Grammar Based Generation: Algorithms, Error Mining and Applications

Claire Gardent
(Joint work with German Kruszewski, Shashi Narayan and Laura Perez-Beltrachini)

CNRS/LORIA, Nancy

January 29, 2014
XRCE Grenoble, France

Computational Grammars

Pros

- Direct modelling of linguistic constructs
- Precise (detailed modelling of interactions)
- Multi-level (morphology, syntax, semantics ..)
- Glass box

Cons

- Brittle, Fail to scale up (Coverage)
- Slow to Process (Efficiency)
- Error Prone (Correctness)

Grammar-Based Surface Realisation (SR)

Efficiency and Coverage

- XTAG and Top-Down, Bottom-Up SR **algorithm** generates Penn Tree Bank sentences in 2.57 seconds
  Average (maximum) word length: 22 (134)

Correctness

- **Error Mining** permits improving correctness and adapting the grammar to the PTB input yielding a BLEU score on covered input of 0.73

Application

- Grammar used to **generate grammar exercises** and their solution (Online WFLEG)

A TD/BU algorithm for Grammar Based Surface Realisation

What is Surface Realisation?

Structured Input from the Generation Challenge Surface Realisation Task

Feature-Based Tree Adjoining Grammar

The Algorithm

Evaluation and Results

S. Narayan and C. Gardent
Structure-Driven Lexicalist Generation
Proceedings of *COLING 2012*, pp 2027 - 2041, Mumbai, India

What is Surface Realisation?

SR maps INPUT DATA to SENTENCES

The input data can be more syntactic or more semantic; a tree or a graph:

- Dependency trees (SR Task)
- OWL triples
- First Order Logic (FOL) Formulae
- Flat semantics (MRSs)
- ...

$\exists x.(Man(x) \wedge \exists y.(Apple(y) \wedge eat(e,x,y) \wedge now(e)))$
$\Rightarrow$ *A man eats an apple*

The SR Shared Task Input Representations

Surface Realisation Task organised by the Generation Challenges 2011

Input derived from the PennTreebank

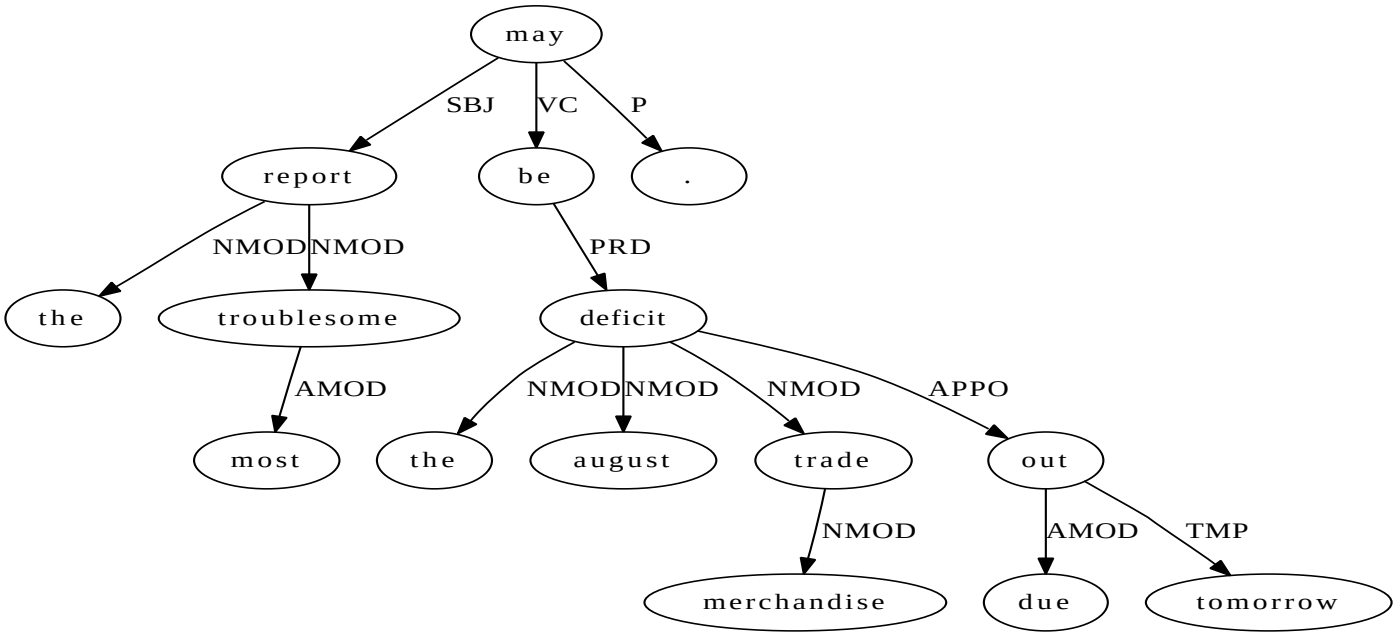
Shallow dependency structures

- Unordered trees
- Edges are labelled with syntactic functions
- Nodes labelled with lemmas, part of speech tags and partial morphosyntactic information

All words of the original sentence are represented by a node in the tree

Example Input

The most troublesome report may be the August merchandise trade deficit due out tomorrow

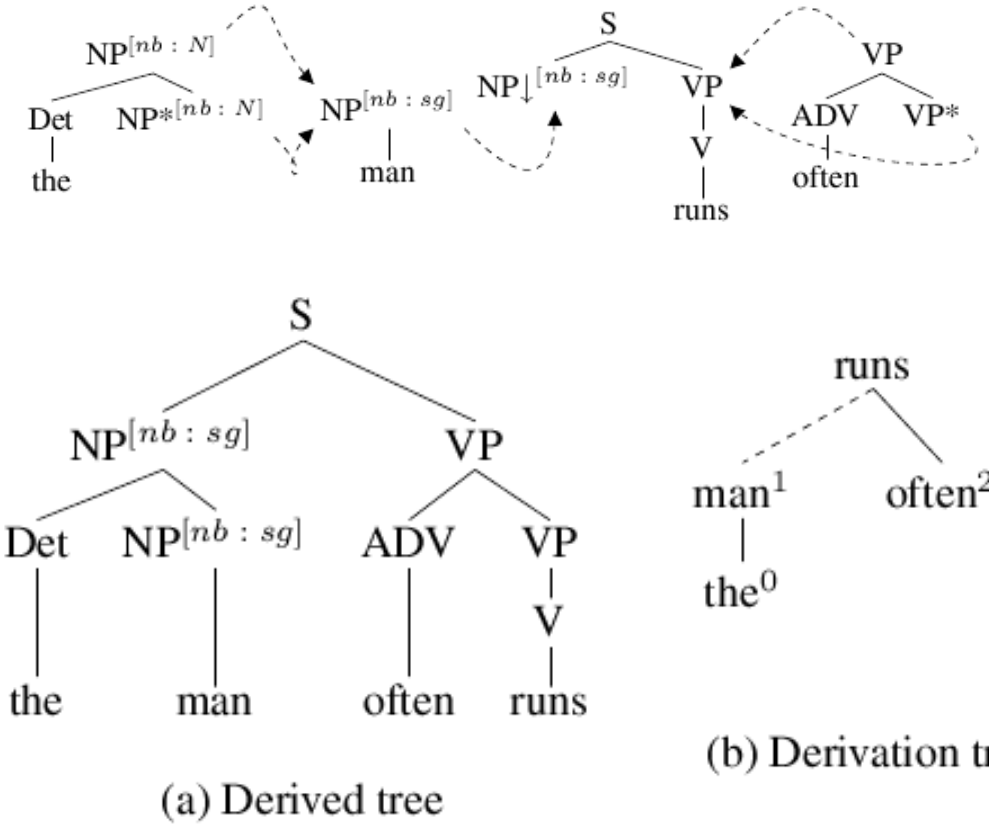


Grammar

Feature-Based Lexicalised Tree Adjoining Grammar (FB-LTAG)

- A set of trees, lexicalised with one or more words and decorated with feature structures
- 2 combining operations: substitution and adjunction
- XMG Reimplementation of XTAG (large coverage of English)

Example FB-LTAG



Grammar-Based Surface Realisation Algorithms

Two main approaches

Head-Driven algorithm

Used for recursively structured input data e.g., logical formulae
Use input structure to guide the search
Cons: Logical Form Equivalence Problem

Lexicalist

Used for unstructured input data (e.g., MRS formula)
Selects lexical entries bottom-up from the input semantic literals
Cons: Computationally expensive (Unordered input, Lexical ambiguity, Intersective modifiers)

Structure-Driven Lexicalist Surface Realisation

Combines techniques and ideas from the head-driven and the lexicalist approach.

