Improving Natural Language Computational Grammars

Claire Gardent[1]
(Joint work with German Kruszewski[2] and Shashi Narayan[3])

(1) CNRS/LORIA, Nancy, (2) INRIA/LORIA, Nancy, (3) UDL/LORIA, Nancy

October 8 2012, Orléans University

Computational Grammars for Natural Language

- Explicit description of the syntax (and semantics of natural language (English, French, Chinese, etc.)
- Linguistic framework: HPSG, LFG, CCG, **TAG**
- Developed semi-automatically by compilation (XMG) or induction (from some existing treebank)
- Large and complex
- Contain errors, gap and inconsistencies

Improving Grammars

Two main approaches for improving grammars

Grammar-based approach

- uses Grammar Engineering techniques
- permits detecting gaps, errors and inconsistencies

Corpus-based approach

- uses Error Mining techniques
- permits identifying the most likely sources of errors

Grammar-Based Evaluation

How?

Grammar Traversal Algorithm (GraDE, Grammar Debugger)

Focus

Systematic exploration of the Grammar and of its Linguistic Coverage

- What sentences does the grammar generate?
- How much does is over- (and under-) generate?
- Can all rules in the grammar be used in at least one derivation?
- Are all possible syntactic realisations of the verb and of its arguments generated and correct?
- Does the grammar correctly capture the interactions between basic clauses and modifiers?
- etc.

Grammar Based Approach

The GraDe algorithm

Experiment: Applying GraDe to SemTAG

Results and Discussion: Grammar Analysis

The GraDe algorithm

Top-Down Grammar Traversal

Outputs the sentences generated by the grammar

User-Defined parameters control the search to ensure

- termination and
- a linguistically guided, systematic, grammar exploration.

Ensuring Termination

Time Out: the process halts when the time bound is reached

Recursion Bound: For each type of recursive rule. The type is the category of the rule lhs e.g., N, NP, S, V, VP, S.

Derivation Depth

Controlling Linguistic Coverage

Number and Type of modifiers
E.g., *1 noun and 1 vp modifier*

Root rule: subcategorisation type; type (recursive or non recursive); "linguistic" features (argument types, voice, etc.)
E.g., *sentences whose main verb is an intransitive verb in the active voice combining with a canonical subject*

Input semantics: Full or Under-specified (core)

(1) a. {`run_v(E M), man_n(M)`}
The man runs, The man ran, A man runs, A man ran, This man runs, My man runs, etc..

b. {`semantics:[A:the_q(C RH SH) B:indiv(C f sg) qeq(RH B) B:man_n(C) G:run_v(E2 C) G:event(E2 pst indet ind)]`}
The man runs

Implementation (GraDe)

Top Down Grammar Traversal of SemTAG with user-defined parameters for controling termination and linguistic coverage

- Conversion SemTAG ($\rightarrow$ Feature-Based Regular Tree Grammar) $\rightarrow$ Definite Clause Grammar (DCG) (Gardent et al. 2011)
- Control: Prolog Constraints in the DCG rules
- Traversal: Depth-First, Left-to-Right traversal of the grammar (DCG)

The SemTAG Grammar

SemTAG: Feature-Based Lexicalised Tree Adjoining Grammar for **French** equipped with a Unification-Based Compositional Semantics (MRSs)

Syntax developed and tested in Parsing Mode [Crabbé 2005]

Core fragment with extensive coverage of verbs and basic coverage of other categories

Grammar Analysis

- Grammar Coherence: Can all rules in the grammar be used so as to derive a constituent?
- Functor/Argument Realisations: Are all possible syntactic realisations of a verb and of its arguments generated and correct?
- Interactions between basic clauses and modifiers
- Morphological and syntactic variants of a given core semantics: Does the grammar correctly account for all such variants ? Are all generated variants correct?

XP1: Checking for Grammar Coherence

For each grammar rule anchored by a verb, can we find at least one derivation?

```
family: n0V
max_adjunctions:
     {N: 1, NP: 0, V:1, VP: 1, ADJ: 0, S: 0}
cat: s
max_results: 1
type: initial
features: [mod:ind,cat:v]
```

XP1: Checking for Grammar Coherence

Tree Family	Trees	Fails	Fails/Trees
CopulaBe	60	1	1%
ilV	2	0	0%
n0V	10	0	0%
n0ClV	9	0	0%
n0ClVn1	45	2	4%
n0ClVden1	36	3	8%
n0ClVpn1	29	3	10%
n0Vn1	84	3	3%
n0Vn1Adj2	24	6	25%
n0Van1	87	3	3%
n0Vden1	38	3	7%
n0Vpn1	30	3	10%
ilVcs1	2	0	0%
n0Vcs1	30	23	74%
n0Vas1	15	10	66%
n0Vn1Adj2	24	0	0%
s0Vn1	72	9	12%
n0Vs1int	15	12	80%
n0Vn1n2	24	0	0%
n0Vn1an2	681	54	7%

Table: Checking for Gaps in the Grammar

Approximately 10% of the grammar rules cannot yield a derivation.

