vouloir, devoir, daigner	1237	5	8	16
Z4000	pouvoir	14291	0	25	253
A16	voir, parler, entendre, perdre, jeter, présenter, ...	9029	157	137	525
P1400	voir, prendre, croire, rappeler, envoyer, remettre, ...	989	34	29	104
PZ001	voir, trouver, arrêter, montrer, pousser, cacher, ...	8151	259	40	165
T1Z00	voir, trouver, croire, entendre, laisser, attendre, ...	28199	310	971	6697
T1Z0Z	voir, tenir, pousser, continuer, cacher, oublier, ...	8635	138	757	6041
A36	prendre, donner, rendre, reprendre, gagner, suffire, ...	4717	52	10	67
P1308	prendre, tenir, brûler, détruire, couper, troubler, ...	525	0	0	0
PZ006	prendre, donner, mettre, tenir, perdre, frapper, ...	9441	314	44	12
PZ308	prendre, glacer, claquer, pincer, froisser, abîmer	412	0	0	0
TZZ00	prendre, savoir, connaître, regarder, tenir, perdre, ...	20109	180	546	3201
TZZ06	prendre, tirer, produire, posséder, épouser, découvrir, ...	6413	84	85	1287
P1000	donner, croire, connaître, aimer, jeter, mourir, ...	3721	316	17	470
P1300	donner, passer, reconnaître, préparer, attirer, créer, ...	1119	3	6	160
PZ000	donner, passer, connaître, entendre, suivre, ouvrir, ...	13904	18	1	29
TZ301	donner, mouiller, implanter	5070	21	83	936
A30	aller, porter, tenir, commencer, arrêter, servir, ...	9895	45	15	219

'Frames des verbes conjugués' page suivante

TABLE E. 4 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
AZ1	aller, arrêter, sortir, manquer, vivre, finir, ...	17015	200	44	244
PZ005	aller, former, rencontrer, taire, enfuir, reproduire	1438	6	2	67
PZ00Z	trouver, passer, ouvrir, lever, tirer, former, ...	8512	22	6	10
Z1a	trouver, souscrire	61	0	0	0
AZ5	passer, arriver, persister, survenir, sommeiller, couver	5593	97	74	593
P300Z	passer, ouvrir, continuer, jouer, travailler, dresser, ...	1459	150	26	0
T1Z40	passer, exiler, transférer, trimballer, transplanter	1196	17	64	439
P1500	dire, douter	214	0	0	1
P3008	dire, expliquer, éclairer, rapporter, redoubler, guérir, ...	1676	257	28	76
T1306	dire, parler, aimer, sentir, écrire, changer, ...	7961	198	241	2175
T3300	dire, présenter, produire, annoncer, conserver, accompagner, ...	14882	81	95	559
P1007	vouloir, extasier, pâmer, ébaudir	56	41	5	16
T1101	vouloir, connaître, revoir, accueillir, joindre, enfermer, ...	7997	29	40	122
TZZ0Z	mettre, toucher, gagner, charger, retenir, coucher, ...	4132	50	165	1036
A10	savoir, croire, jeter, appeler, manquer, penser, ...	12853	125	54	811
AZ4	venir, descendre, voler, décrocher, déferler, dévaler	7441	0	0	35
P1020	venir, ramener, risquer, réfugier, égarer, acheminer	398	38	4	83
T3500	venir, falloir	3467	12	4	15
AZZ	parler, perdre, tomber, ouvrir, partir, tirer, ...	17794	38	87	776
P7000	parler, tenir, causer, ressembler, convenir, provoquer, ...	890	9	6	103
PZ020	porter, rendre, présenter, approcher, avancer, amener, ...	2791	30	6	0
PZ0Z0	porter, retirer, relever, précipiter, écarter, dégager	1538	20	10	0
T1ZZZ	porter, traîner, rejeter	206	2	41	68
TZ100	porter, jeter, quitter, occuper, finir, conduire, ...	18144	433	541	4139
TZ302	porter	1666	4	194	597
TZ306	rendre, laisser, suivre, prononcer, soutenir, inspirer, ...	4622	159	327	1599
TZ30Z	rendre, rouler, couper, percer, réaliser, contracter, ...	2152	62	108	976
T1900	croire, remettre, souffrir, préférer, plaindre, vaincre, ...	3046	25	22	177
P7001	connaître, réunir, revoir, croiser, rallier, côtoyer, ...	222	98	3	9
Z4a	paraître, sembler, apparaître	55	0	0	0
P1001	entendre, recevoir, pousser, fixer, contempler, enfermer, ...	824	172	11	103
T1Z08	entendre, recevoir, perdre, arrêter, frapper, rappeler, ...	9028	396	345	3353
P3001	laisser, garder, accuser, révéler, exposer, déposer, ...	1671	103	15	181
P1406	demander, expliquer	313	0	0	0
T110Z	demander, tomber, rejoindre, surprendre, amuser, utiliser, ...	3626	38	30	138
A96	devenir, sembler, sentir, repousser, redevenir, empoisonner, ...	6886	321	20	436
T1400	regarder, aimer, arrêter, comprendre, sentir, oublier, ...	3829	33	53	326
T1901	regarder, apercevoir, distinguer, remarquer, deviner, parcourir	1681	8	27	2
A3Z	tenir, manquer, couler, retomber, pendre, subsister, ...	1615	5	3	27
TZ900	tenir, toucher, manger, lancer, arracher, animer, ...	2859	58	133	699
T1100	aimer, rappeler, éclater, sauver, craindre, prier, ...	6782	38	26	208
T3Z00	recevoir, avancer, brûler, agiter, marquer, reposer, ...	4502	79	28	157
TZZ08	recevoir, perdre, frapper, battre, courir, fermer, ...	5620	276	279	3172
A31	commencer, placer, couvrir, abonder, poindre, tourbillonner, ...	1448	4	2	54
T130Z	ouvrir, présenter, conserver, retirer, terminer, presser, ...	3727	220	190	1959
P10Z0	jeter, exiler, expatrier	648	18	13	0
T1138	jeter, extraire, proscrire, extirper, bouter, extrader	48	4	8	29
P9001	arrêter	72	0	0	0
T3400	montrer, expliquer, souffrir, indiquer, entraîner, démontrer	949	3	2	26

'Frames des verbes conjugués' page suivante

TABLE E. 4 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
TZ140	appeler, reconduire	60	0	0	0
A70	sortir, traiter, coucher, combattre, lutter, différer, ...	1021	3	2	218
P1030	sortir, revenir, bouger, défiler, trotter, évader	74	11	5	41
T1ZZ0	sortir, approcher, obtenir, embarquer, redescendre	231	4	29	235
T1506	manquer, essayer, tenter, tâcher	185	0	0	1
A11	servir, frapper, toucher, sonner, exercer, étudier, ...	1617	36	27	97
AZ7	mourir, souffrir, rougir, bouger, frémir, déborder, ...	3636	0	12	496
T1Z07	reprendre, célébrer, consacrer, disputer, applaudir, discuter, ...	1119	60	54	636
T1Z01	garder, établir, représenter, refuser, deviner, recueillir, ...	2182	128	98	1187
TZ906	quitter, gagner, saisir, habiter, envahir, dévorer, ...	1569	1	6	44
P100Z	reconnaître, élancer, classer, alanguir	594	18	11	0
TZ308	occuper, contenir, réveiller, supporter, arranger, déployer, ...	1387	99	104	1069
AZ3	lever, débarquer, dérober, jaillir, redescendre, émerger	192	0	1	2
T1Z38	lever, déraciner, débusquer	5	9	1	6
TZZ38	lever	370	8	49	0
A35	continuer, durer, persister, sommeiller, discontinuer, échoir	1296	2	2	105
A1Z	jouer, retourner, rentrer, pleurer, chasser, opérer, ...	1196	57	27	76
TZ800	rencontrer, rejoindre, disperser, dépouiller	1652	1	17	466
TZZ07	descendre, allumer, déchirer, incendier, charrier, flageller	494	49	42	153
P1006	conduire, oublier, tourner, abandonner, défendre, interrompre, ...	1729	186	34	261
TZ320	conduire, ramener	1048	0	0	0
T3100	demeurer, inspirer, étonner, disposer, réveiller, accueillir, ...	4499	223	7	287
T1120	envoyer, déporter	1384	8	0	0
TZ1b6	remplir	7	0	0	0
T1308	tourner, causer, cesser, couvrir, essayer, attacher, ...	5343	328	210	1819
T1340	apporter, exporter	1244	10	25	95
Z30a0	apporter, assigner, administrer, décerner, dédier, léguer, ...	155	0	0	0
ZZ001	produire, multiplier, aborder, entasser, accumuler, amasser, ...	67	233	11	0
P1008	tuer, enivrer, mûrir, immoler, suicider, soûler	63	23	1	11
PZ008	tuer, renouveler, pendre, calmer, supprimer, ruiner, ...	786	72	6	0
TZ305	traverser, précéder	475	0	0	0
ZZ006	traverser, ébranler, endormir, comporter, étrangler, improviser	117	197	3	0
T1Z05	remettre, renvoyer, retarder, reporter, devancer, ajourner	1019	1	8	70
T1Za8	apprendre, souhaiter, conseiller, reprocher, mander, remontrer	139	0	0	360
T1108	courir, employer, rassurer, asseoir, ennuyer, gonfler, ...	1944	156	22	764
P3005	annoncer, décider, achever, déclarer, démentir, emmancher	1041	12	0	3
TZ3Z6	emporter	2	0	0	0
P10Z6	retourner	7	0	0	0
ZZ0a0	suffire, vendre, confier, adresser, comparer, déplaire	560	0	0	28
T13Z6	rentrer	160	1	11	30
ZZ00Z	préparer, éteindre, déployer, redresser, instituer, dessécher	56	318	80	0
TZZ02	amener, transporter	930	0	0	0
T3306	valoir, fournir, sécréter	877	7	2	32
T11Z0	accompagner, escorter, expatrier	62	0	3	7
TZZZ0	accompagner, poursuivre	7	0	0	0
A33	retirer, dépasser, isoler, rejaillir, suinter, exsuder	178	24	0	9
T1908	défendre, peindre, aider, caractériser, secourir, appesantir, ...	1089	34	26	691
TZZ01	représenter, parcourir, lâcher, allonger, déborder, viser	640	6	36	145
P8000	réunir, distinguer, accorder, rejoindre, joindre, marier, ...	2064	165	11	127

'Frames des verbes conjugués' page suivante

TABLE E. 4 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
T1Z02	réunir	423	12	31	354
TZZb0	saisir, entourer	24	0	0	4
PZ030	sauver, relever, échapper, éloigner, dégager, ôter, ...	1188	44	7	48
ZZ0b8	distinguer	5	0	0	0
TZ200	monter, couvrir, délivrer, couronner, piller	1468	5	12	212
TZ101	fixer, entourer, installer, environner, clouer, immobiliser	612	16	7	145
Z10b0	rire, enchanter, affliger, émerveiller, moquer	69	0	0	0
Z7000	ressembler, étreindre	824	8	78	54
T1ZZ8	relever, expédier	250	8	18	179
PZZ08	brûler	9	0	0	0
ZZ6	brûler	8	285	244	0
ZZ030	échapper, écarter, épancher, éclipser	399	23	0	0
P1100	manger, associer, cogner, adjoindre	32	1	2	11
TZZ05	poursuivre, repousser	448	0	28	105
TZ102	mener	4	0	0	0
A20	retenir, délivrer, ronger, manier, aboyer, feindre, ...	250	6	13	19
Z3006	augmenter, accroître, conquérir, regagner, recouvrer, reconquérir	39	74	12	0
Z10bZ	empêcher	3	0	0	0
PZ040	répandre	279	3	1	0
T3906	habiter	2	0	0	0
ZZ906	habiter	1	0	0	0
T1808	mêler, mélanger, emmêler, enchevêtrer, entortiller, panacher	181	4	11	36
ZZ000	étonner, rejoindre, accuser, vaincre, inventer, contracter, ...	131	376	58	398
NZa	résister, appartenir, succéder, contribuer, nuire, adhérer	1084	0	0	825
A17	trembler, périr, rougir, crever, sécher, vibrer	47	0	1	2
TZZa8	indiquer, prescrire	9	0	0	14
A37	dominer, sombrer, flamboyer, vaciller, chavirer, vaguer	340	13	6	2
Z37	dominer, rétrocéder	3	1	0	0
T11Z8	entraîner, convoquer, débaucher, désarçonner	13	4	0	13
Z10	accomplir, ravager, hériter	26	236	9	0
PZ400	imaginer	89	0	0	0
P1050	prouver, promettre, persuader	113	4	0	20
Z1050	prouver	8	0	0	0
Z3000	concevoir, dépasser, franchir, constater, modifier, soupçonner, ...	283	83	27	20
T13Z8	rapporter, glisser, remporter	625	2	14	96
T1907	accuser, célébrer, louer, blâmer, contester, féliciter, ...	546	22	14	134
T1906	juger, surfaire	22	0	0	0
Z1b	juger	46	0	0	0
PZ0Z6	rapprocher	345	5	4	0
ZZb	jouir, regorger	41	0	0	0
ZZ300	créer, attribuer, refroidir, infliger	155	33	18	0
T3900	arracher, intéresser, concerner, élargir, affaiblir, vieillir, ...	740	2	38	1
TZ108	intéresser, enivrer, révolter, ressusciter, corrompre, mûrir, ...	316	29	13	145
NZZ00	devoir	3	0	0	0
Z300Z	acheter, construire, réparer, ménager, enflammer, resserrer, ...	88	416	107	0
T9300	respirer, répliquer, suer, puer	175	12	0	3
A12	accourir, rabattre, regrimper, abouler	510	8	1	27
ZZa	parvenir	18	0	0	27
T13Z2	importer	56	0	0	0

'Frames des verbes conjugués' page suivante

TABLE E. 4 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
P1040	transporter, trimbaler	136	13	0	7
P30Z0	développer, vider	179	13	0	0
Z30Z0	développer, combler, recouvrir, cribler	21	12	0	0
TZ1Z0	emmener, assaillir	267	0	0	0
T1305	reculer, différer, perpétuer, proroger, anticiper	147	2	4	44
TZ5b0	menacer	42	0	0	0
T1301	exposer, publier, projeter, propager, stationner, aventurer, ...	560	39	11	162
TZ120	déposer	192	0	0	0
T8300	envahir, submerger	243	0	0	0
T1700	diviser, citer, énumérer, dénombrer	143	0	2	16
ZZ1	remuer	1	32	2	0
Z1500	douter	19	54	168	1278
Z1Z00	figurer	5	17	30	387
PZ0b0	emparer	372	0	0	24
TZ700	déchirer	117	0	0	0
Z130Z	redoubler	2	13	0	0
T2200	lutter, délivrer	50	0	0	0
Z1000	grandir, diminuer, démanger	15	23	6	0
T1Z20	introduire, cracher, remorquer, piloter	272	3	17	106
ZZ0a6	consacrer	109	0	0	0
TZ908	user, calmer, enlacer	172	11	1	156
T1102	inviter, citer, assigner, mander, embaucher	233	8	6	32
ZZ908	calmer	19	2	0	0
A13	dégager, déboucher, déserter, dévier, décamper, détaler	155	1	5	5
TZZ20	dégager	36	5	6	62
ZZ500	rougir	1	1	1	0
Z16	dissiper, scruter, vaquer	10	327	14	0
T1328	flotter, brouetter	18	0	0	6
P7300	disputer	113	0	0	0
Z30b0	dater, exclure, discerner, emplir, argumenter, déduire, ...	97	0	0	0
ZZ0	triompher	9	300	42	22
N1Za0	soumettre	1	0	0	0
T1Za0	soumettre, confisquer, substituer, inféoder	223	0	0	418
N1ZaZ	proposer	7	0	0	0
T1ZaZ	proposer	61	0	0	63
Z3300	évanouir, décharner, atrophier	9	7	3	0
TZ107	accabler, consoler	168	14	1	135
TZ1Z7	accabler	21	0	0	0
TZ701	distribuer	70	3	6	23
Z3100	émouvoir, enrouer, désaltérer, alanguir, enorgueillir, désillusionner	33	23	2	0
T1320	échouer, convoyer	16	4	4	25
ZZ107	consoler	1	0	0	0
TZ10Z	charmer, fasciner, gifler	126	12	20	37
T1307	punir, déserter, commenter, redire, commémorer	191	7	5	64
TZ1b0	punir, récompenser, débarrasser, munir, dédommager, déshériter	120	0	0	18
T1Z30	débarquer, vider, replier, flanquer, dévisser	135	1	13	48
Z30	prêter, épier	87	138	0	0
TZ400	supposer, flairer	123	7	9	80
T1107	gronder, damner, réprimander, enguirlander, gourmander, sermonner	97	5	6	28

'Frames des verbes conjugués' page suivante

TABLE E. 4 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
T3Z01	environner	52	0	0	0
T3Z06	absorber, obstiner	6	0	0	0
PZZ06	fatiguer	26	0	0	0
P2000	couronner	46	0	0	1
P1Z06	tordre	1	0	0	0
N3b	résonner, résulter, émaner, découler	205	35	37	1
Z1300	expirer	5	18	61	0
ZZ0b0	encombrer, corriger, décorer, démêler, abreuver, enduire	61	0	0	0
P3030	exhaler, déchausser	52	2	0	0
PZ0Z1	déranger	8	0	0	0
ZZ308	contenter, égayer, améliorer, ennoblir	9	11	1	0
P9000	affaiblir, appesantir, dorer, alourdir, alléger, amoindrir	87	1	0	4
TZ1a0	destiner, accoutumer, adapter, acclimater	165	0	0	75
T11b0	remercier, complimenter, inculper, indemniser	134	38	16	25
A80	correspondre, coïncider, concorder, converger, coexister, rimer	28	0	0	8
A86	correspondre, coïncider, cadrer, rimer	143	0	0	16
NZ3a0	procurer, suggérer	2	0	0	0
TZ3a0	procurer, transmettre, susciter, suggérer	124	0	0	144
ZZ108	enivrer, ressusciter	6	3	0	0
TZ3b0	dépouiller, joncher, parsemer	41	0	0	0
ZZZ08	sécher, dégringoler	2	8	0	0
TZ5a0	interdire	16	0	0	0
ZZ008	assombrir, affermir, amollir, alourdir, avilir, purifier	11	42	0	0
A24	avaler, décocher	13	0	0	1
PZZ30	casser	2	0	0	0
T1920	acheminer, voiturer	24	0	0	10
P3007	écrouler, déchaîner	4	0	0	0
TZ3Z8	voiler	1	0	0	0
Z1Z08	déconcerter	8	35	127	1536
Z26	raser	2	20	0	0
NZ3bZ	orner	5	0	0	0
TZ3bZ	orner	90	0	0	0
N1Za8	reprocher	2	0	0	0
NZ1b0	débarrasser, dédommager	3	0	0	0
Z3008	aggraver, copier, terrasser, consolider, contrefaire, régulariser, ...	51	13	7	0
PZ007	consumer, déchaîner, languir	54	0	0	0
Z11	fouler	1	46	8	0
ZZ0Z0	relier	1	46	0	0
A26	ruer, forger, encenser, cadencer, regimber, piéter	20	2	0	5
P7040	ruer, déverser	81	0	1	0
NZ4b0	exiger	2	0	0	0
TZ4b0	exiger	76	0	0	0
Z1Z40	exiler	12	7	0	0
T9106	impatienter, abrutir, ragaillardir, abêtir, efféminer	38	0	0	14
T3308	empourprer	36	3	2	2
Z33	isoler	5	17	0	0
T1800	aligner, éparpiller, répartir, démêler, émietter, récapituler, ...	49	1	1	6
TZ907	flétrir	9	2	2	14
Z3001	stationner, épandre, réinstaller, appendre, thésauriser	7	20	2	0

'Frames des verbes conjugués' page suivante

TABLE E. 4 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
Z1030	trotter, esquiver	7	6	29	14
Z7001	côtoyer, ameuter, coudoyer	6	2	0	0
A23	déceler, dénicher	9	0	0	0
TZ1b8	dégoûter	4	0	0	0
Z1006	contredire, estropier	5	1	0	0
T11b8	convaincre, destituer, exempter, dessaisir, déposséder, spolier	73	4	1	13
Z10aZ	vouer	51	0	0	0
ZZ1a0	habituer, abonner	33	0	0	0
Z1108	suffoquer, saouler	2	7	0	0
T1Zb8	préserver, implorer	48	0	0	11
T1208	pêcher, darder, engluer	36	0	5	7
Z30bZ	garnir, agrémenter, badigeonner	51	0	0	0
T1801	amonceler, superposer, déballer, saupoudrer	31	4	1	3
T7308	investir, infester	40	3	0	4
Z1308	empirer	1	5	3	12
T2300	pointer, paître, brouter, curer, pâturer	24	0	0	3
ZZZ00	agacer	1	3	0	0
Z3Z00	assoupir, étioler	6	3	0	0
T3106	proportionner	6	1	1	1
P8001	superposer, déballer, saupoudrer	11	19	0	0
T1130	déménager, congédier, déloger	38	0	2	17
TZZb6	imprégner	13	0	0	0
P9008	pétrifier, enhardir, sanctifier, immortaliser, idéaliser, pacifier, ...	38	16	2	1
NZ3b0	joncher, parsemer	5	0	0	0
T190Z	symboliser, égratigner, parodier, reluquer	35	0	6	5
Z9008	sanctifier, immortaliser, idéaliser, repopulariser	6	0	0	0
T9101	recoucher, jucher	6	0	0	4
Z9101	recoucher	3	0	0	0
P10b0	enquérir, éprendre, méfier, repentir, gausser, amouracher	64	260	24	53
T11Z6	rassasier	4	0	0	0
A34	appareiller, débouquer	11	0	0	0
T3Z20	déverser	9	0	0	0
ZZi	foisonner	1	0	0	0
P2030	dénicher, départir, fourvoyer, débusquer	26	0	0	2
Z1Z01	dénicher	1	18	7	0
N4a	advenir, incomber	10	3	1	8
ZZ0a8	assujettir	12	0	0	0
N1a	attenter, subvenir	12	87	75	690
T14b0	raffoler, augurer	22	10	3	3
T1701	attrouper, regrouper	11	1	4	0
P2006	acculer, pelotonner	7	0	1	1
T1200	acculer, enfumer	7	6	0	7
ZZ100	chavirer, adjoindre, farcir	7	6	2	0
T1Zb0	suspecter, gaver	3	2	1	29
T13Z1	colporter	2	1	0	2
T11a8	convier	13	2	0	29
Z1306	épeler, jargonner	2	3	6	0
ZZ306	désorganiser	1	2	0	5
T7300	infester, coloniser, ratisser, réoccuper	13	2	1	1