- Select grammar rules bottom up for each input tree node (Lexicalist)
- Uses the structure of the input to guide the search and prune the search space (Structure Driven)

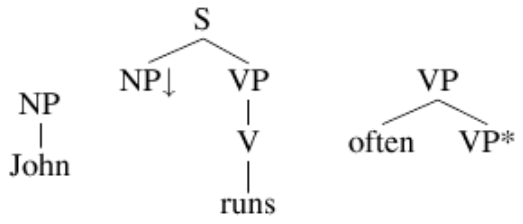
Integrates various optimisations previously proposed for parsing/generation

Parallelised

Structure-Driven Lexicalist Generation

- FB-LTAG **converted to FB-RTG** to construct derivation rather than derived trees
  (Koller and Striegnitz, 2002; Gardent and Perez-Beltrachini 2010)
- Top-down filter using the structure of the input
  (**Head-Driven** algorithm, Shieber et al. 1990)
- Bottom-up **polarity filter** on local input trees.
  (Bonfante 2004; Gardent and Kow 2007).
- **Language model** used to prune competing intermediate structures
  (Bangalore and Rambow 2000; White 2004)
- **Parallelism** used to explore the possible completions of the top-down predictions simultaneously rather than sequentially.

Converting a TAG to an RTG



r1.	NP_S	$\rightarrow$	$john(NP_A)$
r2.	S_S	$\rightarrow$	$runs(S_A\ NP_S\ VP_A\ V_A)$
r3.	VP_A	$\rightarrow$	$often(VP_A)$
r4.	NP_A	$\rightarrow$	ϵ
r5.	S_A	$\rightarrow$	ϵ
r6.	V_A	$\rightarrow$	ϵ
r7.	VP_A	$\rightarrow$	ϵ

The Algorithm

Starts from the root node of the input tree

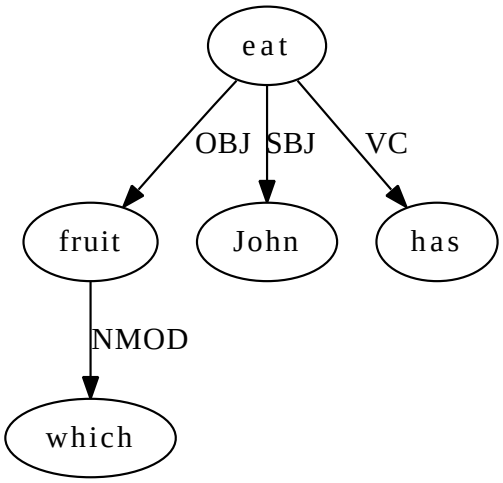
Processes all children nodes *in parallel* spreading lexical selection constraints *top-down* and combining FB-RTG rules *bottom-up*

4 main steps

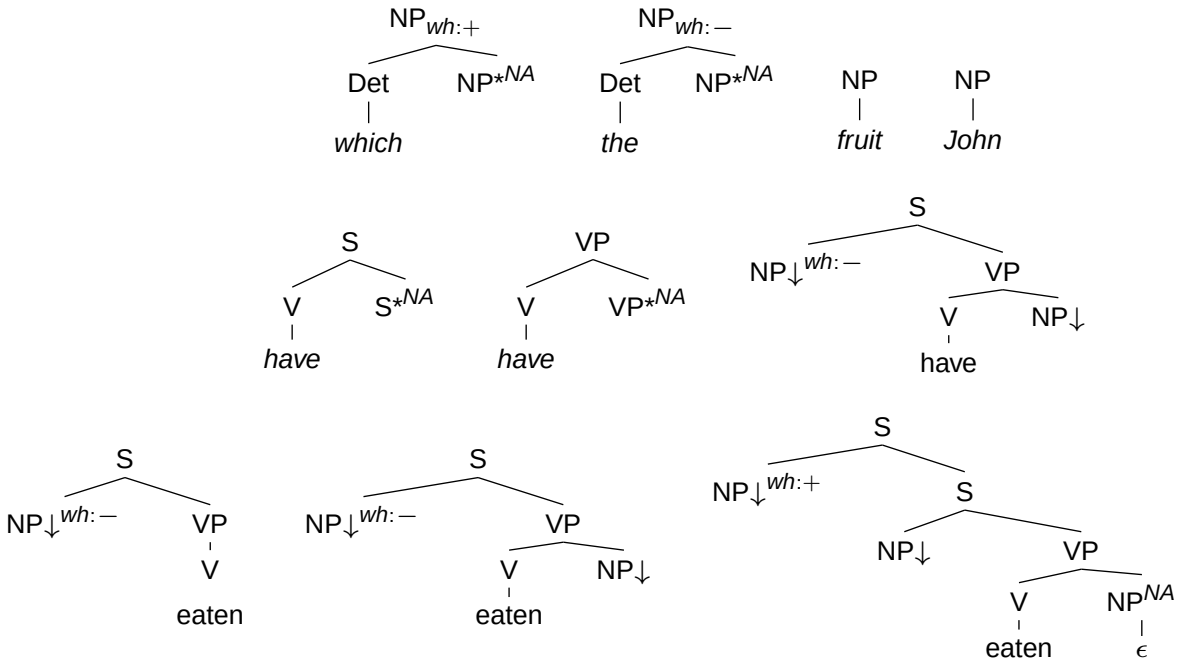
- Bottom-Up Lexical Selection and Top-Down Filtering
- Bottom-Up Local Polarity Filtering
- Bottom-Up Generation
- N-Gram Filtering

Example

Input Dependency Tree



An FB-LTAG


NP$_{wh:+}$
Det
NP*^{NA}
which
NP$_{wh:-}$
Det
NP*^{NA}
the
NP
fruit
NP
John
S
V
S*^{NA}
have
VP
V
VP*^{NA}
have
S
NP↓$^{wh:-}$
VP
V
NP↓
have
S
NP↓$^{wh:-}$
VP
V
eaten
S
NP↓$^{wh:-}$
VP
V
NP↓
eaten
S
NP↓$^{wh:+}$
S
NP↓
VP
V
NP^{NA}
eaten
ϵ

... and the corresponding FB-RTG

$$
\begin{array}{lcl}
NP_A^{[t:T]} & \rightarrow & \mathit{which}(NP_A^{[t:T,b:[wh:+]]}) \\
NP_A^{[t:T]} & \rightarrow & \mathit{the}(NP_A^{[t:T,b:[wh:-]]}) \\
NP_S^{[t:T]} & \rightarrow & \mathit{fruit}(NP_A^{[t:T]}) \\
NP_S^{[t:T]} & \rightarrow & \mathit{John}(NP_A^{[t:T]}) \\
S_A^{[t:T]} & \rightarrow & \mathit{have}(S_A^{[t:T]}) \\
VP_A^{[t:T]} & \rightarrow & \mathit{have}(VP_A^{[t:T]}) \\
S_S^{[t:T,b:B]} & \rightarrow & \mathit{have}(S_A^{[t:T,b:B]}\ NP_S^{[t:[wh:-]]}\ VP_A\ NP_S) \\
S_S^{[t:T,b:B]} & \rightarrow & \mathit{eat}(S_A^{[t:T,b:B]}\ NP_S^{[t:[wh:-]]}\ VP_A) \\
S_S^{[t:T,b:B]} & \rightarrow & \mathit{eat}(S_A^{[t:T,b:B]}\ NP_S^{[t:[wh:-]]}\ VP_A\ NP_S) \\
S_S^{[t:T,b:B]} & \rightarrow & \mathit{eat}(S_A^{[t:T,b:B]}\ NP_S^{[t:[wh:+]]}\ S_A\ NP_S\ VP_A) \\
X_A^{[t:T,b:T]} & \rightarrow & \epsilon
\end{array}
$$

Bottom-Up Lexical Selection and Top-Down Filtering

Lexical Selection: for each input node *n* with lemma *w*, selects all FB-RTG rules which can be lexicalised by *w*

Top-Down Filtering: Only keep those rules whose left-hand side category occurs at least once in the right-hand side of the rules selected by the parent node.

