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)
- ...

$$\exists x. (\text{Man}(x) \land \exists y. (\text{Apple}(y) \land \text{eat}(e, x, y) \land \text{now}(e)))$$
$$\Rightarrow \text{A man eats an apple}$$
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

(a) Derived tree

(b) Derivation tree
Grammar-Based Surface Realisation Algorithms

Two main approaches

<table>
<thead>
<tr>
<th>Head-Driven algorithm</th>
<th>Used for recursively structured input data e.g., logical formulae</th>
<th>Use input structure to guide the search</th>
<th>Cons: Logical Form Equivalence Problem</th>
</tr>
</thead>
</table>

| 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

NP | John
   V | runs
   V | often
   VP | 
   NP | 
   S  |
   VP | 
   V  | 
   VP | 

r1. $NP_S \rightarrow \textit{john}(NP_A)$
r2. $S_S \rightarrow \textit{runs}(S_A NP_S VP_A V_A)$
r3. $VP_A \rightarrow \textit{often}(VP_A)$
r4. $NP_A \rightarrow \epsilon$
r5. $S_A \rightarrow \epsilon$
r6. $V_A \rightarrow \epsilon$
r7. $VP_A \rightarrow \epsilon$
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:+} NP_{wh:-}

Det NP^{NA} Det NP^{NA} NP NP

which the fruit John

S V S^{NA} V VP^{NA} V NP

have have have eaten eaten eaten

NP_{wh:-} VP

NP_{wh:+} NP

VP NP

V NP
... and the corresponding FB-RTG

\[
\begin{align*}
NP_A^{[t:T]} & \rightarrow \text{which}(NP_A^{[t:T,b:[wh:+]]}) \\
NP_A^{[t:T]} & \rightarrow \text{the}(NP_A^{[t:T,b:[wh:-]]}) \\
NP_S^{[t:T]} & \rightarrow \text{fruit}(NP_A^{[t:T]}) \\
NP_S^{[t:T]} & \rightarrow \text{John}(NP_A^{[t:T]}) \\
S_A^{[t:T]} & \rightarrow \text{have}(S_A^{[t:T]}) \\
VP_A^{[t:T,b:B]} & \rightarrow \text{have}(S_A^{[t:T,b:B]} \text{NP}_S^{[t:[wh:-]]} \text{VP}_A \text{NP}_S) \\
S_S^{[t:T,b:B]} & \rightarrow \text{eat}(S_A^{[t:T,b:B]} \text{NP}_S^{[t:[wh:-]]} \text{VP}_A) \\
S_S^{[t:T,b:B]} & \rightarrow \text{eat}(S_A^{[t:T,b:B]} \text{NP}_S^{[t:[wh:-]]} \text{VP}_A \text{NP}_S) \\
S_S^{[t:T,b:B]} & \rightarrow \text{eat}(S_A^{[t:T,b:B]} \text{NP}_S^{[t:[wh:+]]} S_A \text{NP}_S \text{VP}_A) \\
X_A^{[t:T,b:T]} & \rightarrow \epsilon
\end{align*}
\]
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:

\[
\begin{align*}
S_{ST}^{[t:T,b:B]} & \rightarrow \text{eat}(S_{SA}^{[t:T,b:B]} \ NP_{SP}^{[t:[wh:-]]} \ VP_{A}) \\
S_{ST}^{[t:T,b:B]} & \rightarrow \text{eat}(S_{SA}^{[t:T,b:B]} \ NP_{SP}^{[t:[wh:-]]} \ VP_{A} \ NP_{S}) \\
S_{ST}^{[t:T,b:B]} & \rightarrow \text{eat}(S_{SA}^{[t:T,b:B]} \ NP_{SP}^{[t:[wh:+]]} \ SA \ NP_{S} \ VP_{A})
\end{align*}
\]

Rule selection and filtering for has:

\[
\begin{align*}
\sqrt{S_{ST}^{[t:T]}} & \rightarrow \text{have}(S_{SA}^{[t:T]}) \\
\sqrt{VP_{SA}^{[t:T]}} & \rightarrow \text{have}(VP_{SA}^{[t:T]}) \\
\times S_{ST}^{[t:T,b:B]} & \rightarrow \text{have}(S_{SA}^{[t:T,b:B]} \ NP_{SP}^{[t:[wh:-]]} \ VP_{A} \ NP_{S})
\end{align*}
\]
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

