Building an Italian CG bank via incremental statistical parsing

R. Bernardi
KRDB, Free University of Bozen-Bolzano,
bernardi@inf.unibz.it

A. Bolognesi,
Dip. Scienze Matematiche ed Informatiche, Università di Siena
bolognesi2@unisi.it

Abstract

This paper describes the process to construct a categorial grammar derivations bank for Italian by means of an incremental statistical parser based on proof nets. The main research line behind this work lies on the attempt to merge the advantages of a logical approach to linguistic analysis with the ones of statistical and data-driven methods. In particular, we aim to maintain the basic idea of CG framework and overcome its limitations when turning to real life applications. We first introduce the reader to Categorial Type Logic formalism, a particular extension of CG, then we address the issue of enhancing the syntactic categories composition via statistical information along with the overall procedure of building the derivations bank. Finally, we present our first evaluation of the system.

1 Introduction

Categorial Grammar (CG) is a lexicalized formal grammar well known for its tied connection between syntax and semantics. Variants of it have been used to reach wide coverage grammar for English [8] and Dutch [13]. Its elegant syntax-semantics interface has already provided promising preliminary results that could bring to enrich the English CCGbank with semantic information [2]. A semantically annotated treebank would be a highly valuable resource for many language technology applications. We believe CG can help reaching this ambitious goal.

We propose to extend the logical framework shown in [13] following a statistical approach as in [8] to help building a bank of Italian CG derivations. The

Hajič, J. and Nivre, J. (eds.): Proceedings of the TLT 2006, pp. 223–234.

focus of the present paper is on the use of an incremental statistical parser based on proof nets to be trained on a set of CG derivations obtained by starting from the dependency treebank developed at Turin University, TUT [3].

Our work is closely related to the recent developments of the TUT project based on the use of incremental processing and of a Dynamic Version of Tree Adjoining Grammar (DVTAG) [11] to reach a wide coverage Italian grammar. Since we have started from the same training set, a comparison between their DVTAG and our CG might shed lights on similarity and differences between the two formalisms and extend the results presented in [14].

The main research line behind the work presented in this paper lies on the attempt to merge the advantages of a logical approach to linguistic analysis with the ones of statistical and data-driven methods. In particular, we aim to maintain the basic idea of CG framework which captures the essential nature of linguistic structure composition and overcome its limitations when turning to real life applications. So far our attention has been focused on the enhancement of the syntactic categories composition via statistical information, we plan to make a similar move at the semantic level too and exploit the link between syntax and semantics to improve the performance of the parser when applied to large data.

In Section 2 we introduce the version of CG used as formal grammar as well as the proof system adopted and the parser we developed. In Section 3, we briefly describe the dependency tree bank used to derive the training set of proof nets on which we train our parser. Finally, in Section 4 we present our first evaluation.

2 Categorial Type Logic

Categorial Type Logic (CTL) [12] is a family of logics tracing back to CG. As in the latter grammar, in CTL categories are either atomic formulae (e.g. n, s, np, dp for noun, sentence, noun phrase and determiner phrase), or functional formulae (e.g. $(np\backslash s)/dp$ could be assigned to an expression that requires a determiner phrase on the right and a noun phrase on the left to yield a sentence). Differently from CG, in CTL recognizing that a given string of categories, $Cat_1, \ldots Cat_n$, is of a certain category C means to prove that the former derives the latter: $Cat_1, \ldots Cat_n \vdash C$. The logics of the CTL family share logical rules, namely function application and abstraction, and differ with respect to their packages of structural rules. This distinction is reflected on the linguistic theory behind the approach: the main linguistic claim is that the core logic captures the core of linguistic structure composition, while structural rules capture cross-linguistic variations. The parser presented in this paper accounts for the core logic only. The usage of structural rules allows to avoid multiple type lexical assignments when linguistically unjustified and re-

duces the size of the lexicon. They extend the set of derivability relations among categories, allowing to use only the simpler type (the one that derives the others) in the lexicon with obvious advantages both at theoretical linguistics and computational linguistics level. In further works we will employ structural rules to reduce the number of lexical types assigned to a same word. In this paper, we are going to focus on CG functor-argument (fa) structures and on structural and dependency information carried by the syntactic categories.

