Towards Wide-Coverage Semantics for French

Richard Moot
LaBRI (CNRS), SIGNES (INRIA) & U. Bordeaux

CAuLD, 13 december 2010, Nancy

Research partially funded by grants from the Conseil Regional d'Aquitaine: "Itipy" and "Grammaire du Français"

Introduction

Bridge between statistical NLP and syntax/semantics the way I (and many people here) like it!
Don't worry, this will not be a talk of the style I improved on task X from Y% to Y+. 2%
There will be some percentages, but just to show we are up to the level of some of the statistical NLP guys.

- Many wide-coverage parsers for French exist (witness the participation of the Easy and Passage campaigns)
- My goal is not directly to compete with them, but to move towards a wide-coverage parser which produces structures which are more interesting (at least to me!) than shared forests

Introduction

- I will talk about my current research on a wide-coverage categorial grammar for French.
- As we all know, a categorial parse corresponds to a lambda-term in the simply typed lambda calculus.

Introduction

- So sentences analysed with this grammar correspond to lambda terms.
- Since the work of Montague, we know that the simply typed lambda calculus forms a solid base for the semantic analysis of fragments of natural language.

Introduction

- However, we are by no means limited to Montague semantics: Muskens (1994) and de Groote (2006) show that the semantics of categorial grammars are compatible with modern theories of dynamic semantics (DRT in the case of Muskens, and a continuation-based approach in the case of de Groote)

Introduction

- In this talk I will present the Grail parser and the development of a wide-coverage grammar of French as well as the development of two prototype semantic lexicons:
    - one producing DRSs
    - one producing de Groote-style continuation semantics

Introduction

- Wide-coverage semantics in this sense is a relatively new field, which was pioneered for English by Bos e.a. (2004)

Overview

- Grammar Extraction
  - converting a corpus into categorial grammar
  - how to use this grammar for parsing
- Semantics

Grammar Extraction

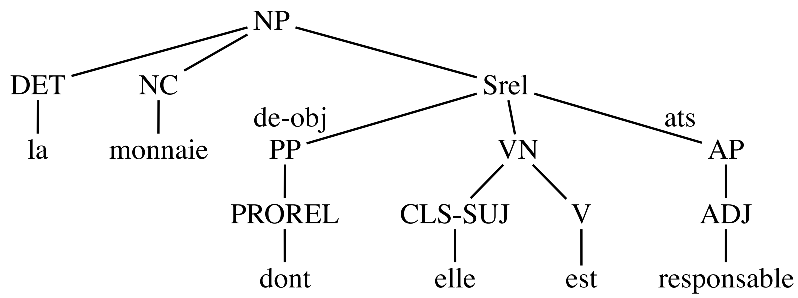
From the Paris VII corpus to a categorial lexicon, while developing several taggers

Grammar Extraction

- Grammar extraction is the conversion of a linguistically annotated corpus (in our case, the Paris VII treebank) into a grammar into a grammar formalism the people doing the conversion really like (in our case, categorial grammar)

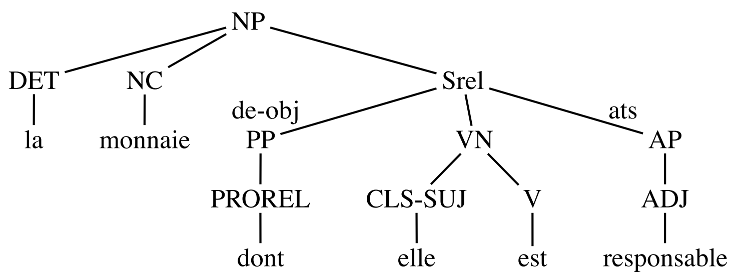
The Paris VII Corpus

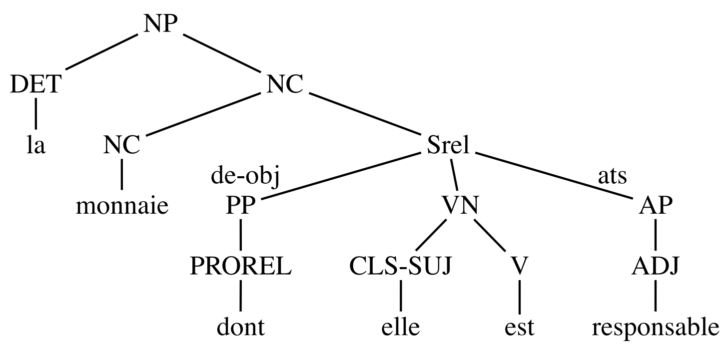
- To the right is a small sentence fragment of the Paris VII corpus, which suffices to illustrate the extraction procedure



