Modeling and parsing natural languages

Benoît Sagot

Atoll, INRIA Rocquencourt – Lattice/Talana, Paris 7 (France)

NCLT Seminar – Dublin – April 3rd, 2006
Objectives

Central question:

How can we model languages so as to conciliate linguistic relevance and algorithmic properties that allow to build efficient parsers?

▶ Which kinds of formalisms are suitable for this task?
▶ Which parsing architecture could lead both to linguistic relevance and efficiency?
▶ How can we build linguistic resources on which such parsers rely?
Objective

I will sketch a few answers to these questions:

- Firstly, I will present two techniques to partially automatize the development of large-coverage syntactic lexicon and grammars.
- Second, I will focus on 2 parsing frameworks for French:
  - SxLfg, an efficient parser for a classical formalism, namely LFG,
  - Meta-RCGs, a polynomial non-linear formalism (and an associated parser) that shows the numerous advantages of closure under intersection.

We will show the complementarity of both approaches, and say a few words on how all their advantages could be put together.
1. Automatic tools and resource development
   - Representing lexical information
   - Automatic acquisition of lexical information
   - Error mining in parsing results
   - The Lefff

2. LFG parsing: SxLfg
   - Preliminaries
   - Robust parsing
   - Evaluation
   - Recent work
   - Conclusions on SxLfg

3. Meta-RCGs
   - RCGs
   - Meta-RCGs
1. Automatic tools and resource development
An NLP lexicon has to represent several kinds of information: morphological, syntactic, and possibly semantic.

- We call **extensional lexicon** a resource that associates with each form a structure that represents all this information.
  - Such a lexicon is typically used by parsers.
- We call **intensional lexicon** a resource that factorizes the information, associating with each lemma a morphological class and a syntactic class.

Hence, we developed a formalism to describe **morphological classes** (conjugation and declension patterns, . . . )

- and a formalism to describe **syntactic classes** (inheritance graph of atomic syntactic properties).
- Both model derivational morphology.

[Sagot et al., 2006].
Developing resources

- The development of large-coverage lexical resources is very costly (approx. 500,000 words for French)
- Therefore, we developed various techniques, in order to
  - acquire lexical information
  - identify errors and lacks in resources
- To ensure high-quality results, a **manual validation** is always needed: the objective of automatic tools is to *minimize human work*, not to avoid it.
- We will describe here two techniques that rely on similar bootstrapping schemes:
  - a technique to automatize the acquisition (or extension) of a morphological lexicon
  - a technique to identify erroneous or missing entries in a morphological and syntactic lexicon
Automatic acquisition of lexical information

We developed a technique to acquire automatically lexical information

- from a raw corpus and a morphological description of the language
- applied to
  - French verbs
  - all open categories of Slovak

[Clément, Sagot and Lang, 2004], [Sagot, 2005c]
The underlying idea is the following:

- Firstly, we use our morphological description as an **ambiguous lemmatizer**: we build all hypothetical lemmas that have at least one inflected form attested in the corpus.
- Then, we **rank these lemmas** according to their likelihood given the corpus (fix-point algorithm).
- Many kinds of information are taken into account (derivational morphology, prefixes, frequency of tags depending on the category, ...).
- **Manual validation** is performed on the best-ranked hypothetical lemmas.
- The whole process is launched anew, and benefits from the manual validation step.
## Manual validation environment