An 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 functors as illustrate in Figure 1.

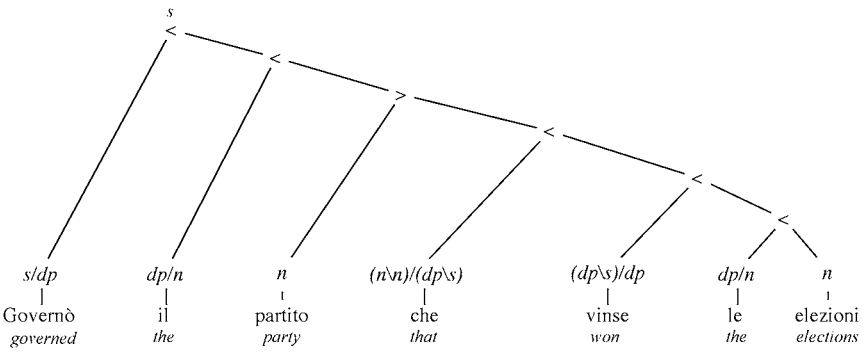

Figure 1: Example of binary tree.

For example, 'vinse' is the functor taking 'le elezioni' as argument on its right (>), whereas 'che vinse le elezioni' is the functor taking 'partito' as argument on its left (<).

Note how much structural information is carried by a category. For example, the category assigned to the relative pronoun 'che' says that: (i) it takes on its right a sentence missing a *dp* in a subject position and that (ii) the composed constituent ('che vinse le elezioni') modifies the noun occurring on the left of the relative pronoun. In this way, it percolates the dependency relation between the verb 'vinse' of the relative clause and the missing subject to the noun 'partito', subject of the main sentence.

Different bracketing of the linguistic structures corresponds to different dependencies among words. For instance, the string 'sede del partito comunista' could receive the two parse trees in Figure 2. For easy of later references to the atomic formulae we need to number them, but numbers do not express any conceptual difference.

In CTL these differences boil down to differences between the matching among

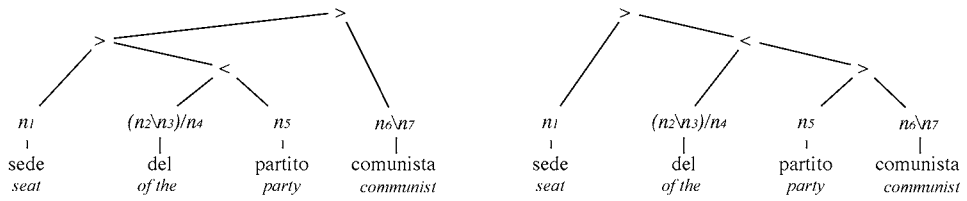

Figure 2: Structural ambiguity example.

atomic formulae. For instance the different dependency relation between 'partito' with 'del' and with 'comunista' in the left and right trees, respectively, is captured by the different matching holding for the atomic formula n_5: it matches n_4 in the left tree and n_6 in the right tree. This intuitive idea has been formalized in the proof system known as proof nets used by our parser and briefly introduced below.

Polarity position The main observation to highlight is the well known notion of polarity position of the implication ($\rightarrow$), namely in $A \rightarrow B$, A is in a negative position (it is what the function is looking for to yield B) and B is in a positive position (it is the value that will be obtained if the A is provided). We mark negative and positive positions by means of $\bullet$ and $\circ$, respectively, $A^\bullet \rightarrow B^\circ$. The counting of polarity satisfies the basic equivalences: $++ = +, -+ = -, -- = +$, hence in a higher order function like $(A^\bullet \rightarrow B^\circ)^\bullet \rightarrow C^\circ$, A is in a positive position, since it is under $\bullet$ twice.

Within the CTL framework, this notion appears both at the level of categories and at the level of derivation due to the connection between $\backslash, /$ and $\vdash$ with $\rightarrow$:

1. functional categories: $B^\circ/A^\bullet$ and $A^\bullet\backslash B^\circ$
2. derivation: $Cat_1^\bullet, \ldots, Cat_n^\bullet \vdash C^\circ$

   notice, how the counting of polarity is applied at the functional category level, for instance in the case of the higher order category assigned to 'che' is $(n^\bullet\backslash n^\circ)/(dp^\circ\backslash s^\bullet)$.