'Frames des verbes conjugués' page suivante

TABLE E. 4 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
P10bZ	abstenir	12	0	0	0
TZ1bZ	gorger	2	0	0	0
T1201	hiverner	3	0	0	0
A18	télégraphier, téléphoner, pagayer	10	0	1	0
Z30b6	doter, émailler, paver	12	0	0	0
A21	outrepasser, butiner, piper, terrir	9	0	1	1
T3800	agglomérer, agglutiner	2	0	0	0
N11bZ	affubler	1	0	0	0
T11bZ	affubler, sevrer	7	0	0	0
Z14a8	riposter	7	0	0	0
AZ8	grésiller	3	0	0	0
Z30aZ	amarrer	3	0	0	0
T2301	butiner	1	0	0	0
P9006	avérer	1	0	0	0
T33a0	occasionner	3	3	1	1
T1302	drainer	1	0	0	0
T1140	remmener, raccompagner	4	0	0	0
A28	renâcler, striduler, sonnailler	4	0	0	0
Z8000	dépareiller	1	0	0	0
T13b0	cercler, ourler	2	90	24	30
Z9000	déchristianiser	1	0	0	0
Z91a0	réabonner	1	0	0	0
T2208	empiéter	1	0	0	0
Z3900	désengourdir	1	0	0	0
Z31b0	désaccoutumer	1	0	0	0
N1b	exciper	1	72	30	28

E. 4.3 289 frames des verbes au participe présent.

TABLE E. 5 – Frames des verbes au participe présent (289 frames)

Frame	verbes	conj.	vpp	vpa	inf.
AZ2	être, tomber, avancer, bondir	142112	11	93	3018
N9a	être, avoir, travailler, entrer	0	2	26	8
T1106	avoir, reconnaître, payer, nommer, observer, choisir, ...	2069	268	55	388
T13b0	avoir, faire, prendre, tenir, gagner, exprimer, ...	2	90	24	30
T1Z06	avoir, prendre, trouver, aimer, commencer, chercher, ...	25652	183	509	4468
Z90	avoir	0	6	26	0
AZ1	faire, sortir, naître, apparaître, régner, souffler, ...	17015	200	44	244
N1c	faire, vivre, marcher, converser, deviser	0	19	12	0
TZ106	faire, former, constituer, bâtir, inquiéter, balancer	11087	62	297	3679
TZ306	faire, rendre, laisser, suivre, prononcer, soutenir, ...	4622	159	327	1599
ZZ0	faire, prendre, manquer, servir, reprendre, frapper	9	300	42	22
TZ500	pouvoir, vouloir, devoir	1237	5	8	16
Z4000	pouvoir, agir	14291	0	25	253
N1a	voir, rester, parler, regarder, arriver, continuer, ...	12	87	75	690
P10c0	voir, marier, entretenir, mesurer, associer, empoigner	0	31	7	0
T11g0	voir, accouder	0	19	2	1
T1Z0Z	voir, mettre, tenir, pousser, sentir, continuer, ...	8635	138	757	6041

'Frames des verbes au participe présent' page suivante

TABLE E. 5 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
ZZ001	voir, cacher, aborder	67	233	11	0
P10a0	prendre, attendre, arrêter, reprendre, former, refuser, ...	0	175	46	166
PZ006	prendre, donner, mettre, perdre, rencontrer, élever, ...	9441	314	44	12
T13a0	prendre, donner, rendre, laisser, ouvrir, garder, ...	0	35	30	202
Z1Z08	prendre	8	35	127	1536
A1b	donner, battre, souffler, siffler	0	6	5	9
N1j	donner, venir, chercher, descendre, persister, fouiller, ...	0	55	28	0
N3g	donner, venir, regarder, tomber, renverser, planer, ...	0	56	51	0
TZ301	donner	5070	21	83	936
N1b	aller, tenir, juger, jouir, sauter, préjuger	1	72	30	28
N3a	aller, passer, plaire, concourir, préluder, répugner	0	10	7	17
T1500	aller, songer	2107	0	7	5
Z1030	aller, sortir, défiler, trotter	7	6	29	14
ZZ005	aller	0	4	10	0
PZ001	trouver, montrer, cacher, écouler, lancer, promener, ...	8151	259	40	165
TZZ00	trouver, savoir, connaître, regarder, élever, battre, ...	20109	180	546	3201
ZZ00Z	trouver, passer, dire, remonter	56	318	80	0
T13g0	passer, jeter, jouer, élever, charger, conserver, ...	0	227	36	0
T1Z40	passer, mener, promener, emmener, exiler	1196	17	64	439
Z300Z	passer, continuer, jouer, travailler, terminer, rapporter, ...	88	416	107	0
P3008	dire, expliquer, dérouler, communiquer, courber, élargir, ...	1676	257	28	76
T1306	dire, parler, aimer, sentir, écrire, changer, ...	7961	198	241	2175
Z1500	dire	19	54	168	1278
P1007	vouloir, extasier	56	41	5	16
P10a6	mettre, essayer, escrimer	0	17	4	6
T11a0	mettre, former, décider, nommer, exercer, éveiller, ...	0	35	14	55
ZZ006	mettre, finir, traverser	117	197	3	0
AZZ	venir, partir, vivre, disparaître, courir, dormir, ...	17794	38	87	776
T3500	venir	3467	12	4	15
Z1020	venir	0	94	5	0
AZ6	rester, paraître, attendre, revenir, écrire, descendre, ...	20762	176	88	1836
A16	parler, agir, lire, causer, songer, chanter, ...	9029	157	137	525
A1c	parler, causer	0	0	21	0
A7g	parler, disputer, discuter, bavarder, converser	0	0	5	0
N1k	parler, lutter, ester	0	0	5	0
Z7000	parler	824	8	78	54
P10Z0	porter, hisser	648	18	13	0
T1ZZZ	porter, traîner, rejeter	206	2	41	68
TZ100	porter, jeter, quitter, finir, abandonner, travailler, ...	18144	433	541	4139
TZ302	porter	1666	4	194	597
P30a0	rendre, ouvrir, apporter, placer, retourner, céder	0	119	54	19
PZ020	rendre, présenter	2791	30	6	0
TZ30Z	rendre, traverser, couper, percer, contracter, piller	2152	62	108	976
T1Z00	croire, entendre, attendre, servir, penser, reconnaître, ...	28199	310	971	6697
T1101	connaître, accompagner, joindre, enfermer, abriter, évoquer	7997	29	40	122
T1Z08	entendre, recevoir, arrêter, frapper, rappeler, rencontrer, ...	9028	396	345	3353
P3001	laisser, révéler, exposer, éventer, épandre, filtrer	1671	103	15	181
P30j0	laisser, changer, cacher, retourner, précipiter, enfoncer, ...	0	376	19	0
P90b0	laisser	0	4	3	0

'Frames des verbes au participe présent' page suivante

TABLE E. 5 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
T13j0	laisser, jeter, cacher, remettre, couper, séparer	0	115	16	0
A40	demander	5472	16	4	37
Z1406	demander, expliquer	0	10	33	489
A96	devenir	6886	321	20	436
T1901	regarder, apercevoir, remarquer, contempler	1681	8	27	2
P10a8	tenir	0	11	10	2
TZ900	tenir, lancer, arracher, creuser, élargir, affaiblir, ...	2859	58	133	699
Z3Z	tenir, croiser	0	25	11	0
P1000	aimer, écouter, admirer, détourner, estimer, citer, ...	3721	316	17	470
ZZ6	attendre, pousser, écrire, changer, payer, craindre	8	285	244	0
P1001	recevoir, remuer, amuser, abriter, recruter	824	172	11	103
TZZ08	recevoir, perdre, frapper, conduire, battre, courir, ...	5620	276	279	3172
ZZ000	suivre, chercher, répondre, comprendre, rentrer, interroger	131	376	58	398
A3e	commencer	0	0	12	0
Z3006	commencer, continuer, compter, augmenter, couler, accommoder	39	74	12	0
P10j0	perdre, avancer, engager, répandre, prolonger, baigner	0	136	13	0
N3b	tomber, percer, procéder, déborder, jaillir, participer	205	35	37	1
T110Z	tomber, servir, instruire, utiliser, convoquer, écarteler	3626	38	30	138
Z110Z	tomber	0	53	9	0
P300Z	ouvrir, dresser, enlever, dessiner, creuser, redresser	1459	150	26	0
T130Z	ouvrir, reprendre, présenter, oublier, retirer, terminer, ...	3727	220	190	1959
ZZZ	ouvrir, tirer, reposer, traîner, piquer	0	48	9	0
T1138	jeter, proscrire, extirper	48	4	8	29
T11b0	jeter, frapper, remettre, payer, éclairer, saluer	134	38	16	25
Z16	jeter, présenter, lire, représenter, disposer, ordonner	10	327	14	0
Z1Z	jeter, repasser, rabattre	0	61	9	0
AZ5	arriver, marcher, fuir, survenir, sommeiller	5593	97	74	593
TZ300	montrer, appeler, sortir, comprendre, présenter, garder, ...	44435	490	514	5097
T1102	appeler, inviter, assigner	233	8	6	32
T34b0	appeler	0	0	1	0
T14a8	sortir, relater	0	6	2	19
A11	servir, toucher, sonner, reparaître, guetter, heurter, ...	1617	36	27	97
A37	mourir, céder, étinceler	340	13	6	2
AZ7	mourir, souffrir, étinceler, rôder, resplendir, piétiner	3636	0	12	496
Z3000	penser, mener, trembler, sauter, rêver, souffler, ...	283	83	27	20
T1Z07	reprendre, descendre, célébrer, consacrer, applaudir, discuter, ...	1119	60	54	636
T1308	pousser, tourner, causer, cesser, attacher, engager, ...	5343	328	210	1819
T1Z01	garder, quitter, établir, abandonner, représenter, fixer, ...	2182	128	98	1187
N1g	frapper, tirer, jouer, agir, avancer, régner, ...	0	16	54	0
P10b0	frapper, éloigner, nourrir, défier, exaspérer, réjouir, ...	64	260	24	53
P30a8	frapper, abandonner, réunir, joindre, suspendre, piquer	0	83	15	0
T13a8	frapper, expliquer, réunir, révéler, joindre, ôter, ...	0	35	21	84
Z11	frapper, exercer	1	46	8	0
P100Z	reconnaître, élancer	594	18	11	0
T19j0	reconnaître	0	3	1	0
Z1Z01	reconnaître, déborder	1	18	7	0
Z96	sentir	0	12	5	0
N9b	revenir, rentrer	0	17	42	0
PZ0Z6	lever	345	5	4	0

'Frames des verbes au participe présent' page suivante

TABLE E. 5 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
TZZ38	lever	370	8	49	0
T13a6	tirer, mesurer	0	11	4	7
TZZ06	tirer, posséder, épouser, découvrir, marquer, attaquer, ...	6413	84	85	1287
PZ008	former, préparer	786	72	6	0
ZZ106	former	0	19	1	0
Z1400	rappeler, remettre, charger, représenter, permettre, revoir	0	17	97	676
T1300	vivre, crier, accomplir, mériter, marcher, interroger, ...	58333	97	138	1304
Z1300	vivre, réussir, crier, marcher, expirer, gémir, ...	5	18	61	0
TZ800	rencontrer, rejoindre, dépouiller	1652	1	17	466
A1g	descendre, tourner, obliquer	0	24	6	0
TZZ07	descendre, allumer, déchirer, incendier, charrier	494	49	42	153
P1006	conduire, oublier, défendre, livrer, étouffer, constituer, ...	1729	186	34	261
Z3100	demeurer, rebondir	33	23	2	0
Z1Z00	envoyer, payer, pleurer, figurer, taper, renifler	5	17	30	387
P30g0	élever, retourner, étendre, fixer, répandre, promener, ...	0	411	32	0
Z10	oublier, emporter, achever, répéter, becqueter	26	236	9	0
P3000	apercevoir, accompagner, éprouver, poursuivre, deviner, détruire, ...	17611	950	45	1041
T1340	apporter	1244	10	25	95
P10d0	battre, révolter, récrier	0	13	3	0
TZZ0Z	toucher, gagner, saisir, retenir, abattre, étaler	4132	50	165	1036
A1q	gagner, craindre, trembler	0	11	5	0
P3006	gagner, emporter, vendre, hausser, prolonger, abattre, ...	6979	623	37	234
T1Zb0	gagner	3	2	1	29
P1020	approcher, ramener, égarer	398	38	4	83
T1ZZ0	approcher, emmener	231	4	29	235
Z12	approcher, retourner	0	8	13	0
T3300	annoncer, exprimer, prouver, révéler, témoigner, renfermer, ...	14882	81	95	559
T13Z6	emporter, rentrer	160	1	11	30
A1Z	rentrer, succomber, nager, tâtonner, flâner, divaguer	1196	57	27	76
ZZ2	avancer	0	2	5	0
T13k0	abandonner	0	0	1	0
AZ0	apparaître, exister, céder, reparaître, triompher, repartir, ...	25236	14	30	904
T1Z02	amener, réunir, renvoyer	423	12	31	354
ZZ020	amener, hisser	0	3	2	0
P30b0	étendre, allonger, aboutir, exhaler, défiler, limiter	0	231	12	1
PZ00Z	étendre, représenter, couper	8512	22	6	10
T13b8	étendre, essuyer, noyer, élaguer	0	31	4	2
T3306	valoir	877	7	2	32
PZ0Z0	retirer, précipiter	1538	20	10	0
ZZ300	causer, respirer, chevaucher	155	33	18	0
N3j	régner, consister	0	4	6	0
T11b8	défendre	73	4	1	13
T11d8	défendre	0	0	2	0
T1908	défendre, aider, caractériser, secourir, appesantir	1089	34	26	691
Z100Z	défendre	0	12	3	0
A3Z	éclater, déborder, luire	1615	5	3	27
Z1000	éclater, reculer, diminuer	15	23	6	0
T18j0	réunir	0	7	1	0
T1ZbZ	payer	0	0	1	7

'Frames des verbes au participe présent' page suivante

TABLE E. 5 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
TZ400	souffrir, supposer	123	7	9	80
ZZ400	souffrir	0	6	4	0
TZ200	couvrir, délivrer	1468	5	12	212
TZ101	fixer	612	16	7	145
ZZ100	sonner	7	6	2	0
N3c	chanter	0	0	1	0
Z1Z06	chanter	0	5	6	0
T1108	employer, entourer, dominer, entraîner, rassurer, diminuer, ...	1944	156	22	764
N1i	pleurer, trembler, repousser, déjeuner, renifler, odorer	0	17	16	0
P7000	ressembler, provoquer, défier, étreindre, répugner	890	9	6	103
PZ030	relever, échapper, éloigner, envoler	1188	44	7	48
T1ZZ8	relever	250	8	18	179
A36	attacher, profiter, grandir, diminuer, resplendir, languir	4717	52	10	67
TZ108	manger, ressusciter, corrompre, mûrir, décomposer, noircir	316	29	13	145
TZZ05	poursuivre	448	0	28	105
A10	espérer, sourire, prier, saluer, dîner, goûter, ...	12853	125	54	811
T1400	espérer, empêcher, souhaiter, envisager, interdire, méditer	3829	33	53	326
Z41	éclairer	0	12	1	0
P30g6	dresser, refléter	0	11	3	0
T33g0	dresser, projeter	0	5	2	0
P30d0	briser	0	3	1	0
P10g0	soulever, serrer, élancer, incliner, attendrir, acheminer	0	72	16	0
T13c0	soulever, discuter	0	1	2	0
PZ040	répandre	279	3	1	0
PZ000	mêler	13904	18	1	29
T1808	mêler, mélanger	181	4	11	36
T3Z00	indiquer, irriter, crisper, ravager, dénoter	4502	79	28	157
T14a0	crier, glisser	0	0	19	18
T11j0	entraîner	0	16	1	0
TZ308	contenir, réveiller, supporter, arranger, déployer, confirmer, ...	1387	99	104	1069
T33a0	mériter	3	3	1	1
P1Z00	imaginer	755	1	5	129
T1100	prier, invoquer, bénir, confesser, habiller, dévisager	6782	38	26	208
T13Z8	rapporter, glisser	625	2	14	96
A30	retentir, épier, scintiller, crépiter, jaunir, fraîchir, ...	9895	45	15	219
T13c8	serrer, souder	0	5	2	0
Z1106	obéir	0	1	1	0
TZZ01	parcourir, lâcher, allonger, viser	640	6	36	145
T14b0	respirer, prévoir, entreprendre	22	10	3	3
A12	accourir	510	8	1	27
T1900	plaindre, redouter, dédaigner, lorgner, courtiser, guigner	3046	25	22	177
T1Z20	traîner, introduire, cracher	272	3	17	106
N1m	reculer, repasser	0	0	3	0
T1305	reculer, ajourner	147	2	4	44
T1301	exposer, projeter, propager, démasquer, épandre, exhiber	560	39	11	162
P8000	marier, séparer, associer, échanger, unir	2064	165	11	127
ZZ1	remuer, pointer	1	32	2	0
T13e0	renouveler	0	0	1	0
T13b6	désigner	0	19	6	2

'Frames des verbes au participe présent' page suivante

TABLE E. 5 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
T19a0	désigner	0	1	1	0
TZ906	dévorer	1569	1	6	44
Z3001	réfléchir	7	20	2	0
T1Z38	rejeter	5	9	1	6
Z3008	redoubler, déboucher, fureter	51	13	7	0
A70	lutter, différer	1021	3	2	218
T1907	célébrer, louer, féliciter, plaisanter, railler, chansonner	546	22	14	134
TZ908	user	172	11	1	156
TZ10Z	précéder	126	12	20	37
TZZ20	dégager	36	5	6	62
ZZ500	rougir	1	1	1	0
T31j0	transformer	0	2	2	0
Z7300	disputer	0	2	2	44
P1400	promettre, proposer, appuyer	989	34	29	104
P1030	enfuir, replier, esquiver, évaporer	74	11	5	41
Z1Z07	applaudir	0	1	1	0
TZ107	accabler	168	14	1	135
P1300	attribuer, pardonner, approprier	1119	3	6	160
TZ701	distribuer	70	3	6	23
T11b6	honorer	0	3	1	1
A13	repartir, jaillir, émerger, rétrograder, gicler	155	1	5	5
Z3300	pâlir	9	7	3	0
Z1306	murmurer, déclamer	2	3	6	0
T1307	punir, déserter, commenter	191	7	5	64
T1130	débarquer, congédier	38	0	2	17
PZ005	envoler	1438	6	2	67
T3900	concerner	740	2	38	1
T1107	gronder, enguirlander, sermonner	97	5	6	28
T1Z05	retarder, devancer, reconduire	1019	1	8	70
N9g	heurter	0	0	1	0
T1320	reporter, embarquer	16	4	4	25
T1Z30	vider, replier, redescendre, flanquer	135	1	13	48
P1100	associer	32	1	2	11
T3400	démontrer, impliquer	949	3	2	26
T3100	tremper, attrister, divertir, effaroucher, hébéter, rechigner	4499	223	7	287
A35	intervenir, discontinuer	1296	2	2	105
Z1308	gratter, érailler	1	5	3	12
Z1907	plaisanter	0	0	2	0
P7040	ruer	81	0	1	0
Z7040	ruer	0	0	1	0
T13c6	combiner	0	0	1	0
Z20	manier	0	7	1	0
T3308	empourprer	36	3	2	2
T11Z0	escorter	62	0	3	7
P1008	mûrir	63	23	1	11
TZ907	flétrir	9	2	2	14
AZ3	émerger	192	0	1	2
A41	tonner	281	3	1	25
P10b8	préserver	0	4	1	0

'Frames des verbes au participe présent' page suivante

TABLE E. 5 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
P7001	bousculer, masser, égrener	222	98	3	9
Z9300	suer	0	0	1	0
T1700	énumérer	143	0	2	16
A31	tourbillonner, affleurer	1448	4	2	54
T3106	proportionner	6	1	1	1
N1d	déclamer, pester, déblatérer	0	4	4	0
T1801	superposer	31	4	1	3
T1208	darder	36	0	5	7
A17	grimacer	47	0	1	2
T190Z	symboliser, égratigner, parodier	35	0	6	5
ZZ308	rôtir	9	11	1	0
N4a	advenir	10	3	1	8
T1701	égrener, regrouper	11	1	4	0
A20	grogner, japper, grommeler, croasser, pâturer	250	6	13	19
P9008	idéaliser, spiritualiser	38	16	2	1
T1800	récapituler	49	1	1	6
N2i	frétiller	0	0	1	0
T13l0	quêter	0	1	1	0
N8g	converger	0	0	2	0
T14a6	télégraphier	0	1	1	0
A21	butiner	9	0	1	1
P2006	pelotonner	7	0	1	1
N3n	serpenter	0	0	4	0
A18	pagayer	10	0	1	0
T7300	quadriller	13	2	1	1
P30c8	embrever	0	8	1	0
P10g6	méprendre	0	1	1	0

E. 4.4 339 frames des verbes au participe passé.

TABLE E. 6 – Frames des verbes au participe passé (339 frames)

Frame	verbes	conjugué	vpp	vpa	infinitif
AZ2	être, tomber, voler	142112	11	93	3018
T1Z06	avoir, prendre, trouver, aimer, suivre, commencer, ...	25652	183	509	4468
Z90	avoir	0	6	26	0
AZ1	faire, sortir, naître, apparaître, coucher, habiter, ...	17015	200	44	244
N1c	faire, sortir, tourner, traiter, briser	0	19	12	0
P10a0	faire, attendre, arrêter, reprendre, occuper, former, ...	0	175	46	166
P30j8	faire, basaner	0	21	0	0
T31q6	faire, naître, tailler	0	29	0	0
TZ106	faire, former, constituer, bâtir, pénétrer, égarer	11087	62	297	3679
TZ306	faire, rendre, laisser, suivre, prononcer, soutenir, ...	4622	159	327	1599
ZZ0	faire, prendre, manquer, servir, reprendre, frapper, ...	9	300	42	22
N1a	voir, trouver, regarder, arriver, chercher, écrire, ...	12	87	75	690
T1Z0Z	voir, mettre, tenir, pousser, sentir, continuer, ...	8635	138	757	6041
ZZ001	voir, retrouver, cacher, produire, réunir, fixer, ...	67	233	11	0
PZ006	prendre, donner, mettre, perdre, rencontrer, élever, ...	9441	314	44	12
T130q	prendre, mettre, employer, usiter	0	9	0	0

