Building a large grammar for Italian

Alessandro Mazzei, Vincenzo Lombardo

Dipartimento di Informatica, Università di Torino
c.Svizzera 185, 10149 Torino, Italy
{mazzei, vincenzo}@di.unito.it

Abstract

We describe the construction of a large lexicalized tree adjoining grammar for Italian, automatically extracted from an annotated corpus. We first introduce the TUT, a dependency style treebank for Italian, then we illustrate the algorithm that we have designed to extract the grammar, and finally we report two experiments about parsing complexity and coverage of the extracted grammar.

1. Introduction

Building a wide coverage grammar plays a key role in the realization of a language understanding system. The traditional methods to develop a wide grammar need a great deal of human-effort (Black et al., 1993), but in the last years with the advent of annotated corpora, the most immediate way to build wide-coverage grammars is to extract them from treebanks. To extract a grammar from a treebank two factors have a primary importance: the type of annotation used in the treebank and the type of the grammatical formalism.

The Turin University Treebank (TUT) is an ongoing project of the University of Turin on the construction of a dependency style treebank for Italian (Bosco et al., 2000): each sentence is semi-automatically annotated with dependency relations that form a tree, and relations are of morphological, syntactic and semantic types. The corpus is very varied, and contains texts from newspapers, magazines, novels and press news. Its current size is 1500 annotated sentences (33.868 words), although in this work we report data on 1200 sentences. In figure 1 there is the annotation for the sentence belonging to the corpus "*La norma non ha mai trovato applicazione*". Each node in the tree contains a terminal word, a number that refers to the position in the linear order of the sentence, and the POS tag of the word. Each label on the edges of the tree represents a head-dependent relation For instance the relation ADVB-RMOD-NEG, that links the dependent adverb "*non*" with the head verb "*trovato*", contains the syntactic information that the adverb is a modifier of the verb.

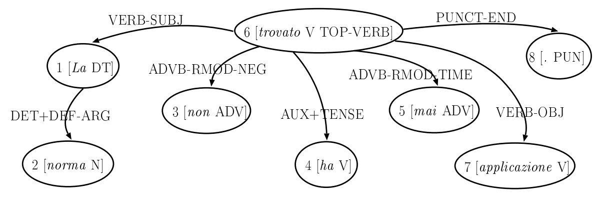

Figure 1: Dependencies tree with basic TUT annotation for the sentence "*La norma non ha mai trovato applicazione*".

Several grammatical formalisms have been proposed with the aim to capture the linguistic information present in the treebanks. Lexicalized Tree Adjoining Grammar (LTAG) is a well known grammar formalism that has interesting mathematical and linguistic properties (Joshi and Schabes, 1997) and has been applied in several applicative tasks. LTAG grammar consists of elementary trees (instead of rules) that are combined through substitution and adjunction to form syntactic trees. Elementary trees can be initial (argumental) trees or auxiliary (modifier) trees. LTAG is a lexicalized formalism because for each elementary tree there is terminal word on the frontier called *anchor*. The anchor of the tree defines the semantic content of the elementary tree: the elementary tree can be seen as an *extended projection* of the anchor. A number of wide coverage LTAGs have been developed for a number of languages (English (Doran et al., 2000), French (Abeillé and Candito, 2000), German (Neumann, 2003)).

We present an algorithm to convert the dependencies trees belonging to the TUT to constituency trees, and then an algorithm to extract from these constituency trees a lexicalized tree adjoining grammar. To our knowledge this is the first attempt to construct a wide coverage LTAG for Italian.

2. From dependencies to constituents

In order to extract the LTAG grammar, we converted the TUT treebank dependency format to a constituency format, and then we adapted the algorithm in (Xia, 2001). This algorithm was originally designed to build a constituency tree close to the trees of Penn treebank (Marcus et al., 1993).