Function application corresponds to the matching of categories of opposite polarity. For instance in the trees given in Figure 2 we have the following situation:

left tree		right tree	
$-(\bullet)$	$+(\circ)$	$-(\bullet)$	$+(\circ)$
n_1	n_2	n_1	n_2
n_3	n_6	n_5	n_6
n_5	n_4	n_7	n_4

This information among the atomic formulae captures the relations represented by the different links in the corresponding dependency trees below.

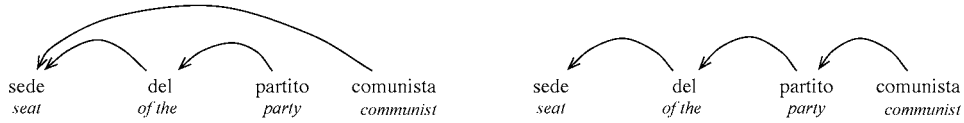

Figure 3: Dependency trees.

2.1 Proof Nets

The main concepts behind the use of proof nets are (i) the order of occurrences and polarity of the leaves (i.e. the atomic sub-formulae of the structure to be parsed) and (ii) the shape of their links. The former is determined by recursively applying the unfolding rules in Figure 4, which formally express the idea of polarity introduced intuitively above (see [19, 14] for further details)[1]

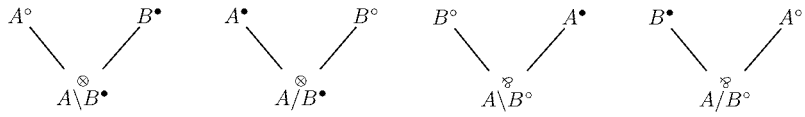

Figure 4: Unfolding rules for building proof nets.

The proof net of the sentence considered above is given in Figure 5.

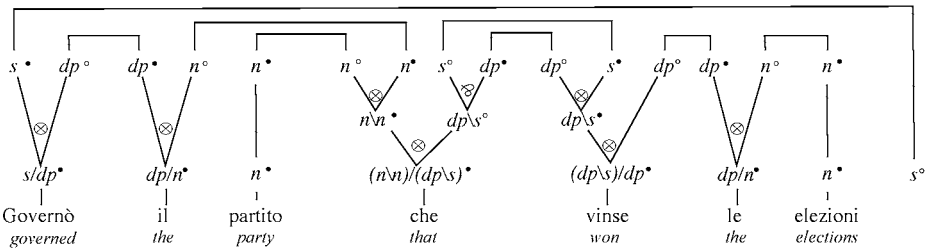

Figure 5: Example of proof net.

An unfolded structure could be completed by means of different links among the leaves. Correctness criteria have been defined to rule out those proof struc-

[1]The unfolding rules simply capture the logical equivalences: $A \rightarrow B = \neg A \vee B$, $\neg(A \vee B) = \neg A \wedge \neg B$ and $\neg\neg A = A$ Think of $\otimes$ and $\parr$ as $\wedge$ and $\vee$ respectively and $\bullet$ as $\neg$, we have that e.g. $(s/dp)^\bullet = (s^\circ \parr dp^\bullet)^\bullet = s^\bullet \otimes dp^\circ$.

tures that do not correspond to logical proofs, hence are not proof nets. We take as formally acceptable only those proof nets that are planar DR-acyclic proof structure [5], i.e. proof nets with no crossing axiom links and with specific paths linking the formulae in the trees.

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. For instance, the two formally correct proof structures of the example considered above 'sede del partito comunista' are given in Figure 6.

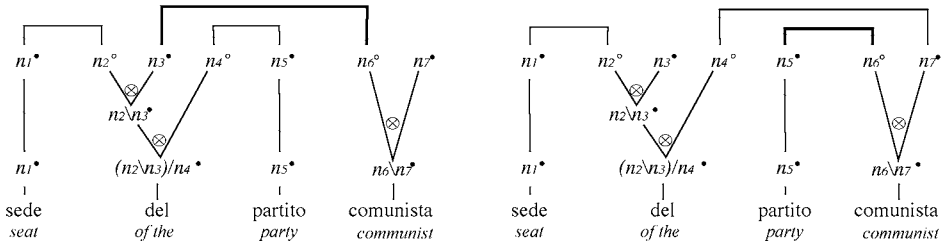

Figure 6: Proof structure for the sentence 'sede del partito comunista'.