Example Top-Down Filtering

Rule selection for *eat*:

$$
\begin{array}{lll}
S_S^{[t:T,b:B]} & \rightarrow & eat(S_A^{[t:T,b:B]}\ NP_S^{[t:[wh:-]]}\ VP_A) \\
S_S^{[t:T,b:B]} & \rightarrow & eat(S_A^{[t:T,b:B]}\ NP_S^{[t:[wh:-]]}\ VP_A\ NP_S) \\
S_S^{[t:T,b:B]} & \rightarrow & eat(S_A^{[t:T,b:B]}\ NP_S^{[t:[wh:+]]}\ S_A\ NP_S\ VP_A)
\end{array}
$$

Rule selection and filtering for *has*:

$$
\begin{array}{lll}
\surd & S_A^{[t:T]} & \rightarrow\ have(S_A^{[t:T]}) \\
\surd & VP_A^{[t:T]} & \rightarrow\ have(VP_A^{[t:T]}) \\
\times & S_S^{[t:T,b:B]} & \rightarrow\ have(S_A^{[t:T,b:B]}\ NP_S^{[t:[wh:-]]}\ VP_A\ NP_S)
\end{array}
$$

Bottom-Up Local Polarity Filtering

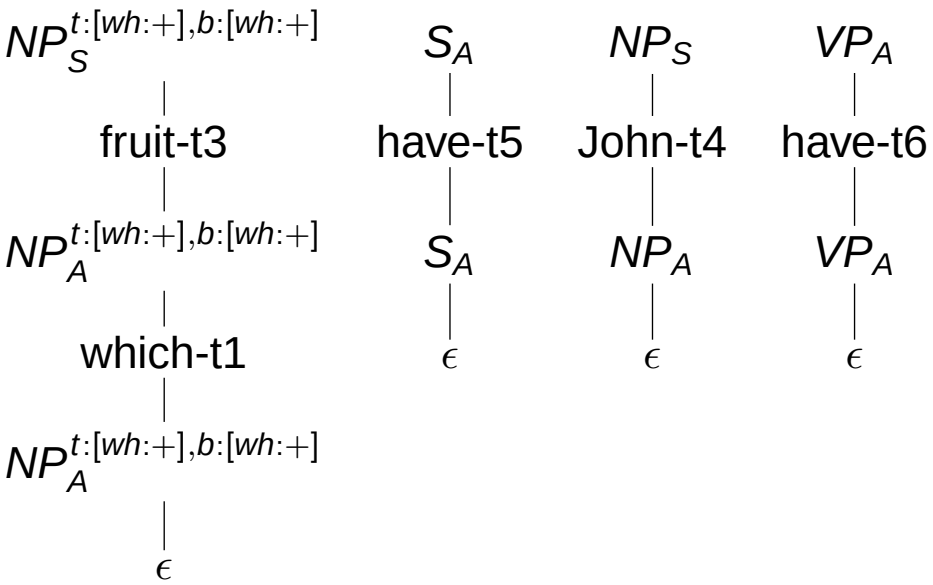
Global Polarity Filtering (Gardent and Kow 2005) filters out

- Sets of rules which cover the input
- but cannot possibly lead to a valid derivation
- either because a substitution node cannot be filled
- or because a root node fails to have a matching substitution site

Local (Structure-Driven) Polarity Filtering: on each local tree

Example of Local Polarity Filtering

$$
\begin{array}{lcll}
\times & S_S^{[t:T,b:B]} & \rightarrow & eat(S_A^{[t:T,b:B]}\ NP_S^{[t:[wh:-]]}\ VP_A) \\
\surd & S_S^{[t:T,b:B]} & \rightarrow & eat(S_A^{[t:T,b:B]}\ NP_S^{[t:[wh:-]]}\ VP_A\ NP_S) \\
\surd & S_S^{[t:T,b:B]} & \rightarrow & eat(S_A^{[t:T,b:B]}\ NP_S^{[t:[wh:+]]}\ S_A\ NP_S\ VP_A)
\end{array}
$$



Bottom-Up Generation and N-Gram filtering

For each local tree in the input, the rule sets passing the local polarity filter are tried out for combination.

Only *the n best scoring n-grams* let through after each bottom-up generation step are kept.
The language model helps finding the most likely ordering of modifiers.

Evaluation and Results

Test data: The SR Data

- Dependency trees derived from the Penn Treebank
- 26 725 inputs
- Average (maximum) word length: 22 (134)
- Average (maximum) branching degree: 4 (18)

Algorithms compared:

- Baseline: A strictly top-down algorithm
  (No time information available for systems participating in SR Task, only coverage and BLEU)
- SEQ: The SDL algorithm without parallelism
- PAR: The SDL algorithm with parallelism

Evaluation Focus: Efficiency (Time)

Evaluation and Results

	Sentences (Length L)			
	$S(0-5)$	$S(6-10)$	$S(11-20)$	$S(All)$
	Total	Total	Total	Total
	1084	2232	5705	13661
BL	**0.85**	10.90	110.07	–
SEQ	1.49	2.84	4.36	4.52
PAR	**1.53**	2.56	2.66	**2.57**

- Maximum arity = 3. Else BL times out.
- Many time out for BL on input longer than 10
- For short sentences (0-5), BL outperforms SDL
- For sentences with more than 5 words, SDL increasingly outperforms BL

Branching factor and Parallelism

	Sentences (Arity)			
	$S(1)$ Total 190	$S(3)$ Total 3619	$S(5)$ Total 2910	$S(6)$ Total 1093
SEQ	**0.89**	3.65	5.24	8.20
PAR	**0.97**	2.63	2.86	3.09

The impact of parallelism increases with the branching factor.

Coverage and BLEU score

Coverage: 81.74%

- No robustness mechanism added.

BLEU score: 0.73 (for covered data)

- No ranking module
- Best statistical system in SR Task: 0.88
- Best symbolic system in SR Task: 0.37

Error Mining as a way to Improve Grammar Correctness

February 10, 2014


S. Narayan and C. Gardent
Error Mining with Suspicion Trees
Proceedings of *COLING 2012*, pp 2011 - 2025,, Mumbai, India


C. Gardent and S. Narayan
Error Mining on Dependency Trees
Proceedings of *ACL 2012*, pp 592 - 600, Jeju Island, Korea

Error Mining (van Noord 2004)

Goal: Identify errors in grammar or lexicon which leads to parsing failure

Method:

- Parse n sentences (S)
- Divide the input set of sentences S into the set of sentences for which parsing succeeds (PASS) and the set of sentences for which parsing fails (FAIL)
- Identify n-grams or words that frequently occur in FAIL (high *Suspicion Score*)

$$S = \frac{ct(w_i \mid FAIL)}{ct(w_i)}$$

Error Mining applied to Generation

Input to generation = Unordered Dependency Trees

Search for subtrees (*Suspicious Forms*) in the input which frequently lead to generation failure and rarely lead to generation success (high *Suspicion Score*).