'Frames des verbes au participe passé' page suivante

TABLE E. 6 – *Suite de la page précédente*

Frame	verbes	conjugué	vpp	vpa	infinitif
T13b0	prendre, toucher, gagner, retirer, secouer, rapporter, ...	2	90	24	30
Z1Z08	prendre, déconcerter, ballotter	8	35	127	1536
A1b	donner, battre, souffler	0	6	5	9
N1j	donner, venir, tenir, tomber, descendre, glisser, ...	0	55	28	0
N3g	donner, venir, gagner, marquer, baisser, renverser, ...	0	56	51	0
P30a0	donner, rendre, présenter, apporter, placer, représenter, ...	0	119	54	19
TZ301	donner, mouiller, implanter	5070	21	83	936
Z36	donner, soutenir, presser, joindre, réduire, nourrir, ...	0	123	0	0
PZ001	trouver, montrer, retrouver, cacher, produire, écouler, ...	8151	259	40	165
TZZ00	trouver, connaître, regarder, élever, battre, emporter, ...	20109	180	546	3201
ZZ00Z	trouver, passer, dire, préparer, étendre, dresser, ...	56	318	80	0
N3a	passer, commencer, chanter	0	10	7	17
T13a0	passer, rendre, jeter, occuper, envoyer, retourner, ...	0	35	30	202
T1Z40	passer, exiler	1196	17	64	439
Z300Z	passer, continuer, jouer, annoncer, saisir, travailler, ...	88	416	107	0
P3008	dire, expliquer, rapporter, troubler, attendrir, essuyer, ...	1676	257	28	76
Z1500	dire	19	54	168	1278
P1007	vouloir, extasier, pâmer	56	41	5	16
T1101	vouloir, connaître, accompagner, accueillir, enfermer, évoquer	7997	29	40	122
TZ500	vouloir, devoir, rougir	1237	5	8	16
P10a6	mettre, tuer, employer, adonner	0	17	4	6
P10c0	mettre, battre, expliquer, épouser, lier, déchirer, ...	0	31	7	0
P10j0	mettre, connaître, entendre, perdre, reconnaître, avancer, ...	0	136	13	0
ZZ006	mettre, finir, conduire, tourner, traverser, établir, ...	117	197	3	0
AZZ	venir, partir, disparaître, monter, rouler, échouer	17794	38	87	776
T3500	venir	3467	12	4	15
Z1020	venir	0	94	5	0
AZ6	rester, paraître, attendre, revenir, écrire, descendre, ...	20762	176	88	1836
T1306	parler, aimer, écrire, changer, avancer, conserver, ...	7961	198	241	2175
Z7000	parler, doubler	824	8	78	54
P10Z0	porter, hisser, exiler	648	18	13	0
TZ100	porter, jeter, quitter, occuper, finir, abandonner, ...	18144	433	541	4139
TZ302	porter	1666	4	194	597
PZ020	rendre, présenter, rabattre, aventurer	2791	30	6	0
TZ30Z	rendre, traverser, couper, percer, contracter, sceller, ...	2152	62	108	976
ZZ6	rendre, attendre, pousser, écrire, changer, payer, ...	8	285	244	0
P7001	connaître, réunir, revoir, grouper, masser, ameuter	222	98	3	9
ZZ000	connaître, entendre, suivre, chercher, comprendre, rentrer, ...	131	376	58	398
T1Z00	entendre, attendre, servir, penser, reconnaître, apprendre, ...	28199	310	971	6697
T1Z08	entendre, perdre, arrêter, frapper, occuper, rencontrer, ...	9028	396	345	3353
Z1001	entendre, éperdre	0	14	0	0
P3001	laisser, garder, révéler, exposer, refléter, hasarder, ...	1671	103	15	181
P30j0	laisser, jeter, changer, cacher, tourner, compter, ...	0	376	19	0
P90b0	laisser, gangrener, râbler	0	4	3	0
T13j0	laisser, jeter, changer, cacher, tourner, fixer, ...	0	115	16	0
A40	demander, importer, advenir	5472	16	4	37
Z1406	demander	0	10	33	489
A96	devenir, sentir, repousser, redevenir, empoisonner, coiffer, ...	6886	321	20	436
N1b	tenir, naître, décider, mériter, juger, tenter, ...	1	72	30	28

'Frames des verbes au participe passé' page suivante

TABLE E. 6 – *Suite de la page précédente*

Frame	verbes	conjugué	vpp	vpa	infinitif
TZ900	tenir, lancer, arracher, creuser, élargir, affaiblir, ...	2859	58	133	699
Z3Z	tenir, attacher, ralentir	0	25	11	0
P1000	aimer, écouter, posséder, admirer, recueillir, blesser, ...	3721	316	17	470
P1001	recevoir, fixer, contempler, remuer, enfermer, abriter, ...	824	172	11	103
T11j0	recevoir, arrêter, rouler, adopter, absorber, consigner	0	16	1	0
TZZ08	recevoir, perdre, frapper, battre, fermer, tromper, ...	5620	276	279	3172
P80c0	suivre, entrelacer	0	7	0	0
Z3006	commencer, continuer, compter, augmenter, couler, accommoder	39	74	12	0
N3b	tomber, glisser, percer, déborder	205	35	37	1
T110Z	tomber, servir, surprendre, amuser, précéder, utiliser, ...	3626	38	30	138
Z110Z	tomber, amuser	0	53	9	0
P300Z	ouvrir, dresser, enlever, renouveler, peindre, dessiner, ...	1459	150	26	0
T130Z	ouvrir, présenter, oublier, retirer, poser, terminer, ...	3727	220	190	1959
ZZZ	ouvrir, tirer, reposer, chasser, traîner, piquer	0	48	9	0
P30g0	jeter, élever, étendre, poser, fixer, répandre, ...	0	411	32	0
T1138	jeter, repousser	48	4	8	29
T13g0	jeter, frapper, tourner, remettre, charger, étendre, ...	0	227	36	0
Z16	jeter, présenter, lire, décider, représenter, oser, ...	10	327	14	0
Z1Z	jeter, étouffer, opérer, mouiller, aviser, déserter	0	61	9	0
P10b0	arrêter, appeler, frapper, occuper, remettre, fixer, ...	64	260	24	53
T31b0	arrêter	0	2	0	0
Z30	arrêter, charger, recommencer, précipiter, enfoncer, rejeter	87	138	0	0
AZ5	arriver, fuir, survenir, moisir	5593	97	74	593
T1102	appeler, inviter, assigner, mander	233	8	6	32
TZ300	appeler, garder, remplir, produire, compter, avancer, ...	44435	490	514	5097
Z1030	sortir	7	6	29	14
A3Z	manquer, déborder, luire	1615	5	3	27
A11	servir, toucher, étudier, consulter, reparaître, enseigner, ...	1617	36	27	97
P90j0	servir, nourrir	0	6	0	0
A37	mourir, dominer, céder, bloquer	340	13	6	2
P3006	reprendre, gagner, emporter, conserver, saisir, atteindre, ...	6979	623	37	234
T18j0	comprendre, réunir, ramasser, englober	0	7	1	0
T1Z01	garder, établir, abandonner, fixer, remarquer, refuser, ...	2182	128	98	1187
N1g	frapper, tirer, jouer, emporter, remonter, presser	0	16	54	0
P30a8	frapper, réunir, joindre, suspendre, préposer, épingler	0	83	15	0
T11b0	frapper, payer, fixer, nommer, éclairer, repousser, ...	134	38	16	25
T13a8	frapper, réunir, enlever, suspendre, tracer, imprimer, ...	0	35	21	84
Z11	frapper, exercer, consulter, manifester, fouler, relâcher	1	46	8	0
P100Z	reconnaître, élancer, classer, asphyxier	594	18	11	0
T1106	reconnaître, payer, nommer, observer, choisir, enseigner, ...	2069	268	55	388
T19j0	reconnaître, nourrir	0	3	1	0
Z1Z01	reconnaître	1	18	7	0
P3000	occuper, apercevoir, remplir, accompagner, éprouver, remarquer, ...	17611	950	45	1041
Z96	sentir, empoisonner, coiffer	0	12	5	0
N9b	revenir, oublier	0	17	42	0
PZ0Z6	lever	345	5	4	0
T1Z38	lever, relever, rejeter	5	9	1	6
TZZ38	lever	370	8	49	0
ZZ3	lever, dérober	0	12	0	0

'Frames des verbes au participe passé' page suivante

TABLE E. 6 – *Suite de la page précédente*

Frame	verbes	conjugué	vpp	vpa	infinitif
PZ030	tirer, relever, échapper, éloigner, dégager, envoler	1188	44	7	48
T13a6	tirer, soulever, mesurer, emprunter, fréter	0	11	4	7
TZZ06	tirer, former, produire, découvrir, marquer, attaquer	6413	84	85	1287
PZ008	former, préparer, pendre, ruiner, sacrifier, crever	786	72	6	0
T11a0	former, gagner, nommer, dresser, ramener, livrer, ...	0	35	14	55
ZZ106	former, enfler	0	19	1	0
A16	changer, lire, causer, chanter, manger, assurer, ...	9029	157	137	525
ZZ400	rappeler, souffrir	0	6	4	0
T1300	vivre, réussir, habiter, accomplir, mériter, interroger, ...	58333	97	138	1304
Z1300	vivre, réussir, habiter, attirer, expirer, dériver	5	18	61	0
TZ800	rencontrer	1652	1	17	466
ZZ005	rencontrer	0	4	10	0
A1g	descendre, tourner	0	24	6	0
T1Z07	descendre, déchirer, consacrer, applaudir, discuter, ménager, ...	1119	60	54	636
ZZZ07	descendre	0	3	0	0
P1006	conduire, oublier, abandonner, défendre, découvrir, interrompre, ...	1729	186	34	261
T3100	demeurer, émouvoir, préoccuper, tremper, affliger, attrister, ...	4499	223	7	287
Z3100	demeurer, rebondir, angoisser, alanguir, efflanquer, débrailler	33	23	2	0
T1120	envoyer, introduire, renvoyer, déporter	1384	8	0	0
Z1Z00	envoyer, payer, figurer, camper	5	17	30	387
Z10	oublier, remettre, emporter, abandonner, achever, répéter, ...	26	236	9	0
T1308	tourner, cesser, essayer, attacher, engager, partager, ...	5343	328	210	1819
T1340	apporter, exporter, transborder	1244	10	25	95
TZZ0Z	toucher, gagner, saisir, retenir, coucher, rouler	4132	50	165	1036
P1008	tuer, supprimer, instruire, noyer, enivrer, étourdir	63	23	1	11
ZZ008	tuer, renouveler, pendre, supprimer, crever, enrichir	11	42	0	0
A1q	gagner, composer, attaquer, peindre, armer	0	11	5	0
Z31	placer	0	13	0	0
ZZZ08	courir, glacer, sécher	2	8	0	0
T3300	annoncer, exprimer, renfermer, évanouir, pâlir, signifier, ...	14882	81	95	559
T11g0	emporter, éclairer, presser, coller, agenouiller, accouder	0	19	2	1
T13Z6	emporter	160	1	11	30
Z1400	charger, représenter, permettre, aligner	0	17	97	676
P10d0	retourner, défendre, soutenir, soulever, protéger, armer	0	13	3	0
Z12	retourner	0	8	13	0
A1Z	rentrer, parvenir, chasser, opérer, débarquer, émigrer	1196	57	27	76
ZZ2	avancer	0	2	5	0
AZ0	apparaître, exister, céder, repartir, débarquer, dégénérer	25236	14	30	904
P1020	amener, ramener, évanouir, réfugier, égarer	398	38	4	83
T33a0	amener, acquérir	3	3	1	1
ZZ020	amener, aventurer	0	3	2	0
P30b0	étendre, retirer, éclairer, imposer, prévoir, enfermer, ...	0	231	12	1
PZ00Z	étendre, représenter, couper	8512	22	6	10
PZ0Z0	retirer, précipiter	1538	20	10	0
Z33	retirer	5	17	0	0
T14b0	achever, mériter, admirer, regretter, redouter, apprécier	22	10	3	3
ZZ300	causer, éloigner, créer, respirer, refroidir, couver	155	33	18	0
T1908	défendre, peindre, dessiner, caractériser, enlacer, appesantir, ...	1089	34	26	691
Z100Z	défendre, élancer	0	12	3	0

'Frames des verbes au participe passé' page suivante

TABLE E. 6 – *Suite de la page précédente*

Frame	verbes	conjugué	vpp	vpa	infinitif
T1Z02	réunir	423	12	31	354
N9a	travailler	0	2	26	8
P10k0	prononcer	0	2	0	0
Z3000	prononcer, mener, sauter, rêver, étudier, souffler, ...	283	83	27	20
ZZ030	sauver, ôter, replier, épancher, évaporer, carrer	399	23	0	0
P10g6	tromper	0	1	1	0
P8000	distinguer, accorder, marier, opposer, séparer, rattacher, ...	2064	165	11	127
T1901	distinguer, mirer	1681	8	27	2
TZ200	monter, couvrir	1468	5	12	212
A31	couvrir, braquer	1448	4	2	54
TZ101	fixer, submerger	612	16	7	145
Z1Z06	chanter	0	5	6	0
T1108	employer, entraîner, asseoir, ennuyer, obliger, assassiner, ...	1944	156	22	764
N1i	pleurer, manger, repousser, embaumer, refouler, bouffer	0	17	16	0
T1ZZ8	relever, expédier	250	8	18	179
ZZ0Z0	relever, écarter	1	46	0	0
A36	attacher, augmenter, baisser, mesurer, gonfler, trancher, ...	4717	52	10	67
T13c8	attacher, cheviller	0	5	2	0
TZ108	manger, intéresser, enivrer, révolter, ressusciter, mûrir, ...	316	29	13	145
T19d8	assurer	0	1	0	0
T11a8	engager	13	2	0	29
T13j8	engager, pratiquer, nouer, draper, hâler	0	7	0	0
A10	espérer, subir, prier, consentir, saluer, goûter, ...	12853	125	54	811
T1400	espérer, empêcher, convenir, souhaiter, envisager, interdire	3829	33	53	326
Z41	éclairer, pincer	0	12	1	0
ZZ1	éclairer, remuer, gratter, pointer	1	32	2	0
Z70	traiter	0	8	0	0
P30g6	dresser	0	11	3	0
T33g0	dresser, propager, réfracter	0	5	2	0
P30d0	briser, acharner	0	3	1	0
T14a8	soutenir, marquer	0	6	2	19
Z20	retenir, manier	0	7	1	0
P10g0	soulever, serrer, élancer, incliner, ramasser, planter, ...	0	72	16	0
P30c0	soulever	0	1	0	0
Z1Z0Z	coucher	0	3	0	0
PZ040	répandre	279	3	1	0
T1301	répandre, exposer, publier, mouiller, projeter, propager, ...	560	39	11	162
ZZ040	répandre	0	8	0	0
P30c8	mêler, couper, tresser, cheviller	0	8	1	0
PZ000	mêler, raconter, mélanger, assortir	13904	18	1	29
T3306	fournir, dévorer, absorber, sécréter	877	7	2	32
T3Z00	indiquer, élancer, irriter, crisper, ravager, timbrer	4502	79	28	157
Z37	dominer	3	1	0	0
T31g0	ramener	0	1	0	0
T31a0	disposer, fonder, obliger	0	4	0	0
TZ308	contenir, réveiller, supporter, arranger, déployer, ruiner, ...	1387	99	104	1069
P1Z00	imaginer	755	1	5	129
P1050	prouver, persuader	113	4	0	20
TZZ07	allumer, déchirer, incendier	494	49	42	153

'Frames des verbes au participe passé' page suivante

TABLE E. 6 – *Suite de la page précédente*

Frame	verbes	conjugué	vpp	vpa	infinitif
T13Z8	rapporter, remporter	625	2	14	96
Z40	convenir, acquérir	0	39	0	0
T1907	accuser, louer, maudire, féliciter, railler, calomnier, ...	546	22	14	134
T1Z20	rapprocher, traîner, introduire	272	3	17	106
Z1106	obéir	0	1	1	0
A12	accourir	510	8	1	27
N4a	importer, acquérir	10	3	1	8
P1040	transporter	136	13	0	7
P30Z0	développer	179	13	0	0
Z30Z0	développer	21	12	0	0
T1Zb0	chasser	3	2	1	29
A30	éviter, prêter, tinter, contraster, jaunir, parfumer, ...	9895	45	15	219
T1ZZ0	emmener, embarquer	231	4	29	235
P1030	écarter, enfuir, replier, esquiver, extirper	74	11	5	41
T11d0	exciter, armer	0	2	0	0
T13g8	renverser	0	3	0	0
T1ZZZ	traîner, rejeter	206	2	41	68
Z1000	reculer, diminuer, déchoir, moucher, débaucher	15	23	6	0
P10q0	réserver, gêner, passionner, dévouer	0	8	0	0
T11a6	surprendre, chauffer	0	2	0	0
ZZZ00	fuir, bouffir	1	3	0	0
P10b6	signer, embarrasser, revêtir, vêtir, accoutrer, harnacher	0	21	0	4
ZZ100	peser, loger	7	6	2	0
T19a0	désigner	0	1	1	0
TZ906	dévorer	1569	1	6	44
P30b6	peindre, glacer, ensanglanter, rehausser, nuancer, galvaniser, ...	0	45	0	0
T13b8	peindre, épuiser, noyer, mouiller, tremper, baigner, ...	0	31	4	2
Z3001	réfléchir	7	20	2	0
P10m0	incliner, planter	0	2	0	0
P10j6	maintenir, emprisonner	0	11	0	0
T11j6	maintenir, confirmer, limiter	0	3	0	0
Z130Z	redoubler, reproduire, courber	2	13	0	0
Z3008	redoubler, guérir, chauffer, croquer, ferrer	51	13	7	0
P10a8	pendre, cramponner	0	11	10	2
TZ908	user, enlacer	172	11	1	156
ZZ908	calmer	19	2	0	0
TZ10Z	précéder, charmer, fasciner	126	12	20	37
TZZ20	dégager	36	5	6	62
Z13	dégager, démancher	0	16	0	0
TZZ01	lâcher, allonger	640	6	36	145
T11c0	séparer, coaliser	0	3	0	0
ZZ500	rougir	1	1	1	0
T31j0	transformer, incarner	0	2	2	0
T1900	redouter, dédaigner, révérer	3046	25	22	177
P7000	provoquer, défier, remplacer, affronter, enlacer	890	9	6	103
Z7300	disputer, contester	0	2	2	44
P70j0	rassembler, grouper, regrouper, enrégimenter	0	9	0	0
T17j0	rassembler, grouper, confédérer	0	3	0	0
P1400	promettre, proposer, appuyer, remémorer	989	34	29	104

'Frames des verbes au participe passé' page suivante

TABLE E. 6 – *Suite de la page précédente*

Frame	verbes	conjugué	vpp	vpa	infinitif
Z1Z07	applaudir	0	1	1	0
A70	différer, divorcer, dialoguer	1021	3	2	218
Z3300	évanouir, pâlir, convulser	9	7	3	0
P3005	déclarer	1041	12	0	3
TZ107	accabler	168	14	1	135
P1300	attribuer, pardonner, approprier	1119	3	6	160
TZ701	distribuer	70	3	6	23
T1320	échouer, embarquer	16	4	4	25
Z1306	murmurer, balbutier, bégayer	2	3	6	0
T1307	punir, déserter, commenter	191	7	5	64
PZ005	envoler	1438	6	2	67
A20	délivrer, crotter	250	6	13	19
P10b8	abriter	0	4	1	0
Z1308	gonfler, disloquer, démâter	1	5	3	12
T3900	concerner, ankyloser	740	2	38	1
TZ400	supposer	123	7	9	80
N3j	glacer, crever, plisser, engainer	0	4	6	0
T13b6	glacer, empoisonner, ensanglanter, dorer, daller, poivrer, ...	0	19	6	2
T1Z05	retarder	1019	1	8	70
ZZ900	élargir	0	9	0	0
P1100	associer	32	1	2	11
T3400	démontrer	949	3	2	26
T11b6	revêtir, accoutrer	0	3	1	1
T1100	bénir, habiller, embaumer, baptiser, affectionner, harceler, ...	6782	38	26	208
T33b0	découper, parfumer, poudrer, transpercer, étoiler, tacheter	0	13	0	0
Z24	avaler	0	1	0	0
A35	intervenir	1296	2	2	105
T1Z30	replier	135	1	13	48
A41	pincer	281	3	1	25
P10b1	habiller, acquitter	0	3	0	0
T9300	écrouler	175	12	0	3
ZZ108	ressusciter	6	3	0	0
Z26	raser, forger, trousser, vermillonner	2	20	0	0
T11b8	enfler, dénuer, rançonner	73	4	1	13
T1701	grouper	11	1	4	0
ZZ308	améliorer, rôtir, durcir, rouiller, coaguler	9	11	1	0
Z1Z40	exiler	12	7	0	0
T3308	empourprer	36	3	2	2
A33	isoler	178	24	0	9
T1305	ajourner, anticiper	147	2	4	44
T13c0	enlacer	0	1	2	0
T33j0	engloutir, arquer, endiabler	0	3	0	0
A13	émerger	155	1	5	5
P9000	appesantir	87	1	0	4
Z1108	suffoquer, avorter	2	7	0	0
Z7001	accoster	6	2	0	0
T7308	investir	40	3	0	4
N1d	excéder	0	4	4	0
T32b0	enrager	0	2	0	0

'Frames des verbes au participe passé' page suivante

TABLE E. 6 – *Suite de la page précédente*

Frame	verbes	conjugué	vpp	vpa	infinitif
P80j0	masser, capitaliser	0	2	0	0
T3106	proportionner	6	1	1	1
Z1301	percher	0	2	0	0
P8001	superposer, déballer, échafauder	11	19	0	0
T1801	superposer	31	4	1	3
T31b6	pétrir	0	1	0	0
P9008	pétrifier, sanctifier, immortaliser, idéaliser, pacifier, tabouer	38	16	2	1
T11Z8	dérouter, démonter, débaucher	13	4	0	13
Z34	appareiller	0	3	0	0
T1808	mélanger, emmêler, entortiller	181	4	11	36
Z1101	cantonner	0	1	0	0
T1107	damner, enguirlander, morguer	97	5	6	28
Z3Z00	enfiévrer	6	3	0	0
T13g6	rabaisser	0	1	0	0
T13Z1	colporter	2	1	0	2
Z3Z06	obstiner	0	1	0	0
Z21	parquer, piper	0	5	0	0
A26	cadencer	20	2	0	5
T1200	cadencer, enfumer	7	6	0	7
T14a6	télégraphier	0	1	1	0
P30j6	diffuser	0	1	0	0
P3030	déchausser, extravaser	52	2	0	0
T1800	inventorier	49	1	1	6
TZ907	éreinter	9	2	2	14
T13l0	énoncer	0	1	1	0
ZZ306	boursoufler	1	2	0	5
P4000	avérer	978	3	0	0
T7300	quadriller	13	2	1	1
Z1006	étriquer	5	1	0	0