Given a level of the dependency tree with a Head and several Dependents, the conversion to constituency relies on three mappings: the projection chain of the terminal category corresponding to the Head, i.e. the chain of non terminal nodes projected by that terminal; the projection chains of the terminal categories corresponding to each Dependent; the attachment of each Dependent projection chain to the Head projection chain. These mappings depend on the POS tags of the nodes in the dependency tree, the argument or modifier role labelled on the edge, the relative position of head and dependent (i.e. whether the dependent word is on the left or the right of the head, respectively).

The algorithm that converts the dependency annotation in the constituency annotation features two stages. In the first stage it builds a binary constituency tree with unlabelled non terminals. Starting from the root, it makes a pre-order visit of the dependency tree. For each node the algorithm creates a new projection chain made of unlabelled non terminals (the base pre-terminal node of the chain is labelled with the POS tag of the dependency node). For each edge in the dependency tree, the label of the edge determines the attachment point of the top of the dependent-related projection chain to some node of the head-related projection chain. In this stage, differently from (Xia, 2001), we build binary constituency trees, with each right branching nodes connecting the several arguments of a head. We did this choose because in Italian there is a really free word order position of the modifier respect to the position of the arguments. Using binary trees we allow a displacement of arguments and modifiers on several heights of the tree. In other words we insert non-necessary constituency nodes to separate the daughters of the head in the dependency tree on different levels of the constituency tree. Using this strategy it is simpler to extract the adjoining trees anchored by the modifier, since the foot and root nodes of the auxiliary tree is yet present in the constituency tree. In the cases of verbs, nouns, or adjectives and adverbs connected with particular relations to the head, we decided to increase by one the number of projection nodes for some heads. For instance in the tree of figure 2 the V node has several intermediate VP projections nodes. In this stage we annotated in the constituency nodes the relations that were present in the correspondent dependency nodes (or a new relation for the right-branching node): this information will be used in the extraction of the LTAG. The second stage of the conversion algorithm labels the nodes of the constituency tree proceedings bottom-up. The frontier nodes, that correspond to terminal words, are assigned the corresponding POS tag of the corresponding terminal. Each node is assigned a label on the basis of six heuristic rules that take in account the label of its daughters. The most relevant rule concerns the label of the head daughter. If a head daughter is labelled VP, and a sibling of the head node is connected to the parent with a *subject* relation, then the algorithm labels the parent node S.

In figure 2 there is the constituency tree obtained applying the conversion algorithm to the dependency tree of figure 1. The tree is quite different by the trees in the Penn treebank. The major difference is that the Penn treebank has a *flat* form for trees, instead our constituency trees are rather *vertical*, because we use only binary trees. Other differences regard some nodes that are present in our constituency trees only to allow the extraction of the correct elementary LTAG trees. This is the case of the node $N1$, inserted in the case of noun phrases with determiner: it is necessary to permit the creation of an initial tree anchored by the noun. Finally our constituency trees carry on the edges all the grammatical relations present in the original dependency trees.

3. Extracting a lexicalized tree adjoining grammar

The constituency trees built from the TUT dependency trees, include some information that normally is not present in a "native" constituency tree. The most important difference is that we do not need a percolation table to discover

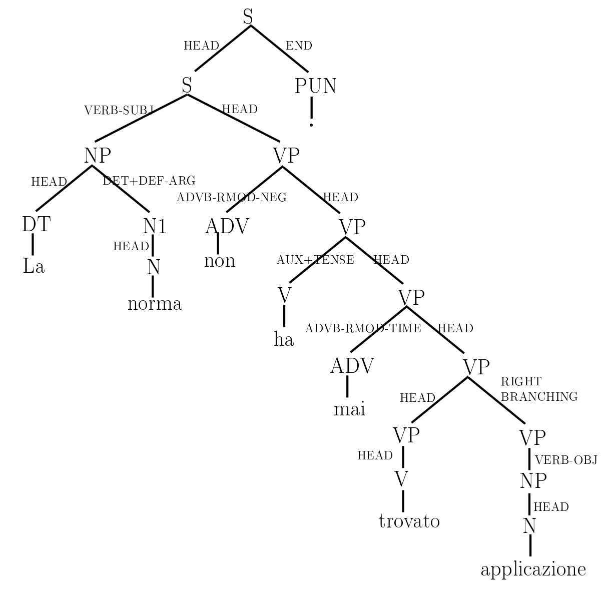

Figure 2: Constituency tree returned as output of the conversion algorithm, starting from the dependency tree of figure 1.

