

Building an Italian CG bank via incremental statistical parsing

ANDREA BOLOGNESI
DIPARTIMENTO DI SCIENZE MATEMATICHE ED INFORMATICHE
“ROBERTO MAGARI”
UNIVERSITY OF SIENA
Email: bolognesi2@unisi.it

Contents

1	Towards a wide-coverage parser for Italian	4
1.1	State of the art: CG induction	5
2	Categorical Grammar Framework reminder	6
2.1	Function-Argument Structures	7
2.2	Polarity of formulae	8
3	Wide-coverage parsing	9
3.1	Proof nets	10
3.2	Proof nets: unfolding rules example	11
3.3	Proof nets: axiom linking	12
3.4	Proof nets: correctness criteria	13
3.5	Proof net example	14
3.6	Structures ambiguity	15
4	Exploiting statistic information	16
4.1	Axiom Links with context	17
4.2	Example: structural ambiguity with probabilities	18
5	Evaluation	19
5.1	Translating TUT Trees to CG Proof Net	20

5.2	Training set	21
5.3	Evaluation	22
6	Conclusions and Future Works	23

1. Towards a wide-coverage parser for Italian

Aim Towards a wide-coverage parser for Italian

1. Towards a wide-coverage parser for Italian

Aim Towards a wide-coverage parser for Italian

How By

- ▶ getting a suitable parser for Italian
- ▶ extracting a Grammar and statistical information from a given Tree-bank,
- ▶ then training the parser on the acquired information obtaining a wide-coverage parser, and
- ▶ finally, parsing raw corpora.

1. Towards a wide-coverage parser for Italian

Aim Towards a wide-coverage parser for Italian

How By

- ▶ getting a suitable parser for Italian
- ▶ extracting a Grammar and statistical information from a given Tree-bank,
- ▶ then training the parser on the acquired information obtaining a wide-coverage parser, and
- ▶ finally, parsing raw corpora.

Resources at our disposal: Italian Treebanks and Large Raw Corpora

- ▶ TUT (Turin University Treebank): Collection of syntactically annotated Dependency Grammar (DG) Italian sentences (1.800 sentences); Downloadable and Open source.
- ▶ CORIS/CODIS, a 100-million-word synchronic corpus of contemporary written Italian.

1.1. State of the art: CG induction

- ▶ We are interested to reach as a second step wide-coverage semantic annotation

1.1. State of the art: CG induction

- ▶ We are interested to reach as a second step wide-coverage semantic annotation
- ▶ Hence, we use a formal grammar known for the tied connection between syntax and semantics, namely Categorical Grammar (CG).

1.1. State of the art: CG induction

- ▶ We are interested to reach as a second step wide-coverage semantic annotation
- ▶ Hence, we use a formal grammar known for the tight connection between syntax and semantics, namely Categorical Grammar (CG).
- ▶ Grammar Induction work within this framework has been carried out for English and Dutch, so far.
 - ▶ CCGBank for English: based on Penn Treebank (1 million word corpus of manually annotated constituents from the Wall Street Journal). [Hockenmaier and Steedman 2003]
 - ▶ CTL for Dutch: based on CG Dependency Tree Bank (1 million words). [Moortgat and Moot 2002, Moot 2003]

1.1. State of the art: CG induction

- ▶ We are interested to reach as a second step wide-coverage semantic annotation
- ▶ Hence, we use a formal grammar known for the tied connection between syntax and semantics, namely Categorical Grammar (CG).
- ▶ Grammar Induction work within this framework has been carried out for English and Dutch, so far.
 - ▶ CCGBank for English: based on Penn Treebank (1 million word corpus of manually annotated constituents from the Wall Street Journal). [Hockenmaier and Steedman 2003]
 - ▶ CTL for Dutch: based on CG Dependency Tree Bank (1 million words). [Moortgat and Moot 2002, Moot 2003]
- ▶ I am currently working on CTL (Categorical Type Logic) which is a family of logics tracing back to CG, but the work presented in this talk is based on the core logic only.

2. Categorical Grammar Framework reminder

- ▶ CG is a lexicalized grammar formalism that aims to capture the core of linguistic structure composition.

2. Categorical Grammar Framework reminder

- ▶ CG is a lexicalized grammar formalism that aims to capture the core of linguistic structure composition.
- ▶ Typically CGs have two function types, depending on whether the function takes its argument from the left or the right:

2. Categorical Grammar Framework reminder

- ▶ CG is a lexicalized grammar formalism that aims to capture the core of linguistic structure composition.
- ▶ Typically CGs have two function types, depending on whether the function takes its argument from the left or the right:
 - ▶ $A \setminus B$ ($A \rightarrow B$), that means “looking for a formula A on the left to yield a B ”

2. Categorical Grammar Framework reminder

- ▶ CG is a lexicalized grammar formalism that aims to capture the core of linguistic structure composition.
- ▶ Typically CGs have two function types, depending on whether the function takes its argument from the left or the right:
 - ▶ $A \backslash B$ ($A \rightarrow B$), that means “looking for a formula A on the left to yield a B ”
 - ▶ B / A ($A \rightarrow B$), that means “looking for an A on the right to yield a B ”

2. Categorical Grammar Framework reminder

- ▶ CG is a lexicalized grammar formalism that aims to capture the core of linguistic structure composition.
- ▶ Typically CGs have two function types, depending on whether the function takes its argument from the left or the right:
 - ▶ $A \setminus B$ ($A \rightarrow B$), that means “looking for a formula A on the left to yield a B ”
 - ▶ B / A ($A \rightarrow B$), that means “looking for an A on the right to yield a B ”
- ▶ A simple example:

Lexicon (Categorization):

2. Categorical Grammar Framework reminder

- ▶ CG is a lexicalized grammar formalism that aims to capture the core of linguistic structure composition.
- ▶ Typically CGs have two function types, depending on whether the function takes its argument from the left or the right:
 - ▶ $A \backslash B$ ($A \rightarrow B$), that means “looking for a formula A on the left to yield a B ”
 - ▶ B / A ($A \rightarrow B$), that means “looking for an A on the right to yield a B ”
- ▶ A simple example:

Lexicon (Categorization): Sara, Ilaria:

2. Categorical Grammar Framework reminder

- ▶ CG is a lexicalized grammar formalism that aims to capture the core of linguistic structure composition.
- ▶ Typically CGs have two function types, depending on whether the function takes its argument from the left or the right:
 - ▶ $A \backslash B$ ($A \rightarrow B$), that means “looking for a formula A on the left to yield a B ”
 - ▶ B / A ($A \rightarrow B$), that means “looking for an A on the right to yield a B ”
- ▶ A simple example:

Lexicon (Categorization): Sara, Ilaria: np

2. Categorical Grammar Framework reminder

- ▶ CG is a lexicalized grammar formalism that aims to capture the core of linguistic structure composition.
- ▶ Typically CGs have two function types, depending on whether the function takes its argument from the left or the right:
 - ▶ $A \backslash B$ ($A \rightarrow B$), that means “looking for a formula A on the left to yield a B ”
 - ▶ B / A ($A \rightarrow B$), that means “looking for an A on the right to yield a B ”
- ▶ A simple example:

Lexicon (Categorization): Sara, Ilaria: np conosce (*knows*):

2. Categorical Grammar Framework reminder

- ▶ CG is a lexicalized grammar formalism that aims to capture the core of linguistic structure composition.
- ▶ Typically CGs have two function types, depending on whether the function takes its argument from the left or the right:
 - ▶ $A \backslash B$ ($A \rightarrow B$), that means “looking for a formula A on the left to yield a B ”
 - ▶ B / A ($A \rightarrow B$), that means “looking for an A on the right to yield a B ”
- ▶ A simple example:

Lexicon (Categorization): Sara, Ilaria: np conosce (*knows*): $(np \backslash s) / np$

2. Categorical Grammar Framework reminder

- ▶ CG is a lexicalized grammar formalism that aims to capture the core of linguistic structure composition.
- ▶ Typically CGs have two function types, depending on whether the function takes its argument from the left or the right:
 - ▶ $A \backslash B$ ($A \rightarrow B$), that means “looking for a formula A on the left to yield a B ”
 - ▶ B / A ($A \rightarrow B$), that means “looking for an A on the right to yield a B ”
- ▶ A simple example:

Lexicon (Categorization): Sara, Ilaria: np conosce (*knows*): $(np \backslash s) / np$
Rules (Assembly):

2. Categorical Grammar Framework reminder

- ▶ CG is a lexicalized grammar formalism that aims to capture the core of linguistic structure composition.
- ▶ Typically CGs have two function types, depending on whether the function takes its argument from the left or the right:
 - ▶ $A \backslash B$ ($A \rightarrow B$), that means “looking for a formula A on the left to yield a B ”
 - ▶ B / A ($A \rightarrow B$), that means “looking for an A on the right to yield a B ”
- ▶ A simple example:

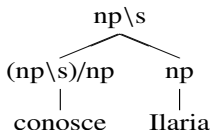
Lexicon (Categorization): Sara, Ilaria: np conosce (*knows*): $(np \backslash s) / np$
Rules (Assembly): “Sara conosce Ilaria”

2. Categorical Grammar Framework reminder

- ▶ CG is a lexicalized grammar formalism that aims to capture the core of linguistic structure composition.
- ▶ Typically CGs have two function types, depending on whether the function takes its argument from the left or the right:
 - ▶ $A \backslash B$ ($A \rightarrow B$), that means “looking for a formula A on the left to yield a B ”
 - ▶ B / A ($A \rightarrow B$), that means “looking for an A on the right to yield a B ”
- ▶ A simple example:

Lexicon (Categorization): Sara, Ilaria: np *conosce* (*knows*): $(np \backslash s) / np$

Rules (Assembly): “Sara *conosce* Ilaria”

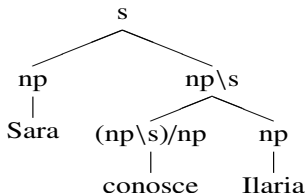


2. Categorical Grammar Framework reminder

- ▶ CG is a lexicalized grammar formalism that aims to capture the core of linguistic structure composition.
- ▶ Typically CGs have two function types, depending on whether the function takes its argument from the left or the right:
 - ▶ $A \backslash B$ ($A \rightarrow B$), that means “looking for a formula A on the left to yield a B ”
 - ▶ B / A ($A \rightarrow B$), that means “looking for an A on the right to yield a B ”
- ▶ A simple example:

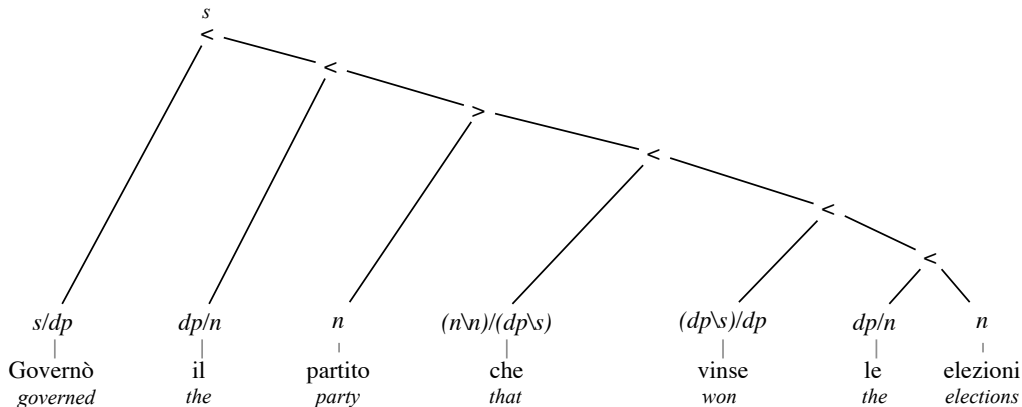
Lexicon (Categorization): Sara, Ilaria: np conosce (*knows*): $(np \backslash s) / np$

Rules (Assembly): “Sara conosce Ilaria”



2.1. Function-Argument Structures

A fa-structure for an expression is a binary tree where the leaf nodes are labeled by lexical expressions (words). We use $<$ and $>$ symbols in the internal nodes to indicate the position of functions.



2.2. Polarity of formulae

When looking at functional application an important point to notice is the **polarity** of the formula, which can be **positive** (\bullet) or **negative** (\circ).

2.2. Polarity of formulae

When looking at functional application an important point to notice is the **polarity** of the formula, which can be **positive** (\bullet) or **negative** (\circ).

Given the functions B/A and $A \setminus B$,

- ▶ A is in a negative position (\circ): it is what functions are looking for to yield B ,

2.2. Polarity of formulae

When looking at functional application an important point to notice is the **polarity** of the formula, which can be **positive** (\bullet) or **negative** (\circ).