Error Mining applied to Generation

Three main modifications w.r.t. Van Noord's approach

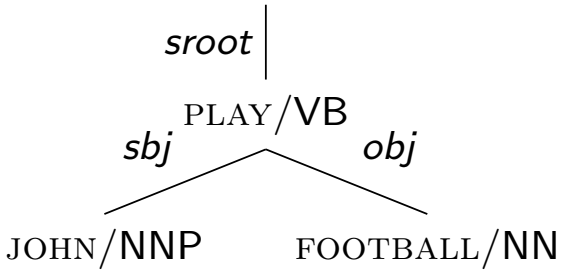
1. The input is a tree. Suspicious forms are subtrees (not strings)
We adapt the HybridTreeMiner algorithm to efficiently enumerate subtrees of the input.

2. The suspicion score takes into account both Pass and Fail (not just Pass).

3. The output of error mining is a tree (not a list).
This output tree highlights the relations between suspicious forms and facilitates grammar correction.

Example Suspicious Form

Suspicious forms are Suspicious subtrees of the SR dependency trees labelled with lemma, parts-of-speech and/or dependency information



WORD	(PLAY, (JOHN), (FOOTBALL))
POS	(VB, (NNP), (NN))
dep	(*sroot*, (*sbj*), (*obj*))
WORD/POS	(PLAY/VB, (JOHN/NNP), (FOOTBALL/NN))
dep-POS	(*sroot*-VB, (*sbj*-NNP), (*obj*-NN))

Enumerating Subtrees

HybridTreeMiner algorithm (Chi et al. 2004): Build *an enumeration tree* whose nodes are all possible subtrees of T and such that, at depth d of this enumeration tree, all possible frequent subtrees consisting of d nodes are listed.

1. Convert the unordered labelled trees to a canonical form called BFCF (Breadth-First Canonical Form)

2. Enumerate the subtrees of the BFCF trees in increasing order of size using two tree operations called join and extension

Adpating the HybridTreeMiner algorithm for Error Mining

Use **support** (nb of FAIL and PASS sentences for a given form f) and suspicion score to prune the search space.

for a larger tree t to be added to the enumeration tree, the suspicion score of all subtrees contained in a new tree t must be smaller or equal to $S(t)$.

$$S(f_n) \geq S(t_{n-1}), \forall t_{n-1} \in t_n$$

Construct the tree breadth first rather than depth first

The enumeration process take 10-15 minutes for a dataset of 123,523 trees.

Suspicion Score, $S_{score}(f)$

Captures the degree to which a form f is associated with failure. It is high when f is *often present in data associated with failure* and/or *often absent in data associated with success*.

$$S_{score}(f) = \frac{1}{2}(\text{Fail}(f) * \ln count(f) + \text{Pass}(\neg f) * \ln count(\neg f))$$

$$\text{Fail}(f) = \frac{count(f|\text{FAIL})}{count(f)}$$

$$\text{Pass}(\neg f) = \frac{count(\neg f|\text{PASS})}{count(\neg f)}$$

Constructing the output Suspicion Tree

We adapt the ID3 decision tree algorithm to yield a tree whose nodes are suspîcious forms and whose structure highlights the relations between suspicious forms .

The Error Mining algorithm recursively partitions the data by

1. selecting the suspicious form with highest suspicion score (*attribute selection*).
2. using this attribute to **split the data** into two subsets, a subset containing that attribute and a subset excluding that attribute (*dataset division*).

Pruning the Suspicion Tree

1. Only consider those forms whose frequency is above a given threshold.

2. Only consider a larger suspicious form if its suspicion rate is larger than the suspicion rate of all smaller forms it contains.

3. Limit depth of tree (max 10).

Building a suspicion tree for a dataset of 123,523 trees takes about one minute.

Experiments and Results

We applied error mining to the results of our SR algorithm on the SR data.

- Corrections:
  - 11 rewrite rules (Gen-1, Dt-4, Adv-1, Inf-3, Aux-1 and Final-1), to adapt the SR data to the input expect by our generator
  - 2 grammar corrections and
  - a few lexicon updates

Test Data	# Failures Before EM	# Failures after EM	Error Reduction
26725	19280 (*72.1%*)	5157 (*19.3%*)	-52.8

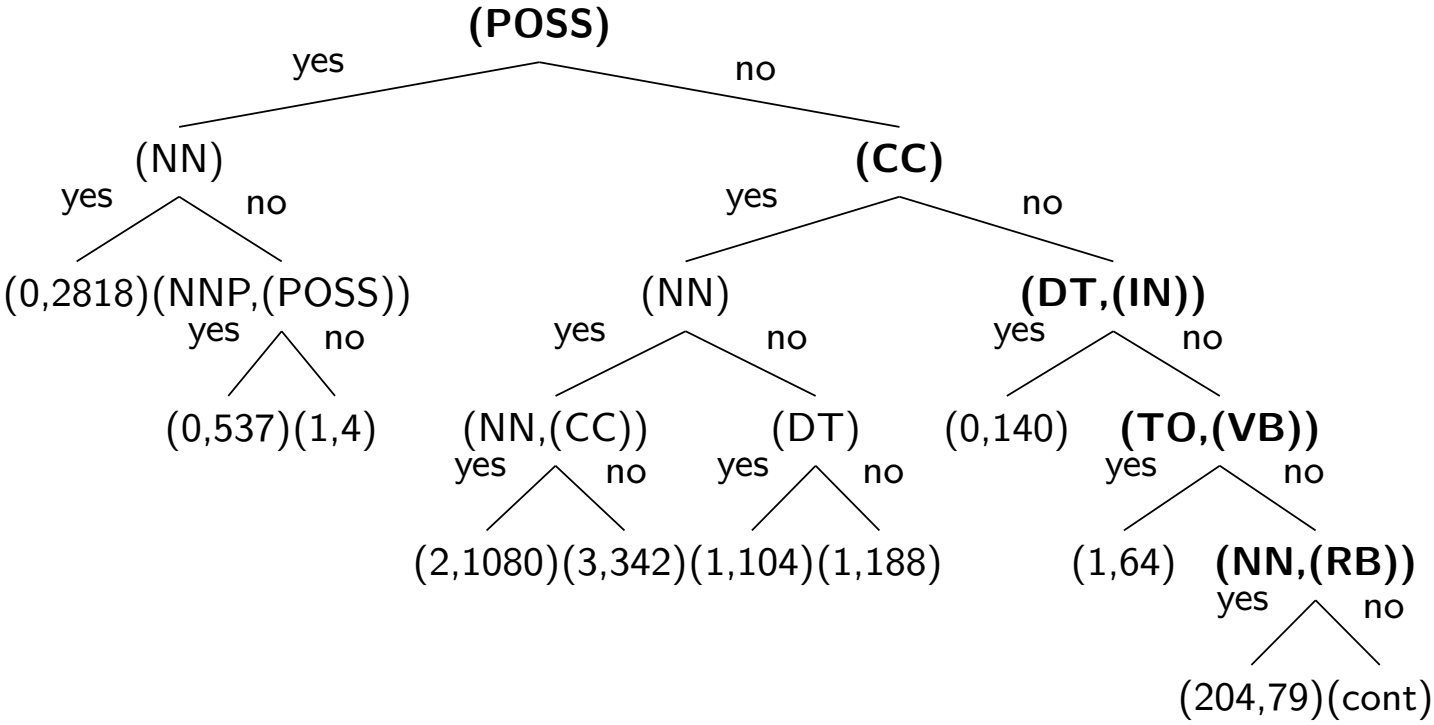
- sentences from minimum size 1 to maximum size 134 with the average size 22 with
- the coverage of 81.74% and the BLEU score 0.73 (for the covered data)