E. 4.5 235 frames des verbes à l'infinitif

TABLE E. 7 – Frames des verbes à l'infinitif (235 frames)

Frame	verbes	conj.	vpp	vpa	inf.
AZ2	être	142112	11	93	3018
T13b0	avoir, prendre, tenir, arracher, réclamer, effacer	2	90	24	30
T1Z06	avoir, aimer, commencer, chercher, manquer, jouer, ...	25652	183	509	4468
P10a0	faire, élever, entraîner, intéresser, déterminer, soumettre, ...	0	175	46	166
T1Zb0	faire, sauver, obtenir, exclure	3	2	1	29
TZ106	faire, former, constituer, bâtir, inquiéter, pénétrer, ...	11087	62	297	3679
T1Z0Z	voir, mettre, pousser, sentir, continuer, cacher, ...	8635	138	757	6041
T1Za6	prendre	0	0	0	68
Z1Z08	prendre	8	35	127	1536
TZ301	donner, mouiller, implanter	5070	21	83	936
TZZa0	donner, attacher	0	0	0	265
N3a	aller, profiter, correspondre, agréer, échoir	0	10	7	17
T1500	aller, songer	2107	0	7	5
N1a	trouver, croire, appeler, penser, apprendre, réfléchir, ...	12	87	75	690
TZZ00	trouver, savoir, connaître, regarder, élever, emporter, ...	20109	180	546	3201

'Frames des verbes à l'infinitif' page suivante

TABLE E. 7 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
T13a8	passer, abandonner, fixer, rattacher, accrocher, relier	0	35	21	84
T1Z40	passer, exiler, transférer, trimballer	1196	17	64	439
TZ3a0	dire, rappeler, réserver, procurer, transmettre, offrir	124	0	0	144
Z1500	dire	19	54	168	1278
P1007	vouloir, extasier, pâmer	56	41	5	16
T13a0	vouloir, laisser, reconnaître, apporter, causer, poser, ...	0	35	30	202
TZ500	vouloir, rougir	1237	5	8	16
T1Za8	mettre, demander, montrer, répondre, cacher, apprendre, ...	139	0	0	360
NZa	venir, manquer, toucher, ressembler, échapper, résister, ...	1084	0	0	825
T3500	venir, falloir	3467	12	4	15
AZ6	rester, paraître, sembler, revenir, rire, briller, ...	20762	176	88	1836
NZb	parler, jouir, profiter	0	0	0	81
T1306	parler, écrire, changer, écouler, répéter, augmenter, ...	7961	198	241	2175
T1Za0	porter, rendre, ouvrir, jeter, présenter, remettre, ...	223	0	0	418
TZ302	porter	1666	4	194	597
TZ30Z	rendre, traverser, couper, percer, contracter, piller, ...	2152	62	108	976
T1Z00	croire, entendre, attendre, servir, penser, reconnaître, ...	28199	310	971	6697
N4a	paraître, apparaître	10	3	1	8
P3001	laisser, révéler, exposer, reporter, hasarder, projeter, ...	1671	103	15	181
TZ306	laisser, suivre, prononcer, soutenir, inspirer, absorber, ...	4622	159	327	1599
A40	demander, importer, advenir	5472	16	4	37
Z1406	demander, expliquer	0	10	33	489
A96	devenir, redevenir, refouler, empester	6886	321	20	436
TZ900	tenir, lancer, arracher, creuser, élargir, affaiblir, ...	2859	58	133	699
ZZa	tenir	18	0	0	27
P1000	aimer, écouter, posséder, admirer, recueillir, blesser, ...	3721	316	17	470
T14b0	attendre	22	10	3	3
P1001	recevoir, contempler, enfermer, recruter, attabler, orienter	824	172	11	103
T13b8	recevoir	0	31	4	2
TZZ08	recevoir, perdre, frapper, conduire, battre, courir, ...	5620	276	279	3172
T110Z	tomber, amuser, utiliser, séduire, convoquer, ensorceler	3626	38	30	138
T130Z	ouvrir, retirer, terminer, presser, recommencer, baisser, ...	3727	220	190	1959
TZ100	jeter, quitter, occuper, finir, abandonner, travailler, ...	18144	433	541	4139
T1Z08	arrêter, retrouver, tuer, préparer, étendre, achever, ...	9028	396	345	3353
AZ5	arriver, persister, survenir, sommeiller	5593	97	74	593
TZ300	montrer, appeler, sortir, comprendre, présenter, garder, ...	44435	490	514	5097
T11a8	appeler, inviter, convoquer	13	2	0	29
N1b	répondre, vivre, descendre, décider, rire, disposer, ...	1	72	30	28
ZZ000	répondre	131	376	58	398
AZZ	partir, disparaître, subsister, voyager, échouer, plonger, ...	17794	38	87	776
P10b6	servir	0	21	0	4
AZ7	mourir, frémir, étinceler, errer, rôder, resplendir, ...	3636	0	12	496
T1Z07	reprendre, célébrer, consacrer, applaudir, discuter, ménager, ...	1119	60	54	636
T1Zb8	garder, tirer, distinguer, relever, laver, distraire	48	0	0	11
P10b0	occuper, permettre, étonner, couper, inquiéter, désespérer, ...	64	260	24	53
P3000	occuper, apercevoir, remplir, accompagner, éprouver, remarquer, ...	17611	950	45	1041
TZZ06	tirer, posséder, épouser, découvrir, marquer, attaquer, ...	6413	84	85	1287
A1b	jouer	0	6	5	9
Z1400	rappeler, remettre, charger, représenter, permettre, revoir, ...	0	17	97	676

'Frames des verbes à l'infinitif' page suivante

TABLE E. 7 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
T1300	vivre, réussir, habiter, crier, accomplir, mériter, ...	58333	97	138	1304
TZ800	rencontrer, rejoindre, dépouiller	1652	1	17	466
TZZ07	descendre, allumer, déchirer, incendier, charrier	494	49	42	153
T11a0	conduire, entraîner, intéresser, condamner, déterminer, abaisser	0	35	14	55
Z1Z00	envoyer, payer, figurer, taper, cogner	5	17	30	387
Z4000	agir	14291	0	25	253
T1308	tourner, cesser, attacher, engager, obtenir, interrompre, ...	5343	328	210	1819
P30a0	apporter, attribuer, procurer, assigner, administrer	0	119	54	19
T1340	apporter	1244	10	25	95
TZZ0Z	toucher, gagner, saisir, retenir, coucher, rouler, ...	4132	50	165	1036
T14a0	lire, glisser, éviter, épargner	0	0	19	18
PZ0b0	approcher, emparer	372	0	0	24
T1ZZ0	approcher, emmener, embarquer	231	4	29	235
T3300	annoncer, exprimer, prouver, révéler, témoigner, renfermer, ...	14882	81	95	559
T1Z01	établir, remarquer, refuser, deviner, recueillir, désigner, ...	2182	128	98	1187
T1ZbZ	charger, couvrir, envelopper	0	0	1	7
AZ0	suffire, apparaître, exister, dormir, périr, céder, ...	25236	14	30	904
ZZ0a0	suffire	560	0	0	28
T13Z6	rentrer	160	1	11	30
AZ1	naître, régner, fonctionner, séjourner, pulluler, trôner, ...	17015	200	44	244
T1ZaZ	préparer, retirer, adresser, proposer	61	0	0	63
T1Z02	amener, réunir, renvoyer, rapatrier	423	12	31	354
TZ1a0	amener, associer, rallier, habituer, accoutumer, allier	165	0	0	75
P3008	expliquer, démontrer, définir, transmettre, copier, alimenter, ...	1676	257	28	76
T3306	valoir	877	7	2	32
T33a0	valoir	3	3	1	1
T1908	défendre, peindre, aider, caractériser, secourir, appesantir, ...	1089	34	26	691
A30	cesser, résister, retentir, éviter, prêter, débuter, ...	9895	45	15	219
P8000	distinguer, accorder, marier, opposer, séparer, rattacher, ...	2064	165	11	127
T13a6	payer, acheter	0	11	4	7
TZ200	monter, couvrir, délivrer	1468	5	12	212
TZ400	souffrir, supposer	123	7	9	80
A31	couvrir, poindre, tourbillonner, germer, progresser	1448	4	2	54
A16	songer, hésiter, jouir, témoigner, douter, déterminer, ...	9029	157	137	525
TZ101	fixer, installer, submerger	612	16	7	145
T15a8	permettre, réclamer	0	0	0	5
T1108	employer, rassurer, asseoir, assassiner, décourager, saigner, ...	1944	156	22	764
P7000	ressembler, provoquer, défier, remplacer, étreindre, affronter	890	9	6	103
T1ZZ8	relever, expédier	250	8	18	179
PZ030	échapper	1188	44	7	48
T1106	nommer, obéir, marier, veiller, enseigner, conseiller, ...	2069	268	55	388
T11b0	assurer, traiter, plaindre, entretenir, louer, délivrer	134	38	16	25
TZZ05	poursuivre	448	0	28	105
A10	espérer, subir, prier, réclamer, consentir, boire, ...	12853	125	54	811
T1400	espérer, empêcher, souhaiter, envisager, excuser, interdire	3829	33	53	326
A35	durer, discontinuer, échoir	1296	2	2	105
T14a8	exprimer, raconter, objecter	0	6	2	19
PZ0bZ	mêler	0	0	0	1
T13b6	mêler	0	19	6	2

'Frames des verbes à l'infinitif' page suivante

TABLE E. 7 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
T1808	mêler, emmêler	181	4	11	36
T3Z00	indiquer, élancer, irriter, ravager, dénoter, assoupir	4502	79	28	157
P1020	ramener, risquer, réfugier, égarer	398	38	4	83
TZ308	contenir, réveiller, supporter, arranger, déployer, confirmer, ...	1387	99	104	1069
P1Z00	imaginer	755	1	5	129
T1100	prier, assister, invoquer, bénir, confesser, habiller, ...	6782	38	26	208
P1050	prouver, persuader	113	4	0	20
P1006	livrer, plaindre, excuser, débattre, qualifier, contredire	1729	186	34	261
T13Z8	rapporter, glisser, remporter	625	2	14	96
P3006	vendre, abattre, accroître, décrire, louer, apprécier, ...	6979	623	37	234
Z73a0	arracher	0	0	0	53
TZ108	intéresser, enivrer, révolter, ressusciter, corrompre, mûrir, ...	316	29	13	145
TZZ01	parcourir, lâcher, allonger, viser, faucher	640	6	36	145
A12	accourir, atterrir, abouler	510	8	1	27
A1Z	parvenir, succomber, émigrer, tâtonner, flâner, délirer	1196	57	27	76
A36	profiter, coûter, fermenter, végéter, décroître, renchérir	4717	52	10	67
P1040	transporter	136	13	0	7
T1101	joindre, enfermer, évoquer, baser, cantonner, inhumer	7997	29	40	122
T1900	plaindre, redouter, dédaigner, révérer, courtiser, lutiner	3046	25	22	177
T1ZZZ	traîner, rejeter	206	2	41	68
T1305	reculer, différer, ajourner, perpétuer, proroger	147	2	4	44
T1301	exposer, publier, propager, aventurer, interposer, récolter, ...	560	39	11	162
TZ906	dévorer	1569	1	6	44
P1500	douter	214	0	0	1
TZ1b0	détourner, détacher, séparer, récompenser, débarrasser, munir	120	0	0	18
A70	lutter, différer, sympathiser, fraterniser, transiger, divorcer, ...	1021	3	2	218
N9a	aider	0	2	26	8
T1Z20	introduire, cracher, remorquer, piloter	272	3	17	106
TZ908	user, calmer, enlacer	172	11	1	156
T11b8	protéger, prévenir, avertir, priver, affranchir, dégoûter	73	4	1	13
T1102	inviter, assigner, embaucher, réassigner	233	8	6	32
TZ10Z	précéder, charmer, fasciner, gifler	126	12	20	37
TZZ20	dégager	36	5	6	62
TZZb0	dégager, libérer	24	0	0	4
T1907	louer, blâmer, féliciter, plaisanter, railler, calomnier, ...	546	22	14	134
AZ4	voler, rétrograder	7441	0	0	35
PZ00Z	taire	8512	22	6	10
T1328	flotter	18	0	0	6
Z7300	disputer, contester	0	2	2	44
PZ001	plaire, déplaire, tapir	8151	259	40	165
P1400	promettre, proposer, appuyer	989	34	29	104
PZ005	enfuir, envoler, égrener	1438	6	2	67
A11	enseigner, guetter, résider, cueillir, comparaître, récolter, ...	1617	36	27	97
P3005	déclarer, démentir	1041	12	0	3
TZ107	accabler, consoler	168	14	1	135
P1300	attribuer, pardonner, procurer, concilier, approprier, infliger	1119	3	6	160
TZ701	distribuer	70	3	6	23
T3100	émouvoir, fâcher, préoccuper, tremper, réjouir, affliger, ...	4499	223	7	287
T1320	échouer, reporter, convoyer	16	4	4	25

'Frames des verbes à l'infinitif' page suivante

TABLE E. 7 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
T1307	punir, commenter, redire	191	7	5	64
T1130	débarquer, déménager, congédier, déloger	38	0	2	17
A20	délivrer, paître, grogner, grommeler, caqueter, meugler, ...	250	6	13	19
T3900	concerner	740	2	38	1
T1107	gronder, damner, réprimander, enguirlander, sermonner, chambrer	97	5	6	28
Z7000	mentir, téléphoner	824	8	78	54
TZZa8	communiquer	9	0	0	14
ZZ0	communiquer	9	300	42	22
P2000	couronner	46	0	0	1
A3Z	luire, sourdre	1615	5	3	27
T1Z05	retarder, devancer, reconduire	1019	1	8	70
T1506	tâcher	185	0	0	1
T1Z30	vider, replier, redescendre, flanquer	135	1	13	48
N3b	résulter	205	35	37	1
P1100	associer	32	1	2	11
T3400	démontrer, impliquer	949	3	2	26
P7001	rallier, côtoyer, masser, ameuter, attrouper	222	98	3	9
A86	correspondre, coïncider, cadrer, rimer	143	0	0	16
A41	pleuvoir, neiger	281	3	1	25
A33	ruisseler, isoler, refluer, rejaillir	178	24	0	9
P1008	enivrer, immoler, suicider	63	23	1	11
PZ006	emprunter, ébrouer	9441	314	44	12
Z1030	défiler, trotter	7	6	29	14
T1920	acheminer	24	0	0	10
T9300	écrouler, suer	175	12	0	3
Z1308	gratter, disloquer	1	5	3	12
Z14b0	aviser	0	0	0	1
TZ506	priver	0	0	0	13
Z3000	prêcher, déménager, radoter	283	83	27	20
T9106	impatienter, abrutir	38	0	0	14
T3308	empourprer	36	3	2	2
TZZbZ	isoler	0	0	0	1
PZ000	raconter, mélanger	13904	18	1	29
T11Z0	escorter	62	0	3	7
TZ907	flétrir, éreinter	9	2	2	14
ZZ007	languir	0	0	0	2
AZ3	émerger	192	0	1	2
P9000	appesantir	87	1	0	4
P1030	évader, extraire, extirper, désarçonner	74	11	5	41
T1208	pêcher	36	0	5	7
T1138	extraire, proscrire, extirper	48	4	8	29
T1801	amonceler, déballer	31	4	1	3
A37	flamboyer, vaguer	340	13	6	2
T7308	investir	40	3	0	4
T1700	énumérer, dénombrer	143	0	2	16
T2300	paître, brouter	24	0	0	3
T3106	proportionner	6	1	1	1
T91a0	assimiler	0	0	0	2
T11Z8	dérouter, débaucher	13	4	0	13

'Frames des verbes à l'infinitif' page suivante

TABLE E. 7 – *Suite de la page précédente*

Frame	verbes	conj.	vpp	vpa	inf.
P9008	enhardir	38	16	2	1
A17	grimacer, trépasser	47	0	1	2
T190Z	symboliser, égratigner, parodier	35	0	6	5
T9101	recoucher	6	0	0	4
T1901	dénicher	1681	8	27	2
A13	dévier, décamper, déguerpir	155	1	5	5
P2030	départir	26	0	0	2
A80	concorder, converger, coexister	28	0	0	8
T1200	acculer, enfumer	7	6	0	7
P10a8	cramponner	0	11	10	2
T13Z1	colporter	2	1	0	2
T1800	émietter, récapituler, inventorier, morceler, compter	49	1	1	6
ZZ306	désorganiser	1	2	0	5
T1Z38	déraciner	5	9	1	6
P10a6	adonner	0	17	4	6
T11b6	vêtir	0	3	1	1
A26	encenser, regimber, suifer	20	2	0	5
P30b0	déduire	0	231	12	1
A21	outrepasser	9	0	1	1
T7300	coloniser	13	2	1	1
A24	décocher	13	0	0	1
P2006	pelotonner	7	0	1	1
T11g0	informer	0	19	2	1
P2001	bauger	0	0	0	1

E. 5 Part Of Speech

E. 5.1 Grew

Le symbole "|" marque les regroupements d'étiquettes créés lors de l'alignement.

```
['ADJ, ADJWH, ADV, ADVWH, CC, CLO, CLR, CLS, CS, DET, DETWH, ET, I, NC,
NPP, P, P+D, P+PRO, PONCT, PREF, PRO, PROREL, PROWH, V, VIMP, VINF, VPP,
 VPR, VS']
```

TABLE E. 8 – part of speech Grew

Grew	Stanford	Talismane
ADJ	NOUN, 610900, 10.23 % ADJ, 4328779, 72.48 % *autres (#48)* : 17.3%	ADJ, 5290926, 88.58 % *autres (#67)* : 11.42%
ADJWH	DET, 9563, 49.31 % ADJ, 3981, 20.53 % PRON, 4850, 25.01 % *autres (#1)* : 5.16%	DETWH, 19333, 100.0 %
ADV	ADV, 3709950, 67.71 % PART, 1117110, 20.39 % *autres (#86)* : 11.91%	ADV, 4900380, 89.42 % *autres (#78)* : 10.58%
ADVWH	SCONJ, 6318, 12.83 % ADV, 41266, 83.83 % *autres (#1)* : 3.33%	ADVWH, 40549, 82.45 % *autres (#6)* : 17.55%
CC	CONJ, 2875114, 95.86 % *autres (#16)* : 4.14%	CC, 2848424, 94.97 % *autres (#15)* : 5.03%
CLO	PRON, 1738962, 82.66 % *autres (#22)* : 17.34%	CLO, 1981711, 94.2 % *autres (#24)* : 5.8%
CLR	PRON, 1106063, 95.82 % *autres (#6)* : 4.18%	CLR, 1095236, 94.89 % *autres (#9)* : 5.11%
CLS	PRON, 4113819, 99.06 % *autres (#22)* : 0.94%	CLS, 3944196, 94.98 % *autres (#28)* : 5.02%
CS	SCONJ, 1166451, 65.32 % PRON, 185071, 10.36 % *autres (#37)* : 24.32%	CS, 1415280, 79.25 % *autres (#46)* : 20.75%
DET	DET, 12113481, 92.29 % *autres (#49)* : 7.71%	DET, 12697515, 96.75 % *autres (#43)* : 3.25%
DETWH	DET, 10420, 66.53 % PRON, 4762, 30.4 % *autres (#1)* : 3.06%	DETWH, 15631, 100.0 %
ET	PROPN, 36618, 21.2 %	NPP, 64747, 37.64 %

TABLE E. 8 – *Part of Speech pour Grew, suite sur la prochaine page*

TABLE E. 8 – *Part of Speech pour Grew, suite sur la prochaine page*

Grew	Stanford	Talismane
	ADJ, 18029, 10.44 % NOUN, 41109, 23.8 % VERB, 34351, 19.89 % *autres (#27)* : 24.67%	ADV, 21736, 12.64 % NC, 43945, 25.55 % *autres (#35)* : 24.17%
I	VERB, 2364, 16.31 % INTJ, 2601, 17.94 % NOUN, 4941, 34.08 % *autres (#8)* : 31.67%	ADV, 5206, 36.15 % I, 2642, 18.34 % NC, 3400, 23.61 % *autres (#3)* : 21.91%
NC	NOUN, 16052513, 93.61 % *autres (#60)* : 6.39%	NC, 16427922, 95.8 % *autres (#78)* : 4.2%
NPP	PROPN, 2350277, 90.65 % *autres (#42)* : 9.35%	NPP, 2353765, 90.8 % *autres (#65)* : 9.2%
P	ADP, 10812816, 94.48 % *autres (#64)* : 5.52%	P, 11072600, 96.75 % *autres (#78)* : 3.25%
P+D	ADP, DET, 1098087, 62.0 % ADP, 481125, 27.17 % *autres (#33)* : 10.83%	P+D, 1659688, 93.7 % *autres (#33)* : 6.3%
P+PRO	NOUN, 10219, 35.38 % ADJ, 14003, 48.48 % *autres (#6)* : 16.13%	P+PRO, 28505, 99.5 % *autres (#1)* : 0.5%
PONCT	PUNCT, 11922461, 99.99 % *autres (#9)* : 0.01%	PONCT, 11922630, 99.99 % *autres (#7)* : 0.01%
PREF	PRON, 965, 63.11 % NOUN, 287, 18.77 % *autres (#2)* : 18.12%	PRO, PONCT, ADJ, 338, 33.3 % VPP, 234, 23.05 % PRO, PONCT, ADV, 248, 24.43 % ADV, PONCT, ADJ, 103, 10.15 % *autres (#1)* : 9.06%
PRO	PRON, 1109516, 83.72 % *autres (#31)* : 16.28%	PRO, 1083967, 81.82 % *autres (#47)* : 18.18%
PROREL	PRON, 1536373, 90.07 % *autres (#9)* : 9.93%	PROREL, 1638349, 96.04 % *autres (#14)* : 3.96%
PROWH	PROPN, 4467, 13.61 % PRON, 26731, 81.45 % *autres (#3)* : 4.94%	PROWH, 19451, 59.6 % NPP, 5142, 15.76 % PROREL, 6337, 19.42 % *autres (#5)* : 5.23%
V	VERB, 6207106, 72.1 % AUX, 2069809, 24.04 % *autres (#18)* : 3.86%	V, 8359997, 97.11 % *autres (#22)* : 2.89%
VIMP	VERB, 53542, 86.1 % *autres (#5)* : 13.9%	VIMP, 19556, 31.33 % V, 37142, 59.51 %

TABLE E. 8 – *Part of Speech pour Grew, suite sur la prochaine page*

TABLE E. 8 – *Part of Speech pour Grew, suite sur la prochaine page*

Grew	Stanford	Talismane
		autres (#5) : 9.15%
VINF	VERB, 2626947, 90.92 % *autres (#15)* : 9.08%	VINF, 2843456, 98.43 % *autres (#19)* : 1.57%
VPP	VERB, 2337720, 84.04 % *autres (#24)* : 15.96%	VPP, 2636522, 94.79 % *autres (#25)* : 5.21%
VPR	VERB, 443284, 82.56 % *autres (#7)* : 17.44%	VPR, 490623, 91.46 % *autres (#9)* : 8.54%
VS	AUX, 96899, 35.05 % VERB, 151863, 54.94 % *autres (#8)* : 10.01%	VS, 220570, 79.83 % *autres (#8)* : 20.17%

Grew offre 29 étiquettes différentes.