Given the functions B/A and $A \setminus B$,

- ▶ A is in a negative position (\circ): it is what functions are looking for to yield B ,
- ▶ and B is in a positive position (\bullet): it is the value that will be obtained if the missing A is provided.

$$A^\circ \rightarrow B^\bullet$$

2.2. Polarity of formulae

When looking at functional application an important point to notice is the **polarity** of the formula, which can be **positive** (\bullet) or **negative** (\circ).

Given the functions B/A and $A \setminus B$,

- ▶ A is in a negative position (\circ): it is what functions are looking for to yield B ,
- ▶ and B is in a positive position (\bullet): it is the value that will be obtained if the missing A is provided.

$$A^\circ \rightarrow B^\bullet$$

For example, in the function $(np_1 \setminus s)/np_2$, np_1 and np_2 have negative polarity (\circ) and s has positive polarity (\bullet):

$$np_2^\circ \rightarrow (np_1^\circ \rightarrow s^\bullet)$$

2.2. Polarity of formulae

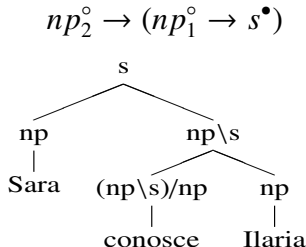
When looking at functional application an important point to notice is the **polarity** of the formula, which can be **positive** (●) or **negative** (○).

Given the functions B/A and $A \setminus B$,

- ▶ A is in a negative position (○): it is what functions are looking for to yield B ,
- ▶ and B is in a positive position (●): it is the value that will be obtained if the missing A is provided.

$$A^\circ \rightarrow B^\bullet$$

For example, in the function $(np_1 \setminus s)/np_2$, np_1 and np_2 have negative polarity (○) and s has positive polarity (●):



2.2. Polarity of formulae

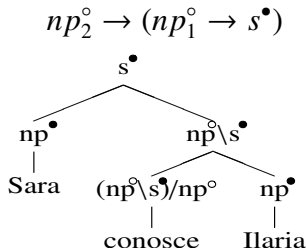
When looking at functional application an important point to notice is the **polarity** of the formula, which can be **positive** (●) or **negative** (○).

Given the functions B/A and $A \setminus B$,

- ▶ A is in a negative position (○): it is what functions are looking for to yield B ,
- ▶ and B is in a positive position (●): it is the value that will be obtained if the missing A is provided.

$$A^\circ \rightarrow B^\bullet$$

For example, in the function $(np_1 \setminus s)/np_2$, np_1 and np_2 have negative polarity (○) and s has positive polarity (●):



2.2. Polarity of formulae

When looking at functional application an important point to notice is the **polarity** of the formula, which can be **positive** (●) or **negative** (○).

Given the functions B/A and $A \setminus B$,

- ▶ A is in a negative position (○): it is what functions are looking for to yield B ,
- ▶ and B is in a positive position (●): it is the value that will be obtained if the missing A is provided.

$$A^\circ \rightarrow B^\bullet$$

For example, in the function $(np_1 \setminus s)/np_2$, np_1 and np_2 have negative polarity (○) and s has positive polarity (●):

$$np_2^\circ \rightarrow (np_1^\circ \rightarrow s^\bullet)$$

Polarity is counted recursively, hence in a higher order such as that for word ‘che’:
 $(n_1 \setminus n_2)/(dp \setminus s)$, n_1 and s have negative polarity (○) while n_2 and dp have positive polarity (●).

$$(dp^\bullet \rightarrow s^\circ) \rightarrow (n_1^\circ \rightarrow n_2^\bullet)$$

3. Wide-coverage parsing

- ▶ Need of statistical parsers (e.g. CCG parser [S. Clark and J. Hockenmaier 2001])

3. Wide-coverage parsing

- ▶ Need of statistical parsers (e.g. CCG parser [S. Clark and J. Hockenmaier 2001])
- ▶ Proposal: add incrementality
 - ▶ Incremental processing means parsing input words from left to right.
 - ▶ It is relevant for language modeling and dynamic formal semantic representation [Mazzei 2005]
 - ▶ Semantic interpretation of partial structures may be incrementally carried out.
 - ▶ It is theoretically possible to “foresee” what word comes after the others on the basis of its types.

3. Wide-coverage parsing

- ▶ Need of statistical parsers (e.g. CCG parser [S. Clark and J. Hockenmaier 2001])
- ▶ Proposal: add incrementality
 - ▶ Incremental processing means parsing input words from left to right.
 - ▶ It is relevant for language modeling and dynamic formal semantic representation [Mazzei 2005]
 - ▶ Semantic interpretation of partial structures may be incrementally carried out.
 - ▶ It is theoretically possible to “foresee” what word comes after the others on the basis of its types.
- ▶ All possible syntactic analysis are generated incrementally: this allows the parser to prune them, by selecting only the most likely ones.

3. Wide-coverage parsing

- ▶ Need of statistical parsers (e.g. CCG parser [S. Clark and J. Hockenmaier 2001])
- ▶ Proposal: add incrementality
 - ▶ Incremental processing means parsing input words from left to right.
 - ▶ It is relevant for language modeling and dynamic formal semantic representation [Mazzei 2005]
 - ▶ Semantic interpretation of partial structures may be incrementally carried out.
 - ▶ It is theoretically possible to “foresee” what word comes after the others on the basis of its types.
- ▶ All possible syntactic analysis are generated incrementally: this allows the parser to prune them, by selecting only the most likely ones.
- ▶ Parsing with CTL can be efficiently performed following the proof nets method.

3. Wide-coverage parsing

- ▶ Need of statistical parsers (e.g. CCG parser [S. Clark and J. Hockenmaier 2001])
- ▶ Proposal: add incrementality
 - ▶ Incremental processing means parsing input words from left to right.
 - ▶ It is relevant for language modeling and dynamic formal semantic representation [Mazzei 2005]
 - ▶ Semantic interpretation of partial structures may be incrementally carried out.
 - ▶ It is theoretically possible to “foresee” what word comes after the others on the basis of its types.
- ▶ All possible syntactic analysis are generated incrementally: this allows the parser to prune them, by selecting only the most likely ones.
- ▶ Parsing with CTL can be efficiently performed following the proof nets method.
- ▶ Hence, we developed an incremental statistical parser based on proof nets.

3.1. Proof nets

Proof nets are a geometrical method of representing proofs.

3.1. Proof nets

Proof nets are a geometrical method of representing proofs.

A proof nets can be built *incrementally* and are compatible with incremental interpretation.

3.1. Proof nets

Proof nets are a geometrical method of representing proofs.

A proof nets can be built *incrementally* and are compatible with incremental interpretation.