the head daughter of a constituent (Magerman, 1995). In fact, this information is a basic issue of a dependency annotation, and it comes for free from the TUT corpus. Similarly, we do not need an argument table to distinguish argument daughters from modifier daughters. Also this information is annotated in the TUT corpus, and it is still present in the constituency tree after the conversion. Then, in the extraction procedure we can use these relations to recognize if a daughter node is the root of an argument subtree, the root of a modifier subtree, or the head daughter. In other words we do not need to recover the derivation tree from the derived tree as in (Xia, 2001), because the dependency tree that we originally used yet contains the derivation tree.

We use a recursive "cut" procedure to extract the elementary trees that can generate that sentence. First we identify the elementary tree anchored by the head daughter of the root node. Then, we call the procedure on the nodes that are the maximal projections of the heads of the arguments, obtaining other initial trees. Finally, we call the procedure on the nodes that are the maximal projection of the heads of the modifier, obtaining auxiliary trees. In figure 3 there are some elementary trees, extracted by the constituency tree of figure 2. We use some *templates* to define the skeleton of the extracted elementary trees. In particular, we use templates to predict auxiliary elementary trees, and a template for initial trees. One of the major difference in our approach with respect to (Xia, 2001) is that we do not use a specific template to build elementary trees anchored by conjunctions, because we trait the conjunction as modifier of the first coordinated word. This approach follows up from the annotation used in TUT for coordination. In fact the two (or more) coordinated words are not dependents of the conjunction word, but the first coordinated word is the head of the conjunction, that is in turn the head of the second coordinated word. The relation from the first conjunct to the conjunction is treated as a modifier relation. Then, in the extracted grammar we have an auxiliary tree anchored by the conjunction word, that includes a substitution node on the right of the anchor for hosting the second element of the coordination.

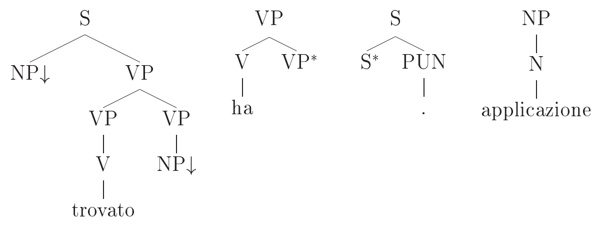

Figure 3: Elementary trees extracted by the constituency tree of figure 2.

As noted in (Xia, 2001) there are three crucial points that we have to consider to compare algorithms for automatic extraction of LTAGs. The first point concerns the treatment of the *empty categories*: if the empty node is an argument, we collapse the tree anchored by the empty node with the elementary tree anchored by his head. A special case is when the empty category is the head-verb of an elementary tree: we encountered this case only for the verb "to be"[1]. In this case we decided to build a bigger elementary tree attaching the elementary headed by the empty verb, with the elementary tree headed by the "nominal predicate". The second point regards the treatment of he *coindexing nodes* (ex. filler-gap nodes). In this case we decided to collapse the substructures which the two nodes belong to, in only one structure. Then we treat this case very similar to the case of sentential form, where several anchors have to be in the same elementary tree. In (Xia, 2001) they pursued a different strategy: they let the coindexing nodes in two different elementary trees, but a feature structure guarantees the simultaneous presence of the two elementary trees in a derivation. The third point of the algorithm concerns the treatment of *punctuation*. Generally we treat the punctuation symbols as modifiers, as they are originally annotated in the TUT. We use a special rule to extract the elementary tree of the last punctuation that close the sentence[2].