In the following we will show how our incremental statistical parser checks the correctness criteria, links the axioms and rank the probability of the proof nets.

2.2 Incremental Parsing

Incremental processing is relevant for language modeling [11] and dynamic formal semantic representation [20]. By parsing the input words from left to right, the language processor may carry out a semantic interpretation of the partial structures. Incremental parsing for CTL family is a challenging approach to natural language processing and has been well discussed in the literature [16, 17, 18].

This section explains our statistical CTL parser based upon the hypothesis of incrementality: words are processed in a left-to-right fashion, and all the correct proof structures are kept at each step, returning a set of proof nets when the sentence processing is ended [17, 5]. After explaining our implementation of incrementality, we will switch to analyze more in depth how we improve parser capabilities with statistical information. Statistic information allows the parser to choose solutions with high likelihood, speeding the parsing operation by pruning bad syntactic representation during the left-to-right processing. Many researchers have

investigated how parsing for CG benefits from statistic information. Our work has been inspired mostly by the results obtained in [9, 15].

In order to build linkages that allows to incrementally check the proof nets integrity we adopted the switch graph method proposed in [5] that however does not handle the planarity criterion. The latter requires axiomatic linkages to be drawn as a planar graph above the sequence of axiomatic formulae. The planar connection of axiomatic polar types of a proof structure is not difficult to implement, for example in [16] a CKY-style algorithm is proposed.

Because an axiomatic linkage corresponds to the bracketing of a string according to a context-free grammar [18], in order to tabulate axiomatic links we used a Earley-style parser [7] over a fixed grammar, here expressed in Backus-Naur form:

$S ::= A\ S$	$S ::= A$	
$A ::= a^{\circ}\ S\ a^{\bullet}$	$A ::= a^{\bullet}\ S\ a^{\circ}$	for every atomic category a
$A ::= a^{\circ}\ a^{\bullet}$	$A ::= a^{\bullet}\ a^{\circ}$	for every atomic category a

S corresponds to those subsequences of axiomatic formulae over which a sub-linkage (or sub proof structure) exists, while A corresponds to those subsequences over which a sub-linkage bracketed by a single axiom link exists. For instance, S may correspond to the subsequence $dp^{\bullet}\ dp^{\circ}\ n^{\bullet}\ n^{\circ}\ n^{\bullet}\ n^{\circ}$ of Figure 5, while A may correspond to the subsequence $n^{\bullet}\ n^{\circ}\ n^{\bullet}\ n^{\circ}$. Any proof structure or sub-linkage constructed with this grammar will satisfy the planarity property of proof nets.

For the running example started in Figure 5 we obtain the following grammar:

1) $S ::= A\ S$	2) $S ::= A$	3) $A ::= dp^{\circ}\ dp^{\bullet}$	4) $A ::= dp^{\circ}\ S\ dp^{\bullet}$
5) $A ::= dp^{\bullet}\ dp^{\circ}$	6) $A ::= dp^{\bullet}\ S\ dp^{\circ}$	7) $A ::= s^{\circ}\ s^{\bullet}$	8) $A ::= s^{\circ}\ S\ s^{\bullet}$
9) $A ::= s^{\bullet}\ s^{\circ}$	10) $A ::= s^{\bullet}\ S\ s^{\circ}$	11) $A ::= n^{\circ}\ n^{\bullet}$	12) $A ::= n^{\circ}\ S\ n^{\bullet}$
13) $A ::= n^{\bullet}\ n^{\circ}$	14) $A ::= n^{\bullet}\ S\ n^{\circ}$		

These rules can be obtained out of the sequence of axiomatic formulae by checking for each literals all the literals on its right, requiring a quadratic time of complexity.

Given a CG lexicon, the parser discussed so far is able to retrieve all the possible proof nets for an input sentence, i.e. all the possible derivations for an input sentence within the CTL logical framework. 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 extracted from a given training Treebank. 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.