Stanford dispersion : Pour 29 POS différents : 47 occurrences (moyenne 1.62).

Talismane Stanford pour Grew : moyenne : 88.85%.

Talismane dispersion : Pour 29 POS différents : 39 occurrences (moyenne 1.34).

Couverture Talismane pour Grew : moyenne : 91.79%.

E. 5.2 Stanford

Le symbole "|" marque les regroupements d'étiquettes créés lors de l'alignement.

```
['ADJ, ADP, ADV, AUX, CONJ, DET, INTJ, NOUN, NUM, PART, PRON, PROPN,
PUNCT, SCONJ, SYM, VERB, X']
```

TABLE E. 9 – Part of Speech Stanford

Stanford	Grew	Talismane
ADJ	ADJ, 4328779, 86.52 % *autres (#25)* : 13.48%	ADJ, 4238665, 84.73 % *autres (#35)* : 15.27%
ADP	P, 10812816, 92.38 % *autres (#22)* : 7.62%	P, 10770814, 92.02 % *autres (#33)* : 7.98%
ADV	ADV, 3709950, 86.2 % *autres (#25)* : 13.8%	ADV, 3613502, 83.98 % *autres (#45)* : 16.02%
AUX	V, 2069809, 81.45 % *autres (#20)* : 18.55%	V, 2060195, 81.07 % *autres (#17)* : 18.93%

TABLE E. 9 – *Part of Speech pour Stanford, suite sur la prochaine page*

TABLE E. 9 – *Part of Speech pour Stanford, suite sur la prochaine page*

Stanford	Grew	Talismane
CONJ	CC, 2875114, 99.57 % *autres (#6)* : 0.43%	CC, 2794215, 96.77 % *autres (#9)* : 3.23%
DET	DET, 12113481, 92.87 % *autres (#18)* : 7.13%	DET, 12103641, 92.8 % *autres (#16)* : 7.2%
INTJ	ADV, 19917, 59.41 % V, 4779, 14.26 % *autres (#7)* : 26.33%	ADV, 23138, 69.17 % V, 6080, 18.18 % *autres (#4)* : 12.66%
NOUN	NC, 16052513, 91.66 % *autres (#27)* : 8.34%	NC, 15957229, 91.12 % *autres (#64)* : 8.88%
NUM	DET, 406584, 56.29 % ADJ, 205106, 28.4 % *autres (#11)* : 15.31%	DET, 293555, 40.64 % ADJ, 242990, 33.64 % NC, 118059, 16.34 % *autres (#16)* : 9.39%
PART	ADV, 1117110, 96.79 % *autres (#9)* : 3.21%	ADV, 1115705, 96.68 % *autres (#8)* : 3.32%
PRON	CLS, 4113819, 40.1 % CLO, 1738962, 16.95 % PROREL, 1536373, 14.98 % CLR, 1106063, 10.78 % PRO, 1109516, 10.82 % *autres (#28)* : 6.37%	CLS, 3961941, 38.62 % CLO, 1741823, 16.98 % PROREL, 1569837, 15.3 % PRO, 1083607, 10.56 % CLR, 1100619, 10.73 % *autres (#44)* : 7.8%
PROPN	NPP, 2350277, 78.01 % NC, 392804, 13.04 % *autres (#32)* : 8.95%	NPP, 2340959, 77.72 % NC, 403507, 13.4 % *autres (#50)* : 8.88%
PUNCT	PONCT, 11922461, 99.92 % *autres (#15)* : 0.08%	PONCT, 11938407, 99.99 % *autres (#7)* : 0.01%
SCONJ	CS, 1166451, 78.72 % *autres (#19)* : 21.28%	CS, 1132564, 76.43 % *autres (#21)* : 23.57%
SYM	DET, 11817, 16.79 % NC, 8499, 12.07 % CC, 46525, 66.1 % *autres (#8)* : 5.04%	DET, 11876, 16.89 % PONCT, 52074, 74.05 % *autres (#5)* : 9.06%
VERB	V, 6207106, 49.95 % VINF, 2626947, 21.14 % VPP, 2337720, 18.81 % *autres (#21)* : 10.09%	V, 6161972, 49.59 % VINF, 2610916, 21.01 % VPP, 2362326, 19.01 % *autres (#22)* : 10.38%
X	NC, 26354, 38.09 % V, 7028, 10.16 % NPP, 8497, 12.28 % ET, 8860, 12.8 %	NC, 23497, 34.05 % V, 7075, 10.25 % NPP, 17691, 25.64 % *autres (#17)* : 30.05%

TABLE E. 9 – *Part of Speech pour Stanford, suite sur la prochaine page*

TABLE E. 9 – *Part of Speech pour Stanford, suite sur la prochaine page*

Stanford	Grew	Talismane
	autres (#19) : 26.67%	

Stanford fournit 17 étiquettes différentes.

Grew dispersion : Pour 17 POS différents : 31 occurrences (moyenne 1.82).

Couverture Stanford pour Grew : moyenne : 88.66%.

Talismane dispersion : Pour 17 POS différents : 30 occurrences (moyenne 1.76).

Couverture Talismane pour Stanford : moyenne : 88.67%.

E. 5.3 Talismane

Le symbole "|" marque les regroupements d'étiquettes créés lors de l'alignement.

```
['ADJ, ADV, ADVWH, CC, CLO, CLR, CLS, CS, DET, DETWH, ET, I, NC, NPP, P,
 P+D, P+PRO, PONCT, PRO, PROREL, PROWH, V, VIMP, VINF, VPP, VPR, VS']
```

TABLE E. 10 – Part of Speech Talismane

Talismane	Grew	Stanford
ADJ	ADJ, 5290926, 90.17 % *autres (#25)* : 9.83%	NOUN, 605094, 10.31 % ADJ, 4238665, 72.23 % *autres (#19)* : 17.45%
ADV	ADV, 4900380, 93.71 % *autres (#24)* : 6.29%	ADV, 3613502, 69.1 % PART, 1115705, 21.34 % *autres (#23)* : 9.56%
ADVWH	ADVWH, 40549, 76.72 % ADV, 6449, 12.2 % *autres (#3)* : 11.07%	SCONJ, 6027, 11.39 % ADV, 42124, 79.61 % *autres (#1)* : 9.0%
CC	CC, 2848424, 98.6 % *autres (#7)* : 1.4%	CONJ, 2794215, 96.72 % *autres (#9)* : 3.28%
CLO	CLO, 1981711, 95.09 % *autres (#8)* : 4.91%	PRON, 1741823, 83.58 % *autres (#7)* : 16.42%
CLR	CLR, 1095236, 95.96 % *autres (#3)* : 4.04%	PRON, 1100619, 96.43 % *autres (#2)* : 3.57%
CLS	CLS, 3944196, 99.29 % *autres (#5)* : 0.71%	PRON, 3961941, 99.73 % *autres (#5)* : 0.27%

TABLE E. 10 – *Part of Speech pour Talismane, suite sur la prochaine page*

TABLE E. 10 – *Part of Speech pour Talismane, suite sur la prochaine page*

Talismane	Grew	Stanford
CS	CS, 1415280, 84.54 % *autres (#7)* : 15.46%	SCONJ, 1132564, 67.65 % *autres (#13)* : 32.35%
DET	DET, 12697515, 98.44 % *autres (#12)* : 1.56%	DET, 12103641, 93.84 % *autres (#16)* : 6.16%
DETWH	ADJWH, 19333, 38.28 % DET, 15443, 30.57 % DETWH, 15631, 30.95 % *autres (#1)* : 0.2%	DET, 33415, 66.07 % ADJ, 6276, 12.41 % PRON, 9935, 19.64 % *autres (#1)* : 1.88%
ET	V, 3875, 19.05 % ET, 9979, 49.05 % *autres (#11)* : 31.9%	VERB, 8262, 40.59 % PROPN, 2084, 10.24 % NOUN, 3961, 19.46 % ADJ, 2052, 10.08 % *autres (#7)* : 19.62%
I	NC, 1202, 18.98 % I, 2642, 41.72 % NPP, 692, 10.93 % VINF, 859, 13.57 % *autres (#5)* : 14.8%	NOUN, 3275, 52.34 % PROPN, 980, 15.66 % VERB, 1288, 20.58 % *autres (#3)* : 11.41%
NC	NC, 16427922, 94.89 % *autres (#26)* : 5.11%	NOUN, 15957229, 92.18 % *autres (#24)* : 7.82%
NPP	NPP, 2353765, 83.76 % *autres (#24)* : 16.24%	PROPN, 2340959, 83.35 % *autres (#20)* : 16.65%
P	P, 11072600, 99.3 % *autres (#14)* : 0.7%	ADP, 10770814, 96.6 % *autres (#22)* : 3.4%
P+D	P+D, 1659688, 97.33 % *autres (#2)* : 2.67%	ADP, DET, 1094200, 64.17 % ADP, 477493, 28.0 % *autres (#11)* : 7.83%
P+PRO	P+PRO, 28505, 100.0 %	NOUN, 10159, 35.63 % ADJ, 13773, 48.3 % *autres (#6)* : 16.07%
PONCT	PONCT, 11922630, 99.49 % *autres (#16)* : 0.51%	PUNCT, 11938407, 99.56 % *autres (#4)* : 0.44%
PRO	PRO, 1083967, 89.28 % *autres (#10)* : 10.72%	PRON, 1083607, 89.23 % *autres (#7)* : 10.77%
PROREL	PROREL, 1638349, 92.16 % *autres (#5)* : 7.84%	PRON, 1569837, 88.29 % *autres (#5)* : 11.71%
PROWH	PROWH, 19451, 57.94 % ADV, 5920, 17.64 % PROREL, 4936, 14.7 % *autres (#2)* : 9.72%	PRON, 30019, 89.5 % *autres (#3)* : 10.5%

TABLE E. 10 – *Part of Speech pour Talismane, suite sur la prochaine page*

TABLE E. 10 – *Part of Speech pour Talismane, suite sur la prochaine page*

Talismane	Grew	Stanford
V	V, 8359997, 97.52 % *autres (#22)* : 2.48%	VERB, 6161972, 71.88 % AUX, 2060195, 24.03 % *autres (#13)* : 4.09%
VIMP	VIMP, 19556, 61.64 % V, 9034, 28.47 % *autres (#8)* : 9.89%	VERB, 27759, 87.8 % *autres (#4)* : 12.2%
VINF	VINF, 2843456, 99.06 % *autres (#13)* : 0.94%	VERB, 2610916, 90.95 % *autres (#12)* : 9.05%
VPP	VPP, 2636522, 92.71 % *autres (#16)* : 7.29%	VERB, 2362326, 83.07 % *autres (#12)* : 16.93%
VPR	VPR, 490623, 95.51 % *autres (#9)* : 4.49%	VERB, 426525, 83.01 % *autres (#7)* : 16.99%
VS	VS, 220570, 90.86 % *autres (#8)* : 9.14%	AUX, 93223, 38.34 % VERB, 125026, 51.42 % *autres (#8)* : 10.24%

Talismane fournit 27 étiquettes différentes.

Grew dispersion : Pour 27 POS différents : 37 occurrences (moyenne 1.37).

Couverture Grew pour Talismane : moyenne : 92.97%.

Stanford dispersion : Pour 27 POS différents : 41 occurrences (moyenne 1.52).

Couverture Stanford pour Talismane : moyenne : 89.42%.

E. 6 Dépendances

E. 6.1 Grew

Le symbole "|" marque les regroupements d'étiquettes créés lors de l'alignement.

```
['_, a_obj, aff, arg, arg.comp, ato, ats, aux.caus, aux.mod, aux.pass,
aux.tps, coord, de_obj, dep, dep.coord, det, dis, mod, mod.app, mod.
cleft, mod.inc, mod.rel, mod.voc, mwe, obj, obj.cpl, obj.p, p_obj.agt,
p_obj.o, ponct']
```

TABLE E. 11 – dépendances Grew

Grew	Stanford	Talismane
_	root, 2244499, 17.03 % punct, 1604506, 12.17 % *autres (#205)* : 70.8%	mod, 3560384, 26.99 % root, 3012491, 22.83 % ponct, 1653035, 12.53 % *autres (#144)* : 37.65%
a_obj	iobj, 452241, 37.74 % case, 274238, 22.88 % dobj, 186978, 15.6 % mark, 122089, 10.19 % *autres (#25)* : 13.59%	mod, 482456, 40.24 % aff, 250149, 20.86 % a_obj, 183767, 15.33 % obj, 182510, 15.22 % *autres (#16)* : 8.34%
aff	dobj, 364374, 95.15 % *autres (#17)* : 4.85%	aff, 347292, 90.65 % *autres (#10)* : 9.35%
arg	case, 8594, 92.73 % *autres (#1)* : 7.27%	mod, 4361, 47.07 % dep, 4246, 45.83 % *autres (#2)* : 7.09%
ato	xcomp, 69070, 39.17 % amod, 38051, 21.58 % *autres (#25)* : 39.26%	mod, 79775, 45.16 % obj, 43269, 24.49 % ats, 21392, 12.11 % *autres (#15)* : 18.24%
ats	root, 283580, 26.38 % conj, 141210, 13.13 % *autres (#47)* : 60.49%	obj, 397600, 36.93 % ats, 392331, 36.44 % mod, 149734, 13.91 % *autres (#33)* : 12.72%
aux.caus	root, 38514, 23.18 % conj, 34578, 20.81 % acl, 28308, 17.04 % acl :relcl, 21781, 13.11 % advcl, 17374, 10.46 % *autres (#11)* : 15.39%	aux_caus, 46958, 28.28 % root, 30556, 18.4 % _, 17421, 10.49 % *autres (#8)* : 42.84%
aux.mod	aux, 354, 100.0 %	obj, 199, 59.76 % prep, 134, 40.24 %
aux.pass	auxpass, 249245, 54.84 % aux, 111314, 24.49 % cop, 74279, 16.34 % *autres (#16)* : 4.33%	aux_pass, 271656, 59.74 % aux_tps, 129228, 28.42 % *autres (#13)* : 11.84%
aux.tps	aux, 1044761, 90.83 % *autres (#24)* : 9.17%	aux_tps, 989890, 86.05 % *autres (#16)* : 13.95%
coord	cc, 1929989, 95.19 % *autres (#19)* : 4.81%	coord, 1883988, 92.91 % *autres (#12)* : 7.09%
de_obj	case, 317634, 53.44 %	mod, 482050, 81.09 %

TABLE E. 11 – *Dépendances pour Grew, suite sur la prochaine page*

TABLE E. 11 – *Dépendances pour Grew, suite sur la prochaine page*

Grew	Stanford	Talismane
	mark, 115590, 19.45 % *autres (#23)* : 27.11%	*autres (#18)* : 18.91%
dep	case, 4095017, 77.86 % *autres (#79)* : 22.14%	mod, 2793113, 53.09 % dep, 2132987, 40.55 % *autres (#53)* : 6.36%
dep.coord	conj, 1662197, 58.86 % case, 390086, 13.81 % *autres (#61)* : 27.33%	dep_coord, 2181378, 77.22 % *autres (#54)* : 22.78%
dep.coord|mod.inc	conj, 232, 100.0 %	sub, 242, 100.0 %
dep.coord|mod.rel	conj, 2034, 66.64 % acl :relcl, 405, 13.27 % aux, 375, 12.29 % *autres (#1)* : 7.8%	dep_coord, 2825, 92.35 % *autres (#2)* : 7.65%
det	det, 9936121, 77.26 % nmod :poss, 2105792, 16.37 % *autres (#73)* : 6.37%	det, 12474849, 96.99 % *autres (#49)* : 3.01%
dis	mark, 4521, 89.28 % *autres (#2)* : 10.72%	mod, 4531, 87.78 % *autres (#4)* : 12.22%
mod	amod, 3343009, 18.67 % case, 2960586, 16.53 % advmod, 2795595, 15.61 % *autres (#279)* : 49.18%	mod, 14938931, 83.35 % *autres (#155)* : 16.65%
mod.app	appos, 43366, 40.25 % conj, 38892, 36.1 % *autres (#22)* : 23.65%	mod, 95621, 88.16 % *autres (#15)* : 11.84%
mod.cleft	acl :relcl, 22884, 73.82 % *autres (#13)* : 26.18%	mod_rel, 23233, 75.16 % mod, 4717, 15.26 % *autres (#6)* : 9.58%
mod.inc	conj, 20280, 44.37 % acl, 5713, 12.5 % *autres (#16)* : 43.13%	mod, 35356, 77.06 % *autres (#10)* : 22.94%
mod.rel	acl :relcl, 848259, 67.55 % *autres (#29)* : 32.45%	mod_rel, 855698, 68.15 % mod, 179724, 14.31 % *autres (#19)* : 17.53%
mod.voc	nsubj, 627, 35.67 % nmod, 570, 32.42 % root, 222, 12.63 % appos, 197, 11.21 % *autres (#1)* : 8.08%	mod, 933, 52.86 % suj, 597, 33.82 % *autres (#2)* : 13.31%
mwe	nmod, 573, 34.79 %	mod, 782, 39.3 %

TABLE E. 11 – *Dépendances pour Grew, suite sur la prochaine page*

TABLE E. 11 – *Dépendances pour Grew, suite sur la prochaine page*

Grew	Stanford	Talismane
	case, 532, 32.3 % mark, 420, 25.5 % *autres (#1)* : 7.41%	prep, 648, 32.56 % obj, 291, 14.62 % *autres (#2)* : 13.52%
obj	dobj, 3716155, 64.65 % *autres (#63)* : 35.35%	obj, 4332951, 75.36 % *autres (#48)* : 24.64%
obj.cpl	ccomp, 333941, 21.64 % advcl, 332949, 21.57 % aux, 169678, 10.99 % conj, 156945, 10.17 % *autres (#28)* : 35.62%	sub, 1076650, 69.77 % *autres (#22)* : 30.23%
obj.p	nmod, 9792461, 75.07 % *autres (#62)* : 24.93%	prep, 12286066, 94.17 % *autres (#53)* : 5.83%
p_obj.agt	case, 173265, 98.47 % *autres (#4)* : 1.53%	mod, 168941, 95.96 % *autres (#8)* : 4.04%
p_obj.o	case, 545700, 67.94 % *autres (#36)* : 32.06%	mod, 637033, 79.3 % obj, 113207, 14.09 % *autres (#25)* : 6.6%
ponct	punct, 10226010, 99.98 % *autres (#8)* : 0.02%	ponct, 10224626, 99.97 % *autres (#8)* : 0.03%
ponct|ponct	punct, 10471, 100.0 %	ponct, 10472, 100.0 %
suj	nsubj, 6521773, 83.06 % *autres (#66)* : 16.94%	suj, 6692626, 85.23 % *autres (#60)* : 14.77%
suj|suj	nsubj, 330, 100.0 %	suj, 303, 100.0 %

Grew fournit 30 étiquettes différentes (hors regroupement créé lors de l'alignement).

Stanford dispersion : Pour 34 DEP différentes : 63 occurrences (moyenne : 1.85).

Talismane Stanford pour Grew : moyenne : 80.35%.

Talismane dispersion : Pour 36 DEP différentes : 57 occurrences (moyenne : 1.58).

Couverture Talismane pour Grew : moyenne : 88.01%.