$\times S_{[t:T,b:B]} \rightarrow eat(S_{A[t:T,b:B]} N_{S[t:[wh:]]} P_{S[t:[wh:]]} V_{P_A})$

$\checkmark S_{[t:T,b:B]} \rightarrow eat(S_{A[t:T,b:B]} N_{S[t:[wh:]]} P_{S[t:[wh:]]} V_{P_A} N_{P_S})$

$\checkmark S_{[t:T,b:B]} \rightarrow eat(S_{A[t:T,b:B]} N_{S[t:[wh:]]} P_{S[t:[wh:]]} S_{A N_{P_S} V_{P_A}})$

NP$^{t:[wh:],[wh:]}$

fruit-t3

have-t5

John-t4

have-t6

NP$^{t:[wh:],[wh:]}$

which-t1

NP$^{t:[wh:],[wh:]}$

NP$^{t:[wh:],[wh:]}$

NP$^{t:[wh:],[wh:]}$

NP$^{t:[wh:],[wh:]}$
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

<table>
<thead>
<tr>
<th></th>
<th>Sentences (Length L)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S(0 – 5)</td>
<td>S(6 – 10)</td>
</tr>
<tr>
<td>Total</td>
<td>1084</td>
<td>2232</td>
</tr>
<tr>
<td>BL</td>
<td>0.85</td>
<td>10.90</td>
</tr>
<tr>
<td>SEQ</td>
<td>1.49</td>
<td>2.84</td>
</tr>
<tr>
<td>PAR</td>
<td>1.53</td>
<td>2.56</td>
</tr>
</tbody>
</table>

- 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

<table>
<thead>
<tr>
<th></th>
<th>Sentences (Arity)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S(1)</td>
<td>S(3)</td>
<td>S(5)</td>
<td>S(6)</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>Total</td>
<td>Total</td>
<td>Total</td>
</tr>
<tr>
<td></td>
<td>190</td>
<td>3619</td>
<td>2910</td>
<td>1093</td>
</tr>
<tr>
<td>SEQ</td>
<td>0.89</td>
<td>3.65</td>
<td>5.24</td>
<td>8.20</td>
</tr>
<tr>
<td>PAR</td>
<td>0.97</td>
<td>2.63</td>
<td>2.86</td>
<td>3.09</td>
</tr>
</tbody>
</table>

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)

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

```
sroot |
   play/VB
   sbj   obj
   john/NNP   football/NN
```

<table>
<thead>
<tr>
<th>WORD</th>
<th>(play, (john), (football))</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS</td>
<td>(vb, (nnp), (nn))</td>
</tr>
<tr>
<td>dep</td>
<td>(sroot, (sbj), (obj))</td>
</tr>
<tr>
<td>WORD/POS</td>
<td>(play/vb,</td>
</tr>
<tr>
<td></td>
<td>(john/nnp), (football/nn))</td>
</tr>
<tr>
<td>dep-POS</td>
<td>(sroot-vb,</td>
</tr>
<tr>
<td></td>
<td>(sbj-nnp), (obj-nn))</td>
</tr>
</tbody>
</table>
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
Adapting 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.

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

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

$$\text{Pass}(\neg f) = \frac{\text{count}(\neg f|\text{PASS})}{\text{count}(\neg f)}$$
Constructing the output Suspicion Tree

We adapt the ID3 decision tree algorithm to yield a tree whose nodes are suspicious 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

<table>
<thead>
<tr>
<th>Test Data</th>
<th># Failures Before EM</th>
<th># Failures after EM</th>
<th>Error Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>26725</td>
<td>19280 (72.1%)</td>
<td>5157 (19.3%)</td>
<td>-52.8</td>
</tr>
</tbody>
</table>

- 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 correct 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)
  yes   no
  (0, 155)

(ALL/PDT)
  yes   no
  (0, 140)

(THE/DT)
  yes   no
  (0, 23) (2, 9)

(DT, (IN))
  yes   no
  (JJ, (DT))

(IN)
  yes   no
  (POSS)
  (0, 15)

(RB, (IN))
  yes   no
  (0, 38) (cont.)