The extraction algorithm

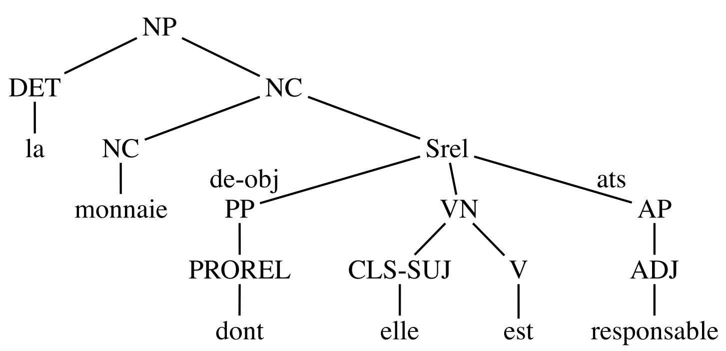
1. Binarize the annotation

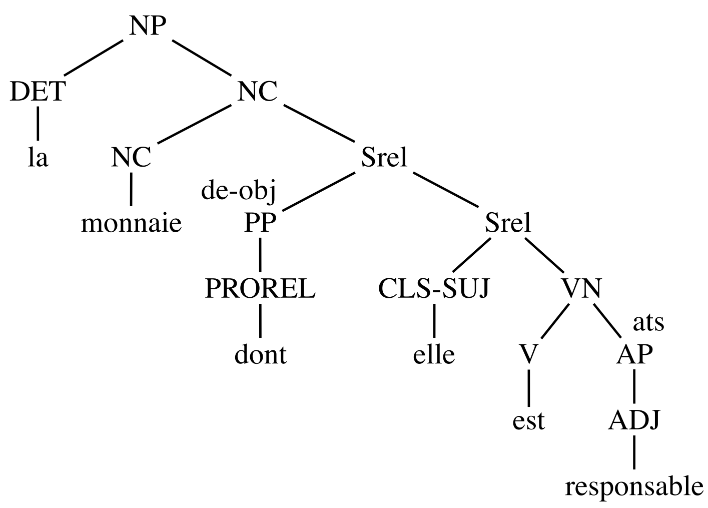




The extraction algorithm

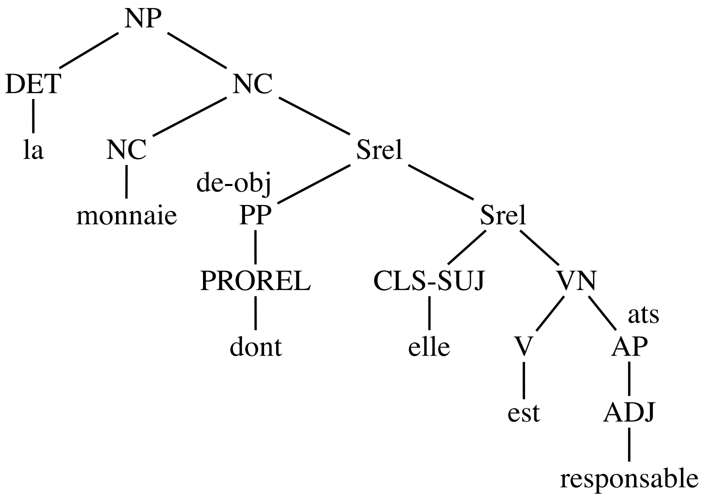
1. Binarize the annotation

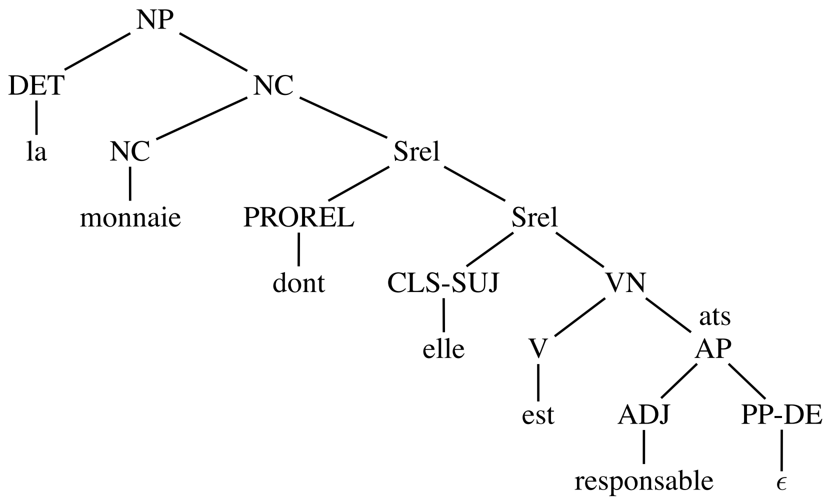




The extraction algorithm

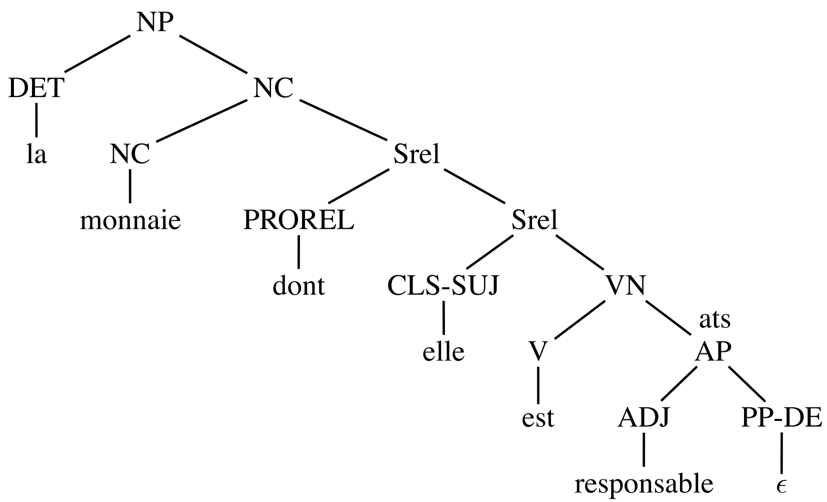
1. Binarize the annotation inserting traces for wh words