The main concepts behind the use of proof nets are

- ▶ the order of occurrences and polarity of the leaves and
- ▶ the shape of their links.

3.1. Proof nets

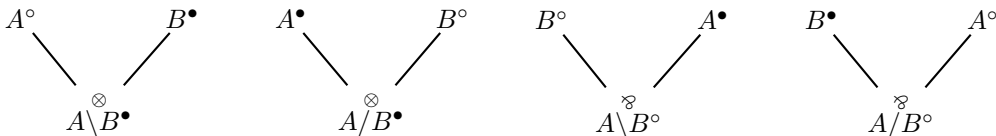
Proof nets are a geometrical method of representing proofs.

A proof nets can be built *incrementally* and are compatible with incremental interpretation.

The main concepts behind the use of proof nets are

- ▶ the order of occurrences and polarity of the leaves and
- ▶ the shape of their links.

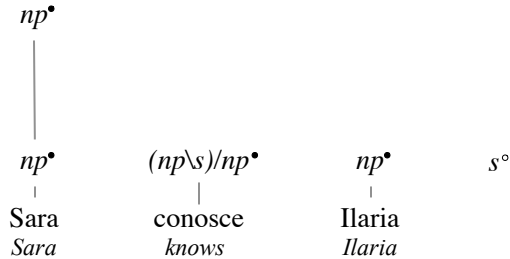
The former is determined by recursively applying unfolding rules, which formally express the idea of polarity introduced intuitively above.



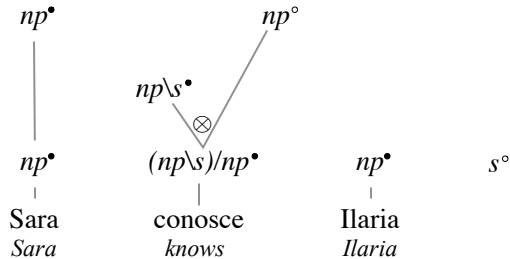
3.2. Proof nets: unfolding rules example

np^\bullet	$(np \setminus s) / np^\bullet$	np^\bullet	s°
Sara	conosce	Ilaria	
<i>Sara</i>	<i>knows</i>	<i>Ilaria</i>	

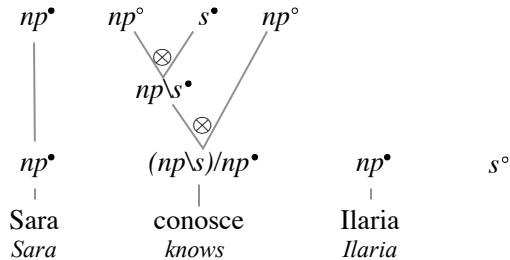
3.2. Proof nets: unfolding rules example



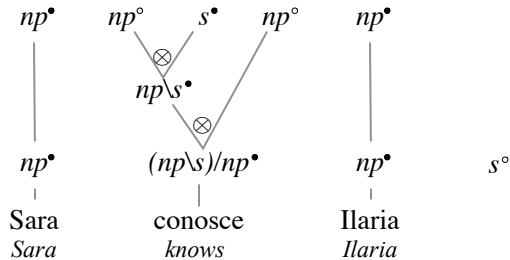
3.2. Proof nets: unfolding rules example



3.2. Proof nets: unfolding rules example

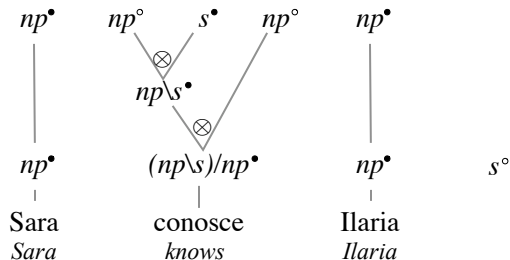


3.2. Proof nets: unfolding rules example



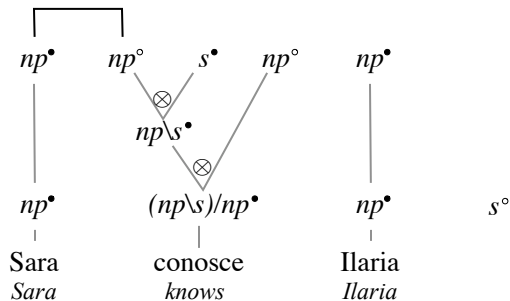
3.3. Proof nets: axiom linking

After lexical unfolding, matching pairs of axiomatic polar formulae are linked together (X° and X^\bullet , for a basic category X) satisfying planarity property.



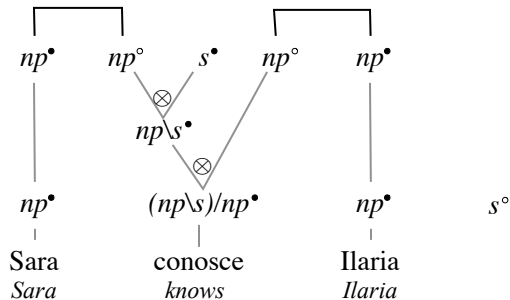
3.3. Proof nets: axiom linking

After lexical unfolding, matching pairs of axiomatic polar formulae are linked together (X° and X^\bullet , for a basic category X) satisfying planarity property.



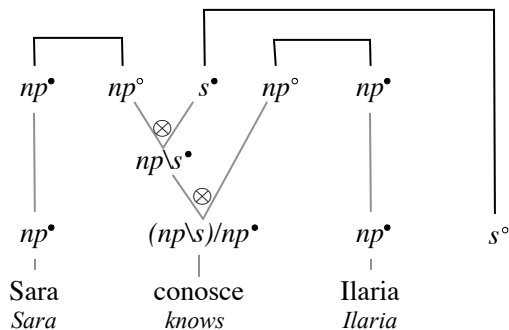
3.3. Proof nets: axiom linking

After lexical unfolding, matching pairs of axiomatic polar formulae are linked together (X° and X^\bullet , for a basic category X) satisfying planarity property.



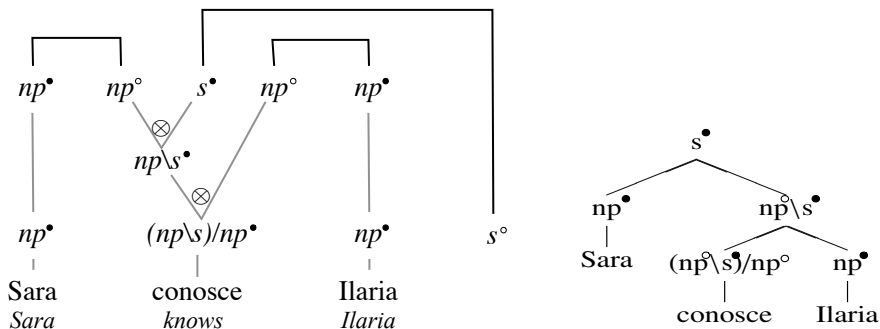
3.3. Proof nets: axiom linking

After lexical unfolding, matching pairs of axiomatic polar formulae are linked together (X° and X^\bullet , for a basic category X) satisfying planarity property.