Given two unfolded structures S_1 and S_2 whose roots and leaves are the categorial types T_1 and T_2 and the atomic formulae $a_1^1 \ldots a_1^n$ and $a_2^1 \ldots a_2^k$, respectively.

Let them be connected by an axiom link between a_1^i and a_2^j with polarity p_i and p_j, respectively, we call *extended axiom link* the tuple $\alpha = \langle T_1, p_1, i, T_2, p_2, j \rangle$.
Let D be the set of all the extended axiom links, we define the function $prob$: $D \rightarrow [0,1]$ as following

$$prob(\alpha) = \begin{cases} freq(\langle T_1, p_1, i, T_2, p_2, j \rangle) \,/\, freq(\langle T_1, p_1, i \rangle) & \text{if } p_1 = \circ \\ freq(\langle T_1, p_1, i, T_2, p_2, j \rangle) \,/\, freq(\langle T_2, p_2, j \rangle) & \text{if } p_2 = \circ \end{cases}$$

where $freq$ is a function which returns the frequency of its input, that is the number of times its input has appeared. Therefore, for each extended axiom link α, its probability is calculated as the ratio between the frequency of α and the overall frequency of the root categorial type containing the positive atomic formula of the axiom link.

Finally, we define the probability of a given proof net P ($prob(P)$) as the product of all extended axiom links of P, that is $prob(P) = \Pi_{\forall \alpha \in P} prob(\alpha)$.

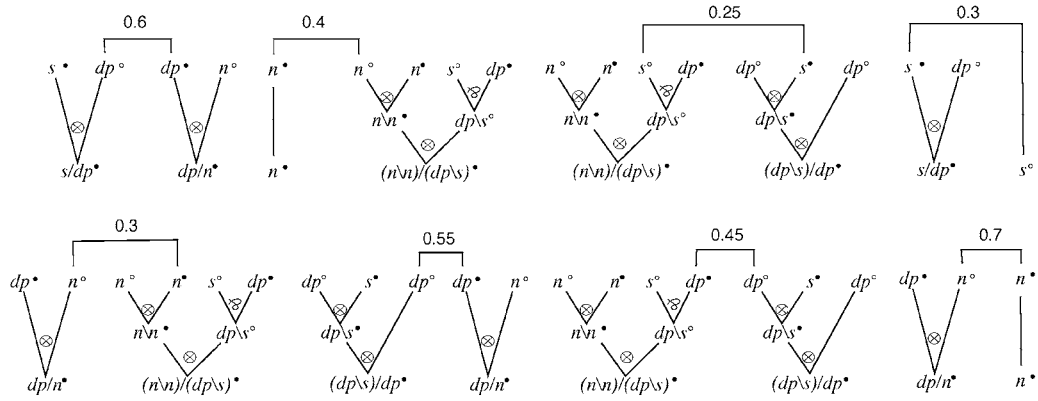

Figure 7: Example of probability weight assignment for axiom links

We used TUT as training Treebank to collect the statistical information for each possible extended axiom link. Figure 7 shows an example of probability weight assignments for the extended axiom links of the example seen so far. Given these values, the probability of the proof net of Figure 5 is equal to $0.9355x10^{-3}$.

3 Training set: Converted Turin University Treebank

The Turin University Treebank (TUT) [3] is a publicly available corpus of ca. 1800 sentences[2]. The annotation format is based on the dependency paradigm centered upon the notion of predicate-argument structure. The relations of TUT trees are annotated by following a model called Augmented Relational Structure (ARS) which allows for a clear distinction of various components in each relation. Each relation is therefore implemented as a feature structure that can include values for a morpho-syntactic, a functional-syntactic and a semantic component.

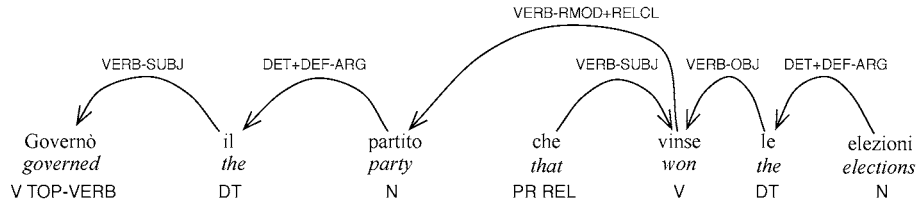

Figure 8: Example of TUT dependency annotation.