E. 6.2 Stanford

Le symbole "|" marque les regroupements d'étiquettes créés lors de l'alignement.

```
['acl, acl:relcl, advcl, advmod, amod, appos, aux, auxpass, case, cc,
ccomp, compound, conj, cop, csubj, dep, det, discourse, dislocated, dobj
, expl, goeswith, iobj, mark, mwe, name, neg, nmod, nmod:poss, nsubj,
nsubjpass, nummod, parataxis, punct, reparandum, root, vocative, xcomp']
```

TABLE E. 12 – dépendances Stanford

Stanford	Grew	Talismane
acl	obj.p, 998256, 42.46 % mod, 651102, 27.7 % _, 395917, 16.84 % *autres (#17)* : 13.0%	prep, 991761, 42.19 % mod, 819798, 34.87 % *autres (#22)* : 22.94%
acl :relcl	mod.rel, 848259, 53.46 % _, 226706, 14.29 % *autres (#18)* : 32.26%	mod_rel, 816069, 51.44 % mod, 255191, 16.09 % *autres (#22)* : 32.47%
advcl	obj.p, 435620, 36.92 % obj.cpl, 332949, 28.22 % _, 155016, 13.14 % *autres (#16)* : 21.72%	prep, 433152, 36.72 % sub, 301933, 25.6 % mod, 155484, 13.18 % *autres (#17)* : 24.49%
advmod	mod, 2795595, 79.41 % _, 494973, 14.06 % *autres (#17)* : 6.53%	mod, 3053098, 86.76 % *autres (#46)* : 13.24%
amod	mod, 3343009, 85.05 % *autres (#19)* : 14.95%	mod, 3521139, 89.59 % *autres (#31)* : 10.41%
appos	mod, 478211, 44.59 % _, 204551, 19.07 % *autres (#25)* : 36.34%	mod, 614908, 57.37 % obj, 157172, 14.66 % *autres (#31)* : 27.97%
aux	aux.tps, 1044761, 49.63 % _, 445293, 21.15 % *autres (#19)* : 29.22%	aux_tps, 1035061, 49.17 % root, 296943, 14.11 % *autres (#19)* : 36.72%
auxpass	aux.pass, 249245, 67.3 % _, 57301, 15.47 % *autres (#12)* : 17.22%	aux_pass, 201005, 54.3 % aux_tps, 53268, 14.39 % root, 37320, 10.08 % *autres (#12)* : 21.23%
case	dep, 4095017, 41.09 % mod, 2960586, 29.7 % *autres (#23)* : 29.21%	mod, 6952370, 69.76 % dep, 2170532, 21.78 % *autres (#33)* : 8.46%
cc	coord, 1929989, 64.61 % _, 958310, 32.08 %	coord, 2805329, 93.91 % *autres (#18)* : 6.09%

TABLE E. 12 – *Dépendances pour Stanford, suite sur la prochaine page*

TABLE E. 12 – *Dépendances pour Stanford, suite sur la prochaine page*

Stanford	Grew	Talismane
	autres (#13) : 3.31%	
ccomp	obj.cpl, 333941, 45.33 % _, 102850, 13.96 % *autres (#17)* : 40.71%	sub, 308455, 41.9 % mod, 96653, 13.13 % obj, 95150, 12.92 % *autres (#19)* : 32.05%
compound	mod, 53841, 43.05 % _, 16189, 12.95 % obj.p, 15969, 12.77 % *autres (#15)* : 31.23%	mod, 64412, 51.55 % prep, 15290, 12.24 % *autres (#23)* : 36.22%
conj	dep.coord, 1662197, 34.78 % _, 817364, 17.1 % obj.p, 777150, 16.26 % mod, 763442, 15.97 % *autres (#27)* : 15.89%	dep_coord, 1528465, 31.98 % mod, 1139939, 23.85 % prep, 785106, 16.43 % *autres (#41)* : 27.73%
cop	_, 528398, 46.76 % obj.cpl, 147141, 13.02 % mod, 127373, 11.27 % *autres (#17)* : 28.95%	root, 375370, 33.22 % mod, 225538, 19.96 % sub, 150922, 13.36 % *autres (#16)* : 33.46%
csubj	_, 3100, 39.33 % obj.cpl, 1333, 16.91 % mod, 1254, 15.91 % *autres (#7)* : 27.86%	mod, 2130, 26.87 % root, 1640, 20.69 % sub, 1166, 14.71 % *autres (#7)* : 37.73%
dep	_, 18353, 43.13 % mod, 7306, 17.17 % suj, 6651, 15.63 % *autres (#11)* : 24.08%	mod, 14124, 33.18 % suj, 7291, 17.13 % _, 5974, 14.03 % obj, 5211, 12.24 % *autres (#12)* : 23.42%
det	det, 9936121, 90.16 % *autres (#17)* : 9.84%	det, 9967966, 90.45 % *autres (#23)* : 9.55%
discourse	mod, 6680, 70.42 % _, 2806, 29.58 %	mod, 8389, 86.6 % *autres (#4)* : 13.4%
dobj	obj, 3716155, 57.67 % _, 949458, 14.73 % *autres (#24)* : 27.6%	obj, 3816524, 59.23 % aff, 1149851, 17.85 % *autres (#40)* : 22.92%
expl	mod, 90458, 59.15 % det, 28532, 18.66 % *autres (#11)* : 22.2%	aff, 65481, 42.92 % det, 29426, 19.29 % obj, 16772, 10.99 % mod, 15473, 10.14 % *autres (#10)* : 16.65%
goeswith	mod, 101, 100.0 %	mod, 100, 100.0 %

TABLE E. 12 – *Dépendances pour Stanford, suite sur la prochaine page*

TABLE E. 12 – *Dépendances pour Stanford, suite sur la prochaine page*

Stanford	Grew	Talismane
iobj	a_obj, 452241, 47.72 % _, 147999, 15.62 % obj, 124271, 13.11 % *autres (#12)* : 23.56%	obj, 265071, 27.98 % aff, 244371, 25.79 % a_obj, 193273, 20.4 % _, 107705, 11.37 % *autres (#14)* : 14.46%
mark	mod, 1194620, 40.98 % _, 615501, 21.12 % obj, 392553, 13.47 % *autres (#19)* : 24.44%	mod, 1971403, 67.64 % obj, 382471, 13.12 % *autres (#24)* : 19.24%
mwe	_, 117837, 37.12 % mod, 87783, 27.65 % dep, 35597, 11.21 % *autres (#16)* : 24.01%	mod, 178898, 56.45 % *autres (#22)* : 43.55%
name	mod, 158113, 65.74 % _, 29286, 12.18 % *autres (#14)* : 22.08%	mod, 190586, 79.34 % *autres (#16)* : 20.66%
neg	mod, 1633515, 93.52 % *autres (#11)* : 6.48%	mod, 1617548, 92.63 % *autres (#14)* : 7.37%
nmod	obj.p, 9792461, 86.93 % *autres (#30)* : 13.07%	prep, 9681112, 85.96 % *autres (#52)* : 14.04%
nmod :poss	det, 2105792, 98.14 % *autres (#8)* : 1.86%	det, 2110127, 98.35 % *autres (#9)* : 1.65%
nsubj	suj, 6521773, 86.4 % *autres (#28)* : 13.6%	suj, 6221400, 82.44 % *autres (#49)* : 17.56%
nsubjpass	suj, 264675, 84.89 % *autres (#13)* : 15.11%	suj, 252952, 81.19 % *autres (#21)* : 18.81%
nummod	det, 311268, 57.23 % mod, 174804, 32.14 % *autres (#20)* : 10.63%	det, 262848, 48.31 % mod, 240559, 44.21 % *autres (#14)* : 7.48%
parataxis	mod, 172406, 36.0 % _, 165402, 34.54 % *autres (#17)* : 29.46%	mod, 248856, 52.01 % obj, 61844, 12.93 % *autres (#18)* : 35.06%
punct	ponct, 10226010, 85.62 % _, 1604506, 13.43 % *autres (#19)* : 0.95%	ponct, 11935299, 99.87 % *autres (#15)* : 0.13%
root	_, 2244499, 65.34 % *autres (#21)* : 34.66%	root, 1955105, 56.93 % *autres (#31)* : 43.07%
xcomp	_, 166737, 22.75 % obj, 135002, 18.42 % mod, 103618, 14.14 %	obj, 350155, 47.79 % mod, 125159, 17.08 % *autres (#22)* : 35.13%

TABLE E. 12 – *Dépendances pour Stanford, suite sur la prochaine page*

TABLE E. 12 – *Dépendances pour Stanford, suite sur la prochaine page*

Stanford	Grew	Talismane
	autres (#15) : 44.69%	

Stanford fournit 38 étiquettes différentes (hors regroupement créé lors de l'alignement).

Grew dispersion : Pour 35 DEP différentes : 73 occurrences (moyenne : 2.09).

Couverture Stanford pour Grew : moyenne : 80.09%.

Talismane dispersion : Pour 35 DEP différentes : 67 occurrences (moyenne : 1.91).

Couverture Talismane pour Stanford : moyenne : 78.99%.

E. 6.3 Talismane

Le symbole "|" marque les regroupements d'étiquettes créés lors de l'alignement.

```
['_, a_obj, aff, arg, ato, ats, aux_caus, aux_pass, aux_tps, comp, coord
, de_obj, dep, dep_coord, det, mod, mod_rel, obj, p_obj, ponct, prep,
root, sub, suj']
```

TABLE E. 13 – dépendances Talismane

Talismane	Grew	Stanford
_	_, 813617, 34.42 % mod, 416976, 17.64 % suj, 318889, 13.49 % *autres (#29)* : 34.45%	nsubj, 369584, 15.65 % dobj, 249568, 10.57 % conj, 247114, 10.46 % *autres (#56)* : 63.31%
a_obj	a_obj, 183767, 69.92 % obj, 27001, 10.27 % *autres (#10)* : 19.81%	iobj, 193273, 73.62 % *autres (#15)* : 26.38%
aff	obj, 486763, 29.96 % aff, 347292, 21.38 % _, 321239, 19.77 % a_obj, 250149, 15.4 % *autres (#13)* : 13.49%	dobj, 1149851, 70.77 % iobj, 244371, 15.04 % *autres (#29)* : 14.19%
arg	_, 505, 44.14 % mod, 283, 24.74 % obj.cpl, 260, 22.73 % *autres (#1)* : 8.39%	acl :relcl, 218, 25.38 % acl, 188, 21.89 % conj, 172, 20.02 % xcomp, 151, 17.58 % advcl, 130, 15.13 %
ato	_, 10527, 30.47 % ato, 7143, 20.67 % mod, 6593, 19.08 % *autres (#12)* : 29.77%	xcomp, 6080, 17.78 % nsubj, 5853, 17.12 % nmod, 3492, 10.21 % *autres (#19)* : 54.88%
ats	ats, 392331, 65.53 % mod, 97261, 16.24 % *autres (#14)* : 18.23%	root, 167044, 27.9 % conj, 85196, 14.23 % amod, 61135, 10.21 % *autres (#26)* : 47.65%
aux_caus	aux.caus, 46958, 58.06 % obj.p, 17556, 21.71 % *autres (#11)* : 20.23%	acl, 28859, 35.75 % advcl, 12260, 15.19 % conj, 11175, 13.84 % aux, 8639, 10.7 % *autres (#16)* : 24.51%
aux_pass	aux.pass, 271656, 68.72 % aux.tps, 53514, 13.54 %	auxpass, 201005, 50.84 % aux, 105121, 26.59 %

TABLE E. 13 – *Dépendances pour Talismane, suite sur la prochaine page*

TABLE E. 13 – *Dépendances pour Talismane, suite sur la prochaine page*

Talismane	Grew	Stanford
	autres (#12) : 17.74%	cop, 49758, 12.59 % *autres (#19)* : 9.98%
aux_tps	aux.tps, 989890, 79.93 % aux.pass, 129228, 10.44 % *autres (#11)* : 9.63%	aux, 1035061, 83.57 % *autres (#26)* : 16.43%
comp	_, 570, 37.9 % suj, 431, 28.66 % dep.coord, 205, 13.63 % mod, 183, 12.17 % *autres (#1)* : 7.65%	nsubj, 573, 54.16 % conj, 242, 22.87 % acl :relcl, 152, 14.37 % *autres (#1)* : 8.6%
coord	coord, 1883988, 64.84 % _, 898610, 30.93 % *autres (#15)* : 4.23%	cc, 2805329, 96.55 % *autres (#28)* : 3.45%
de_obj	mod, 10513, 33.07 % de_obj, 9335, 29.37 % _, 5562, 17.5 % *autres (#7)* : 20.06%	mark, 10637, 33.7 % case, 7094, 22.47 % iobj, 6963, 22.06 % *autres (#8)* : 21.77%
dep	dep, 2132987, 75.61 % mod, 315970, 11.2 % *autres (#23)* : 13.19%	case, 2170532, 76.96 % case, det, 374044, 13.26 % *autres (#41)* : 9.78%
dep_coord	dep.coord, 2181378, 80.56 % *autres (#25)* : 19.44%	conj, 1528465, 56.46 % case, 440817, 16.28 % *autres (#40)* : 27.26%
det	det, 12474849, 97.43 % *autres (#17)* : 2.57%	det, 9967966, 77.85 % nmod :poss, 2110127, 16.48 % *autres (#33)* : 5.67%
mod	mod, 14938931, 58.94 % _, 3560384, 14.05 % dep, 2793113, 11.02 % *autres (#56)* : 16.0%	case, 6952370, 27.44 % amod, 3521139, 13.9 % advmod, 3053098, 12.05 % *autres (#139)* : 46.62%
mod_rel	mod.rel, 855698, 67.01 % _, 190694, 14.93 % *autres (#17)* : 18.05%	acl :relcl, 816069, 63.91 % *autres (#30)* : 36.09%
obj	obj, 4332951, 60.97 % _, 873277, 12.29 % *autres (#27)* : 26.75%	dobj, 3816524, 53.71 % *autres (#39)* : 46.29%
p_obj	mod, 7216, 44.51 % p_obj.agt, 3250, 20.05 % p_obj.o, 2152, 13.27 % *autres (#5)* : 22.17%	case, 12900, 79.96 % *autres (#6)* : 20.04%

TABLE E. 13 – *Dépendances pour Talismane, suite sur la prochaine page*

TABLE E. 13 – *Dépendances pour Talismane, suite sur la prochaine page*

Talismane	Grew	Stanford
ponct	ponct, 10224626, 84.74 % _, 1653035, 13.7 % *autres (#24)* : 1.56%	punct, 11935299, 98.86 % *autres (#31)* : 1.14%
prep	obj.p, 12286066, 95.11 % *autres (#28)* : 4.89%	nmod, 9681112, 74.95 % *autres (#40)* : 25.05%
root	_, 3012491, 89.08 % *autres (#21)* : 10.92%	root, 1955105, 57.82 % cop, 375370, 11.1 % *autres (#32)* : 31.08%
sub	obj.cpl, 1076650, 70.36 % _, 209770, 13.71 % *autres (#21)* : 15.93%	ccomp, 308455, 20.16 % advcl, 301933, 19.73 % conj, 185869, 12.15 % aux, 154942, 10.13 % *autres (#29)* : 37.84%
suj	suj, 6692626, 88.29 % *autres (#25)* : 11.71%	nsubj, 6221400, 82.07 % *autres (#37)* : 17.93%

Talismane fournit 24 étiquettes différentes (hors regroupement créé lors de l'alignement).

Grew dispersion : Pour 24 DEP différentes : 53 occurrences (moyenne : 2.21).

Couverture Grew pour Talismane : moyenne : 84.71%.

Stanford dispersion : Pour 24 DEP différentes : 53 occurrences (moyenne : 2.21).

Couverture Stanford pour Talismane : moyenne : 75.17%.

E. 7 Codification syntaxico-sémantique des verbes (LVF)

Ces informations sont directement reprises de celles fournies par Dubois et Dubois-Charlier (1997).

Ce qui suit ***reprend*** *les informations sus-mentionnées (pdf INFORMATIONS_LVF, téléchargeable à* `http://rali.iro.umontreal.ca/LVF+1/documents/INFORMATIONS_LVF.pdf` *(page 5, consulté le 27 mars 2020).*

La représentation des types de sujets, d'objets et de circonstants au moyen des schèmes de construction syntaxique

Les informations syntaxiques basiques sont représentées, dans la rubrique CONSTRUCTION, par des schèmes de construction syntaxique codés sous la forme d'une suite de caractères alphanumériques, selon les conventions suivantes :

- L'appartenance aux types traditionnels est notée par une lettre majuscule.

Code	Type	Construction
A	intransitif	Sujet + Circonstant
N	transitif indirect	Sujet + Complément Prépositionnel (CompPrep)
T	transitif	Sujet + Objet direct + CompPrep + Circonstant
P	pronominal	Sujet + Objet direct + CompPrep + Circonstant

TABLE E. 14 – codage de la transitivité et de l'intransitivité

- Notation pour la nature du sujet et des compléments. Sujet = 1^{er} caractère après A, N, T, P. Objet = $2^{ème}$ caractère après T et P.

1	humain
2	animal
3	chose
4	complétive ou chose

5	complétive ou inf
7	pluriel humain
8	pluriel chose
9	humain ou chose

Exemple *croire 01 [T1400] On croit que tu dis la vérité*

TABLE E. 15 – codage de la nature du sujet et des compléments

- Notation pour les compléments prépositionnels et les circonstants ($2^{ème}$ caractère pour N et A, $3^{ème}$ et $4^{ème}$ caractère pour T et P)
Une lettre minuscule code la préposition.

a à	d contre	i de	l auprès	n divers mouvements (cheminer le long de)
b de	e par	j dans	m devant	q pour
c avec	g sur, vers	k pour		

Exemple : *avertir 01 [T11b0] On avertit Pierre de mon arrivée*

TABLE E. 16 – codage des prépositions

Un chiffre code le type de complément

1	locatif (où l'on est)	*bivouaquer qp*
2	locatif de destination	*accourir qp*
3	locatif d'origine	*décamper de qp*
4	double locatif	*conduire de qp à qp*
5	temps	*durer deux heures, persévérer longtemps*
6	modalité (manière, mesure, quantité)	*aller bien*, *chausser du 37*, *manger beaucoup*
7	cause	*mourir d'un cancer*
8	instrumental, moyen	*montrer par un geste*, *blesser avec une arme*

Exemple : *conduire 04 [T3140] Ces empreintes conduisent l'enquêteur au voleur*

TABLE E. 17 – codage du type de complément

E. 8 Proto-analyses, exemple

Nous avons créé un ensemble de données qui a servi d'appui à la création des corpus de travail ayant permis la génération de phrases depuis des représentations de sens vers du texte.

Voici deux exemples complets des informations stockées quant à deux phrases ayant servi de support aux explications fournies. Afin de clarifier la mise en page, nous avons supprimé des informations de fin de ligne vides. Par exemple `["7", "haut", "haut", "N", "NC", "s=c", "6", "obj.p", "_", "_"]` devient `["7", "haut", "haut", "N" , "NC", "s=c", "6", "obj.p"]`.

Les informations en doublon en fin de chaîne comme `"2", "det"` dans `"1", "Un", "un", "DET", "DET", "n=s|g=m", "2", "det", "2", "det" ]` sont réduites. Cet exemple devient `"1", "Un", "un", "DET", "DET", "n=s|g=m", "2", "det" ]`