3.3. Proof nets: axiom linking

After lexical unfolding, matching pairs of axiomatic polar formulae are linked together (X° and X^\bullet , for a basic category X) satisfying planarity property.



3.4. Proof nets: correctness criteria

A same unfolded structure could be completed by means of different links among the leaves.

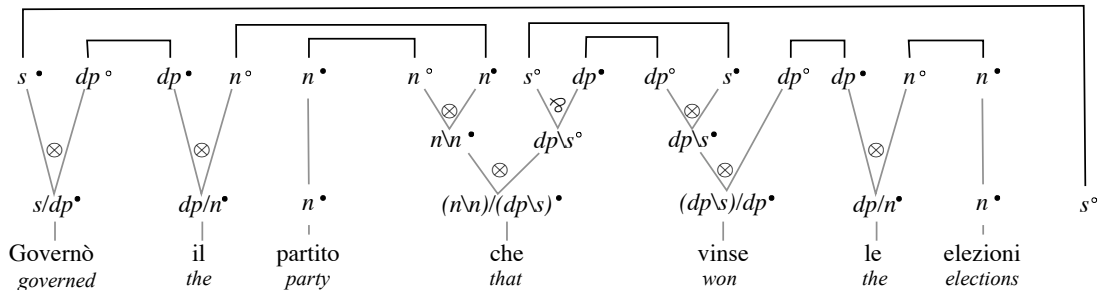
Correctness criteria have been defined to rule out those proof structures that do not correspond to logical proofs, hence are not proof nets.

We take as formally acceptable only those proof structures which verify correctness criteria:

- ▶ proof structures that have no crossing axiom links and
- ▶ with specific paths linking the formulae in the trees.

Correctness criteria can be checked incrementally.

3.5. Proof net example



3.6. Structures ambiguity

Finally, note that for a same structure there could be more than one axiom linking possibility.

This means that there could be more than one proof net that satisfies the correctness criterion.

3.6. Structures ambiguity

Finally, note that for a same structure there could be more than one axiom linking possibility.

This means that there could be more than one proof net that satisfies the correctness criterion.

In some cases this is due to real ambiguity of natural language structures, in others to syntactic ambiguities that would be ruled out if semantics is also taken into consideration.

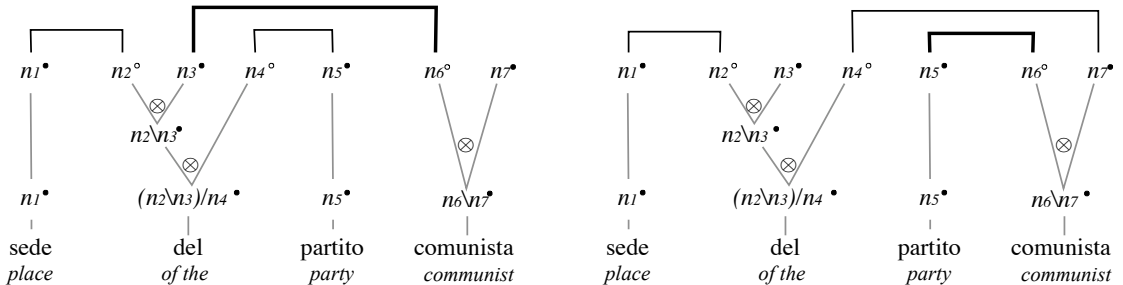
3.6. Structures ambiguity

Finally, note that for a same structure there could be more than one axiom linking possibility.

This means that there could be more than one proof net that satisfies the correctness criterion.

In some cases this is due to real ambiguity of natural language structures, in others to syntactic ambiguities that would be ruled out if semantics is also taken into consideration.

Furthermore, among the former, there are attachments that are more plausible than others.



4. Exploiting statistic information

- ▶ Given a CG lexicon, the parser discussed so far is able to retrieve all the possible proof nets for an input sentence.

4. Exploiting statistic information

- ▶ Given a CG lexicon, the parser discussed so far is able to retrieve all the possible proof nets for an input sentence.
- ▶ Of course many of these solutions are undesired, because they refer to wrong bracketing, representing wrong dependency assignments.

4. Exploiting statistic information

- ▶ Given a CG lexicon, the parser discussed so far is able to retrieve all the possible proof nets for an input sentence.
- ▶ Of course many of these solutions are undesired, because they refer to wrong bracketing, representing wrong dependency assignments.
- ▶ We use statistic information to assign a weight to each solution and choose the solution with highest likelihood.

4. Exploiting statistic information

- ▶ Given a CG lexicon, the parser discussed so far is able to retrieve all the possible proof nets for an input sentence.
- ▶ Of course many of these solutions are undesired, because they refer to wrong bracketing, representing wrong dependency assignments.
- ▶ We use statistic information to assign a weight to each solution and choose the solution with highest likelihood.
- ▶ To this end, we improved the incremental parser by assigning each axiom link with a probability value.

4. Exploiting statistic information

- ▶ Given a CG lexicon, the parser discussed so far is able to retrieve all the possible proof nets for an input sentence.
- ▶ Of course many of these solutions are undesired, because they refer to wrong bracketing, representing wrong dependency assignments.
- ▶ We use statistic information to assign a weight to each solution and choose the solution with highest likelihood.
- ▶ To this end, we improved the incremental parser by assigning each axiom link with a probability value.
- ▶ These values are collected from a given corpus, we will discuss in next slides.

4. Exploiting statistic information

- ▶ Given a CG lexicon, the parser discussed so far is able to retrieve all the possible proof nets for an input sentence.
- ▶ Of course many of these solutions are undesired, because they refer to wrong bracketing, representing wrong dependency assignments.
- ▶ We use statistic information to assign a weight to each solution and choose the solution with highest likelihood.
- ▶ To this end, we improved the incremental parser by assigning each axiom link with a probability value.
- ▶ These values are collected from a given corpus, we will discuss in next slides.
- ▶ During the parsing axiom links are placed from left to right upgrading with their own probability the whole probability of the proof structure they belong to.