Figure 8 shows an example of TUT dependency annotation for the sentence 'Governò il partito che vinse le elezioni'[3]. Each node in the dependency structure contains a terminal word and the PoS tag of the word. Each label on the edges represents a head-dependent relation, following the ARS model. For example, the relation DET+DEF-ARG, that links the dependent noun 'partito' with the head determiner 'il', contains the syntactic information that the noun is argument of the determiner.

3.1 Translating TUT Trees to CTL Proof Net

In order to obtain a CG lexicon suitable for parsing within CTL formalism, we converted TUT dependency format to binary constituency format. Thus, we translated TUT trees into binary trees which represent CG derivations through application rules, and therefore correspond uniquely to CTL proof nets. This operation required a preprocessing of the starting Dependency Structure, however we won't go into the details of this process since it is out of the scope of this paper.

[2] http://www.di.unito.it/~tutreeb/index.html

[3] This sentence does not belong to TUT, it is the adaptation of the example seen so far to the TUT representation format. Moreover, the image presented here graphically differs from the graphic tree representation done by TUT's authors, but it is easier to compare with fa-structures and proof nets and still equivalent to the original ones.

The next step in building a CTL lexicon consists in extracting categorial types by means of type inference algorithms which create a set of word-type pairs. [4] proposes a categorial type induction algorithm that takes as input functor-argument structures. The algorithm follows two steps: type assignment and type unification. The type assignment operation works top-down starting from the type of the root node of an fa-structure and applying two simple rules based on the direction of the function-argument relation, either < or >. These rules are depicted in Figure 9. The type unification reduces the set of type assignments to a word by identifying type assignments that can be unified (see [1].)

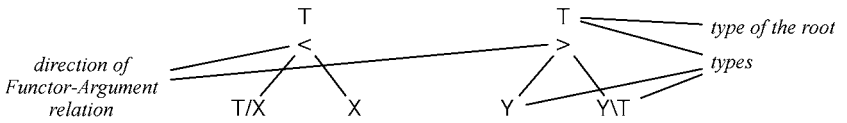

Figure 9: Type assignment

We instantiate atomic categories using the grammatical relations and the PoS information given in TUT. By running the unification algorithm we induce a lexicon containing all the types obtained per each word.

4 Evaluation

There exists a one to one correspondence between binary tree obtained from TUT and CTL proof nets. For example, the binary tree depicted in Figure 1 corresponds to the proof net in Figure 5. By translating trees from TUT treebank into CTL derivations we are already building a first CTL derivations bank. Therefore statistical information can be extracted in order to obtain a database of weighted axiom links, see by means of example Figure 7.

In order to run a first experiment, we have chosen to start from a subset of TUT that contains dependency structures with a low level of structural complexity. To this end, we have adopted the structural complexity definition proposed in [10]: the structural complexity of a dependency structure is the total length of the dependency links in the structure, where the length of a dependency link is one plus the number of words between the head and the dependent. This made possible to execute a first grammar induction and collect a first database of axiom links probability.

From the 1800 trees of TUT we extracted 443 trees with structural complexity less than 70, obtaining our initial gold standard. Then we translate these trees into a CTL derivations bank as explained in Section 3.1.

So far, we have extracted statistical information only for the first 400 trees, leading to the creation of the training set of trees. The remaining 43 trees formed the test set. The induced lexicon consists of 1909 words, 480 categories, with an average of two categories per word.

We have run a first experiment to evaluate the parser performance with respect to the constituents bracketing along with the indication of functor positions, expressed in fa-structures by means of < and > labels. The experiment consists of two steps: first we tested precision and recall for the bracketing performances only, and then for the bracketing and functor labels. The incremental parsing with statistical information leaded to precision and recall results shown in Table 1.

bracketed precision (BP)	bracketed recall (BR)
0.818	0.815
labelled precision (LB)	labelled recall (LR)
0.787	0.782

Table 1: Evaluation results

In the majority of the cases, mistakes were due to a high use of adjectives and adverbs which pushed the parser to choose bad sub-bracketings. Most of these mistakes will be overcome by extending the training treebank, others will require semantic information to be taken into account.