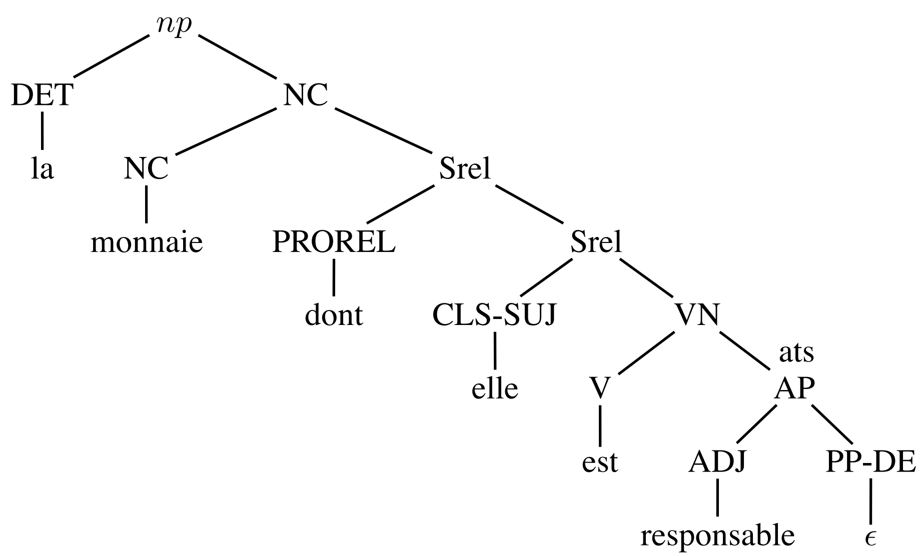
The extraction algorithm

2. Assign formulas



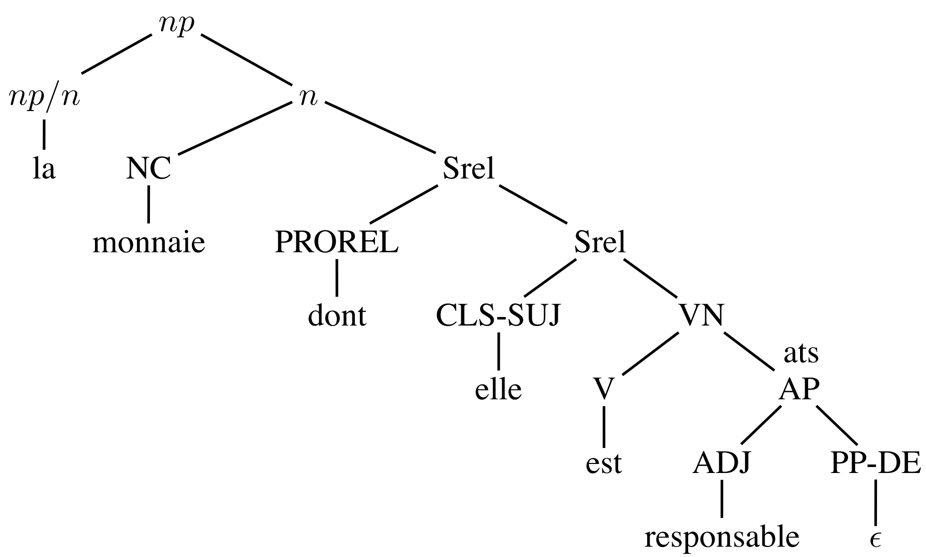
The extraction algorithm

2. Assign formulas



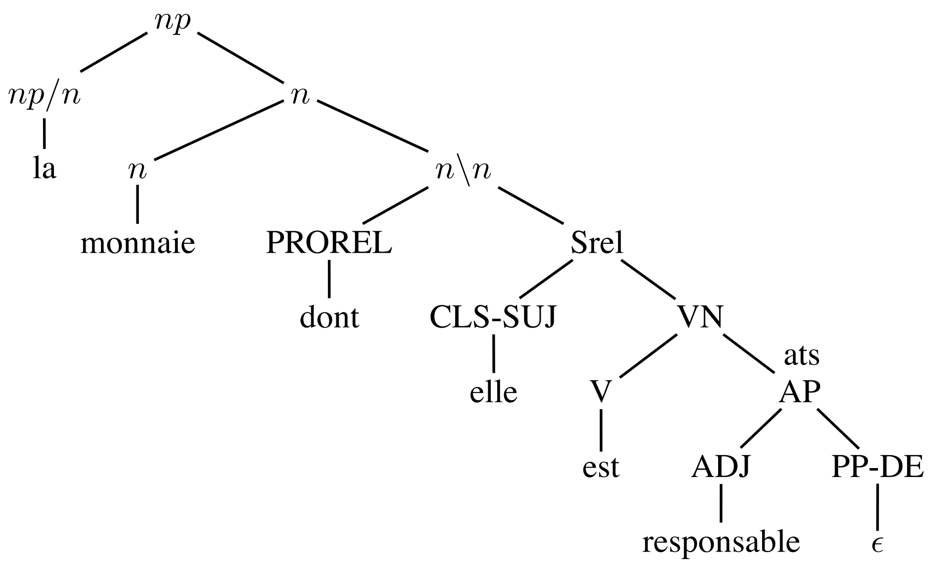
The extraction algorithm

2. Assign formulas



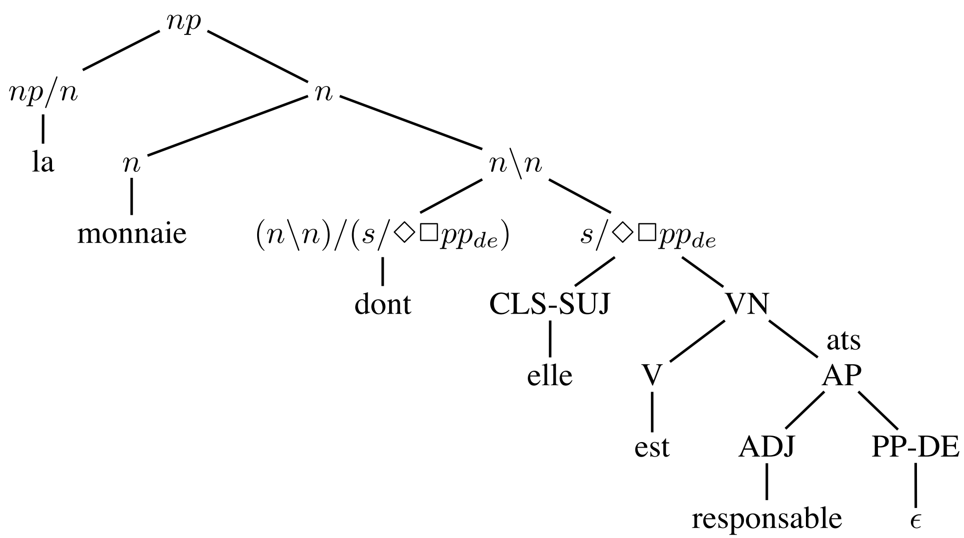
The extraction algorithm

2. Assign formulas



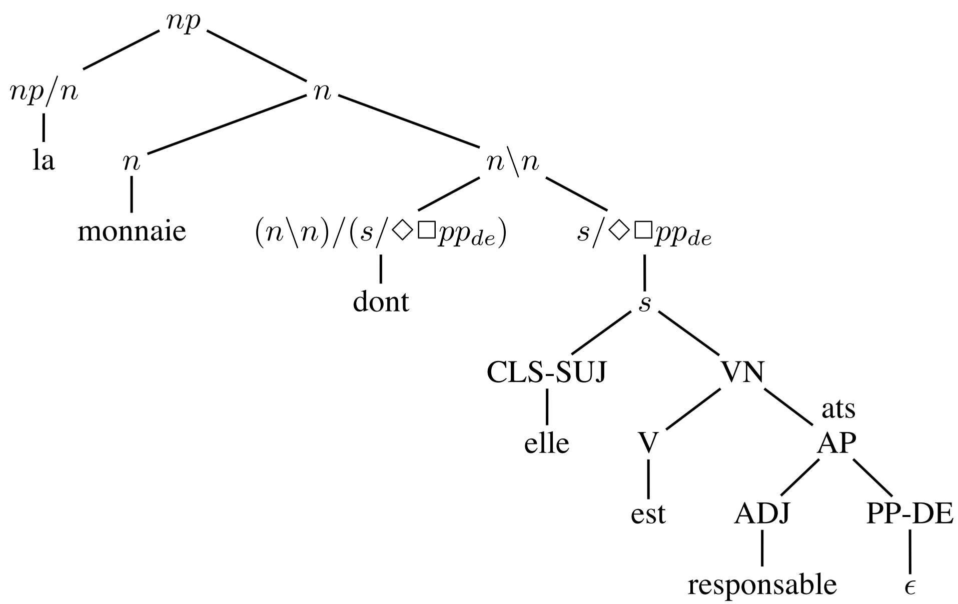
The extraction algorithm

2. Assign formulas



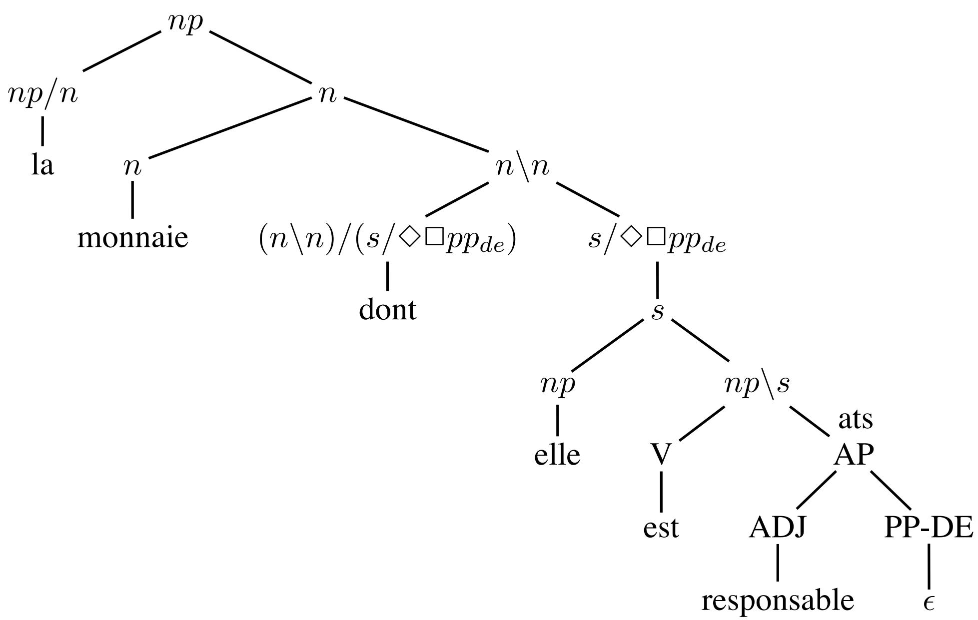
The extraction algorithm

2. Assign formulas



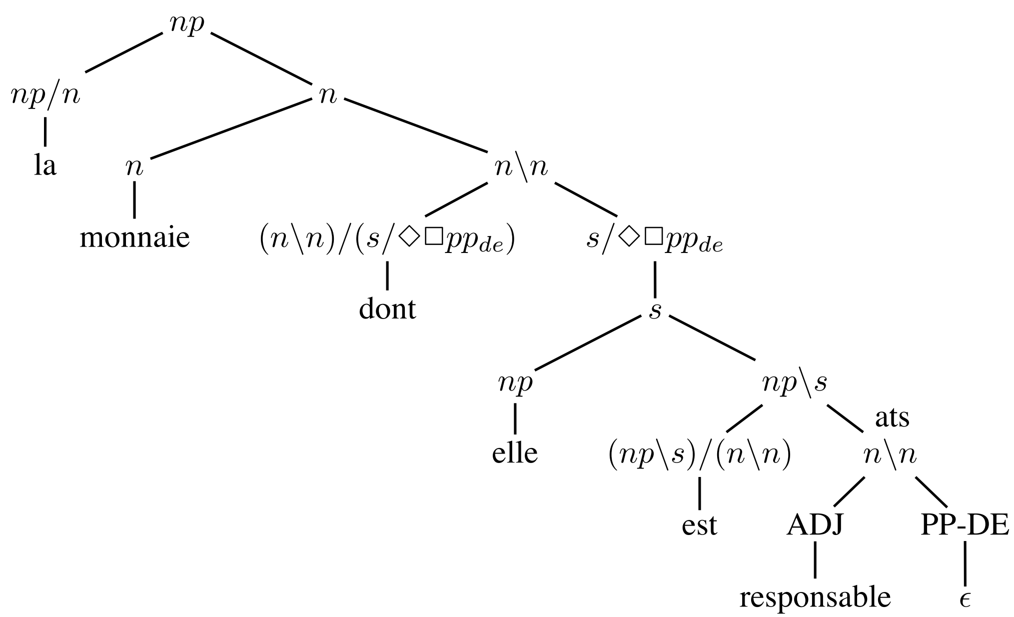
The extraction algorithm

2. Assign formulas



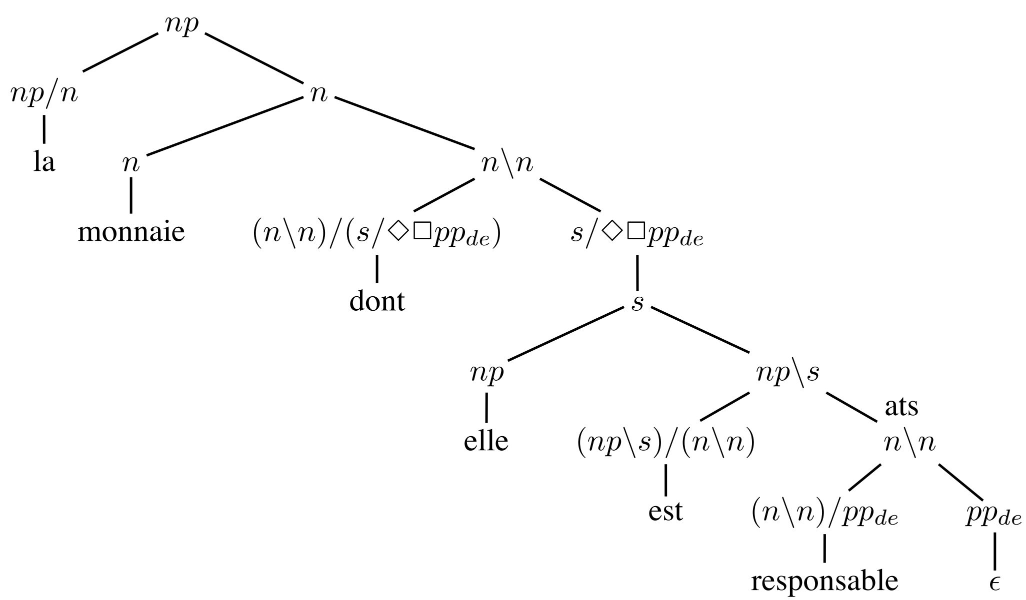
The extraction algorithm

2. Assign formulas



The extraction algorithm

2. Assign formulas



Grammar Extraction

- A lot of useful information (such as the position of “traces” of extracted elements) is not annotated but very useful for the grammar and needs to be added by hand.
- In addition, the extracted grammar has received a very significant amount of manual cleanup

The extracted grammar

- On the basis of the 382.145 words and 12.822 sentence of the treebank, the extraction algorithm extracts 883 different formulas, of which 664 occur more than once.
- Many frequent words are assigned <u>many</u> different formulas
- This is a significant bottleneck for parsing

The extracted grammar

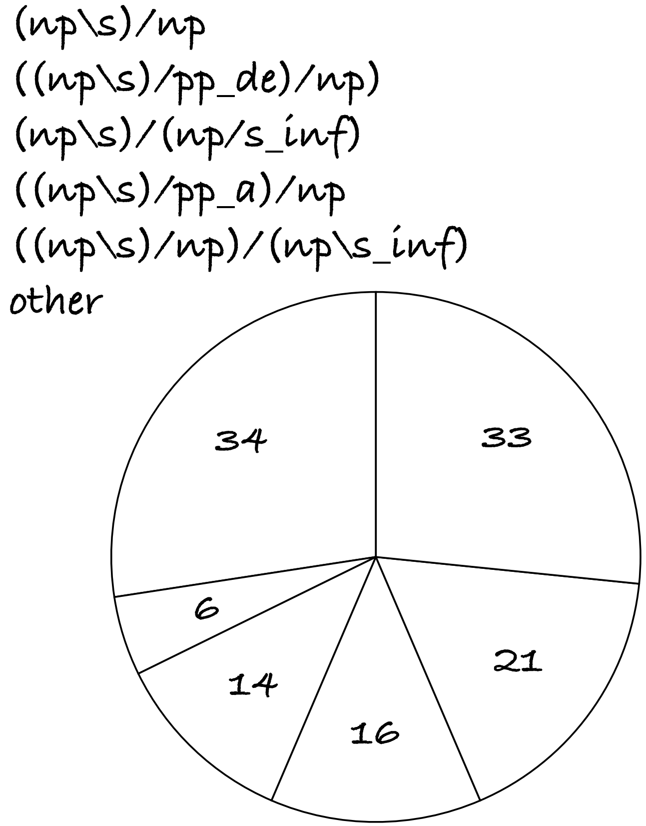
Word	POS	#
et	conj	71
,	ponct	62
à	prp	55
plus	adv	44
ou	conj	42
est	verb	39
être	inf	36
en	prp	34
a	verb	31

POS	#
adv	206
conj	92
prp	149
ponct	89
verb	175

An illustration of some of the most ambiguous words and part-of-speech tags.