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
(S) 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
(S) 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

1. Sentence generation
   GraDe

2. Sentence selection
   GramEx query language

3. Exercise generation
   GramEx activity types

Generated Sentences

Grammar (SemTAG) and Lexicon
input constraints

Source sentences
Exercise items
Creating a grammar exercise

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

Pedagogical goal: Plural form of irregular nouns
Exercise type: Fill-in-the-blank.

1. Select sentences
   ⇒ NP[\text{num} = \text{pl} & \text{plural} = \text{irreg}]
   (+canonical order)

2. Process the selected sentence
   NP[\text{num} = \text{pl}] ⇒ blank
   NP[\text{lemma} = \text{bijou}] ⇒ hint

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

{CanonicalObject, CanonicalSubject, ActiveVerb}
Selecting appropriate sentences

GramEx's boolean constraint language: Signature

morpho-syntactic (feature structures)
syntactic (tree properties –XMG metagrammar)

S

NP

Tammy

V\[\text{tense=pst mode=ind}\] \{\text{CanSubj, CanObj, Active}\}

VP

a

D\[\text{num=sg}\]

NP\[\text{num=sg}\]

une

NP\[\text{num=sg gen=f}\]

voix

Adj\[\text{flexion=irreg}\] \{\text{Epith}\}

doce
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
  ⇒ 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 ]

\(\checkmark\) Tammy a une voix douce (Tammy has a soft voice)

\(\times\) Tammy a une jolie voix (Tammy has a nice voice)

---

**Pedagogical goal:** Prepositions with infinitives

POBJinf \(\wedge\) CLAUSE

POBJinf \(\equiv\) (DE-OBJinf \(\lor\) A-OBJinf)

CLAUSE \(\equiv\) Vfin \(\land\) \(\neg\)Mod \(\land\) \(\neg\)CCoord \(\land\) \(\neg\)Sub

\(\checkmark\) Tammy refuse de chanter (Tammy refuses to sing)

\(\times\) Jean dit que Tammy refuse de chanter (John says that Tammy refuses to sing)
Transformation-based grammar exercises
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.)

✗ (K) *Tex a fait la tarte.*
   (Tex has baked the pie.)

✗ (K) *La tarte a été faite par Tex.*
   (The pie has been baked by Tex.)

✗ (K) *C’est par Tex que la tarte sera faite.*
   (It is Tex who will bake the pie.)

✗ (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?

-faire: \{Active, CleftSubj, CanObj\}
  (num: sg, tse: pst, mode: ind, pers: 3)

tex: \{ProperNoun\}
  (fun: subj, gen: fem, num: sg, pers: 3)

tarte: \{Noun\}
  (fun: obj, gen: fem, num: sg)

\la: \{DefDet\}
  (gen: fem, num: sg)

- 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
  \[ \text{Active, CleftSubj, CanObj} \leftrightarrow \text{Passive, CleftAgent, CanSubj} \]

- e.g. NP/Pronoun
  \[ \text{CanSubj} \leftrightarrow \text{CliticSubj} \]

- e.g. Assertion/YN-Question
  \[ \text{questionMark} \leftrightarrow \emptyset \]
Meaning Preserving Transformations
Same core meaning (e.g. active/passive)

(Q) C’est Tex qui a fait la tarte.  ↔  (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)
 ↔  (K) La tarte a été faite par Tex.
(The pie has been baked by Tex)

\[ \alpha\text{-faire}: \{ \text{Active,CleftSubj,CanObj} \} \]
\[ \alpha\text{-tex}: \{ \cdots \} \]
\[ \beta\text{-avoir}: \{ \cdots \} \]
\[ \alpha\text{-tarte}: \{ \cdots \} \]
\[ \beta\text{-la}: \{ \cdots \} \]
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

Qui dort?
Who sleeps?

Quel petit tatou dort?
Which small armadillo sleeps?

Quel tatou dort?
Which armadillo 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!