<table>
<thead>
<tr>
<th>Slovak</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>byť</td>
<td>v-byť</td>
</tr>
<tr>
<td>európsky, a, e</td>
<td>adj-l</td>
</tr>
<tr>
<td>ten článok, bez článku, o článku (lok), tie články, s, so článkami</td>
<td>nc-mid:Gu:0:Leu:ami</td>
</tr>
<tr>
<td>ten štát, bez štátu, o štáte (lok), tie štáty, s, so štáti</td>
<td>nc-mid:Gu:0:Leu:mi</td>
</tr>
<tr>
<td>ta únia, tie únie, sto únií, o úniách (lok)</td>
<td>nc-fiv</td>
</tr>
<tr>
<td>členský, á, é</td>
<td>adj-b</td>
</tr>
<tr>
<td>mať, mám, majú</td>
<td>v-ať</td>
</tr>
<tr>
<td>ta komisia, tie komisie, sto komisií, o komisiách (lok)</td>
<td>nc-fiv</td>
</tr>
<tr>
<td>to právo</td>
<td>nc-no</td>
</tr>
<tr>
<td>môcť, môžem, môžu, mohol</td>
<td>v-môcť</td>
</tr>
<tr>
<td>ta rada, tie rady</td>
<td>nc-fdv</td>
</tr>
<tr>
<td>ta oblasť, tie oblasti</td>
<td>nc-fm2</td>
</tr>
<tr>
<td>to opatrenie</td>
<td>nc-nie</td>
</tr>
<tr>
<td>ta návrha, tie návrhy</td>
<td>nc-fdv</td>
</tr>
<tr>
<td>ta krajina, tie krajiny</td>
<td>nc-fdv</td>
</tr>
<tr>
<td>spoločný, á, é</td>
<td>adj-b</td>
</tr>
<tr>
<td>to rámcne</td>
<td>nc-ne</td>
</tr>
<tr>
<td>ta politika, tie politiky</td>
<td>nc-fdv</td>
</tr>
<tr>
<td>iný, á, é</td>
<td>adj-b</td>
</tr>
<tr>
<td>ten cieľ, tí cieľi, s, so cieľmi</td>
<td>nc-mac:i:0:mi</td>
</tr>
</tbody>
</table>
Error mining in parsing results

Lexical resources have incorrect/incomplete information. Some of these problems can be automatically identified when the lexicon is used in parsers:

- we identify **unparsable sentences**
- we use a similar fix-point algorithm to select one word (or sequence of words) that is the **main suspect**: it most likely caused the failure
  - either because its lexical entry is incorrect/incomplete
  - or because it lexicalizes a phenomenon that is badly dealt with in the grammar or in the pre-syntactic processing chain
- we benefit from the fact that our team works both on large-coverage LFG and TAG parsers for French, but use the same morphological and syntactic lexicon, the Lefff
  → we can eliminate most problems due to the grammars

[Sagot and de La Clergerie 2006a], [Sagot and de La Clergerie 2006b]
1. Automatic tools and resource development
2. LFG parsing: SxLfg
3. Meta-RCGs
Conclusion

Representing lexical information
Automatic acquisition of lexical information
Error mining in parsing results
The Lefff

Environment for error mining results

Browsing errors for results [iter=200]

Enter rank (or start:end:key) | 34 | Mail this page

edit comment

manque la construction attributive (demeurer<subj,accomp>)

Statistical info on demeure/demeure

<table>
<thead>
<tr>
<th>rank</th>
<th>#occ.</th>
<th>#failed</th>
<th>%failed weight</th>
<th>%failed sentences</th>
<th>orate</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>870</td>
<td>705</td>
<td>24.64%</td>
<td>81.15%</td>
<td>7.27</td>
</tr>
</tbody>
</table>

history:

| #iteration | weight | 24.64% | 24.64% | 24.64% | 24.65% | 24.65% | 24.65% | 24.65% | 24.68% | 24.69% | 24.71% | 24.73% | 24.75% | 24.78% | 24.81% | 24.83% |
|-------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 200         |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 199         |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 195         |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 185         |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 175         |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 165         |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 155         |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 145         |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 135         |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 125         |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 115         |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 105         |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 95          |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 90          |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |

Lefff info for demeure

Failed sentences with demeure/demeure as most probable cause for failure

- [mondedipl01#19948] L’armée demeure une force majeure
- [mondedipl02#22126] LE FN demeure l’unique parti à défendre les négationnistes dans son programme.
- [mondedipl04#7744] Le pétrole demeure l’enjeu principal.
- [mondedipl01#19379] L’EUROPE demeure un projet à deux vitesses.
- [mondedipl01#19984] Certes, l’indonésie demeure la grande puissance régionale.
- [mondedipl01#28830] L’histoire demeure cependant la principale discipline d’enseignement.
- [mondedipl05/15643] En mer Rouge et dans la corne de l’Afrique, la situation demeure très incertaine.
- [mondedipl02#17949] Le père demeure le chef exclusif de la famille.
Thanks to these techniques, and others, we developed a **large-coverage morphological and syntactic lexicon for French**, the **Lefff** (Lexique des formes fléchies du français)