The extracted grammar

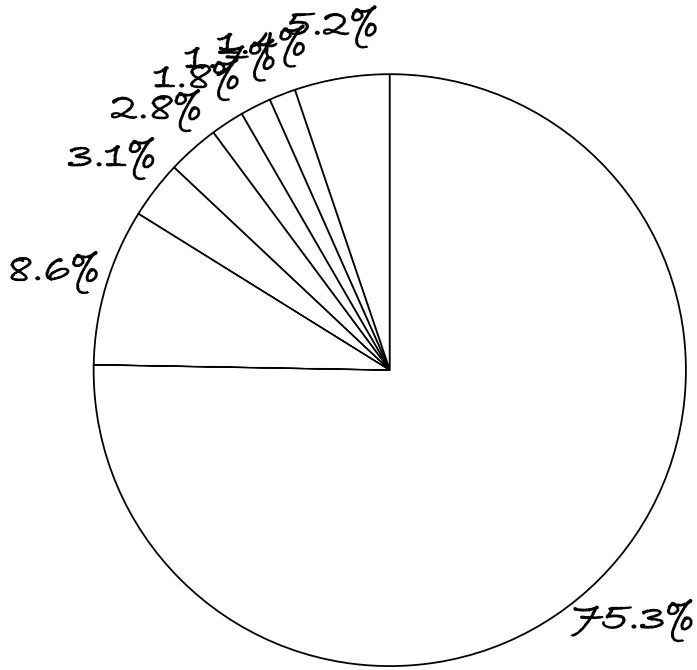
- Formula assignments to the present tense form “fait”
- 124 occurrences in the corpus, with 19 different formulas assigned to it.



The extracted grammar

- Formula assignments to the comma “,”
- 21,398 occurrences, 62 different formulas.

no formula
(np\np)/np
(n\n)/n
(np\np)/n
(s\s)/s
((np\s)\(np\s))/(np\s))
((n\n)\(n\n))\(n\n))
other



The extracted grammar

- The sum up, we have produced a categorial grammar for French, which is essentially a very big lexicon.
- The size of this lexicon, coupled with high lexical ambiguity, makes direct exploitation for parsing difficult.
- A fairly standard solution is to use a supertagger to estimate the most likely sequence of formulas for the given words.

Supertagging

- Supertagging is essentially part-of-speech tagging but with richer structure hence "super" tags.

- Like part-of-speech tagging, we use superficial contextual information and statistical estimation to decide the most likely tag.

Supertagging

- So what is the context for a supertagger?
- Typically, it consists of the current word, the surrounding words, the current and surrounding POS tags and the previous supertags.

Context for “de”

np/n	n	?		
DET	NC	P	NPP	NPP
la	voiture	de	Prince	Charles

Supertagging

- The basic procedure for finding the sequence of formulas then becomes
  - Find the correct POS tag sequence
  - Find the correct supertag sequence

Context for “de”

np/n	n	?		
DET	NC	P	NPP	NPP
la	voiture	de	Prince	Charles

Supertagging

- Estimation is done using maximum entropy models
- Very standard and easy to modify (ie. we can add any information we think is useful and let the estimation algorithm decide which ones really are).
- Good performance and efficient training (Clark & Curran 2004).

Context for "de"

np/n	n	?		
DET	NC	P	NPP	NPP
la	voiture	de	Prince	Charles

Any information which we can easily obtain, of course. If we think a word having an even number of letters is useful, we can add it.

POS/Supertagging

- Note, that, though Part-of-Speech tagging helps, an incorrect POS-tag can actually hurt the supertagger.

np/n	n	(np\s)/np	np/n	n
DET	NC	V	DET	N
la	petite	brise	la	glace

- Errors in DET-N versus CLO-V POS-tags are difficult for the supertagger to recover from.

np/n	n/n	n	(np\s)/((np\s)/np)	(np\s)/np
DET	ADJ	NC	CLO	V
la	petite	brise	la	glace

POS/Supertagging

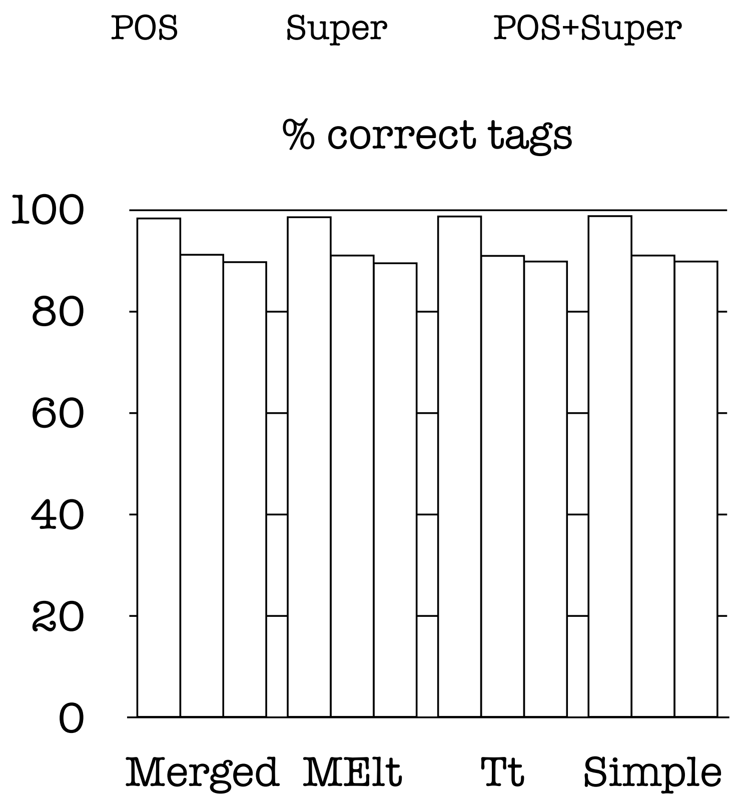
- Other difficult words for the POS-tagger include "que" (which can be a conjunction, an adverb or a relative pronoun)

- However, in general, the POS-tag information helps (as we will see)

np/n	n	(n\n)/s	np/np	np	np\s
DET	NC	CC	ADV	NPP	V
le	fait	que	que	Marie	dort

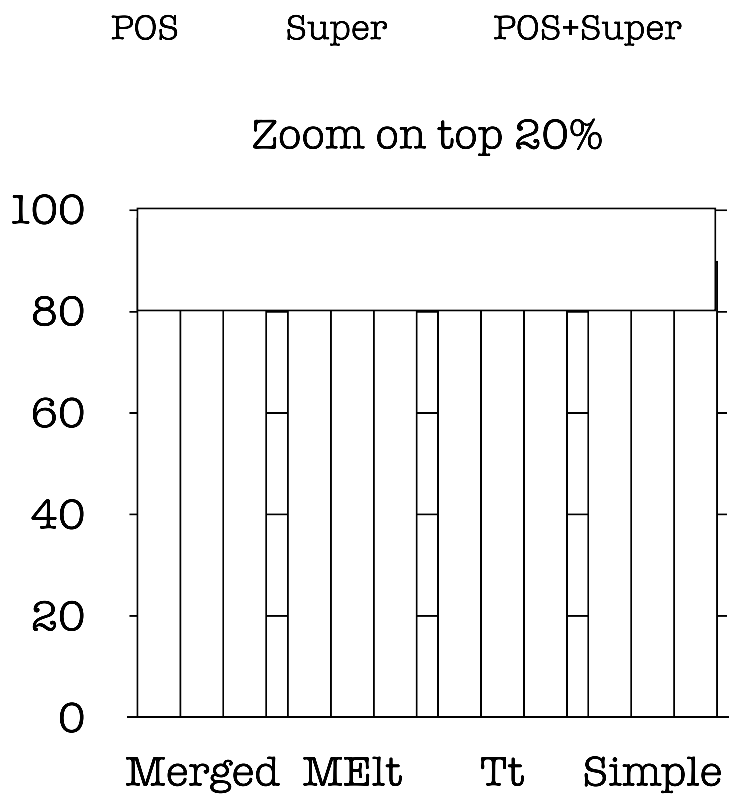
np/n	n/n	(n\n)/(s/np)	np	(np\s)/np
DET	ADJ	PROREL	NPP	V
le	chien	que	Marie	aime