Suspicion tree and Grammar Correction

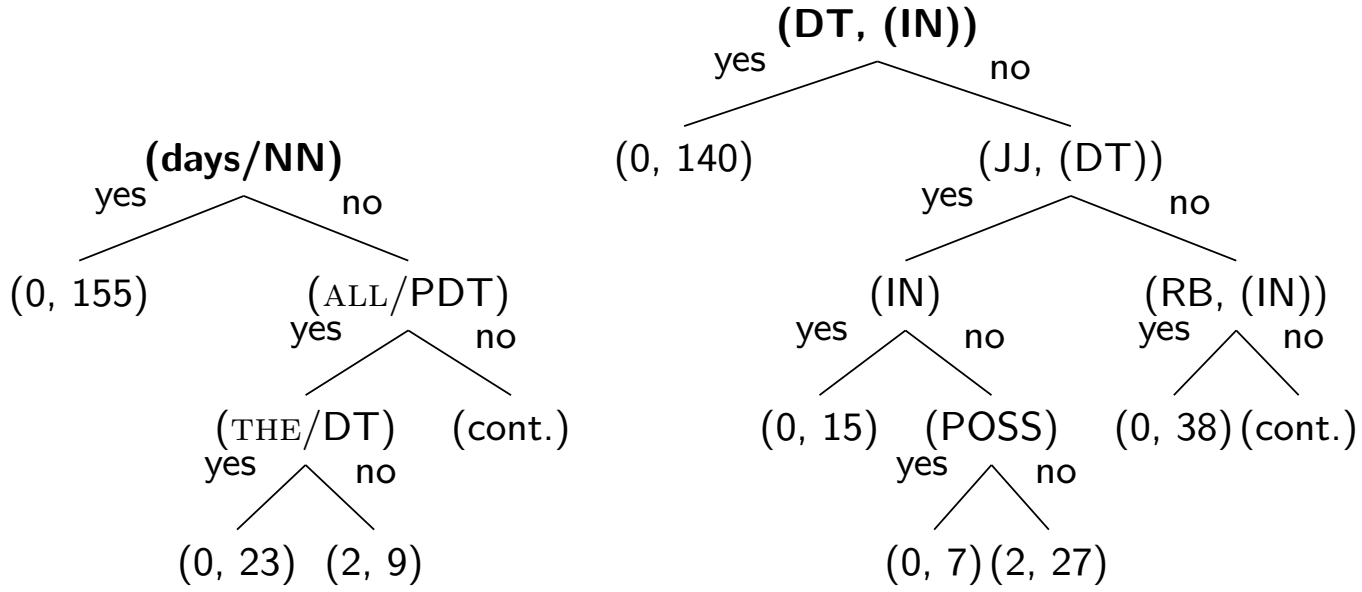
The structure of the suspicion tree helps the linguist correect the grammar by

- ordering suspicious forms from the most to the least suspicious
- showing forms that are suspicious independently of context and require a single correction
- showing forms that are suspicious independently of context and require several corrections (several subcases)
- showing forms that are suspicious in some but not all contexts

From the most to the least Suspicious Form

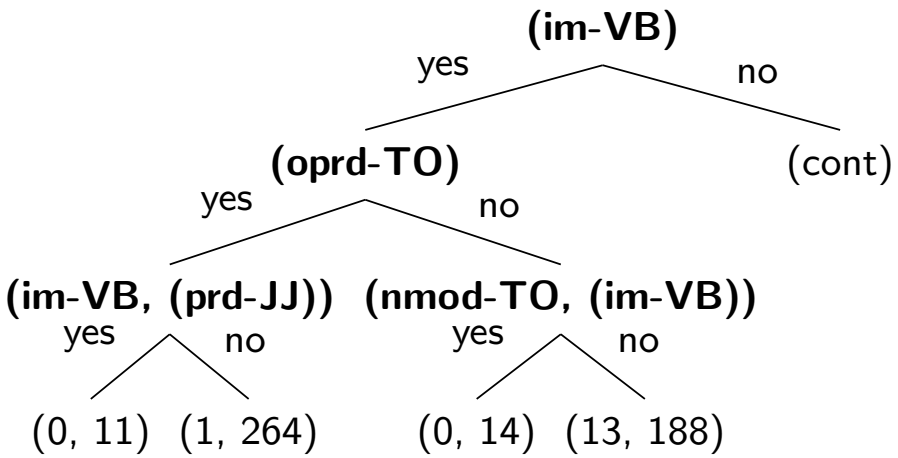


Forms that are suspicious independently of context and require a single correction



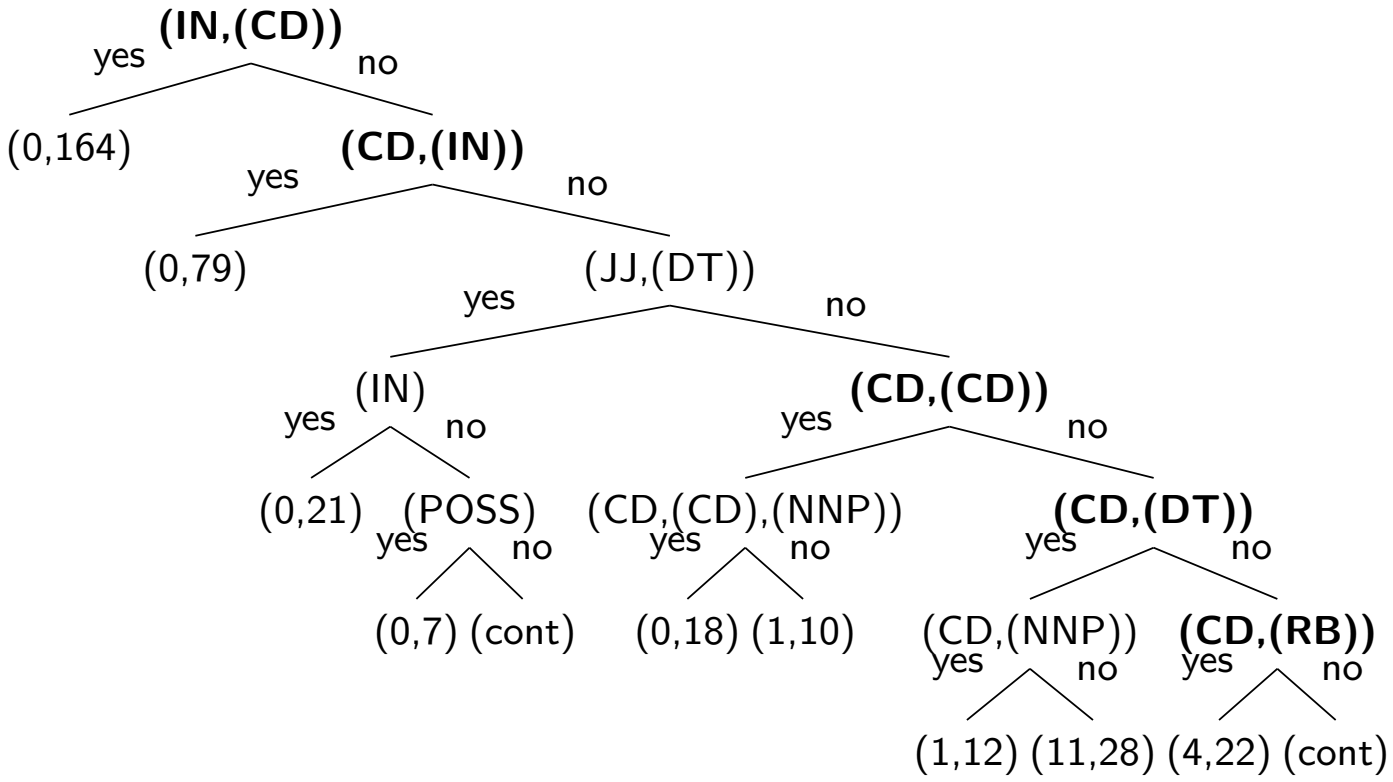
days/NN. Incorrect TAG family. Lexicon update
(DT,(IN)) e.g., *some of us*. DT in Input, PRP in TAG. Rewrite Input

Forms that are suspicious independent of context but require several corrections



im-VB: Infinitival verbs.
oprd-TO: Infinitival verbs which are complement of a control or raising verb.
im-VB,(prd-JJ): Infinitival verbs subcategorising for an adjectival complement.
im-VB: Infinitival verbs modifying a noun.