Les informations sont rangées, les évènements sont étiquetés par le nom de l'arbre sémantico-syntaxique auquel ils ont été rattaché, la clé de chacun étant le token auquel ils ont été trouvés (les tokens ne sont pas réutilisés, ils servent juste à permettre la gestion de multiples évènements à même structure.


```
"Un rayon blanc, tombant du haut du ciel, anéantit cette comédie.": {
"titre": "Oeuvres de Arthur Rimbaud Vers et proses",
"auteur": "Arthur Rimbaud",
"source": "books/56708-0.txt",
"local_file": "../preparsing/outputs/results_70/063_analyses.txt",
"fichier": "arbres_extraits_0141.json",
"flat tokenized": "Un rayon blanc , tombant du haut du ciel , anéantit
                   cette comédie .",
"book number": 62,
"analyse number": 2801,
"events": {
```

```
"N3b": {
  "5": {
    "event":
      [ "5", "tombant", "tomber", "V", "VPR", "m=part|t=pst", "2",
        "mod" ],
    "sujet": [ "chose",
               [ "2", "rayon", "rayon", "NC", "NC", "n=s|g=m", "11",
                 "suj", "11", "suj" ] ],
    "compl-prep": [ "de",
                    [ "de", ["7", "haut", "haut", "N", "NC", "s=c", "6",
                      "obj.p" ] ] ],
    "contraintes": {
      "conjugaison": {
        "mode": "vpa", "auxiliaires": [],
        "nombre": "inconnu",
        "informations": {
          "socle": [["present", 3, "indicative", "progressive" ]],
          "temps": "présent", "mode": "participe", "personne": "3" } }
"T3Z00": {
  "11": {
    "event": [ "11", "anéantit", "anéantir", "V", "V", "m=ind" ],
    "sujet": [ "chose",
      [ "2", "rayon", "rayon", "N", "NC", "s=c", "11", "suj", ] ],
    "objet": [ "chose ou humain",
      [ "13", "comédie", "comédie", "N", "NC", "s=c", "11", "obj" ] ],
    "constraints": {
      "conjugaison": {
        "mode": "conj", "auxiliaires": [],
        "nombre": "singulier", "informations": {
          "socle": [["present", 3, "indicative", "imperfective",
                     "singular" ],
                    ["past", 3, "indicative", "perfective",
                    "singular" ]],
          "temps": "passé_simple | présent", "mode": "indicatif",
          "personne": "3", "nombre": "singular" } } } } },
"arguments": {
  "2": {
    "argument":
       [ "2", "rayon", "rayon", "N", "NC", "s=c", "11", "suj" ],
    "modifiers": [ [ "1", "Un", "un", "D", "DET", "_", "2", "det" ],
      [ "3", "blanc", "blanc", "A", "ADJ", "_", "2", "mod" ] ]
  },
  "7": {
```

```
      "argument":
         [ "7", "haut", "haut", "N", "NC", "s=c", "6", "obj.p" ],
      "modifiers": [ [ "de",
          ["9", "ciel", "ciel", "N", "NC", "s=c", "8", "obj.p" ] ] ]
    },
    "13": {
      "argument":
         [ "13", "comédie", "comédie", "N", "NC", "s=c", "11", "obj" ],
      "modifiers":
         [ [ "12", "cette", "ce", "D", "DET", "_", "13", "det" ] ] } },
  "arguments hors events": {
    "9": {
      "argument":
         [ "9", "ciel", "ciel", "N", "NC", "s=c", "8", "obj.p" ],
      "modifiers":
         [ [ "8", "du", "le", "D", "DET", "n=s|g=m", "7", "det" ] ]
    }
  }
},
"analyses": {
  "grew": [
    ["1", "Un", "un", "D", "DET", "_", "2", "det" ],
    ["2", "rayon", "rayon", "N", "NC", "s=c", "11", "suj" ],
    ["3", "blanc", "blanc", "A", "ADJ", "_", "2", "mod" ],
    ["4", ",", ",", "PONCT", "PONCT", "_", "2", "ponct" ],
    ["5", "tombant", "tomber", "V", "VPR", "m=part|t=pst", "2", "mod" ],
    ["6", "du", "*du", "P+D", "P+D", "_", "5", "mod" ],
    ["7", "haut", "haut", "N", "NC", "s=c", "6", "obj.p" ],
    ["8", "du", "*du", "P+D", "P+D", "_", "7", "dep" ],
    ["9", "ciel", "ciel", "N", "NC", "s=c", "8", "obj.p" ],
    ["10", ",", ",", "PONCT", "PONCT", "_", "2", "ponct" ],
    ["11", "anéantit", "anéantir", "V", "V", "m=ind", "_", "." ],
    ["12", "cette", "ce", "D", "DET", "_", "13", "det" ],
    ["13", "comédie", "comédie", "N", "NC", "s=c", "11", "obj" ],
    ["14", ".", ".", "PONCT", "PONCT", "_", "11", "ponct" ]
 ],
  "talismane": [
    ["1", "Un", "un", "DET", "DET", "n=s|g=m", "2", "det" ],
    ["2", "rayon", "rayon", "NC", "NC", "n=s|g=m", "11", "suj" ],
    ["3", "blanc", "blanc", "ADJ", "ADJ", "n=s|g=m", "2", "mod" ],
    ["4", ",", ",", "PONCT", "PONCT", "", "3", "ponct" ],
    ["5", "tombant", "tomber", "VPR", "VPR", "t=G", "2", "mod" ],
    ["6", "du", "de", "P+D", "P+D", "n=s|g=m", "5", "mod" ],
```

```
    ["7", "haut", "haut", "NC", "NC", "n=s|g=m", "6", "prep" ],
    ["8", "du", "de", "P+D", "P+D", "n=s|g=m", "7", "dep" ],
    ["9", "ciel", "ciel", "NC", "NC", "n=s|g=m", "8", "prep" ],
    ["10", ",", ",", "PONCT", "PONCT", "", "9", "ponct" ],
    ["11", "anéantit", "anéantir", "V", "V", "n=s|t=J,P|p=3", "0",
     "root" ],
    ["12", "cette", "cette", "DET", "DET", "n=s|g=f", "13", "det" ],
    ["13", "comédie", "comédie", "NC", "NC", "n=s|g=f", "11", "obj" ],
    ["14", ".", ".", "PONCT", "PONCT", "", "13", "ponct" ]
],
  "stanford": [
    ["1", "Un", "_", "_", "DET", "_", "2", "det" ],
    ["2", "rayon", "_", "_", "NOUN", "_", "0", "root" ],
    ["3", "blanc", "_", "_", "ADJ", "_", "2", "amod" ],
    ["4", ",", "_", "_", "PUNCT", "_", "2", "punct" ],
    ["5", "tombant", "_", "_", "VERB", "_", "2", "acl" ],
    ["6", "de_le", "_", "_", "ADP", "_", "7", "case" ],
    ["7", "haut", "_", "_", "NOUN", "_", "5", "nmod" ],
    ["8", "de_le", "_", "_", "ADP", "_", "9", "case" ],
    ["9", "ciel", "_", "_", "NOUN", "_", "7", "nmod" ],
    ["10", ",", "_", "_", "PUNCT", "_", "2", "punct" ],
    ["11", "anéantit", "_", "_", "VERB", "_", "2", "acl" ],
    ["12", "cette", "_", "_", "DET", "_", "13", "det" ],
    ["13", "comédie", "_", "_", "NOUN", "_", "11", "dobj" ],
    ["14", ".", "_", "_", "PUNCT", "_", "2", "punct" ]
],
  "tokens_dones": [true, true, true, false, true, false, true, false,
                   false, false, true, true, true, false]
}
}

"Mille flammes s’ échappaient par les balcons où tourbillonnaient des
flots d’ étincelles.": {
"titre": "Belle-Rose",
"auteur": "Amédée Achard",
"source": "books/17808-8.txt",
"local_file": "../preparsing/outputs/results_70/053_analyses.txt",
"fichier": "arbres_extraits_0118.json",
"flat tokenized": "Mille flammes s’ échappaient par les balcons où
tourbillonnaient des flots d’ étincelles .",
"book number": 52,
"analyse number": 3751,
"events": {
  "PZ030": {
```

```
   "4": {
     "event":
       [ "4", "échappaient", "échapper", "V", "V", "m=ind", "_", "." ],
     "sujet": [ "animal ou chose ou humain",
       [ "2", "flammes", "flamme", "N", "NC", "s=c", "4", "suj" ]
],
     "contraintes": {
       "conjugaison": {
         "mode": "conj", "auxiliaires": [],
         "nombre": "pluriel", "informations": {
           "socle":
              [ [ "past", 3, "indicative", "imperfective", "plural" ] ],
           "temps": "imparfait", "mode": "indicatif", "personne": "3",
           "nombre": "plural"
         }
       }
     },
     "modifiers": [ [ "par",
         [ "7", "balcons", "balcon", "N", "NC", "s=c", "5", "obj.p" ]
]
]
   }
 },
 "A31": {
   "9": {
     "event":
       [ "9", "tourbillonnaient", "tourbillonner", "V", "V", "m=ind",
         "7", "mod.rel" ],
     "sujet": [ "chose",
       [ "11", "flots", "flot", "N", "NC", "s=c", "9", "suj" ]
],
     "circonstant": [ "locatif (où on est)",
       [ "7", "balcons", "balcon", "N", "NC", "s=c", "5", "obj.p" ]
],
     "contraintes": {
       "relative objet": [
         [ "9", "tourbillonnaient", "tourbillonner", "V", "V", "m=ind",
           "7", "mod.rel" ]
],
       "conjugaison": {
         "mode": "conj", "auxiliaires": [],
         "nombre": "pluriel", "informations": {
           "socle":
```

```
              [ [ "past", 3, "indicative", "imperfective", "plural" ] ],
            "temps": "imparfait", "mode": "indicatif", "personne": "3",
            "nombre": "plural" } } } } },
  "arguments": {
    "2": {
      "argument": [ "2", "flammes", "flamme", "N", "NC", "s=c", "4",
                    "suj" ],
      "modifiers": [ [ "1", "Mille", "*Mille", "D", "DET", "_", "2",
                    "det" ]
]
    },
    "7": {
      "argument": [ "7", "balcons", "balcon", "N", "NC", "s=c", "5",
                    "obj.p" ],
      "modifiers": [ [ "6", "les", "le", "D", "DET", "_", "7", "det" ]
]
    },
    "11": {
      "argument": [ "11", "flots", "flot", "N", "NC", "s=c", "9", "suj" ],
      "modifiers": [ [ "10", "des", "un", "D", "DET", "_", "11", "det" ],
        [ "de",
          [ "13", "étincelles", "étincelle", "N", "NC", "s=c", "12",
            "obj.p" ]
]
]
    }
  }
},
"analyses": {
  "grew": [
    [ "1", "Mille", "*Mille", "D", "DET", "_", "2", "det" ],
    [ "2", "flammes", "flamme", "N", "NC", "s=c", "4", "suj" ],
    [ "3", "s'", "se", "CL", "CLR", "s=refl", "4", "a.obj" ],
    [ "4", "échappaient", "échapper", "V", "V", "m=ind", "_", "." ],
    [ "5", "par", "par", "P", "P", "_", "4", "mod" ],
    [ "6", "les", "le", "D", "DET", "_", "7", "det" ],
    [ "7", "balcons", "balcon", "N", "NC", "s=c", "5", "obj.p" ],
    [ "8", "où", "où", "PRO", "PROREL", "s=rel", "9", "mod" ],
    [ "9", "tourbillonnaient", "tourbillonner", "V", "V", "m=ind",
      "7", "mod.rel" ],
    [ "10", "des", "un", "D", "DET", "_", "11", "det" ],
    [ "11", "flots", "flot", "N", "NC", "s=c", "9", "suj" ],
    [ "12", "d'", "de", "P", "P", "_", "11", "dep" ],
```

```
    [ "13", "étincelles", "étincelle", "N", "NC", "s=c", "12", "obj.p" ],
    [ "14", ".", ".", "PONCT", "PONCT", "_", "4", "ponct" ]
],
 "talismane": [
    [ "1", "Mille", "mille", "DET", "DET", "s=card|n=p", "2", "det" ],
    [ "2", "flammes", "flamme", "NC", "NC", "n=p|g=f", "4", "suj" ],
    [ "3", "s'", "se", "CLR", "CLR", "n=p,s|p=3", "4", "aff" ],
    [ "4", "échappaient", "échapper", "V", "V", "n=p|t=I|p=3", "0",
      "root" ],
    [ "5", "par", "par", "P", "P", "", "4", "mod" ],
    [ "6", "les", "les", "DET", "DET", "n=p", "7", "det" ],
    [ "7", "balcons", "balcon", "NC", "NC", "n=p|g=m", "5", "prep" ],
    [ "8", "où", "où", "PROREL", "PROREL", "", "9", "suj", "9", "suj" ],
    [ "9", "tourbillonnaient", "tourbillonner", "V", "V", "n=p|t=I|p=3",
      "7", "mod.rel" ],
    [ "10", "des", "des", "DET", "DET", "n=p", "11", "det" ],
    [ "11", "flots", "flot", "NC", "NC", "n=p|g=m", "9", "obj" ],
    [ "12", "d'", "de", "P", "P", "", "9", "mod", "9", "mod" ],
    [ "13", "étincelles", "étincelle", "NC", "NC", "n=p|g=f", "12",
      "prep" ],
    [ "14", ".", ".", "PONCT", "PONCT", "", "13", "ponct" ]
],
 "stanford": [
    [ "1", "Mille", "_", "_", "NUM", "_", "2", "nummod" ],
    [ "2", "flammes", "_", "_", "NOUN", "_", "0", "root" ],
    [ "3", "s'", "_", "_", "PRON", "_", "4", "dobj" ],
    [ "4", "échappaient", "_", "_", "VERB", "_", "2", "acl" ],
    [ "5", "par", "_", "_", "ADP", "_", "7", "case" ],
    [ "6", "les", "_", "_", "DET", "_", "7", "det" ],
    [ "7", "balcons", "_", "_", "NOUN", "_", "4", "nmod" ],
    [ "8", "où", "_", "_", "PRON", "_", "9", "nmod" ],
    [ "9", "tourbillonnaient", "_", "_", "VERB", "_", "7", "acl:relcl" ],
    [ "10", "des", "_", "_", "DET", "_", "11", "det" ],
    [ "11", "flots", "_", "_", "NOUN", "_", "9", "dobj" ],
    [ "12", "d'", "_", "_", "ADP", "_", "13", "case" ],
    [ "13", "étincelles", "_", "_", "NOUN", "_", "11", "nmod" ],
    [ "14", ".", "_", "_", "PUNCT", "_", "2", "punct" ],
"tokens_dones": [true, true, false, true, true, true, true, true, true,
                 true, true, false, false, false]
```

E. 9 Traits de sortie, proposition

Voici les traits que nous proposons pour les termes en sorties dans le cadre de la génération à partir de sens. Les traits utilisés en entrée sont présentés page 113.

Propriété	Valeurs possibles	Exemple	Explication
nature	pronom, adverbe, verbe, nom, adposition, nom propre, ponctuation, auxiliaire, adjectif, conjonction de coordination, déterminant, x, conjonction de subordination, interjection	le chat dort.	le est un déterminant, chat un nom, dort est un verbe, le point une ponctuation.
genre	masculin, féminin, commun	le chat a mangé	mangé est au genre commun
		le chat est mangé	mangé est au genre masculin
nombre	singulier, pluriel	les chats mangent	chat est au pluriel.
nombre	singulier, pluriel	les chats mangent	chat est au pluriel.
mode	indicatif, infinitif, participe, conditionnel, subjonctif, impératif	les chats ont mangé	mangé est un participe.
temps	présent, passé, passé-simple, imparfait, futur	les chats ont mangé	ont est au présent.
personne	0, 1, 2, 3	le chien mange	chien est un mot à la 3ème personne
sujet	humain, chose...	La femme dort.	dort a un sujet humain.
objet	humain, chose...	La femme mange du pain.	mange a un objet chose.
préposition	à, dans, en, vers...	On avertit Pierre de mon arrivée	avertit a un complément introduit par "de"
circonstant	modalité, instrumental,-moyen...	il montre l'oiseau par un geste	montre a un complément instrumental,-moyen.
position sujet	1, 2...	La femme dort.	dort a un sujet position 2 (femme).
position objet	1, 2...	La femme mange du pain.	avertit objet position 5 (pain).
position préposition	1, 2...	On avertit Pierre de mon arrivée	avertit complément position 6 (arrivée).

Suite page suivante

TABLE E. 18 – *Suite de la page précédente*

Propriété	Valeurs possibles	Exemple	Explication
position circonstant	1, 2...	il montre l'oiseau par un geste	montre : complément position 7 (geste).
divers	indéfini, event, relatif, aux.tps, défini, invariable, determinant, CLS, interrogatif, aux.pass, CLO, CLR, numérique, bruit, aux.caus, réfléchi, aux.mod	La pomme est mangée.	est est un auxiliaire passif, mangée est un évènement...
position HEAD	1, 2...	la femme mange.	la a cette propriété valorisée à "2", femme à "3", mange à "0", et le point à "3" .

TABLE E. 18 – Propriétés encodées dans les facteurs en sortie

Annexe F

Ensemble d'étiquettes (*Tagset*)

F. 1 Penn Treebank Tagset

source : https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	*to*
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

F. 2 Universal Dependencies

source : http://universaldependencies.org/en/dep/ et https://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/trees/UniversalEnglishGrammaticalRelations.html

Code	Correspondance	Exemple
acl	clausal modifier of noun	points to establish are.. points -acl→ establish, I admire the fact that you are honest (fact -acl→honest)
acl:relcl	relative clause modifier	I saw the man you love (man-rel :acl→love)
advcl	adverbial clause modifier	They heard about you missing classes (heard -advcl→ missing), The accident happened as the night was falling (happened -advcl→ falling)
advmod	adverbial modifier	Genetically modified food (modified -advmod→ genetically), less often (often-advmod→less)
amod	adjectival modifier	Anything else for me ? (anything -amod→ else), We can go somewhere nice (somewhere -amod→nice) , Sam eats red meat (meat-amod→red)
appos	appositional modifier	Same, my brother, arrived (Sam -appos→ brother)
aux	auxiliary	Reagan has died (died -aux→has)
auxpass	passive auxiliary	Kennedy was killed (killed →auxpass→was)
case	case marking	I saw a cat with a telescope (telescope-case→with), The school's grounds(school-case→'s)
cc	coordination	And then we left (left-cc→and)
cc:preconj	preconjunct	Both the boys and the girls are here (boys -cc :preconj→both)
ccomp	clausal complement	He says you like to swim (says -ccomp→like)
compound	compound	phone book (book ←compound-phone)
compound:prt	phrasal verb particle	They shut down the station (shut-compound :prt→down) [This relation excludes literal/directional uses of prepositions/particles, such as up, down, in, out, etc. These would typically become an ADV with the relation advmod]
conj	conjunct	Bill is big and honest (big-and→honest)
cop	copula	Bill is honest (honest ←cop-is)
csubj	clausal subject	What she said makes sense (makes←csubj-said)
Suite page suivante		

Suite. ..		
csubjpass	clausal passive subject	That she lied was suspected by everyone (lied←csubjpass-suspected)
dep	dependent	Then , as if to show that he could ,. .. (show←dep-if)
det	determiner	The man is here (man-det→the), Which book do you prefer ?(book-det→which), You 've all won ! (all-det→You)
det:predet	predeterminer	All the boys are here (boys-det :predet→All)
discourse	discourse element	Iguazu is in Argentina :) is-discourse→ :)
dislocated	dislocated elements	The Mezza Luna : you should try it. (office -dislocated→ me) This is our office , me and Sam (it -dislocated→Luna)
dobj	direct object	She gave me a raise (gave -dobj→ raise)
expl	expletive	There is a ghost in the room (is -expl→ there)
foreign	foreign words	I guess that c' est la vie (c'-foreign→est , c'-foreign→la, c'-foreign→vide)
goeswith	goes with	They come here with out legal permission (out-goeswith→with)
iobj	indirect object	She gave me a raise (gave-iobj→me)
list	list	Steve Jones Phone : 555-9814 Email : jones@abc.edf (Steve-list→Phone, Steve-list→Email)
mark	marker	I tried to finish it (finish-mark→to)
mwe	multi-word expression	I like dogs as well (as-mwe→well), I like dogs as well as cats (as^1-mwe→well, as^1-mwe→as^2)
name	name	Carl XVI Gustaf (Carl-name→XVI, Carl-name→Gustaf)
neg	modificateur de négation	Bill is not a scientist : not ←neg- scientist
nmod	nominal modifier	the Chair's office (génitif : Chair ←nmod- office), give the toys to the children (complément prépositionnel : give -nmod→ children)
nmod:npmod	noun phrase as adverbial modifier	The director is 65 years old (old-nmod :npmod→ years) , Shares eased a fraction (eased-nmod :npmod→ fraction)
nmod:poss	possessive nominal modifier	Marie 's book (book-nmod :poss→Marie)
nmod:tmod	temporal modifier	Last night , I swam in the pool (swam-nmod :tmod→night)
Suite page suivante		

Suite. ..		
nsubj	nominal subject	Clinton defeated Dole (defeated-nsubj→Clinton), Sue is a true patriot (patriot-nsubj→Sue), Agatha is in trouble. (trouble←nsubj-Agatha), There is a ghost in the room. (is-nsubj→ghost)
nsubjpass	passive nominal subject	Dole was defeated by Clinton (defeated-nsubjpass→Dole)
nummod	numeric modifier	Sam ate 3 sheep (3 ←nummod- sheep), Sam spent forty dollars (forty ←nummod- dollars)
parataxis	parataxis	Let 's face it we 're annoyed (Let-parataxis→ annoyed)
punct	punctuation	Go home ! (Go -punct→ !)
remnant	remnant in ellipsis	Marie went to Paris , not Miriam (Marie -remnant→ Miriam)
reparandum	overridden disfluency	Go to the righ- to the left. (left -reparandum→ righ.)
root	root	. ..
vocative	vocative	Guys , take it easy ! (take -vocative→ Guys)
xcomp	open clausal complement	I consider him honest (consider -xcomp→ honest)

Bibliographie

Azevedo, F. A., Carvalho, L. R., Grinberg, L. T., Farfel, J. M., Ferretti, R. E., Leite, R. E., Lent, R., Herculano-Houzel, S., et al. (2009). Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5) :532–541.

Bahdanau, D., Cho, K., et Bengio, Y. (2014a). Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Bahdanau, D., Cho, K., et Bengio, Y. (2014b). Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., et Schneider, N. (2013). Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186. Association for Computational Linguistics.

Banerjee, S. et Lavie, A. (2005). Meteor : An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Basile, V. et Bos, J. (2011). Towards generating text from discourse representation structures. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 145–150. Association for Computational Linguistics.

Bastings, J., Titov, I., Aziz, W., Marcheggiani, D., et Sima'an, K. (2017). Graph convolutional encoders for syntax-aware neural machine translation. *arXiv preprint arXiv :1704.04675*.

Beck, D., Haffari, G., et Cohn, T. (2018). Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 273–283, Melbourne, Australia. Association for Computational Linguistics.

Bengio, Y., Simard, P., Frasconi, P., et al. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2) :157–166.

Berant, J. et Liang, P. (2014). Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, volume 1, pages 1415–1425.

Bernardinelli, Y., Randall, J., Janett, E., Nikonenko, I., König, S., Jones, E. V., Flores, C. E., Murai, K. K., Bochet, C. G., Holtmaat, A., et al. (2014). Activity-dependent structural plasticity of perisynaptic astrocytic domains promotes excitatory synapse stability. *Current Biology*, 24(15) :1679–1688.

Black, P. E. (2004). Ratcliff/obershelp pattern recognition. *Dictionary of Algorithms and Data Structures*, 17.

Bohnet, B., Wanner, L., Mille, S., et Burga, A. (2010). Broad coverage multilingual deep sentence generation with a stochastic multi-level realizer. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 98–106. Association for Computational Linguistics.

Bolshakov, I. A. et Gelbukh, A. (2004). Synonymous paraphrasing using wordnet and internet. In *International Conference on Application of Natural Language to Information Systems*, pages 312–323. Springer.

Bontcheva, K. et Wilks, Y. (2004). Automatic report generation from ontologies : the miakt approach. In *International conference on application of natural language to information systems*, pages 324–335. Springer.

Bordes, A., Chopra, S., et Weston, J. (2014). Question answering with subgraph embeddings. *arXiv preprint arXiv :1406.3676*.

Boyd, A. et Meurers, D. (2011). Data-driven correction of functionwords in non-native english. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 267–269.

Callison-Burch, C. (2008). Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 196–205. Association for Computational Linguistics, Association for Computational Linguistics.

Cao, K. et Clark, S. (2019). Factorising AMR generation through syntax. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2157–2163, Minneapolis, Minnesota. Association for Computational Linguistics.

Chan, W., Kitaev, N., Guu, K., Stern, M., et Uszkoreit, J. (2019). KERMIT : Generative insertion-based modeling for sequences. *CoRR*, abs/1906.01604.

Chandrasekar, R. et Srinivas, B. (1997). Automatic induction of rules for text simplification. *Knowledge-Based Systems*, 10(3) :183–190.

Chen, M. et Gimpel, K. (2019). Smaller text classifiers with discriminative cluster embeddings. *arXiv preprint arXiv :1906.09532*.

Chen, M., Lampouras, G., et Vlachos, A. (2018). Sheffield at e2e : structured prediction approaches to end-to-end language generation. Technical report, E2E Challenge System Descriptions.

Chen, M., Tang, Q., Wiseman, S., et Gimpel, K. (2019). Controllable paraphrase generation with a syntactic exemplar. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5972–5984, Florence, Italy. Association for Computational Linguistics.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., et Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Chollet, F. et al. (2015). Keras. https://keras.io.

Colin, É. (2015). Cerebellum modeling. Master's thesis, Université de Lorraine.

Colin, E. (2017). Création automatique d'une grammaire syntaxico-sémantique. In *19es REncontres jeunes Chercheurs en Informatique pour le TAL (RECITAL 2017)*, volume 1, pages 28–41.

Colin, E. et Gardent, C. (2018). Generating syntactic paraphrases. *EMNLP*.

Colin, E. et Gardent, C. (2019). Generating text from anonymised structures. *INLG*.

Colin, E., Gardent, C., M'rabet, Y., Narayan, S., et Perez-Beltrachini, L. (2016). The webnlg challenge : Generating text from dbpedia data. In *Proceedings of the 9th International Natural Language Generation conference*, pages 163–167.

Coster, W. et Kauchak, D. (2011). Learning to simplify sentences using wikipedia. In *Proceedings of the workshop on monolingual text-to-text generation*, pages 1–9. Association for Computational Linguistics.

Curran, J. R., Clark, S., et Bos, J. (2007). Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 33–36. Association for Computational Linguistics.

Dai, A. M. et Le, Q. V. (2015). Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

Devlin, J., Chang, M.-W., Lee, K., et Toutanova, K. (2018). Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv :1810.04805*.

Domhan, T. et Hieber, F. (2017). Using target-side monolingual data for neural machine translation through multi-task learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1500–1505, Copenhagen, Denmark. Association for Computational Linguistics.

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M., et Hon, H.-W. (2019). Unified language model pre-training for natural language understanding and generation. *CoRR*, abs/1905.03197.

Dorr, B., Zajic, D., et Schwartz, R. (2003). Hedge trimmer : A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text summarization workshop-Volume 5*, pages 1–8. Association for Computational Linguistics.

Dubois, J. et Dubois-Charlier, F. (1997). *Les verbes français*. Larousse.

Duboue, P. A. et McKeown, K. R. (2003). Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 121–128. Association for Computational Linguistics.

Duma, D. et Klein, E. (2013). Generating natural language from linked data : Unsupervised template extraction. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 83–94, Potsdam, Germany. Association for Computational Linguistics.

Dušek, O. et Jurcicek, F. (2015). Training a natural language generator from unaligned data. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, pages 451–461. Association for Computational Linguistics.

Dušek, O., Novikova, J., et Rieser, V. (2018). Findings of the e2e nlg challenge. *arXiv preprint arXiv :1810.01170*.

Dušek, O., Novikova, J., et Rieser, V. (2020). Evaluating the state-of-the-art of end-to-end natural language generation : The e2e nlg challenge. *Computer Speech & Language*, 59 :123–156.

Elder, H. et Hokamp, C. (2018). Generating high-quality surface realizations using data augmentation and factored sequence models. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 49–53. Association for Computational Linguistics.

Er, M. J., Zhang, Y., Wang, N., et Pratama, M. (2016). Attention pooling-based convolutional neural network for sentence modelling. *Information Sciences*, 373 :388–403.