POS/Supertagger Results



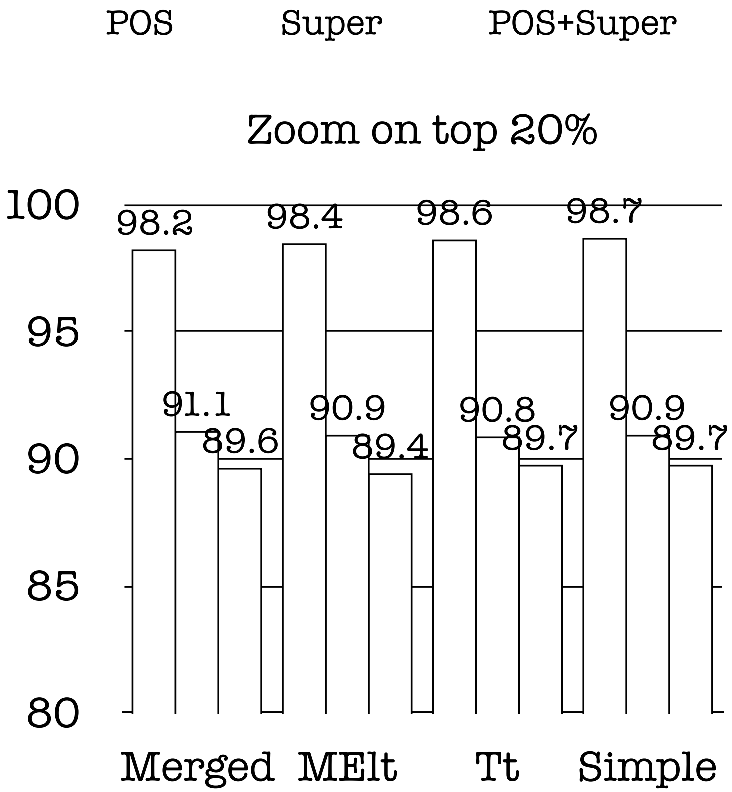
- A plot of POS/ Supertagger results for the four different tagsets.
- **POS+Super** gives the % correct supertags given the POS-tag <u>assigned by the tagger</u>, **Super** is the correct supertag given the <u>correct</u> POS-tag.

POS/Supertagger Results



- A plot of POS/ Supertagger results for the four different tagsets.
- **POS+Super** gives the % correct supertags given the POS-tag assigned by the model, **Super** is the correct supertag given the correct POS-tag.

POS/Supertagger Results



- A plot of POS/ Supertagger results for the four different tagsets.
- **POS+Super** gives the % correct supertags given the POS-tag assigned by the model, **Super** is the correct supertag given the correct POS-tag.

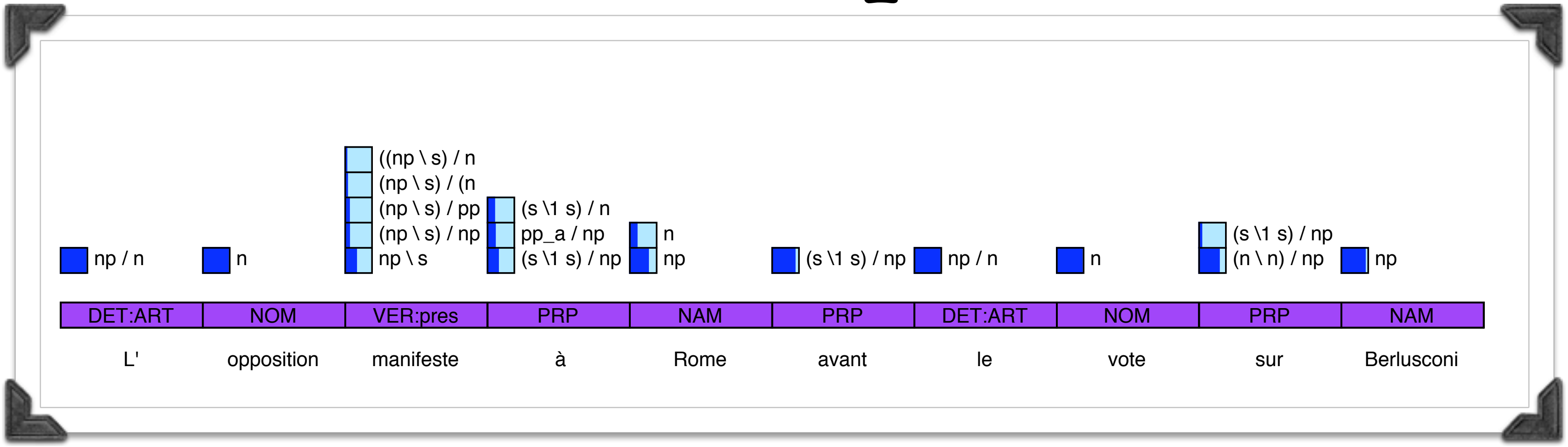
Multiple Solutions

- Though these results are comparable to the best supertaggers for English, in practice, even at around 91% correct supertags, we do not cover enough sentences of the corpus.
- A standard solution is to look at supertags within a range depending on the best supertag.
- This is called the β value.

Multiple Solutions

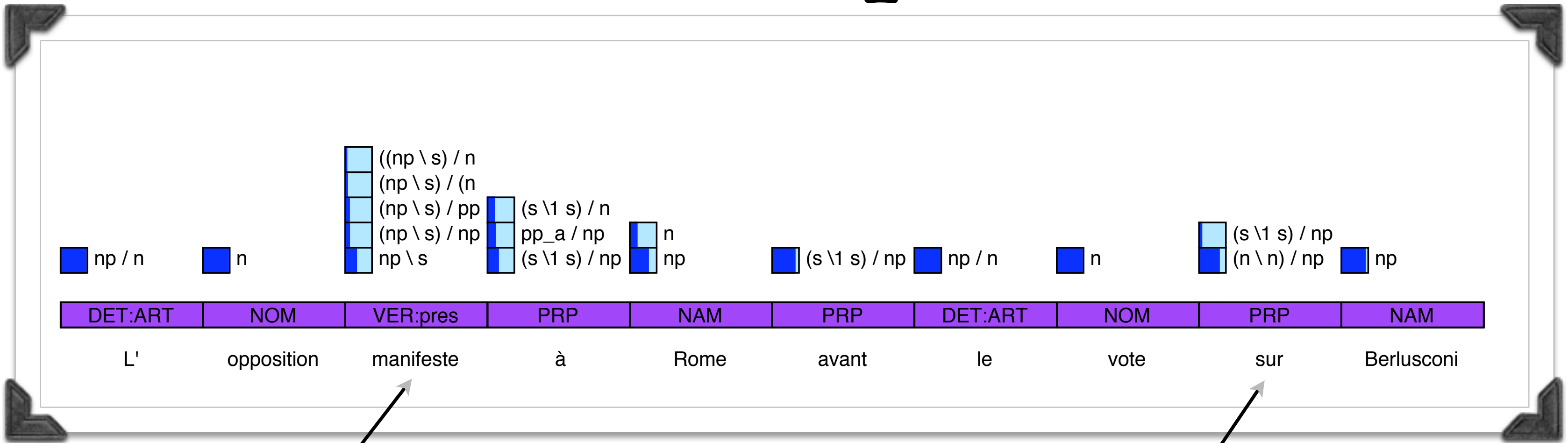
- Roughly speaking: if p is the probability of the best supertag, we will assign all supertags of probability > βp
- So, the less we are sure of our first supertag, the more alternatives we add.
- On average, a β of 0.1 gives 2.7 supertags per word, 0.05 gives 3.1 and 0.01 gives 4.7

Example



- Here is an example with β=0.1
- We can see that many "easy" words get assigned a single supertag whereas difficult words (here: verbs and prepositions) get assigned many tags.