4.1. Axiom Links with context

Let us consider the categorial type $n \setminus n$ which represents a function asking for an argument type n .

4.1. Axiom Links with context

Let us consider the categorial type $n \setminus n$ which represents a function asking for an argument type n . This argument may compare inside a bigger type, which represents its context.

4.1. Axiom Links with context

Let us consider the categorial type $n \setminus n$ which represents a function asking for an argument type n .

This argument may compare inside a bigger type, which represents its context.

For instance, n could be part of the type $(n \setminus n) / n$.

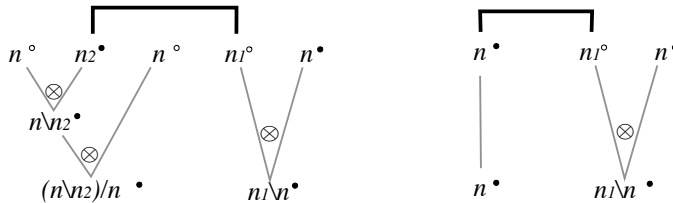
4.1. Axiom Links with context

Let us consider the categorial type $n \setminus n$ which represents a function asking for an argument type n .

This argument may compare inside a bigger type, which represents its context.

For instance, n could be part of the type $(n \setminus n) / n$.

Possible axiom links with context which involve $n \setminus n$ are



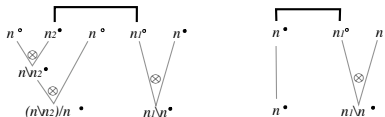
4.1. Axiom Links with context

Let us consider the categorical type $n \setminus n$ which represents a function asking for an argument type n .

This argument may compare inside a bigger type, which represents its context.

For instance, n could be part of the type $(n \setminus n) / n$.

Possible axiom links with context which involve $n \setminus n$ are



$$\text{prob} \left[\begin{array}{c} \overbrace{n^\circ \quad n2^\circ \quad n^\circ \quad n1^\circ \quad n^\circ} \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ n \setminus n2^\circ \quad n^\circ \quad n1^\circ \quad n^\circ \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ (n \setminus n2) / n \bullet \quad n \setminus n1^\circ \end{array} \right] =$$

4.1. Axiom Links with context

Let us consider the categorial type $n \setminus n$ which represents a function asking for an argument type n .

This argument may compare inside a bigger type, which represents its context.

For instance, n could be part of the type $(n \setminus n) / n$.

Possible axiom links with context which involve $n \setminus n$ are

The diagram illustrates axiom links with context for the type $n \setminus n$. It consists of two rows of diagrams.

The top row shows two basic axiom links:

- Left: A diagram with nodes n° , n_2^\bullet , n° , n_1^\bullet , and n^\bullet . A bracket connects n° and n_2^\bullet . Below n° and n_2^\bullet is $n \setminus n_2^\bullet$. Below n_1^\bullet and n^\bullet is $n \setminus n_1^\bullet$. A larger bracket connects $(n \setminus n_2^\bullet)$ and $n \setminus n_1^\bullet$, with the result $(n \setminus n_2) / n^\bullet$ below it.
- Right: A diagram with nodes n^\bullet , n_1^\bullet , and n^\bullet . A bracket connects n^\bullet and n_1^\bullet . Below n_1^\bullet and n^\bullet is $n \setminus n^\bullet$.

The bottom row shows a more complex equation:

$$\text{prob} \left[\begin{array}{c} n^\circ \quad n_2^\bullet \quad n^\circ \quad n_1^\bullet \quad n^\bullet \\ \diagdown \quad \diagup \quad \diagdown \quad \diagup \\ n \setminus n_2^\bullet \quad n \setminus n_1^\bullet \\ \diagdown \quad \diagup \\ (n \setminus n_2) / n^\bullet \quad n \setminus n_1^\bullet \end{array} \right] = \frac{\text{freq} \left[\begin{array}{c} n^\circ \quad n_2^\bullet \quad n^\circ \quad n_1^\bullet \quad n^\bullet \\ \diagdown \quad \diagup \quad \diagdown \quad \diagup \\ n \setminus n_2^\bullet \quad n \setminus n_1^\bullet \\ \diagdown \quad \diagup \\ (n \setminus n_2) / n^\bullet \quad n \setminus n_1^\bullet \end{array} \right]}{\text{freq} \left[\begin{array}{c} n_1^\bullet \quad n^\bullet \\ \diagdown \quad \diagup \\ n \setminus n_1^\bullet \end{array} \right]} =$$

4.1. Axiom Links with context

Let us consider the categorical type $n \setminus n$ which represents a function asking for an argument type n .

This argument may compare inside a bigger type, which represents its context.

For instance, n could be part of the type $(n \setminus n) / n$.

Possible axiom links with context which involve $n \setminus n$ are

The diagram illustrates axiom links with context involving the type $n \setminus n$. It consists of several parts:

- A tree diagram at the top left with a bracket labeled 0.15 above it. The tree has root $(n \setminus n2) / n \cdot$ with children n° and $n \setminus n2^\bullet$. $n \setminus n2^\bullet$ has children $n2^\bullet$ and n° . $n \setminus n2^\bullet$ also has a child $n \setminus n1^\bullet$ which has children $n1^\circ$ and n^\bullet .
- A tree diagram at the top right with a bracket above it. The tree has root $n \setminus n1^\bullet$ with children n^\bullet and $n1^\circ$.
- A fraction of two tree diagrams. The numerator is the tree diagram from the top left, enclosed in large square brackets and labeled $prob$ to its left. The denominator is a tree diagram with root $n \setminus n1^\bullet$ and children $n1^\circ$ and n^\bullet , enclosed in large square brackets and labeled $freq$ to its left. The fraction is followed by $=$ and 0.15 .
- A curved arrow points from the 0.15 label in the top right to the 0.15 label in the fraction.

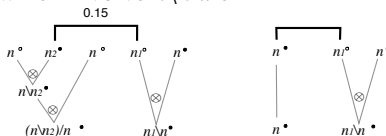
4.1. Axiom Links with context

Let us consider the categorial type $n \setminus n$ which represents a function asking for an argument type n .

This argument may compare inside a bigger type, which represents its context.

For instance, n could be part of the type $(n \setminus n) / n$.

Possible axiom links with context which involve $n \setminus n$ are



$$\text{prob} \left[\begin{array}{c} \left[\begin{array}{c} n^\circ \quad n^\circ \quad n^\circ \\ \diagdown \quad \diagup \\ n \setminus n1^\circ \end{array} \right] \\ \diagdown \quad \diagup \\ n^\circ \quad n^\circ \end{array} \right] =$$

4.1. Axiom Links with context

Let us consider the categorial type $n \setminus n$ which represents a function asking for an argument type n .