Fellbaum, C. (1998). Towards a representation of idioms in wordnet. In *Usage of WordNet in Natural Language Processing Systems*.

Flanigan, J., Dyer, C., Smith, N. A., et Carbonell, J. (2016). Generation from abstract meaning representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 731–739, San Diego, California. Association for Computational Linguistics.

Galanis, D. et Androutsopoulos, I. (2007). Generating multilingual descriptions from linguistically annotated owl ontologies : the naturalowl system. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, pages 143–146. Association for Computational Linguistics.

Galanis, D., Karakatsiotis, G., Lampouras, G., et Androutsopoulos, I. (2009). An open-source natural language generator for owl ontologies and its use in protégé and second life. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics : Demonstrations Session*, pages 17–20. Association for Computational Linguistics.

Ganitkevitch, J. et Callison-Burch, C. (2014). The Multilingual Paraphrase Database. In *LREC*, pages 4276–4283.

Ganitkevitch, J., Callison-Burch, C., Napoles, C., et Van Durme, B. (2011). Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1168–1179. Association for Computational Linguistics.

Gardent, C., Shimorina, A., Narayan, S., et Perez-Beltrachini, L. (2017). The WebNLG challenge : Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Guillaume, B., Bonfante, G., Masson, P., Morey, M., et Perrier, G. (2012). Grew : un outil de réécriture de graphes pour le tal (grew : a graph rewriting tool for nlp)[in french]. In *Proceedings of the Joint Conference JEP-TALN-RECITAL 2012, volume 5 : Software Demonstrations*, pages 1–2.

Haidar-Ahmad, L. (2017). *Extraction d'axiomes et de règles logiques à partir de définitions de wikipédia en langage naturel*. PhD thesis, École Polytechnique de Montréal.

Hassan, S., Csomai, A., Banea, C., Sinha, R., et Mihalcea, R. (2007). Unt : Subfinder : Combining knowledge sources for automatic lexical substitution. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 410–413. Association for Computational Linguistics.

Hebb, D. O. et Hebb, D. (1949). *The organization of behavior*, volume 65. Wiley New York.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02) :107–116.

Hochreiter, S. et Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8) :1735–1780.

Hu, B., Lu, Z., Li, H., et Chen, Q. (2014). Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050.

Hu, Z., Yang, Z., Liang, X., Salakhutdinov, R., et Xing, E. P. (2017). Controllable text generation. *arXiv preprint arXiv :1703.00955*.

Iyyer, M., Wieting, J., Gimpel, K., et Zettlemoyer, L. (2018). Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Kaljurand, K. et Fuchs, N. E. (2007). Verbalizing owl in attempto controlled english. In *OWLED*, volume 258.

Kauchak, D. et Barzilay, R. (2006). Paraphrasing for automatic evaluation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 455–462. Association for Computational Linguistics.

Kingma, D. P. et Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv :1312.6114*.

Klein, G., Kim, Y., Deng, Y., Senellart, J., et Rush, A. M. (2017). Opennmt : Open-source toolkit for neural machine translation. In *Proc. ACL*.

Koehn, P. (2005). Europarl : A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses : Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180.

Koncel-Kedziorski, R., Bekal, D., Luan, Y., Lapata, M., et Hajishirzi, H. (2019). Text generation from knowledge graphs with graph transformers. *arXiv preprint arXiv :1904.02342*.

Konstas, I., Iyer, S., Yatskar, M., Choi, Y., et Zettlemoyer, L. (2017). Neural AMR : Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Kovács, Á., Ács, E., Ács, J., Kornai, A., et Recski, G. (2019). BME-UW at SRST-2019 : Surface realization with interpreted regular tree grammars. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019)*, pages 35–40, Hong Kong, China. Association for Computational Linguistics.

Kozlowski, R., McCoy, K. F., et Vijay-Shanker, K. (2003). Generation of single-sentence paraphrases from predicate/argument structure using lexico-grammatical resources. In *Proceedings of the second international workshop on Paraphrasing-Volume 16*, pages 1–8. Association for Computational Linguistics.

Lampouras, G. et Androutsopoulos, I. (2013). Using integer linear programming for content selection, lexicalization,and aggregation to produce compact texts from owl ontologies. In *Proceedings of the 14th European Workshop on Natural Language Generation*.

Lawrence, C., Kotnis, B., et Niepert, M. (2019). Attending to future tokens for bidirectional sequence generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1–10, Hong Kong, China. Association for Computational Linguistics.

Li, Y., Tarlow, D., Brockschmidt, M., et Zemel, R. (2015). Gated graph sequence neural networks. *arXiv preprint arXiv :1511.05493*.

Li, Z., Jiang, X., Shang, L., et Liu, Q. (2019). Decomposable neural paraphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3403–3414, Florence, Italy. Association for Computational Linguistics.

Liao, K., Lebanoff, L., et Liu, F. (2018). Abstract meaning representation for multi-document summarization. *arXiv preprint arXiv :1806.05655*.

Lin, D. et Pantel, P. (2001). Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4) :343–360.

Luong, M.-T., Pham, H., et Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Ma, L., Lu, Z., et Li, H. (2016). Learning to answer questions from image using convolutional neural network. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Madnani, N., Ayan, N. F., Resnik, P., et Dorr, B. J. (2007). Using paraphrases for parameter tuning in statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 120–127. Association for Computational Linguistics.

Madnani, N. et Dorr, B. J. (2010). Generating phrasal and sentential paraphrases : A survey of data-driven methods. *Computational Linguistics*, 36(3) :341–387.

Madnani, N., Tetreault, J., et Chodorow, M. (2012). Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 182–190. Association for Computational Linguistics.

Mallinson, J., Sennrich, R., et Lapata, M. (2017). Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics : Volume 1, Long Papers*, volume 1, pages 881–893.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., et McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics : system demonstrations*, pages 55–60.

Marcheggiani, D. et Perez-Beltrachini, L. (2018). Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 1–9, Tilburg University, The Netherlands. Association for Computational Linguistics.

Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S., et Zamparelli, R. (2014). Semeval-2014 task 1 : Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 1–8.

May, J. et Priyadarshi, J. (2017). Semeval-2017 task 9 : Abstract meaning representation parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 536–545. Association for Computational Linguistics.

McCulloch, W. S. et Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4) :115–133.

McKeown, K. R. (1983). Paraphrasing questions using given and new information. *Computational Linguistics*, 9(1) :1–10.

Mihalcea, R. et Csomai, A. (2007). Wikify ! : Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 233–242, New York, NY, USA. ACM.

Mille, S., Belz, A., Bohnet, B., Graham, Y., Pitler, E., et Wanner, L. (2018). The first multilingual surface realisation shared task (sr'18) : Overview and evaluation results. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 1–12.

Mille, S., Belz, A., Bohnet, B., Graham, Y., et Wanner, L. (2019). The second multilingual surface realisation shared task (sr'19) : Overview and evaluation results. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019)*, pages 1–17.

Moussallem, D., Wauer, M., et Ngomo, A.-C. N. (2018). Machine translation using semantic web technologies : A survey. *Journal of Web Semantics*, 51 :1–19.

Napoles, C., Gormley, M., et Van Durme, B. (2012). Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100. Association for Computational Linguistics.

Narayan, S. et Gardent, C. (2014). Hybrid simplification using deep semantics and machine translation. In *HAL*.

Narayan, S., Reddy, S., et Cohen, S. B. (2016). Paraphrase generation from latent-variable pcfgs for semantic parsing. *arXiv preprint arXiv :1601.06068*.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., McDonald, R. T., Petrov, S., Pyysalo, S., Silveira, N., et others (2016). Universal Dependencies v1 : A Multilingual Treebank Collection. In *LREC*.

Novikova, J., Dušek, O., et Rieser, V. (2017). The e2e dataset : New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206. Association for Computational Linguistics.

Oberheim, N. A., Takano, T., Han, X., He, W., Lin, J. H., Wang, F., Xu, Q., Wyatt, J. D., Pilcher, W., Ojemann, J. G., et al. (2009). Uniquely hominid features of adult human astrocytes. *The Journal of Neuroscience*, 29(10) :3276–3287.

Palmer, M., Gildea, D., et Kingsbury, P. (2005). The proposition bank : An annotated corpus of semantic roles. *Computational linguistics*, 31(1) :71–106.

Palmer, M., Knight, K., et Bonial, C. (2014). Designing abstract meaning representations for machine translation.

Papineni, K., Roukos, S., Ward, T., et Zhu, W.-J. (2002). Bleu : a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Pascanu, R., Mikolov, T., et Bengio, Y. (2012). Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, 2.

Pennington, J., Socher, R., et Manning, C. (2014). Glove : Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Perez-Beltrachini, L., Gardent, C., et Kruszewski, G. (2012). Generating grammar exercises. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 147–156. Association for Computational Linguistics.

Poria, S., Cambria, E., et Gelbukh, A. (2016). Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems*, 108 :42–49.

Post, M. (2018). A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation : Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Pouget-Abadie, J., Bahdanau, D., Van Merrienboer, B., Cho, K., et Bengio, Y. (2014). Overcoming the curse of sentence length for neural machine translation using automatic segmentation. *arXiv preprint arXiv :1409.1257*.

Pourdamghani, N., Knight, K., et Hermjakob, U. (2016). Generating english from abstract meaning representations. In *Proceedings of the 9th International Natural Language Generation conference*, pages 21–25.

Power, R. (2010). Complexity assumptions in ontology verbalisation. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 132–136. Association for Computational Linguistics.

Power, R. et Third, A. (2010). Expressing owl axioms by english sentences : dubious in theory, feasible in practice. In *Proceedings of the 23rd International Conference on Computational Linguistics : Posters*, pages 1006–1013. Association for Computational Linguistics.

Prakash, A., Hasan, S. A., Lee, K., Datla, V., Qadir, A., Liu, J., et Farri, O. (2016). Neural paraphrase generation with stacked residual LSTM networks. *arXiv preprint arXiv :1610.03098*.

Puzikov, Y. et Gurevych, I. (2018). E2E NLG challenge : Neural models vs. templates. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 463–471, Tilburg University, The Netherlands. Association for Computational Linguistics.

Qin, P., Xu, W., et Guo, J. (2016). An empirical convolutional neural network approach for semantic relation classification. *Neurocomputing*, 190 :1–9.

Quirk, C., Brockett, C., et Dolan, W. (2004). Monolingual machine translation for paraphrase generation. *EMNLP*, 149.

Ribeiro, L. F., Gardent, C., et Gurevych, I. (2019). Enhancing amr-to-text generation with dual graph representations. *arXiv preprint arXiv :1909.00352*.

Riezler, S., Vasserman, A., Tsochantaridis, I., Mittal, V., et Liu, Y. (2007). Statistical machine translation for query expansion in answer retrieval. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 464–471.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., et Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1) :61–80.

See, A., Liu, P. J., et Manning, C. D. (2017). Get to the point : Summarization with pointer-generator networks. *arXiv preprint arXiv :1704.04368*.

Semeniuta, S., Severyn, A., et Barth, E. (2017). A hybrid convolutional variational autoencoder for text generation. *arXiv preprint arXiv :1702.02390*.

Shannon, C. E. et Weaver, W. (1949). *The Mathematical Theory of Communication, by CE Shannon (and Recent Contributions to the Mathematical Theory of Communication), W. Weaver*. University of illinois Press.

Shimorina, A., Khasanova, E., et Gardent, C. (2019). Creating a corpus for Russian data-to-text generation using neural machine translation and post-editing. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 44–49, Florence, Italy. Association for Computational Linguistics.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., et Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Song, L., Peng, X., Zhang, Y., Wang, Z., et Gildea, D. (2017). Amr-to-text generation with synchronous node replacement grammar. *arXiv preprint arXiv :1702.00500*.

Song, L., Zhang, Y., Wang, Z., et Gildea, D. (2018). A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 1616–1626, Melbourne, Australia. Association for Computational Linguistics.

Stent, A. et Bangalore, S. (2014). *Natural language generation in interactive systems*. Cambridge University Press.

Stevens, R., Malone, J., Williams, S., Power, R., et Third, A. (2011). Automating generation of textual class definitions from owl to english. In *Journal of Biomedical Semantics*, volume 2, page S5. BioMed Central.

Sun, X. et Mellish, C. (2007). An experiment on "free generation" from single rdf triples. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, ENLG '07, pages 105–108, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sutskever, I., Martens, J., et Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

Sutskever, I., Vinyals, O., et Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Takase, S., Suzuki, J., Okazaki, N., Hirao, T., et Nagata, M. (2016). Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1054–1059.

Touzet, C. (1992). *Réseaux de neurones artificiels : introduction au connexionnisme (Artificial neural nets : introduction to connectionism)*. EC2.

Trisedya, B. D., Qi, J., Zhang, R., et Wang, W. (2018). Gtr-lstm : A triple encoder for sentence generation from rdf data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 1627–1637.

Tu, Z., Lu, Z., Liu, Y., Liu, X., et Li, H. (2016). Modeling coverage for neural machine translation. *arXiv preprint arXiv :1601.04811*.

Urieli, A. (2013). *Robust French syntax analysis : reconciling statistical methods and linguistic knowledge in the Talismane toolkit*. PhD thesis, Université de Toulouse II le Mirail.

van der Lee, C., Gatt, A., van Miltenburg, E., Wubben, S., et Krahmer, E. (2019). Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 355–368.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, \., et Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., et Bengio, Y. (2018). Graph Attention Networks. *International Conference on Learning Representations*. accepted as poster.

Vinyals, O., Fortunato, M., et Jaitly, N. (2015). Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Vogiatzis, D., Galanis, D., Karkaletsis, V., Androutsopoulos, I., et Spyropoulos, C. (2008). A conversant robotic guide to art collections. In *Proceedings of the 2nd Workshop on Language Technology for Cultural Heritage Data, Language Resources and Evaluation Conference, Marrakech, Morocco*, pages 55–60.

Watanabe, W. M., Junior, A. C., Uzêda, V. R., Fortes, R. P. d. M., Pardo, T. A. S., et Aluísio, S. M. (2009). Facilita : reading assistance for low-literacy readers. In *Proceedings of the 27th ACM international conference on Design of communication*, pages 29–36. ACM.

Wen, T.-H., Gasic, M., Mrkšić, N., Su, P.-H., Vandyke, D., et Young, S. (2015). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721. Association for Computational Linguistics.

Wieting, J., Bansal, M., Gimpel, K., et Livescu, K. (2015a). Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv :1511.08198*.

Wieting, J., Bansal, M., Gimpel, K., Livescu, K., et Roth, D. (2015b). From paraphrase database to compositional paraphrase model and back. *arXiv preprint arXiv :1506.03487*.

Wieting, J. et Gimpel, K. (2017). Paranmt-50m : Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. *arXiv preprint arXiv :1711.05732*.

Woodsend, K. et Lapata, M. (2011). Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the conference on empirical methods in natural language processing*, pages 409–420. Association for Computational Linguistics.

Wubben, S., Van Den Bosch, A., et Krahmer, E. (2010). Paraphrase generation as monolingual translation : Data and evaluation. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 203–207. Association for Computational Linguistics.

Wubben, S., Van Den Bosch, A., et Krahmer, E. (2012). Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics : Long Papers-Volume 1*, pages 1015–1024. Association for Computational Linguistics.

Xu, W., Napoles, C., Pavlick, E., Chen, Q., et Callison-Burch, C. (2016). Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4 :401–415.

Yamada, K. et Knight, K. (2001). A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530.

Zerva, C. et Kopaneli, A. (2012). *Named-Entity Recognition and Text Enrichment using Semantic Web*. PhD thesis, National Technical University of Athens - School of Electrical and Computer Engineering, division Computer Science.

Zhang, X. et Lapata, M. (2017). Sentence simplification with deep reinforcement learning. *arXiv preprint arXiv :1703.10931*.

Zhao, S., Niu, C., Zhou, M., Liu, T., et Li, S. (2008). Combining multiple resources to improve SMT-based paraphrasing model. *Proceedings of ACL-08 : HLT*, pages 1021–1029.

Zhu, Y., Wan, J., Zhou, Z., Chen, L., Qiu, L., Zhang, W., Jiang, X., et Yu, Y. (2019). Triple-to-text. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR'19*.

Zhu, Z., Bernhard, D., et Gurevych, I. (2010). A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd international conference on computational linguistics*, pages 1353–1361. Association for Computational Linguistics.

Zou, L., Huang, R., Wang, H., Yu, J. X., He, W., et Zhao, D. (2014). Natural language question answering over rdf : a graph data driven approach. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 313–324. ACM.

Résumé

Nous nous sommes attelée à participer à la recherche autour d'un thème clé, la génération de paraphrases sur support neuronal. Nos perspectives sont éducatives : créer des exercices de grammaire pour le français. La paraphrase est une opération de reformulation : la reproduction d'un signifié sous une forme différente. Le potentiel en recherche va bien au-delà des possibles en terme d'enseignement. Nous pensons en particulier à la consolidation de représentations computationnelles, à celle de la robustesse de systèmes de questions réponses, aux capacités à lexicaliser de l'information sémantique (par des données argent). Les capacités des modèles séquence vers séquence sont aussi d'un intérêt qui nous a paru remarquable. Nos travaux tendent à attester que ces réseaux ne sont pas de simples répétiteurs mais peuvent apprendre la syntaxe.
Nous avons tout d'abord, en combinant divers modèles, montré que la représentation de l'information sous de multiples formes (en utilisant de la donnée formelle (RDF), couplée à du texte pour l'étendre ou le réduire, ou encore seulement du texte) permet d'exploiter un corpus sous différents angles, augmentant la diversité des sorties, exploitant les leviers syntaxiques mis en place. Nous nous sommes penché sur un problème récurrent, celui de la qualité des données, et avons obtenu des paraphrases avec une haute adéquation syntaxique (jusqu'à 98% de couverture de la demande) et un très bon niveau linguistique. Nous obtenons jusqu'à 83.97 points de BLEU [82], 78.41 de plus que la moyenne de nos lignes de base, sans levier syntaxique. Ce taux indique un meilleur contrôle des sorties, pourtant variées et de bonne qualité en l'absence de levier.
Ce premier travail a été réalisé grâce à une ressource parallélisant représentations (RDF) et textes, en anglais : WebNLG (Gardent et al., 2017). Nous avions pour idée de travailler depuis du texte brut en passant, pour la génération de phrases, par la production d'une représentation du sens de ce texte qui puisse servir d'entrée à la génération de paraphrases. Le passage à du texte en français était aussi pour nous un impératif. Travailler depuis du texte brut, en automatisant les procédures, nous a permis de créer un corpus de plus de 450 000 couples représentations/phrases, grâce auquel nous avons appris à générer des textes massivement corrects (92% sur la validation qualitative). Anonymiser ce qui n'est pas fonctionnel a participé notablement à la qualité des résultats (68.31 de BLEU, soit +3.96 par rapport à la ligne de base, qui était la génération depuis des données non anonymisées).
Ce second travail peut se voir appliquer l'intégration d'un levier syntaxique guidant les sorties. Ce qui a été notre approche dans la deuxième partie de cette thèse (générer sans contrainte) viendrait se combiner à un modèle contraint. En appliquant une fouille d'erreur, cela permettrait la constitution d'une base d'apprentissage "argent" associant des représentations à des textes et à des contraintes reflétant les structures syntaxiques contenues. Cette base pourrait être démultipliée par une ré-application d'une génération

82. BLEU (Papineni et al., 2002) : qualité d'un texte sur une échelle de 0 (pire) à 100 (meilleur)

sous contrainte pour réaliser l'objectif appliqué de la thèse.
Notre travail laisse ouvert cet objectif, mais plus encore de grandes questions de recherche. La représentation formelle de l'information dans un cadre linguistique particulier à une langue est une tâche ardue. Cette thèse offre des pistes de méthodes pour automatiser cette opération. Par ailleurs, nous n'avons pu traiter que des phrases relativement courtes. L'utilisation de modèles neuronaux plus récents (Transformer avec pré-apprentissage) permettrait sans doute d'étendre l'approche à de plus longues phrases. Enfin, la fouille d'erreur est une démarche essentielle. L'anonymisation offre des outils. L'usage de traits adéquats en sortie permettrait des vérifications poussées.

Mots-clés: apprentissage profond, acquisition de la langue, syntaxe, génération

Abstract

We set out to participate in research around a key theme, the generation of paraphrases on neural support. Our perspectives are educational, to create grammar exercises for French. Paraphrasing is an operation of reformulation : the reproduction of a signified in a different form. The research potential goes far beyond what is possible in terms of teaching. We are thinking in particular of the consolidation of computational representations, of the robustness of question-answer systems, of the ability to lexicalize semantic information (using silver data). The capabilities of sequence-to-sequence models are also of remarkable interest. Our work tends to attest that these networks are not simple repeaters but can learn syntax.
First, by combining various models, we have shown that the representation of information in multiple forms (using formal data (RDF), coupled with text to extend or reduce it, or only text) allows us to exploit a corpus from different angles, increasing the diversity of outputs, exploiting the syntactic levers put in place. We also addressed a recurrent problem, that of data quality, and obtained paraphrases with a high syntactic adequacy (up to 98% coverage of the demand) and a very good linguistic level. We obtain up to 83.97 points of BLEU-4 [83], 78.41 more than our baseline average, without syntax leverage. This rate indicates a better control of the outputs, which are varied and of good quality in the absence of syntax leverage.
Our idea was to be able to work from raw text : to produce a representation of its meaning. The transition to French text was also an imperative for us. Working from plain text, by automating the procedures, allowed us to create a corpus of more than 450,000 sentence/representation pairs, thanks to which we learned to generate massively correct texts (92% on qualitative validation). Anonymizing everything that is not functional contributed significantly to the quality of the results (68.31 of BLEU, i.e. +3.96 compared to the baseline, which was the generation of text from non-anonymized data).
This second work can be applied the integration of a syntax lever guiding the outputs. What was our baseline at time 1 (generate without constraint) would then be combined

83. BLEU : quality of a text (scale from 0 (worst) to 100 (best), Papineni et al. (2002))

with a constrained model. By applying an error search, this would allow the constitution of a silver base associating representations to texts. This base could then be multiplied by a reapplication of a generation under constraint, and thus achieve the applied objective of the thesis.
Our work leaves this goal open ; it leaves open important research questions. The formal representation of information in a language-specific framework is a challenging task. This thesis offers some ideas on how to automate this operation. Moreover, we were only able to process relatively short sentences. The use of more recent neural models (Transform with pre-training) would probably allow to extend the approach to longer sentences. Finally, error mining is an essential step. Anonymization offers tools. The use of appropriate output strokes would allow for extensive checks.

Keywords: deep learning, language acquisition, syntax, surface realization