Example

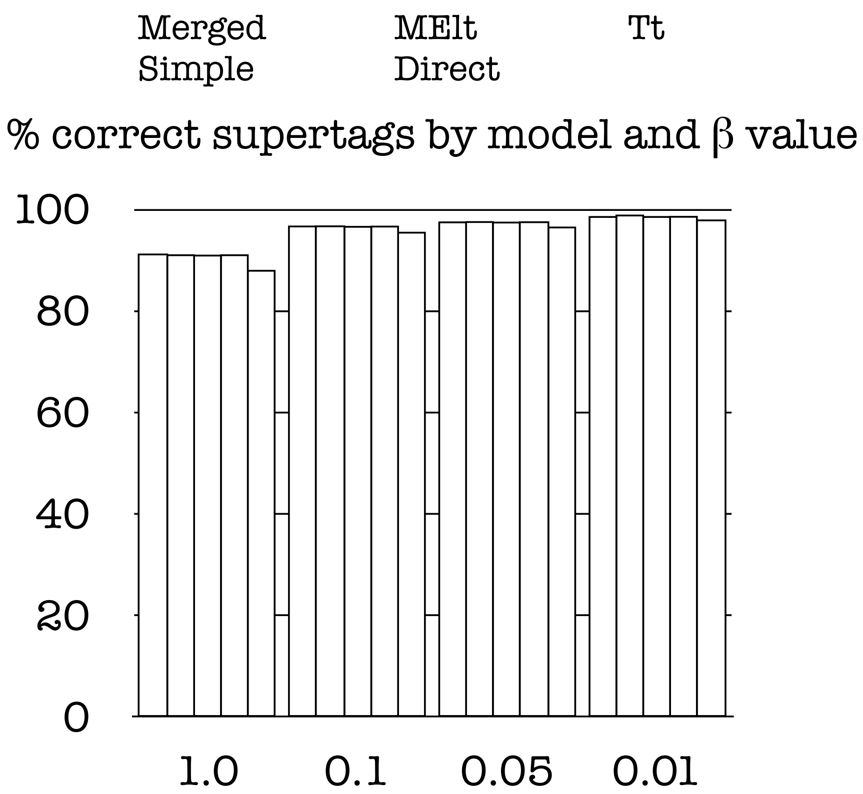


"manifeste"	%
np\s	43.6%
(np\s)/np	15.7%
(np\s)/pp_a	15.3%
(np\s)/(np\s_{ainf})	7.7%
((np\s)/np)/pp_a)	5.1%

"sur"	%
(n\n)/np	79.1%
(s\$_1$s)/np	9.4%

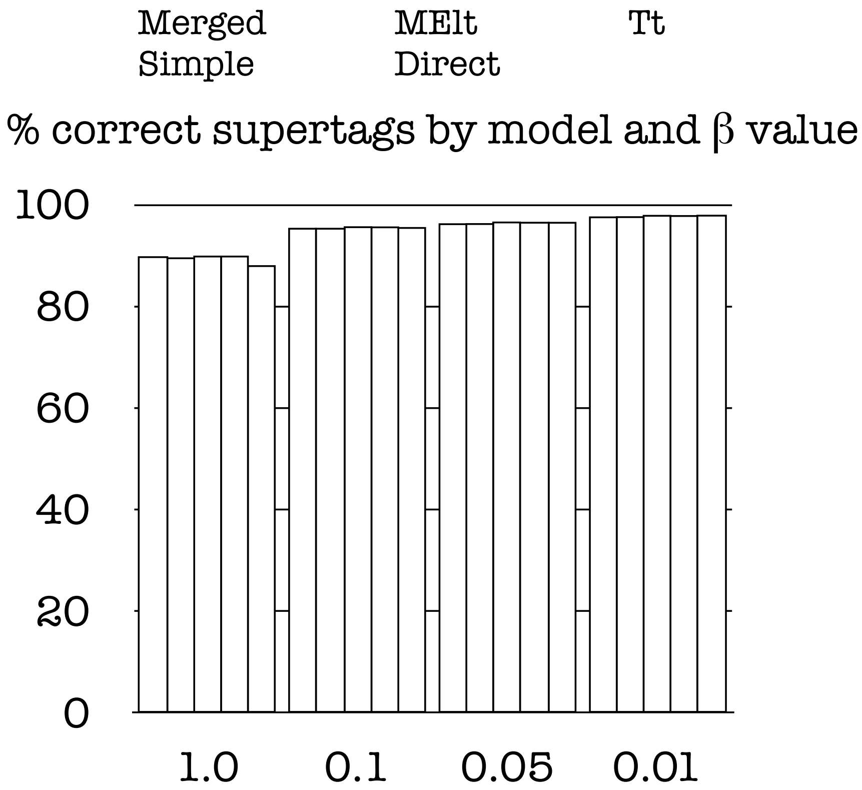
Remark: this is very typical of prepositions, they are either arguments (of verbs, or, more rarely, at least in our analysis, of nouns) or modifiers (of VPs/sentences, so-called adverbial uses, or of nouns)
Adverbial uses are assigned to take scope at the sentence-level instead of at the VP level: this is a simplification, but semantically, we just need the event/state variable of the verb and the subject variable (some adverbs, like "ensemble" or "tous" do clearly need the subject variable, of course!

POS/Supertagger Results



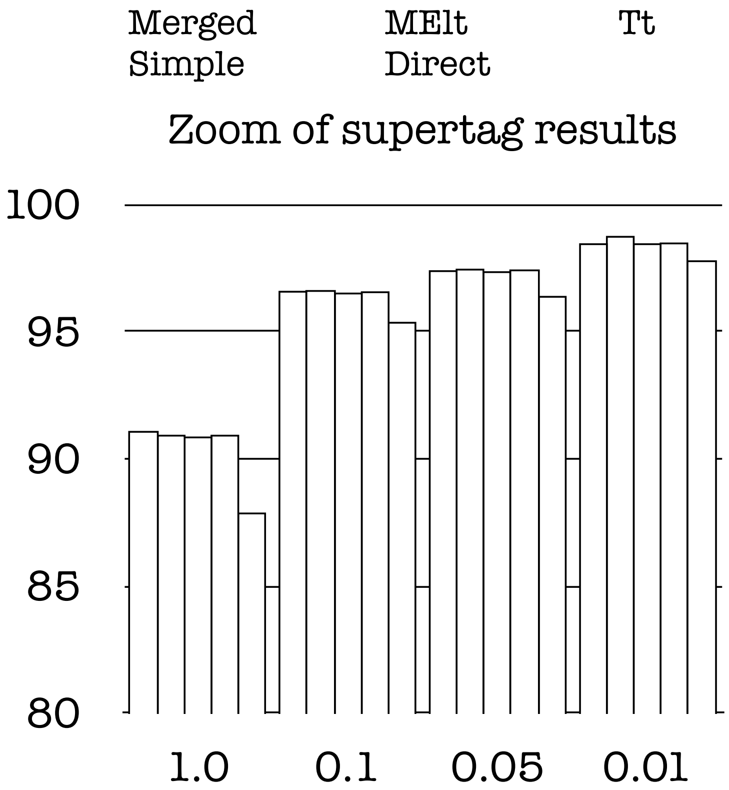
- Results with the use of different values of β.
- In a sense, the β value allows us to trade coverage for efficiency: at higher values of β, we parse more sentences, but we do so more slowly.

POS/Supertagger Results



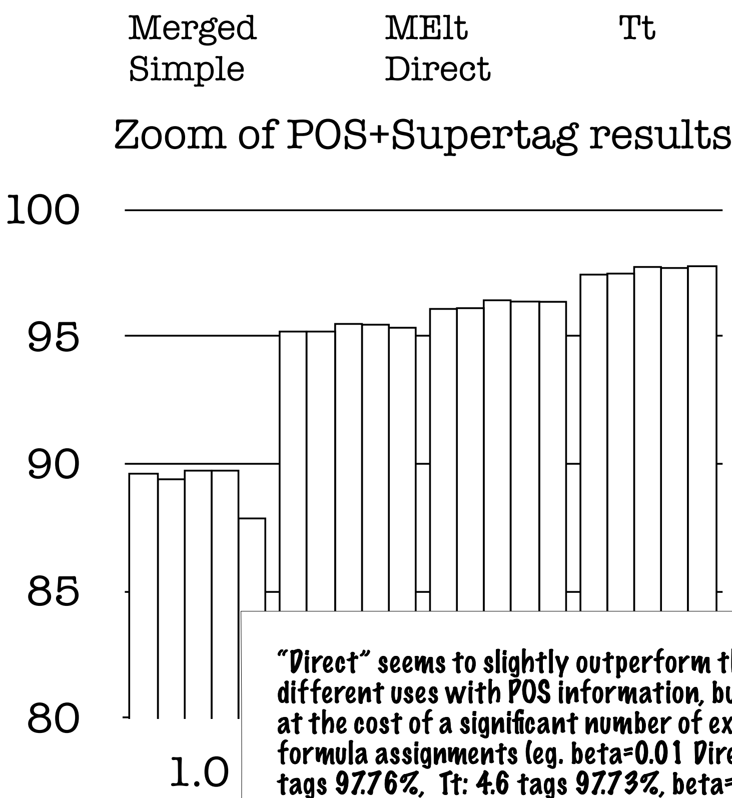
- As before, there is a slight decrease in performance once we switch from "gold" POS tag to tags assigned by the tagger.
- Eg. for the Treetagger tagset, it is -1.0% at β=0.1 and -0.5% at β=0.01

POS/Supertagger Results



- A comparison of the Supertagger and the combined POS/ Supertagger.
- Same results as the previous slides, but with a zoom on the top 20 percentile.
- Direct is the result of the Supertagger without POS info.