- 500,000 lexical entries...
- ...representing 400,000 wordforms
- used in all our tools (LFG and TAG parsers,...), as well as by other teams
- **freely available** (downloadable at www.lefff.net)
- ongoing collaborations to use a consensual representation model for syntactic information (with Claire Gardent, Laurence Danlos, Lionel Clément and others) and to compare this resources to the very few other resources for French (with Claire Gardent)

[Sagot et al., 2005], [Sagot et al., 2006], [Danlos and Sagot, 2006], [Danlos, Sagot and Salmon-Alt, 2006, submitted]
2. LFG parsing: SxLfg
Architecture

- The structure of the LFG formalism is the following:
  - a **CFG backbone** that builds **c-structures**
  - **unification-based decorations** that define **f-structures**
- CFG parsing techniques are well-known
- Hence, the main difficulty is to compute efficiently unification-based decorations (f-structures)
- We proceed in the following way:
  - Firstly, the CFG parse forest is constructed
    *A CFG parse forest is a compact factorized (polynomial-size) representation of all parse trees*
  - Secondly, computations can be performed over the forest, e.g., f-structures computation
  - Robustness mechanisms at each level
  - Rule-based disambiguation

[Boullier and Sagot, 2005a], [Boullier and Sagot, 2005b], [Boullier and Sagot, 2006]
We choose this architecture because

- CFG backbones of large-coverage LFG grammars are often very ambiguous
- such an architecture will allow to replace the CFG backbone by more powerful syntactic backbones (TAG, RCG, . . . )
The first pass computes the CFG parse forest thanks to a CFG parser

- from Pierre Boullier’s system, called Syntax (evolution of the parser presented in (Boullier, 2003)).
- It’s an Earley-like parser with a regular guide,
- which relies on a left-corner automaton,
- It deals with the output of our pre-syntactic processing chain SxPipe (which we use without morphosyntactic tagging, although the parser can deal with this information) [Sagot and Boullier, 2005a], [Sagot and Boullier, 2005b]
- we added powerful and customizable error recovery and error correction mechanisms
f-structures computation

To deal with the exponential complexity of f-structures computation,

- the evaluation is performed bottom-up on the parse forest of c-structures: this is a **synthetized evaluation**, all f-structures associated with a given node are computed once and never modified later
- we make an extensive use of **computation and structure sharing** as well as **lazy evaluation**
- we use **internal disambiguation** techniques...
We define a set of **heuristics** to eliminate non-optimal f-structure rules associated to a node of the forest

- they can be applied at the root of the forest, once all parses are built: we call this **global disambiguation**
- but they can be applied on any node of the forest, as soon as all its f-structures are computed: we call this heuristic pruning **internal disambiguation**

Some node names (CFG non-terminal symbol) are associated with a (different) list of some of these heuristics that are applied so as to eliminate f-structures.
Robustness

At all levels, we introduce robustness techniques to guarantee an output whatever the input:

▶ error recovery and correction in the CFG parser
▶ various error tolerance mechanisms during f-structures computation
▶ possibility to output partial f-structures (which cover only parts of the input)
▶ over-segmentation of non-parsable inputs
Robustness: an example

Jean essaye de...

John tries to...

\[
\begin{align*}
\text{pred} &= \text{'essayer} <\text{subj, de-vcomp}'>, \ v[2..3] \\
\text{subj} &= \begin{cases}
\text{pred} = \text{'Jean} <\text{(subj)}'>, \ \text{pn}[1..2] \\
\text{det} = + \\
\text{hum} = + \\
A_{ij} = \{ R_{9}^{182}, R_{26}^{177}, R_{28}^{170} \} & F_{68}
\end{cases}
\\
\text{de-vcomp} &= \begin{cases}
\text{pred} = \text{'de} <\text{obj}...>' , \ \text{prep}[3..4] \\
vcomp &= \begin{cases}
\text{subj} = \emptyset_{F_{68}} \\
A_{ij} = \{ R_{84}^{162} \}_{2} & F_{69}
\end{cases} \\
pcase = \text{de} \\
A_{ij} = \{ \} & F_{70}
\end{cases}
\end{align*}
\]
Using SxLfg to build a parser for French

- We use SxLfg to build a **parser for French** with:
  - our large-coverage LFG grammar for French we developed (starting from Lionel Clément’s grammar)
  - our large-coverage syntactic lexicon, Lefff (500,000 entries, 400,000 wordforms)

- To evaluate SxLfg, we parsed **10,000,000 words of journalistic corpus** (*Le Monde*), pre-processed by SxPipe.