4. Testing parser performances

Using the protocol defined in (Sarkar et al., 2000), we have extracted a LTAG from all the sentences belonging to the TUT up to length 21, and we have used this grammar to evaluate the parsing complexity for LTAG. We were able to replicate their experimental results with a very smaller grammar and on different language. In the TUT there are 460 sentences with length up to 21: we have extracted from these sentences a LTAG with 2333 elementary trees, corresponding to 429 tree templates. In the experiment described in (Sarkar et al., 2000), they extracted a very large grammar from the sections 02-21 of the Wall Street Journal Penn Treebank II corpus (Marcus et al., 1993), with 123039 elementary trees, corresponding to 6789 tree templates. To compute the parsing time we used, as in (Sarkar et al., 2000), the LEM parser, a head-corner parser for LTAG developed in the XTAG project (Doran et al., 2000). The LEM parser achieves the theoretical lowest time complexity with respect to the length of the sentence, computing the parses in $O(n^6)$ complexity. Since we do not use any probabilistic model of the language, the parsing time is the amount of time used by the parser to compute all the possible parses of a sentence.

The figure 4 reports the relation between the parsing time and the sentence length. The figure 5 reports the relation between the parsing time and the number of elementary trees anchored by sentence words. Comparing the two figures one can argue that the lexical ambiguity of the sentence, i.e. the total number of the grammar trees anchored by sentence words, plays a basic role in the computation of the parsing time. One can see a major *regularity* in the figure 5. In particular, lexical ambiguity overcomes the sentence length in predicting the parsing time of the sentence, and this holds for an Italian LTAG, a smaller LTAG with respect to the grammar used in (Sarkar et al., 2000) (429 vs. 6789 tree templates).

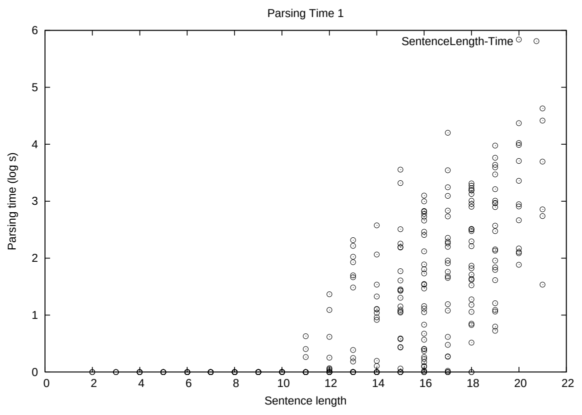

Figure 4: Parsing time respect to sentence length.

5. Coverage measures

A second type of tests concern the coverage of the extracted LTAG. We split the TUT in two sets, then we extract an LTAG from the first set (learning set), and compute the coverage of the extracted grammar on the second set (test set). We say that the LTAG grammar *covers* a sentence belonging to the TUT if the grammar derives the sentence, and the tree produced by the grammar in the derivation is equal to the tree produced converting the dependency tree to constituency tree (golden tree). This type of *structural consistency* (Carroll et al., 1998) is chosen to focus our study about the "syntactic information" contained directly in the grammar extracted, and indirectly in the treebank. The use of a measure depending on coverage and not depending on parsing, allows to abstract the results from the specific model of the language used by the parser.

[1]An example of sentence in the corpus for this phenomena is *Valona in mano ai dimostarnti.*

[2]We are not able to take in account the special coindexing of same punctuation symbols, as open and close parenthesis, cf. (Xia, 2001).

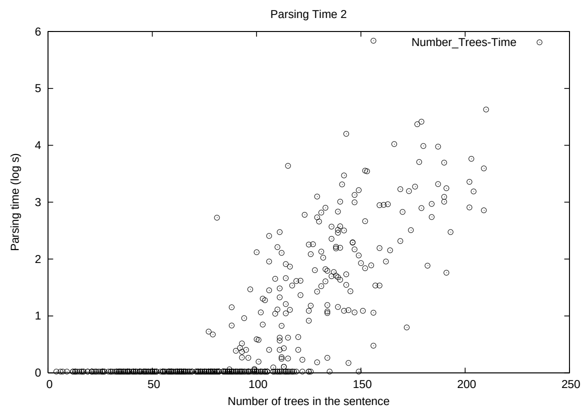

Figure 5: Parsing time respect to number of elementary trees anchored by words of the sentence.