Forms that are suspicious in some but not all contexts



CD: noun or determiner. Determiner but not noun in TAG lexicon.

N.B., CD does not appear in suspicion tree as it generates fine when used

Conclusion

- Proposed a novel approach to error mining which supports a linguistically meaningful error analysis.
- Permits quickly identifying the main sources of errors while providing a detailed description of the various subcases of these sources if any.
- We applied it to the analysis of undergeneration in a grammar based surface realisation algorithm.
- The approach is generic in that permits mining trees and strings for suspicious forms of arbitrary size and arbitrary conjunctions of labelling.

Using Surface Realisation to Generate Grammar Exercises

February 10, 2014

C. Gardent and L. Perez-Beltrachini
Using FB-LTAG Derivation Trees to Generate Transformation-Based Grammar Exercises
Proceedings of *TAG+11*, 2012, Paris, France

L. Perez-Beltrachini, C. Gardent and G. Kruszewski
Generating Grammar Exercises
Proceedings of *The 7th Workshop on Innovative Use of NLP for Building Educational Applications*, NAACL-HLT Workshop, Montreal, Canada, June 2012.

Grammar Exercises

Built from a single sentence.

[FIB] Complete with an appropriate personal pronoun.

(S) *Elle adore les petits tatous*

(She loves the small armadillos)

(Q) _____ adore les petits tatous (gender=fem)

(K) elle

[Shuffle] Use the words below to make up a sentence.

(S) *Tammy adore les petits tatous*

(Tammy loves the small armadillos)

(Q) tatous / les / Tammy / petits / adore

(K) Tammy adore les petits tatous.

Grammar Exercises

Built from a pair of syntactically related sentences

[Reformulation] Rewrite the sentence using passive voice

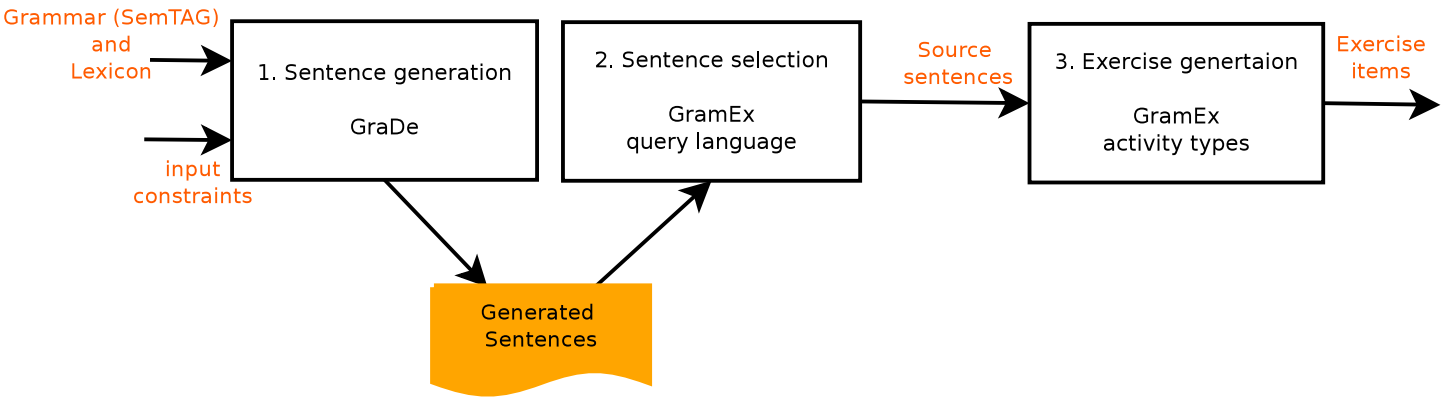
(Q) C'est Tex qui a fait la tarte.
(It is Tex who has baked the pie.)

(K) C'est par Tex que la tarte a été faite.
(It is Tex by whom the pie has been baked.)

Active/Passive, NP/Pronoun, Assertion/Wh-Question, Assertion/YN-Question

SemTAG based generation for language learning

The *GramEx* framework: selecting sentences and building exercises

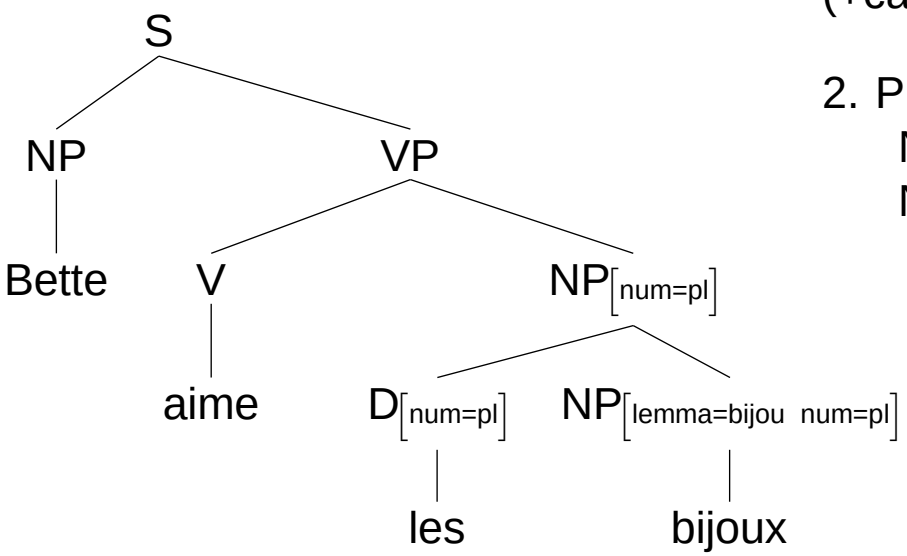


Creating a grammar exercise

Pedagogical goal: Plural form of irregular no
Exercise type: Fill-in-the-blank.