This argument may compare inside a bigger type, which represents its context.

For instance, n could be part of the type $(n \setminus n) / n$.

Possible axiom links with context which involve $n \setminus n$ are

$$\begin{array}{c}
 \begin{array}{ccc}
 \begin{array}{c}
 \text{0.15} \\
 \hline
 \begin{array}{c}
 n^\circ \quad n^\circ \quad n^\circ \\
 \diagdown \quad \diagup \\
 n \setminus n^\circ \\
 \diagdown \quad \diagup \\
 (n \setminus n) / n^\circ
 \end{array}
 \end{array}
 &
 &
 \begin{array}{c}
 \begin{array}{c}
 n^\circ \quad n^\circ \\
 \diagdown \quad \diagup \\
 n \setminus n^\circ
 \end{array}
 \end{array}
 \end{array}
 \\
 \\
 \begin{array}{c}
 \begin{array}{c}
 \text{prob} \\
 \left[\begin{array}{c}
 \begin{array}{c}
 n^\circ \quad n^\circ \quad n^\circ \\
 \diagdown \quad \diagup \\
 n \setminus n^\circ
 \end{array}
 \end{array}
 \right]
 \end{array}
 =
 \frac{
 \begin{array}{c}
 \begin{array}{c}
 \text{freq} \\
 \left[\begin{array}{c}
 \begin{array}{c}
 n^\circ \quad n^\circ \quad n^\circ \\
 \diagdown \quad \diagup \\
 n \setminus n^\circ
 \end{array}
 \end{array}
 \right]
 \end{array}
 }{
 \begin{array}{c}
 \begin{array}{c}
 \text{freq} \\
 \left[\begin{array}{c}
 \begin{array}{c}
 n^\circ \quad n^\circ \\
 \diagdown \quad \diagup \\
 n \setminus n^\circ
 \end{array}
 \end{array}
 \right]
 \end{array}
 }
 =
 \end{array}
 \end{array}
 \end{array}$$

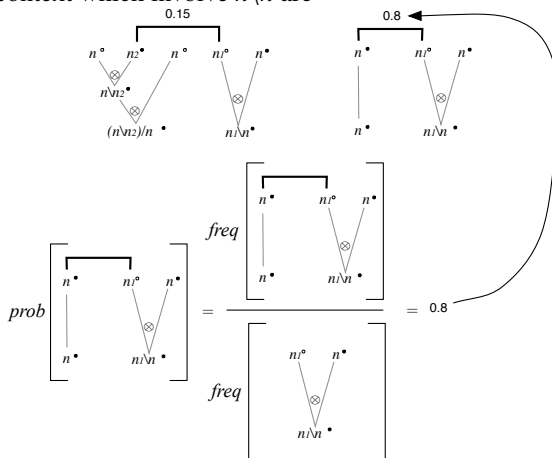
4.1. Axiom Links with context

Let us consider the categorial type $n \setminus n$ which represents a function asking for an argument type n .

This argument may compare inside a bigger type, which represents its context.

For instance, n could be part of the type $(n \setminus n) / n$.

Possible axiom links with context which involve $n \setminus n$ are



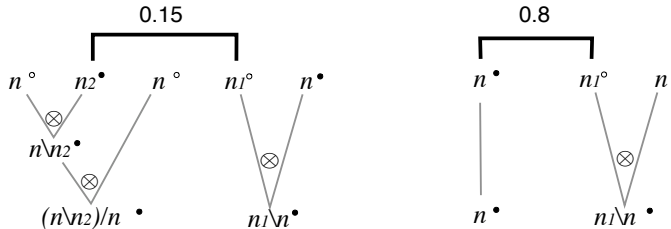
4.1. Axiom Links with context

Let us consider the categorial type $n \setminus n$ which represents a function asking for an argument type n .

This argument may compare inside a bigger type, which represents its context.

For instance, n could be part of the type $(n \setminus n) / n$.

Possible axiom links with context which involve $n \setminus n$ are



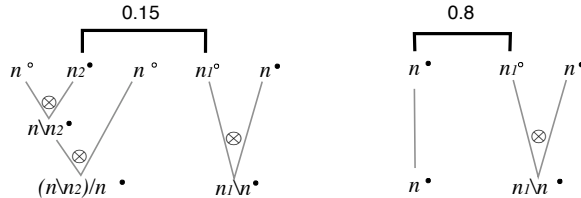
4.1. Axiom Links with context

Let us consider the categorical type $n \setminus n$ which represents a function asking for an argument type n .

This argument may compare inside a bigger type, which represents its context.

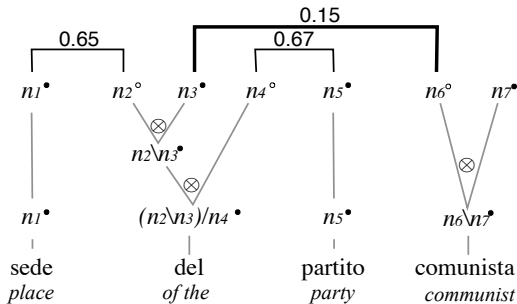
For instance, n could be part of the type $(n \setminus n) / n$.

Possible axiom links with context which involve $n \setminus n$ are

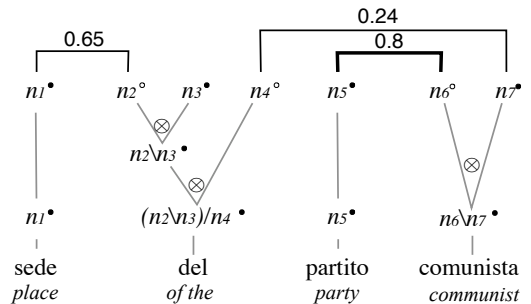


Hence, the function $n \setminus n$ takes a simple type n more probably than taking the argument n in the more complex context $(n \setminus n) / n$, where it is already modified by another function.