XP2: Functor/Argument Dependencies

Which syntactic realisations does the grammar generate for a given syntactic functor (verb type) ?

```
family: n0V
adjunctions:
     {N: 1, NP: 0, V:1, VP: 0, ADJ: 0, S: 0/1}
cat: s
max_results: all
features: [mod:ind]
```

Example Output (family: n0V)

Elle chante (She sings), La tatou chante-t'elle? (Does the armadillo sing?), La tatou chante (The armadillo sings), Chacun chante -t'il (Does everyone sing?), Chacun chante (Everyone sings), Chante-t'elle? (Does she sing?) Chante -t'il? (Does he sing?), Chante! (Sing!), Quel tatou chante ? (Which armadillo sing?), Quel tatou qui chante dort? (Which armadillo who sings sleep?) Tammy chante-t'elle? (Does Tammy sing?), Tammy chante (Tammy sings), une tatou qui chante chante (An armadillo which sings sings), C'est une tatou qui chante (It is an armadillo which sings), ...

Output

55 distinct MRSs, 65 distinct sentences of which only 28 were correct.

Examples of incorrect cases

(2) a. Chacun chante-t'elle? (*Everyone sings?*)
The agreement between the inverted subject clitic and the subject fails to be enforced

b. Chantée chacun? (*Sung everyone?*)
The inverted nominal subject fails to require a verb in the indicative mode

c. La tatou qui chante-t'elle? (*The armadillo which does she sing?*)
the inverted subject clitic fails to be disallowed in embedded clauses

XP3: Interaction with Modifiers

Does the grammar correctly account for the interactions between basic clauses and modifiers ?

Query GraDE for derivations rooted in n0V (intransitive verbs) and with 1N, 2N, 1V or 1VP adjunction.

0	1S	1VP	1V	1N	2N
36	170	111	65	132	638

Figure: Number of sentences output per query

Interaction with Modifiers

The inverted subject clitic fails to be constrained to occur directly after the verb:

(3) Semble-t'il chanter? / * Semble chanter t'il? (*Does he seems to sing?*)

The order and compatibility of determiners are unrestricted:

(4) * Un quel tatou, *Quel cette tatou, Ma quelle tatou (*Un which armadillo, Which this armadillo, My which armadillo*)

XP4: Morphological and Syntactic Variants

Does the grammar correctly account for the morphological and syntactic variants of a given core semantics ?

```
semantics: {run(E M), man(M)}
output: all
family: VERB_FAMILY
max_adjunctions:
     {N: 2, NP: 0, V:0, VP: 0, ADJ: 0, S: 1}
cat: v
features: [mod:ind]
```

Producing Variants

Tree Family	MRS	Sent.	S/MRS
ilV	7	52	7.4
n0V	65	161	2.4
n0ClV	30	62	2.0
n0ClVn1	20	25	1.25
n0ClVden1	10	15	1.5
n0ClVpn1	40	63	1.57
n0Vn1	20	110	5.5
n0Van1	30	100	3.33
n0Vden1	5	15	3.00
n0Vpn1	25	76	3.04
ilVcs1	1	1	1.00
n0Vcs1	200	660	3.3

Conclusion

GraDe permits exploring the sentences generated by a grammar from different viewpoints:

- to find gaps or inconsistencies in the rule system;
- to systematically analyse the grammar account of functor/argument dependencies;
- to explore the interaction between base constructions and modifiers;
- and to verify the completeness and correctness of syntactic and morphological variants.

Strengths and Weaknesses

Grammar Based Evaluation

- Permits analysing the consistency, the linguistic coverage and the accuracy of the grammar
- Requires a manual analysis of the sentences output by GraDE
- Provides only weak support for detecting under-generation. Constructs not handled by the grammar are easier to identify by using the grammar to parse or generate text.

Corpus-Based Evaluation (Error Mining)

How?

- Using Error Mining Techniques
- Generalised to trees (rather than n-grams)
- Applied to generation (rather than parsing)

Focus

Identify the most likely sources of errors (*Suspicious Forms*)

Error Mining using Parsing

(van Noord, ACL 2004)

- 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*)

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

Generalising to Trees

(Gardent and Narayan, ACL 2012)

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*).

$$S(f) = \frac{1}{2}\left(\frac{\#(f \mid FAIL)}{\#(f)} * \log \#(f) + \frac{\#(f \mid PASS)}{\#(\neg f)} * \log \#(\neg f)\right)$$

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.