⇓

Bette aime le bijou.
C'est Bette qui aime les bijoux.
Bette aime les bijoux. ✓



$\{CanonicalObject, CanonicalSubject, ActiveVerb\}$

1. Select sentences
   $\Rightarrow$ NP[$num = pl \& plural = irreg$]
   (+canonical order)

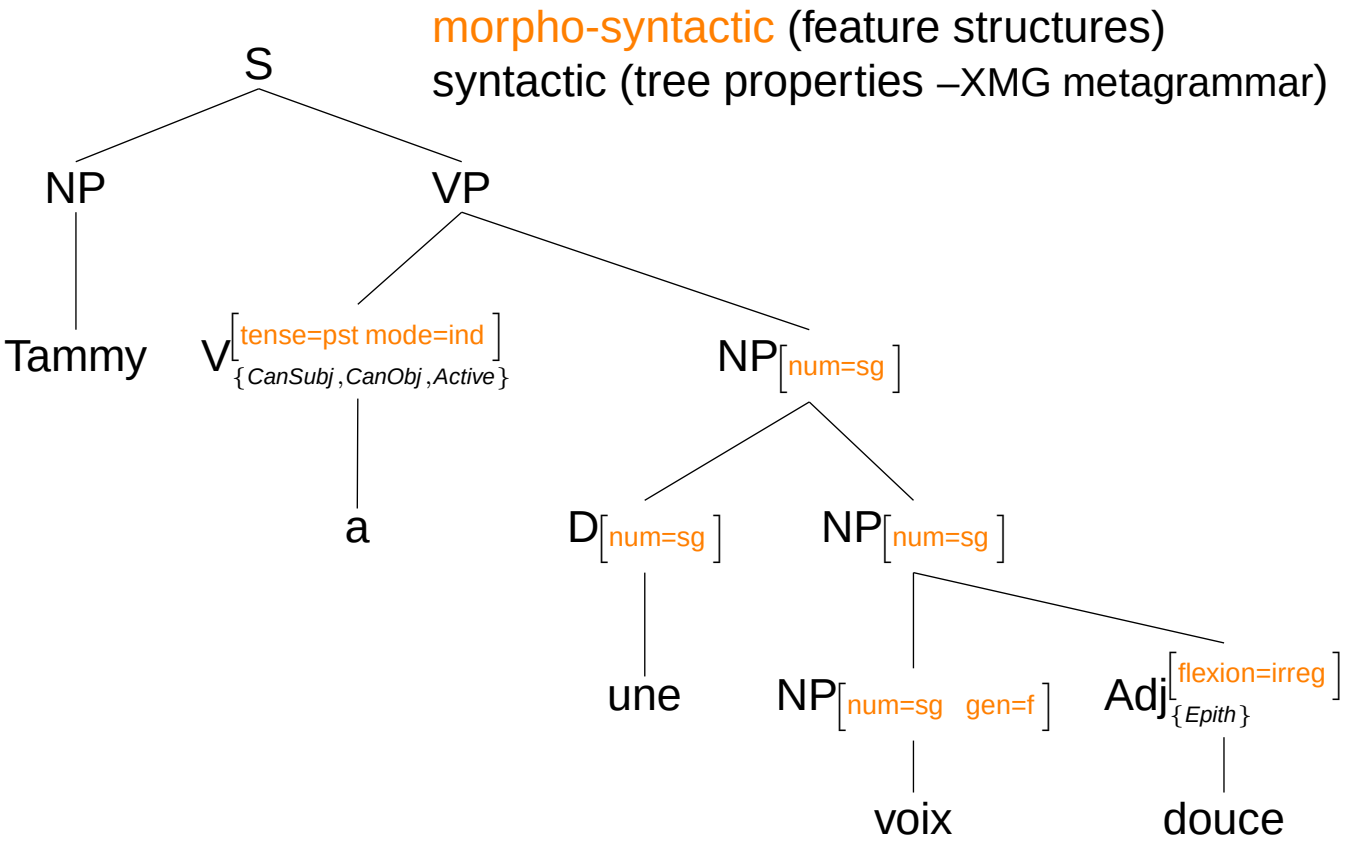
2. Process the selected sentence
   NP[$num = pl$] $\Rightarrow$ blank
   NP[$lemma = bijou$] $\Rightarrow$ hint

⇓

(S) Bette aime les bijoux.
(Q) Bette aime les _____. (bijou)
(K) bijoux

Selecting appropriate sentences

GramEx's boolean constraint language: Signature



Selecting appropriate sentences

GramEx's boolean constraint language: syntax and use

- Boolean constraint language:
  - conjunction, disjunction and negation of morpho-syntactic and syntactic properties
- Describe the linguistic requirements of pedagogical goals
  - $\Rightarrow$ linguistic characterization of appropriate source sentences

Selecting appropriate sentences

GramEx's boolean constraint language: an example

Pedagogical goal: *Pre/post nominal irregular adjectives*

[Epith $\wedge$ flexion: irreg]

✓ *Tammy a une voix douce* (Tammy has a soft voice)

X *Tammy a une jolie voix* (Tammy has a nice voice)

Pedagogical goal: *Prepositions with infinitives*

POBJinf $\wedge$ CLAUSE

POBJinf $\equiv$ (DE-OBJinf $\vee$ A-OBJinf)

CLAUSE $\equiv$ Vfin $\wedge \neg$Mod $\wedge \neg$CCoord $\wedge \neg$Sub

✓ *Tammy refuse de chanter* (Tammy refuses to sing)

X *Jean dit que Tammy refuse de chanter* (John says that Tammy refuses to sing)

Transformation-based grammar exercices

Finding syntactically related sentences (e.g. active/passive)

(Q) C'est Tex qui a fait la tarte.

(It is Tex who has baked the pie.)

X *(K) Tex a fait la tarte.*

(Tex has baked the pie.)

X *(K) La tarte a été faite par Tex.*

(The pie has been baked by Tex.)

X *(K) C'est par Tex que la tarte sera faite.*

(It is Tex who will bake the pie.)

X *(K) Est-ce que la tarte a été faite par Tex ?*

(Has the pie been baked by Tex ?)

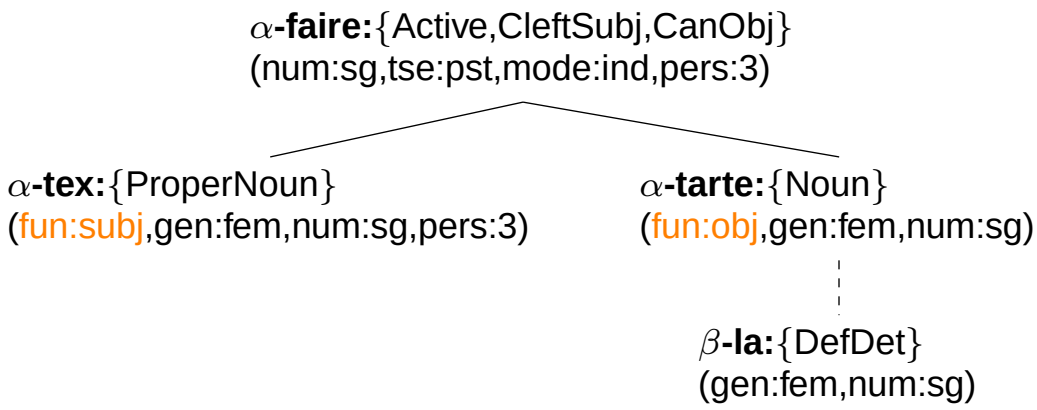
✓ *(K) C'est par Tex que la tarte a été faite.*

(It is Tex by whom the pie has been baked.)

Creating transformation-based grammar exercises

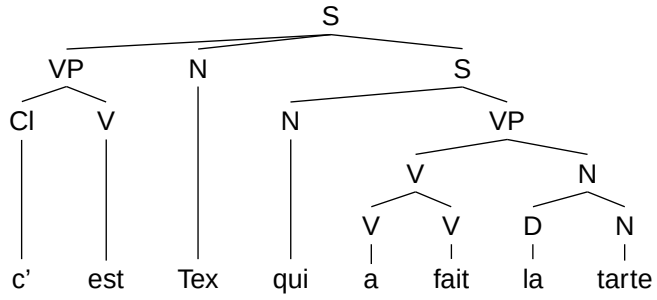
- To identify pairs of sentences that are identical up to a single syntactic transformation:
  - Use the information contained in SemTAG derivation trees
  - Define tree filters on pairs of SemTAG derivation trees
  - Retrieve sentences pairs that match those tree filters

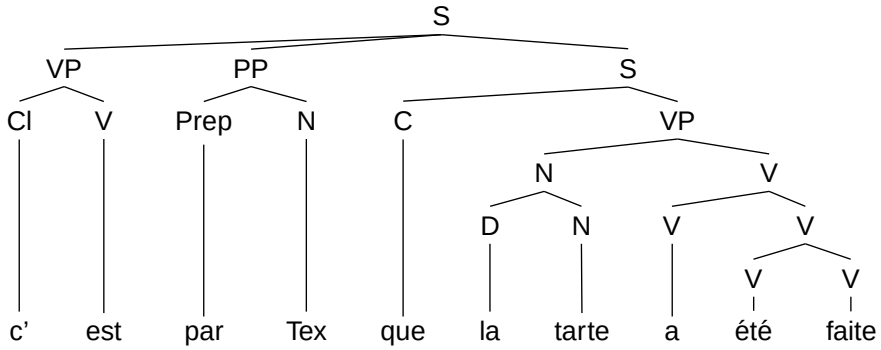
Why SemTAG derivation trees?