In one experiment the TUT is split in a learning set containing 1069 sentences with length beyond 10, and a test set containing 131 sentences with length below 10. We computed that only 34 out of 131 sentences of the test set are covered by the LTAG extracted by the learning set. We repeated this experiment using the same learning and test sets, but replacing the anchors of the trees and the terminal words in the test sentences with the their respective POS tag: in this case the grammar covered 126 out of 131 of the test sentences. This result proves that the extracted LTAG has the right "template" trees to cover the sentences in the test set, but these templates are not connected to the lexical items appearing in the test sentences. This experiment shows that if we want to really use the extracted grammar to parse unseen sentences, we need a "smoothing" technique to assign the right template to the words of the sentences. In other words we have the syntactic structures, expressed as elementary trees in the treebank, but we are not able to connect an unseen word to the right syntactic structures (cf. (Srinivas and Joshi, 1999)).

Starting with this result we have performed several experiments to test the coverage of the "unanchored" syntactic structure. In the whole treebank we replaced the terminal words with their respective POS tag, and we split the treebank in several learning and test sets, each time changing the relative sizes of the sets. The table 1 reports the results of these experiments[3]. The coverage of the grammar decreases very softly with respect to the decrease of the size of the learning set: when half of the corpus is in the test set we obtain that 86% of the sentence are derived by the grammar. These results reveal that many elementary trees are repeated in the treebank, and that much syntactic information is repeated several times throughout the trees of the treebank.

Learning Set	Test Set	Coverage
95%	5%	92%
90%	10%	92%
80%	20%	90%
70%	30%	89%
60%	40%	88%
50%	50%	86%

Table 1: Coverage results with several ratios of learning/test sets

6. Conclusion

We described two algorithms to convert the TUT dependency trees to constituency tree and to extract a LTAG by these trees. We reported the results of two experiments on the extracted grammar. The results confirm some complexity measures about parsing of LTAG, and report some data about the redundancy of syntactic information present in the treebanks. In future work we intend to investigate on the relation between the coverage of the extracted LTAG, and the domain of the sentences in the treebank.

[3]In these trials we used a version of the TUT with 1235 sentences. We obtained the values averaging on 5 independent random runs.

7. References

Abeillé, A. and M. Candito, 2000. Ftag: a lexicalized tree adjoining grammar for french. In A. Abeillé and O. Rambow (eds.), *Tree Adjoining Grammars*. Chicago Press, pages 305–330.

Black, E., R. Garside, and G. Leech, 1993. *Statistically-driven computer grammars of English: The IBM/Lancaster approach*. Rodopi, Amsterdam, The Netherlands.

Bosco, C., V. Lombardo D. Vassallo, and L. Lesmo, 2000. Building a treebank for italian: a data-driven annotation schema. In *LREC00*. Athens.

Carroll, J., T. Briscoe, and A. Sanlippo, 1998. Parser evaluation : A survey and a new proposal. In *LREC98*.

Doran, C., B. Hockey, A. Sarkar, B. Srinivas, and F. Xia, 2000. Evolution of the xtag system. In A. Abeillé and O. Rambow (eds.), *Tree Adjoining Grammars*. Chicago Press, pages 371–405.

Joshi, A. and Y. Schabes, 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa (eds.), *Handbook of Formal Languages*. Springer, pages 69–123.

Magerman, D., 1995. Statistical decision-tree models for parsing. In *ACL95*.

Marcus, M., B. Santorini, and M. A. Marcinkiewicz, 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330.

Neumann, G., 2003. An uniform method for automatically extracting stochastic lexcalized tree grammars from treebanks and HPSG. In A. Abeillé (ed.), *Building and Using Parsed Corpora*. KLUWER, pages 351–366.

Sarkar, A., F. Xia, and A. Joshi, 2000. Some experiments on indicators of parsing complexity for lexicalized grammars. In *COLING00*.

Srinivas, B. and A. Joshi, 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Xia, F., 2001. *Automatic Grammar Generation from two Different Perspectives*. Ph.D. thesis, Computer and Information Science Department, Pensylvania University.