Adpating the HybridTreeMiner algorithm for Error Mining

Use 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

Applying Error Mining Using Generation

(Gardent and Narayan, ACL 2012)

- Input to generation: a set of Unordered Shallow Dependency Trees (T) provided by the Generation Challenge's Surface Realisation Task
- Generate from T using a sentence generator based on XXTAG, an XTAG grammar for **English** developed using XMG (Alahverdzhieva, 2007)
- Divide T into the set of inputs for which generation succeeds (PASS) and the set of inputs for which generation fails (FAIL)
- Identify subtrees (in the input) that frequently occur in FAIL and rarely in PASS (high *Suspicion Score*)
  $S(f) = \frac{1}{2}(\frac{\#(f|FAIL)}{\#(f)} * \log \#(f) + \frac{\#(f|PASS)}{\#(\neg f)} * \log \#(\neg f))$

Mining for Various Trees and Labels

Our EM algorithm supports **multiple views on the data**. It permits mining the data for:

- tree patterns of arbitrary size
- using different types of labelling information (POS tags, dependencies, word forms and any combination thereof)

Mining Trees of Size 1 with a single Label

Example 1: Mining on POS tags alone (displaying only trees of size 1 sorted by decreasing suspicion rate). Permits identifying suspicious *syntactic categories*. POS (possessive determiner e.g., *his, John's*) and CC (coordination).

POS	Sus	Sup	Fail	Pass
POS	0.99	0.38	3237	1
CC	0.99	0.21	1774	9
CD	0.39	0.16	1419	2148
NNP	0.35	0.32	2749	5014
NN	0.30	0.81	6798	15663
IN	0.30	0.16	1355	3128
DT	0.09	0.12	1079	10254

Mining Trees of Size 1 with a single Label

Example 2: Mining on words alone. Permits identifying suspicious *words*.

- Most suspicious form: *\$* (Sus=1)
  Assigned the POS tag \$ in the input data, a POS tag which is unknown to our system and not documented in the SR Task guidelines

Example 3: Mining on dependencies alone. Permits identifying suspicious *constructs*.

- APPOS: appositions, $S = 0.19$
- TMP: temporal modifiers, $S = 0.54$
- DEP: “others”, $S = 0.61$

Mining on Trees of Arbitrary Depth

Permits characterising the *context* in which a suspicious form occurs.

Dataset: NP chunks

TP1	**cd**(IN,RBR)	*more than 10*
TP2	IN(**cd**)	*of 1991*
TP3	NNP(**cd**)	*November 1*
TP4	CD(NNP(**cd**))	*Nov. 1, 1997*

Problem: in our lexicon, cardinals are classified as determiners not nouns. Thus they fail to combine with prepositions (e.g., *of 1991, than 10*) and with proper names (e.g., *November 1*).

Improving the Generator

Suspicious forms point to various types of errors inducing different types of modifications.

- Mismatches between input provided and input format expected by the generator (e.g., POS).
  Fixed by rewriting the input to the expected format
- Missing or erroneous lexical entries.
  Modify lexicon (e.g., cardinals classified both as determiners and as noun)
- Errors in the grammar
  Modify grammar (e.g., coordination failure due to error in metagrammar that propagates to all coordination trees)
- Error in the generation algorithm
  Modify algorithm (e.g., unification failure for some cases of multiple adjunctions)

The Impact of Error Mining on Generation

	NP-4	NP-ALL	S-6	S-ALL
Input Data	23468	123523	3877	26725
Init Fail	25.3%	21.7%	-	-
NP-Final	2.7%	13.0%	44.0%	72.1%
S-Final	-	-	10.3%	61.5%
Final	2.1%	11.3%	9.5%	61.2%
Decrease	**23.2**	**10.4**	**33.7**	**10.9**

Remaining error cases: Coordination, Verbs with particles, Verbs with Sentential Arguments

Ongoing and Future Work

Improve GraDe Efficiency: Yapp tabling vs NLP tabular algorithms

Combine GraDE with Error Mining. Use language models to automatically sort bad from good output.

Improve and test both grammars (French and English)

- Generating Grammar Exercices. High Precision and extensive linguistic coverage. Limited Lexicon. Core Syntax.
- SR Task: High Precision (BLEU score) and large lexical and syntactic coverage. Large lexicon. Arbitrary Corpus syntax.

Merci!