- The aim of the experience is to evaluate SxLfg (the parser generator), and not (only) the parser itself. Hence, we are more interested in parsing times w.r.t. the size of the resulting structures (as opposed to, e.g., precision measures).

[Sagot and Boullier, 2006]
1. Automatic tools and resource development
2. LFG parsing: SxLfg
3. Meta-RCGs

Conclusion

Preliminaries
Robust parsing
Evaluation
Recent work
Conclusions on SxLfg
Conclusions on SxLfg

CFG backbone ambiguity
1. Automatic tools and resource development
2. LFG parsing: SxLfg
3. Meta-RCGs
Conclusion

Preliminaries
Robust parsing
Evaluation
Recent work
Conclusions on SxLfg
Conclusions on SxLfg

CFG parser efficiency

![CFG parser efficiency graph](image)
f-structures computation efficiency
In ongoing work with Pierre Boullier, we go beyond this purely LFG architecture, in order to improve parsing time and precision, without the use of a treebank.

The idea is to use many modules that perform computations over the CFG parse forest, and can prune it to ease the work of following modules, as, e.g., f-structure computation.

If the forest has been pruned too much, we allow to go back to a previous state of the forest and try less a violent pruning before computing f-structures.
We developed a rule-based **chunker** module:

- some non-terminal symbols are known to span **chunks**,  
- **pruning rules** of the following form are used: if there is a parse with chunks that have some characteristics, then we eliminate parses from the forest with less satisfying chunks  
- If requested, a **unique parse** tree can be chosen (chunk types are ordered, depending on their length, and we try to maximize chunk lengths).  
- Precision results are **state-of-the-art** (even without any fine-grained contextual information)
We developed a syntax to **cascade modules**, e.g.:

- precision gets a bit higher, and average parsing time is divided by 2 (still without probabilistic models)
Bootstrapping probabilistic models

- We developed an $n$-best module as well, with different possible levels of contextuality.
- Since there is no PTB for French, we apply our bootstrapping scheme again:
  - We first parse a large corpus without any probabilistic model (as shown until now).
  - We use the result as an annotated corpus, and learn models from it.
  - We use these models to parse the corpus again, hopefully faster and with a better precision.
- Results are only preliminary at this point.
- We probabilized only the forest (c-structures), we currently work on probabilizing f-structures as well.
In order to build reasonably efficient and robust parsers, it is possible to use

- large-coverage deep non-probabilistic grammars
- with a linear backbone (CFG, TAG, . . .)
- and unification-based decorations
  → we call them two-stage formalisms

Error tolerance, recovery and robustness techniques do work
Conclusions on SxLfg

But the classical drawbacks of such two-stage formalisms remain:

- The **boundary between structure** (→ backbone) **and attributes** (→ decorations) is somewhat arbitrary
  - only one kind of linguistic constraint can be encoded in the backbone (here, constituency),
  - other constraints must be dealt with in the decorations (dependency, . . .)

- **Tree structures are often not sufficient** (discontinuous constituents, coordination, . . .)

- Unification-based decorations still have an **exponential complexity**, hence the need for “internal disambiguation”

- The **coexistence of two different mechanisms** leads to efficiency and relevance problems for parsers, in particular for robustness techniques (no CFG recovery because of an error during f-structures computation . . .)
Conclusions on SxLfg: looking towards non-linear formalisms

On the contrary, some powerful backbone formalisms exist, in particular formalisms that are **closed under intersection**, that are called **non-linear formalisms**:

- every word is a resource that can be consumed many times
- each “consumption” can deal with a specific kind of linguistic constraint (morphology, constituency, syntactic dependency, semantic dependency, . . .)
- resulting structures can be **graphs**
Conclusions on SxLfg: looking towards non-linear formalisms