4.2. Example: structural ambiguity with probabilities



prob= 0.065



prob= 0.125

5. Evaluation

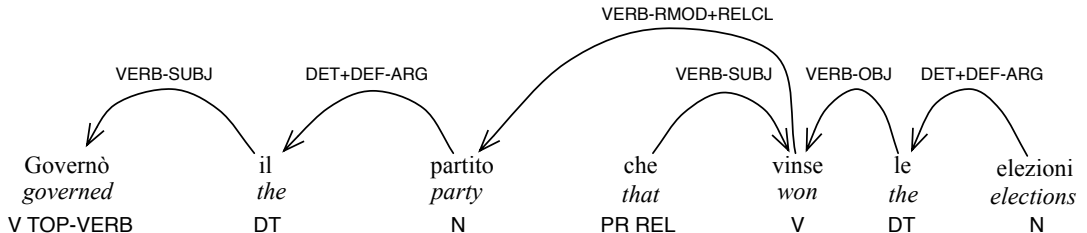
In order to run the incremental parser on a given corpus we must start from a grammar and we need to collect statistical information for each possible axiom link.

5. Evaluation

In order to run the incremental parser on a given corpus we must start from a grammar and we need to collect statistical information for each possible axiom link.

To this end we used TUT:

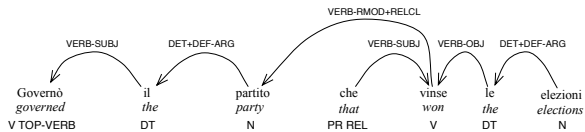
- ▶ The Turin University Treebank (TUT) [Bosco 2003] is a publicly available corpus of ca. 1800 sentences.
- ▶ The annotation format is based on the dependency paradigm centered upon the notion of predicate-argument structure.



5.1. Translating TUT Trees to CG Proof Net

In order to obtain a CG lexicon suitable for parsing:

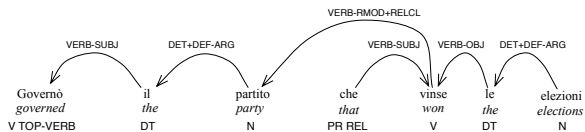
we converted TUT dependency format



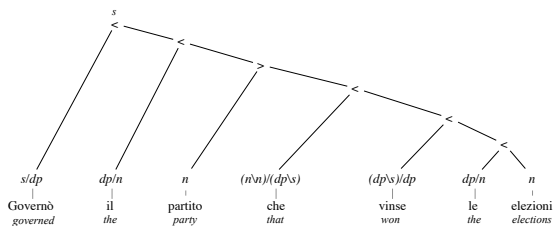
5.1. Translating TUT Trees to CG Proof Net

In order to obtain a CG lexicon suitable for parsing:

we converted TUT dependency format

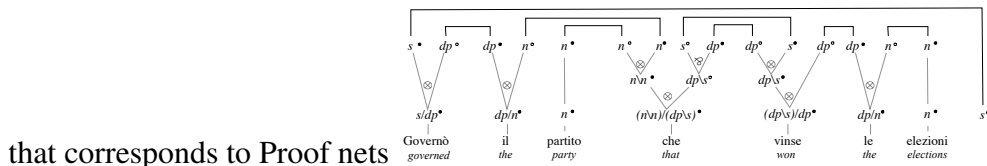
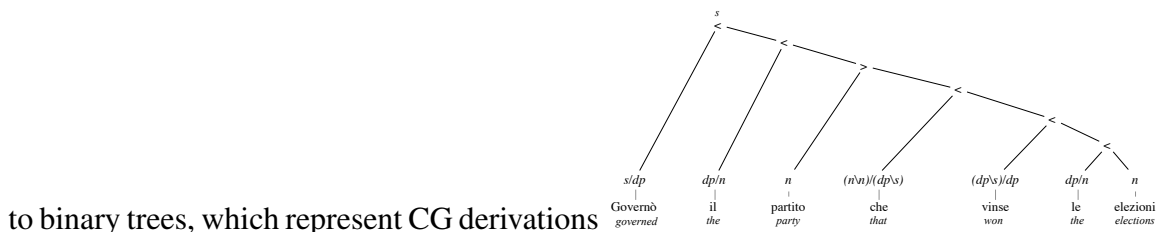
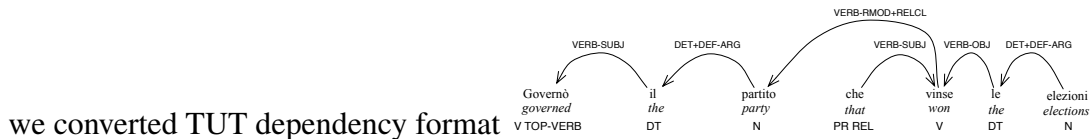


to binary trees, which represent CG derivations



5.1. Translating TUT Trees to CG Proof Net

In order to obtain a CG lexicon suitable for parsing:



5.2. Training set

Proof nets hold useful information:

- ▶ categorial types: set of word-type pairs are created and a lexicon is obtained.
- ▶ statistical information can be extracted from proof nets, obtaining weighted axiom links.

5.2. Training set

Proof nets hold useful information:

- ▶ categorial types: set of word-type pairs are created and a lexicon is obtained.
- ▶ statistical information can be extracted from proof nets, obtaining weighted axiom links.

Training and Test sets:

- ▶ so far, we have worked on a sub-corpus of TUT, consisting of 443 trees\sentences.
- ▶ we divided this set into
 - ▶ the training set, composed of 400 proof nets labeled with statistical information
 - ▶ the test set, composed of the remaining 43 sentences.
- ▶ The induced lexicon consists of 1909 words, 480 categories, with an average of two categories per word.

5.3. Evaluation

- ▶ Precision is the ratio of correctly proposed constituents over the number of all proposed constituents,
- ▶ Recall is the ratio of correctly proposed constituents over the number of constituents in the gold standard.
- ▶ Labelled precision (LP) and labelled recall (LR) count a proposed constituent as correct if the gold standard contains a constituent with the same span and label.
- ▶ Bracketed precision (BP) and bracketed recall (BR) consider a constituent as correct if the gold standard just contains a constituent with the same span.

labelled precision (LB) 0.787	labelled recall (LR) 0.782	labelled F-score 0.785
bracketed precision (BB) 0.818	bracketed recall (BR) 0.815	bracketed F-score 0.816

6. Conclusions and Future Works

- ▶ We will use the induced lexicon and the weighted axiom links to convert the remaining part of TUT,
- ▶ then both the lexicon and the database of weighted axiom links will be extended,
- ▶ and we will test the parser on a bigger amount of data.

6. Conclusions and Future Works

- ▶ We will use the induced lexicon and the weighted axiom links to convert the remaining part of TUT,
- ▶ then both the lexicon and the database of weighted axiom links will be extended,
- ▶ and we will test the parser on a bigger amount of data.
- ▶ Finally, we are studying how to add semantic representations to the parser, exploiting the syntax-semantic interface of the CG framework.