POS/Supertagger Results

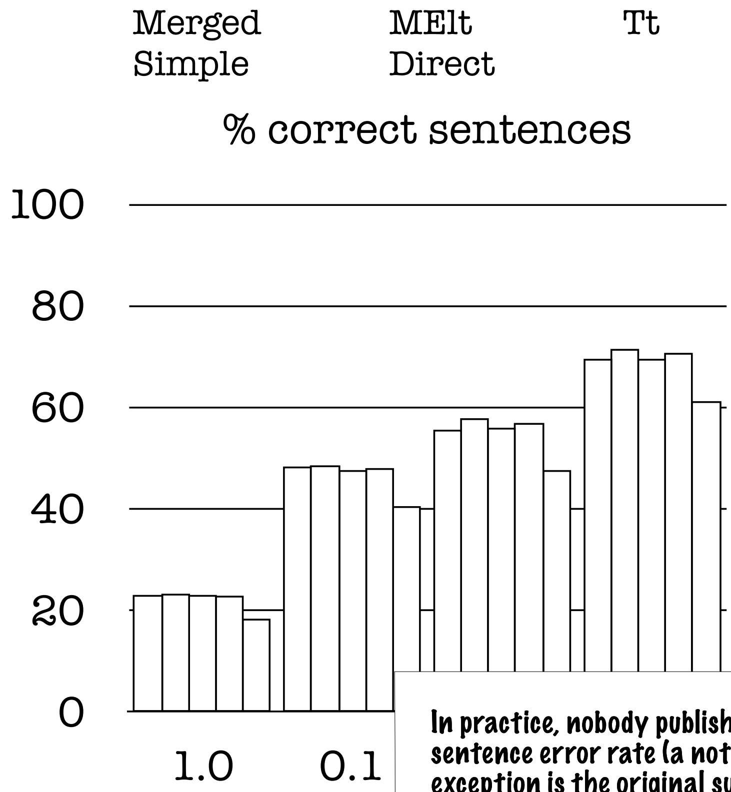


"Direct" seems to slightly outperform the different uses with POS information, but this is at the cost of a significant number of extra formula assignments (eg. beta=0.01 Direct: 5.6 tags 97.76%, Tt: 4.6 tags 97.73%, beta=0.001 Direct 12.4 tags, 98.42% Tt: 9.1 tags 98.40%). So, though incorrect POS tags can sometimes hurt performance, even at high beta levels, the important reduction in the number of tags per word outweighs (IMHO) the slight reduction in correct tags.

- A comparison of the Supertagger and the combined POS/ Supertagger.
- Same results as the previous slides, but with a zoom on the top 20 percentile.

Direct is the result of the Supertagger without POS info.

POS/Supertagger Results



In practice, nobody publishes there per sentence error rate (a notable exception is the original supertagging paper). This is because in general, they tend to be quite unflattering (eg. 98.2% correct POS tags corresponds to 65.1% correct sentences, the figures for beta=0.01 indicate a similar picture)

- Finally, here is the percentage of sentences which are assigned the correct sequence of supertags for the different settings of β and the different POS models.
- Note that we number of sentences for which a parse is found is actually better (around 85% at β=0.01)

Semantics

On the development of two different semantic lexicons for the wide-coverage grammar

Formulas as Types

- As is well-known, formulas in categorial grammars correspond to types in the simply typed lambda calculus
- Proofs (parses) correspond to lambda terms.
- By substituting lambda terms from the lexicon, we obtain a Montague-style meaning of analysed sentences.

Formulas as Types

- The translation of formulas to terms is the following
- The only thing to note is that we use a "lifted" type for noun phrases: $(e \rightarrow t) \rightarrow t$ instead of the more usual e
- This choice is will simplify things later on.

formula	type
type(np) =	$(e \rightarrow t) \rightarrow t$
type(s) =	t
type(n) =	$e \rightarrow t$
type(A/B) =	type(B) $\rightarrow$ type(A)
type(B\A) =	type(B) $\rightarrow$ type(A)

Formulas as Types

word	formula	lambda term
Jean	np	$\lambda P.P(j)$
Marie	np	$\lambda P.P(m)$
dort	np\s	$\lambda S.(S\ \lambda x.dort(x))$
aime	(np\s)/np	$\lambda O \lambda S.(S\ \lambda x.\ O(\lambda y.\ aime(x,y)))$
chaque	np/n	$\lambda P \lambda Q \forall x\ P(x) \rightarrow Q(x)$
homme	n	$\lambda x.homme(x)$

- This is a very basic extensional Montague grammar lexicon for categorial grammar.
- Only the verb types are slightly more complicated than usual.

Of course this has the disadvantage that we do not treat scope ambiguity but fix it at subject wide scope readings.
A simple but laborious solution would be to multiply verb semantics

Formulas as Types

word	formula	lambda term
Jean	np	$\lambda P.([j\|] \oplus P(j))$
Marie	np	$\lambda P.([m\|] \oplus P(m))$
dort	np\s	$\lambda S.(S\ \lambda x.[\|dort(x)])$
aime	(np\s)/np	$\lambda O \lambda S.(S\ \lambda x.\ O(\lambda y.\ [\|aime(x,y)]))$
chaque	np/n	$\lambda P \lambda Q[\|[x\|P(x)] \rightarrow [\|Q(x)]]$
homme	n	$\lambda x.[\|homme(x)]$
il	np	$\lambda P.([x\|x = ?] \oplus P(x))$

- DRT: $t := s \rightarrow (s \rightarrow t)$
- [x|...] add reference marker “x” to the context
- x = ? select an appropriate marker from the context

Formulas as Types

word	formula	lambda term
Jean	np	$\lambda P e \phi. (((P\ j)\ e)\ \lambda e' \phi(j::e'))$
Marie	np	$\lambda P e \phi. (((P\ m)\ e)\ \lambda e' \phi(m::e'))$
dort	np\s	$\lambda S.(S\ \lambda x e \phi. (dort(x) \wedge (\phi\ e)))$
aime	(np\s)/np	$\lambda O \lambda S.(S\ \lambda x. O(\lambda y e \phi. (aime(x,y) \wedge (\phi\ e)))$
chaque	np/n	$\lambda P Q e. (\forall x\ (P\ x) \rightarrow ((Q\ x)\ (x::e)))$
homme	n	$\lambda x e \phi. homme(x) \wedge (\phi\ e)$
il	np	$\lambda P e.((P\ (sel_m\ e))\ e)$

- Montegovian Dynamics: $t := s \rightarrow (s \rightarrow t) \rightarrow t$ (de Groote)
- x::e add “x” to the context “e”
- sel::e select an appropriate term from the context “e”

Formulas as Types

word	formula	lambda term
Jean	np	$\lambda P e \phi.\ (((P\ j)\ e)\ \lambda e' \phi(j{::}e'))$
Marie	np	$\lambda P e \phi.\ (((P\ m)\ e)\ \lambda e' \phi(m{::}e'))$
dort	np\s	$\lambda S.(S\ \lambda x e \phi.\ (dort(x) \wedge (\phi\ e)))$
aime	(np\s)/np	$\lambda O \lambda S.(S\ \lambda x.\ O(\lambda y e \phi.\ (aime(x,y) \wedge (\phi\ e)))$
chaque	np/n	$\lambda P Q e \phi.\ (\forall x \neg((P\ x)\ e)\ (\lambda e' \neg((Q\ x)\ (x{::}e'))\ (\lambda e''.T))) \wedge (\phi\ e)$
homme	n	$\lambda x e \phi. homme(x) \wedge (\phi\ e)$
il	np	$\lambda P e.((P\ (sel_m\ e))\ e)$

- Montegovian Dynamics: $t := s \rightarrow (s \rightarrow t) \rightarrow t$ (de Groote)
- x::e add “x” to the context
- sel::e select an appropriate term from the context

Towards Wide-Coverage Semantics

- In order to move beyond a simple lexicon listing a limited number of words, it suffices to remark that many of the “open class” words (eg. names, nouns, verbs) follow a general schema to obtain their lexical semantics.
- For example, a noun “n” generally has $\lambda x.n(x)$ as its semantics.