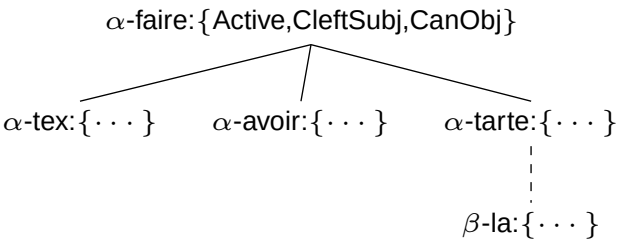


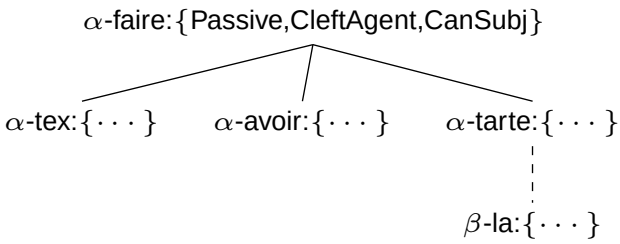
- Detailed syntactic information
- Informational content of the sentence

Why SemTAG derivation trees?





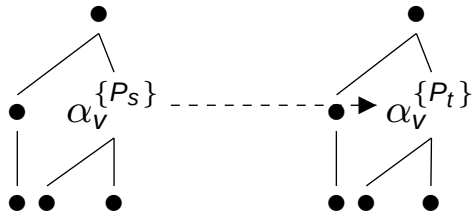




- More abstract description than derived trees

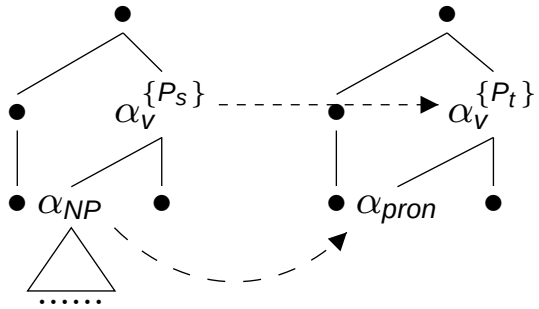
Derivation Tree Filters

Tree filter types



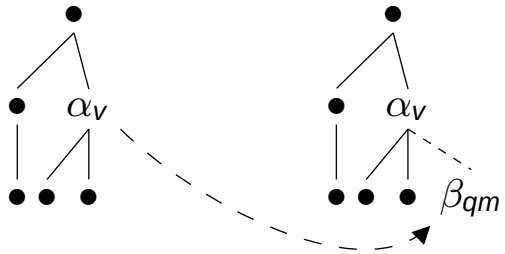
e.g. `active/passive`

$\bullet_s\{\text{Active,CleftSubj,CanObj}\}$
$\leftrightarrow \bullet_t\{\text{Passive,CleftAgent,CanSubj}\}$



e.g. `NP/Pronoun`

$\bullet_s\{\text{CanSubj}\} \leftrightarrow \bullet_t\{\text{CliticSubj}\}$



e.g. `Assertion/YN-Question`

$\varnothing \leftrightarrow \bullet_q\{\text{questionMark}\}$

Meaning Preserving Transformations

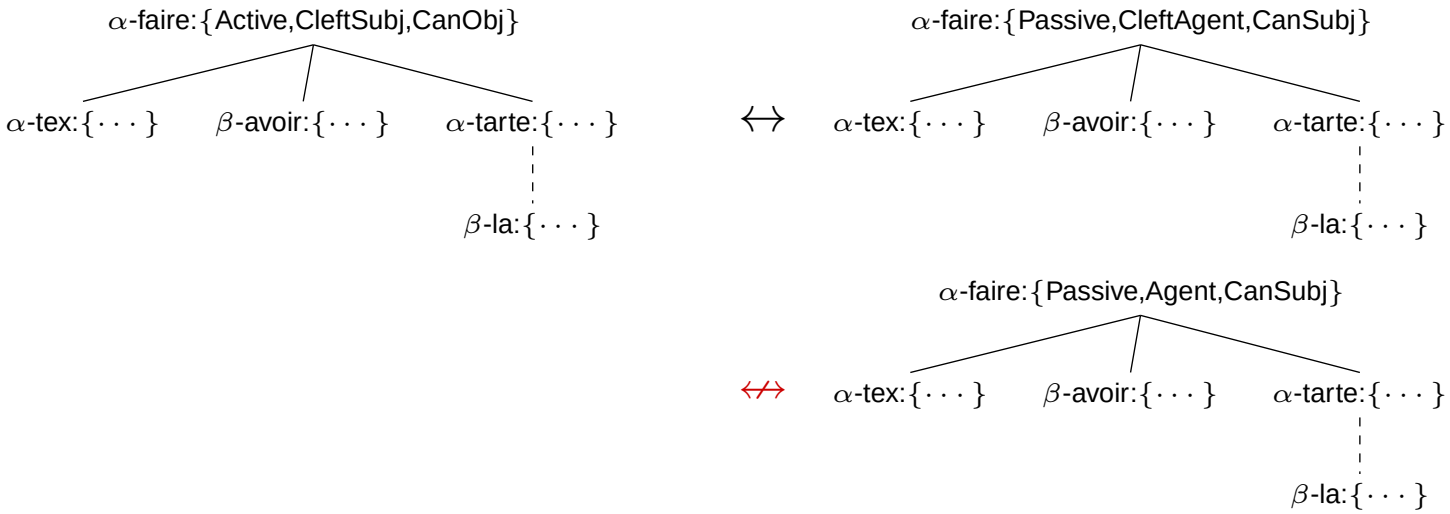
Same core meaning (e.g. active/passive)

(Q) *C'est Tex qui a fait la tarte.* $\leftrightarrow$ (K) *C'est par Tex que la tarte a été faite.*

(It is Tex who has baked the pie) (It is by Tex that the pie has been baked)

$\not\leftrightarrow$ (K) *La tarte a été faite par Tex.*

(The pie has been baked by Tex)



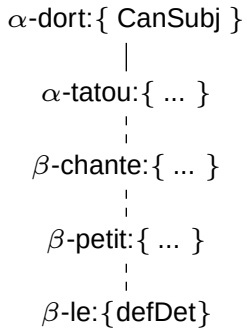
$\not\leftrightarrow$

$\bullet_s\{\text{Active,CleftSubj,CanObj}\}$

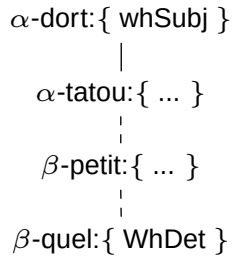
$\leftrightarrow \bullet_t\{\text{Passive,CleftAgent,CanSubj}\}$

Meaning Altering Transformations

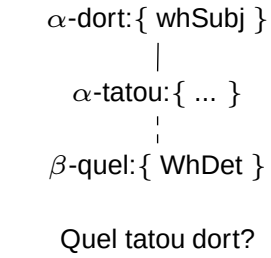
Related core meaning: content deleted, added or replaced
(e.g. Assertion/Wh-Question)



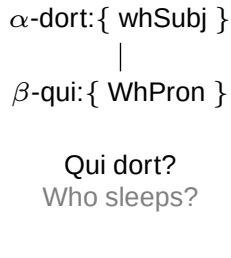
Le petit tatou qui chantera dort.
The small armadillo that will sing sleeps



Quel petit tatou dort?
Which small armadillo sleeps?



Quel tatou dort?
Which armadillo sleeps?



Qui dort?
Who sleeps?

Main results

- Correctness and productivity
  - Manual annotation of a sample of generated exercises
    - using SemFraG and lexicon tailored to *Tex's French Grammar* vocabulary
    - around 80% of the automatically generated exercises are correct
  - 52 input formulae $\Rightarrow$ around 5000 exercises
- Use *GramEx* to generate exercises for I-FLEG (serious game) and WFLEG
  - I-FLEG has been evaluated with 15 students at Saarbrucken University who played during 40 minutes solving grammar exercises.

Thanks!