- Non-linear formalisms have **complementary advantages** to those of two-stage formalisms such as LFG
  - they allow to give the same status to **different kinds of interacting linguistic properties**, including, e.g., lexical semantics, without the use of decorations
  - they are not limited to tree structures,
  - some of these formalisms have a **polynomial complexity**, such as RCGs and sLMGs (and this remains true if one adds decorations that are defined over finite or linear-sized spaces)

- But they are less used, and no such formalism for linguistic description is really used. Moreover, efficient parsing is less trivial.
3. Meta-RCGs
Range Concatenation Grammars: general presentation

- **Range Concatenation Grammars, RCGs** are a polynomial non-linear rewriting formalism introduced by Pierre Boullier that can be used
  - as such,
  - as implementation formalism for any polynomially parsable formalism (CFG, TAG, MC-TAG, LCFRS, ...)
- The advantage of non-linearity, as said before, is both:
  - to make all kinds of constraints interact in the grammar without the need for decorations (morphology, constituency, dependency, lexical semantics, predicate-argument semantics, and even discourse structure...)
  - Classical views (constituency, dependency) can be extracted by **projection** of the full parse
  - to build graph structures (not only trees)

[Sagot and Boullier, 2004], [Sagot, 2005a]
Let us consider the following context-free rule:

\[ S \rightarrow NP \text{ VP} \]

Symbols NP and VP are linked by two different operators:

- **concatenation**: the portion of string covered by S is made out of two parts, one covered by NP and one covered by VP.
- **conjunction**: S is true on the portion of string it covers if this is true as well for NP and VP.

In a dual way, one can say that NP is a notation that denotes both:

- A one-argument **predicate** that denotes a property of its argument ("being an NP"),
- A **range** of the input string which is the unique argument of this predicate.
Nothing prevents us from separating these two concepts:

- we still denote predicates by S, NP, VP
- we denote by Subj and Pred the ranges that are arguments of NP and VP respectively
- we denote by a space the concatenation of two ranges: $Subj \ Pred$ is the concatenation of $Subj$ and $Pred$, and thus it is the argument of S:

$$S(Subj \ Pred) \rightarrow NP(Subj) \ VP(Pred)$$

One can close this rule by the following: *to check if a range is an S, it is sufficient to split it into 2 ranges and to check if the first one is an NP and the second one a VP.*
Empty right hand sides

To express a context-free lexical rule such as

\[ V \rightarrow \text{veut} \]

one just needs to say that “\text{veut}” is a V:

\[ V(\text{“veut”}) \rightarrow \varepsilon \]

One can close this rule by the following: \textit{to check if “veut” is a V, there is nothing to do}
Predicates with more than one-arguments and non-linear rules

Now that predicates and ranges are separated, one can introduce predicates with more than one argument:

▶ without any change:

\[
S(\text{Subj Pred}) \quad \rightarrow \quad S2(\text{Subj}, \text{Pred})
\]
\[
S2(\text{Subj}, \text{Pred}) \quad \rightarrow \quad \text{NP}(\text{Subj}) \text{ VP}(\text{Pred})
\]

▶ more interestingly, using non-linearity (trying to model *Pierre veut dormir*):

\[
P(\text{Subj Verb Cplts}) \quad \rightarrow \quad V(\text{Verb}) \text{ Subj}(\text{Subj}, \text{Verb})
\]
\[
\text{ VP}(\text{Verb Cplts}, \text{Subj}, \text{Verb})
\]
\[
\text{Subj}(\text{Np}, \text{Verb}) \quad \rightarrow \quad \text{NP}(\text{Np})
\]
\[
\text{VP}(\text{Verb Inf}, \text{Subj}, \text{Verb}) \quad \rightarrow \quad \text{CtrlV}(\text{Verb}) \text{ VComp}(\text{Inf}, \text{Verb}, \text{Subj})
\]
\[
\text{VComp}(\text{Inf}, \text{Verb}, \text{Subj}) \quad \rightarrow \quad \text{Vinf}(\text{Inf}) \text{ Subj}(\text{Subj, Inf})
\]
Parsing *Pierre veut dormir*

With a couple of lexical rules, this allows to parse *Pierre veut dormir* (*Pierre wants to sleep*):