Towards Wide-Coverage Semantics

- So the basic idea behind wide-coverage semantics is very simple:
  - the lexicon lists words which require special treatment (eg. conjunctions “et” and auxiliary verbs like “être” and “avoir”)
  - other words are assigned a lambda term based on their root form and POS tag

So the general motto is: if you want to add more information to the semantic lexicon, there are two basic (non-exclusive) solutions: 1) you list the different cases 2) you train a (reliable) tagger
Solution 1 would be an option for distinguishing subjet/object control verbs and Solution 2 would be an option for Named Entities (and their types: persons, places, enterprises)

Towards Wide-Coverage Semantics

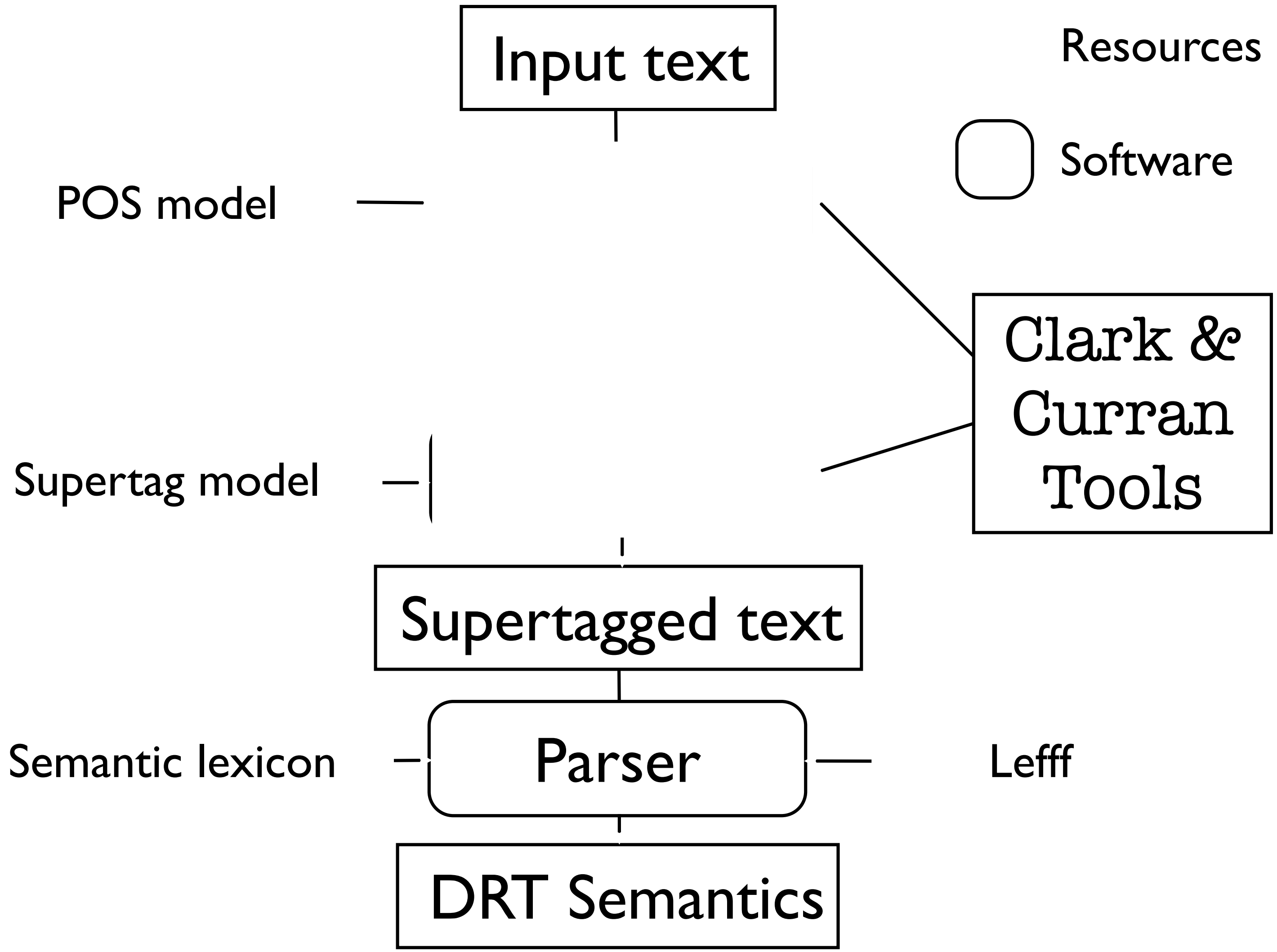
Example entries

"dormir" is a state rather than an event, however, the current system does not distinguish between different types of eventualities.

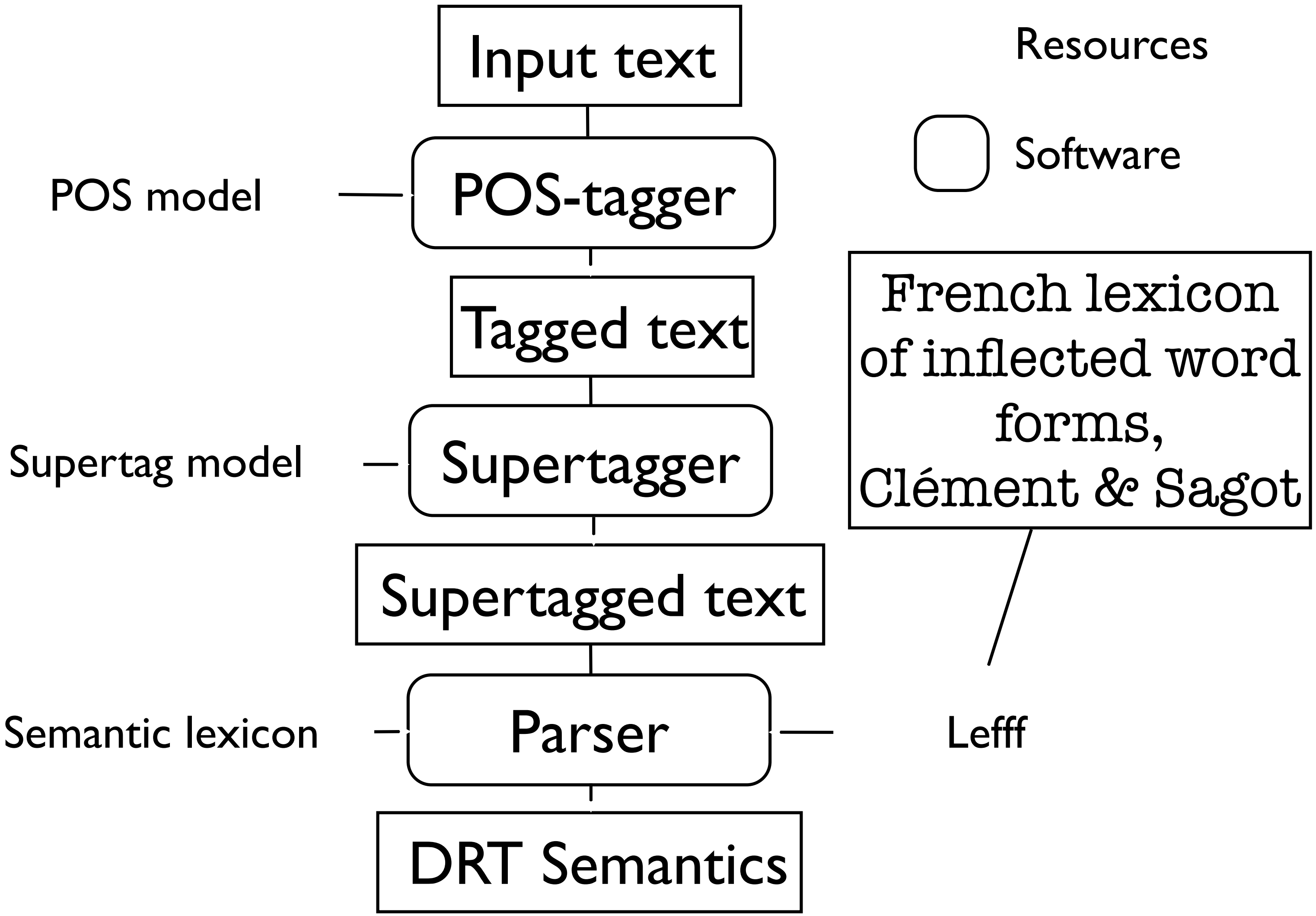
$$\text{dort} : np\backslash_0 s - \lambda L_0 e_0 . L_0(\lambda z_0 . \begin{array}{|l|}\hline e_0 \\ \hline \mathbf{event}(e_0) \\ \mathbf{dort}(e_0, z_0) \\ \hline \end{array})$$

$$\text{pousser} : ((np\backslash_0 s)/_0(np\backslash_0 s_{ainf}))/_0 np - \lambda x_0 y_0 z_0 x_1 . x_0(\lambda y_1 . z_0(\lambda z_1 . \begin{array}{|l|}\hline d_2 \\ \hline \mathbf{pousser\_à}(x_1) \\ \mathbf{agent}(x_1, z_1) \\ \mathbf{patient}(x_1, y_1) \\ \mathbf{theme}(x_1, y_2) \\ y_2 : y_0(z_2, x_0) \\ \hline \end{array}))$$

Grail & Friends



Grail & Friends



Demo

- All talk and no demo make Jack a dull boy.
- All talk and no demo make Jack a dull boy.

Give a demo of the system with today's headlines from "Google Actualités"

Conclusion

- I have described the development of a wide-coverage categorial grammar for French and first steps towards using it for wide-coverage <u>semantics</u>
- All software and resources are available under LGPL (with the unfortunate exception of the annotated corpus, which is bound by the same conditions as the Paris VII treebank).

Future Work

- A very long list, but I will mention some of the more important tasks.

Future Work - Parser

- Improve the accuracy of the extracted grammar and the parser
- Improve the efficiency of parser (eg. by using tree automata)

(as in Noémie-Fleur's talk, of course !)

- Add a component for multi-word expressions.

Future Work - Semantics

- Incorporate a Named-Entity component.
- Incorporate a rudimentary analysis of tense/aspect and discourse structure.
- Others (eg. word sense disambiguation)
- General problem: lack of annotated data

Future Work - Semantics

- Open questions:
  - how "deep" can we go with wide-coverage semantics?
  - what are appropriate evaluation measure?