5 Future Works

We will use the induced lexicon and the weighted axiom links to convert the remaining part of TUT, extend both the lexicon and the database of weighted axiom links, and test the parser. Furthermore, we will work on the evaluation of parsing long distance dependency relations along with the work carried out in [6]. There, it is shown how to capture long distance dependencies occurring in English, similarly we plan to deepen our study of these structures in Italian exploiting the system presented in this paper. Finally, we are studying how to add dynamic semantic composition to improve the parser performance.

References

[1] R. Bernardi, A. Bolognesi, F. Tamburini, and M. Moortgat. Categorial type logic meets dependency grammar to annotate an italian corpus. In *Recent Advances in Dependency Grammars Workshop*, pages 57–64, Geneva, 2004. COLING 2004.

[2] J. Bos. Towards Wide-Coverage Semantic Interpretation. In *Proceedings of Sixth International Workshop on Computational Semantics IWCS-6. Pages 42-53.*, 2005.

[3] C. Bosco. *A grammatical relation system for treebank annotation.* PhD thesis, Computer Science Department, Turin University, 2003.

[4] W. Buszkowski and G. Penn. Categorial grammars determined from linguistic data by unification. *Studia Logica 49, pp. 431-454.*, 1990.

[5] B. Carpenter and G. Morrill. Switch Graphs for Parsing Type Logical Grammars. In *Proceedings of the International Workshop on Parsing Technology, IWPT05, Vancouver*, 2005.

[6] S. Clark, J. Hockenmaier, and M. Steedman. Building Deep Dependency Structures with a Wide-Coverage CCG Parser. In *40th Annual Meeting on Association for Computational Linguistics*, 2001.

[7] J. Earley. An Efficient Context-Free Parsing Algorithm. In *Communications of the ACM*, 1969.

[8] J. Hockenmaier. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar.* PhD thesis, School of Informatics, University of Edinburgh., 2003.

[9] J. Hockenmaier and M. Steedman. Generative Models for Statistical Parsing with Combinatory Categorial Grammar proceedings of the 40th annual meeting of the association for computational linguistics, 2002.

10] D. Lin. On the structural complexity of natural language sentences. In *Proceedings of the 16th conference on Computational linguistics*, pages 729–733, Morristown, NJ, USA, 1996. Association for Computational Linguistics.

11] A. Mazzei and V. Lombardo. Building a Wide Coverage Dynamic Grammar. *AI*IA 2005: 303-314*, 2005.

12] M. Moortgat. Categorial Type Logics. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 93–177. Elsevier, Amsterdam, 1996.

13] M. Moortgat and R. Moot. Using the Spoken Dutch Corpus for type-logical grammar induction. In *Third International Language Resources and Evaluation Conference*, 2002.

14] R. Moot. *Proof Nets for Linguistic Analysis.* PhD thesis, UiL OTS, Utrecht University, 2002.

15] R. Moot. Parsing corpus-induced type-logical grammars. In R. Bernardi and M. Moortgat, editors, *Workshop on Linguistic Corpora and Logic Based Grammar Formalisms*, 2003.

16] G. Morrill. Memoisation of Categorial Proof Nets: Parallelism in Categorial Processing. Technical report, Dept. de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 1996.

17] G. Morrill. Incremental processing and acceptability. In *Computational Linguistics*, volume 26, pages 319–338, 2000.

18] G. Penn. A Graph-Theoretic Approach to Sequent Derivability in the Lambek Calculus. In *Electronic Notes in Theoretical Computer Science 53*, 2002.

19] D. Roorda. *Resource Logics: Proof-theoretical Investigations.* PhD thesis, Universiteit van Amsterdam, 1991.

20] R. Schwitter and M. Tilbrook. Dynamic Semantics at Work. In *K. Hasida and K. Nitta (eds.), "New Frontiers in Artificial Intelligence: Joint Proceedings of the 17th and 18th Annual Conferences of the Japanese Society for Artificial Intelligence.", LNCS, Springer.*, 2006.