- **consituency view:** we consider the first arguments of predicates $S$, $NP$, $ProperNoun$, $VP$, $CtrlV$, $Vinf$, and we link them according to → :

- **syntactic dependency view:** we link the first argument to the second argument for predicates $Subj$ et $VComp$.
Lexical semantic constraints

- Non-linearity allows to take into account **additional constraints**, such as **lexical semantics**: one can replace
  \[
  \text{Subj}(Np, \text{Verb}) \rightarrow \text{NP}(Np)
  \]
  by
  \[
  \begin{align*}
  \text{Subj}(Np, \text{Verb}) & \rightarrow \text{NP}(Np) \text{ Agent}(Np, \text{Verb}) \\
  \text{Agent}(Np, "veut") & \rightarrow \text{Animate}(Np) \\
  \text{Animate}("Pierre") & \rightarrow \varepsilon
  \end{align*}
  \]

- Thus, we added both:
  - a **predicate-argument semantics relation**: Agent
  - a **lexical semantics constraint** (**restriction of selection**): only an animate NPs can be the agent of **vouloir**
Some properties

- Polynomial parsing time (parser available)
- in fact, RCGs cover exactly all what is doable in polynomial time ($PTIME$)
- The set of languages that can be defined by an RCG is a set that is closed under the following operations on languages:
  - union, concatenation, Kleene star
  - intersection (non-linearity)
  - complementation (negative predicates available)
  - ... but not substitution
Meta-RCGs

- Some linguistic aspects lack in RCGs:
  - homonyms
  - morphological features
  - head(s) of a syntagm
  - appropriate treatment of long distance dependencies
- ⇒ we developed a linguistic formalism on top of RCGs, called **Meta-RCG**, that extends the syntax of RCGs to take into account these notions, without extending the expressive power:
  - Any Meta-RCG can be converted into an equivalent RCG
  - We developed a medium-coverage Meta-RCG for French, which uses the (large-coverage) *Lefff* as a morphological lexicon, and a toy syntactic and semantic lexicon

[Sagot, 2005a]
Constituency view of a Meta-RCG parse

Un avocat mange un avocat (A lawyer eats an advocado)
Dependency view of a Meta-RCG parse

**Pierre veut une bière et dormir (Pierre wants a beer and [wants to] sleep)**

- Syntactic dependencies
Les entreprises dans lesquelles le Japon veut que la Commission accepte que l’Europe investisse fabriquent des ordinateurs (The companies in which Japan wants that the Commission accepts that Europe invests produce computers)
Conclusions
Conclusions

- Different applications of the same booststrapping idea have proven efficient, in particular for lexical acquisition and error identification in linguistic resources.
- SxLfg shows that it is possible to perform large-coverage non-probabilistic deep (LFG) parsing, with a reasonable efficiency and robustness techniques.
- Meta-RCGs show that it is possible to make all kind of constraints interact in the grammar, without the use of unification-based decorations, thanks to non-linearity.
- An other way to benefit from non-linearity could be to replace the CFG backbone of LFG by an RCG backbone.
- But quantitative information is very important as well:
  - It can be built from scratch by bootstrapping techniques
  - But it can also be learned from existing treebanks
Conclusions

All these tools and methods have a common goal:

perform high-precision, high-speed and high-coverage parsing with deep linguistically relevant models
Extending the method to improve TAG grammar induction

With Alexis Nasr, we intend to try and adapt this method
- to improve TAG grammar induction (dealing with hapax and missing trees)
- using the following equivalences (*quasi-tree = tree before lexicalization*)

<table>
<thead>
<tr>
<th>inflected form</th>
<th>TAG quasi-tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>lemma</td>
<td>set of quasi-trees with the same semantic frame</td>
</tr>
<tr>
<td>inflection class</td>
<td>set of possible syntactic transformations</td>
</tr>
<tr>
<td>&quot;-er&quot; 1st group verbs (Fr)</td>
<td>transitive constructions</td>
</tr>
<tr>
<td>present, 1st pers. singular</td>
<td>transitive construction with left extraction</td>
</tr>
</tbody>
</table>

- The idea is to learn **classes of quasi-trees**, so as to keep hapax and to acquire unseen quasi-trees if they belong to a well-attested class, and to eliminate hapax (or rare trees) that do not belong to any class.