Transformed Subcategorization Frames in Chunk Parsing

Leonardo Lesmo & Vincenzo Lombardo
Dipartimento di Informatica and Centro di Scienza Cognitiva
Università di Torino
C.so Svizzera, 185 - I-10149 Italy
{lesmo, vincenzo}@di.unito.it

Abstract

This paper describes an approach to treebank development which relies on the manual development of annotation tools. The overall process of tree annotation is described, and a special emphasis is put on the description of the last tool which has been built, i.e. a dependency-based robust chunk parser. The modularization of the parser and the central role of verbal subcategorization is presented. The first experimental results, carried out on a corpus of 645 sentences are reported and discussed.

1. Introduction

It is well known that the amount of linguistic resources available for different languages vary considerably in extent (see, e.g., (ATALA, 1999) for report of treebanks for German, Spanish, Chinese, Polish). Currently, Italian, despite its historical and cultural value, stands rather low in the scale. Although some wide-scale projects are under way (e.g. CLIPS: Lessico Computazionale dell'Italiano Parlato e Scritto, Computational Lexicon of Spoken and Written Italian), there is a shortage of POS annotated corpora. The situation of procedural resources, as POS taggers and wide-coverage robust parsers is even worse.

The construction of annotated corpora is a particularly labor-intensive and time-consuming task usually performed by human annotators with the help of software tools. Usually the annotation process occurs in two phases: a fully automated Part Of Speech (POS) tagging and a syntactic annotation that can be performed in different ways. For example, in the Penn Treebank (Marcus et al., 1993) syntactic bracketing consists of the manual correction of an automated parsing output where the annotator possibly "glues" together disconnected syntactic chunks. In the NEGRA treebank (Brants et al., 1999) the annotation is an interactive process that the annotator can stop at any point to correct or alter the structure automatically generated. In general, syntactic annotation schemata of existing treebanks try to achieve a trade-off between representation accuracy and corpus data coverage satisfying some requirements of theory-independence (Skut et al., 1997) (XTAG Project: www.cis.upenn.edu/~xtag) and annotation consistency between the analyses of different phenomena (Marcus et al., 1994). Data coverage requires a formalism that is able to represent all the types of specific linguistic phenomena occurring in the corpus, catching most of the peculiarities of the represented natural language.

Recent work in corpus-based linguistic analysis has pointed out the necessity of including predicate-argument structures in treebank annotation schemata (Marcus et al., 1994) (Skut et al., 1997). This permits the collection of large data bases of subcategorization frames and their relative statistics, that are of valuable help for the accuracy of parsing models (Collins, 1997), and in the implementation of IE systems (Vilain, 1999).

In the last five years, the Turin University NLP working group has engaged in a first effort in building NL resources for Italian. The research proceeded in steps, and has a threefold goal:
- to develop annotated corpora
- to develop tools for annotation
- to develop linguistic knowledge bases

While the first two objectives are clear enough, the third one requires some words of explanation. We believe (in agreement with, e.g. (Skut et al., 1997)) that the parsing task could be made easier by exploiting pieces of well-founded linguistic knowledge. In particular, we have paid considerable attention to the use of subcategorization information, especially for verbs, but also for nouns and some adjectives. Of course, subcategorization data, as other pieces of linguistic knowledge, is not readily available for Italian; but our mostly manual approach to parser development enabled (or forced) us to build also a depository of subcategorization information. So, as a temporary result of the work, which is under way, we have today available not only 1200 parsed sentences, but also a set of about 360 verbs for which accurate subcategorization data are available.

In short, the current process of treebank development is based on the following steps:

1. A text (without any form of preprocessing) is submitted to a tokenizer. The tokenizer keeps apart standard words from punctuation marks, special symbols (e.g. @), and numbers. The tokenizer will not be discussed in the present paper.
2. The output of the tokenizer is submitted to a morphological analyzer. Italian is a rather highly inflected language, so that word stems have to be separated from suffixes. The analyzer uses an (augmented) automaton for hypothesizing suffixes, and a morphological dictionary (including about 24.000 lemmas). The augmentation aims at collecting syntactic information (as gender and number) which

is usually carried by the suffix. An average of 1.9 interpretations for each token survive to the dictionary filter. As the previous one, this step will not be discussed here.

3. The output of the morphological analyser undergoes POS disambiguation. A rule-based POS tagger chooses the best syntactic interpretation for each token. The POS tagger has been presented elsewhere (Boella & Lesmo 1998), and will be briefly reviewed in the next section. The current performance is about 3.4% of errors on a corpus of newspaper articles on various topics, but, unexpectedly, much better results have been obtained recently on different data.
4. The resulting disambiguated text is manually inspected to identify and correct tagging errors.
5. The correct POS-tagged text is submitted to the parser. The parser is the main topic of the paper; it will be discussed in Section 5.

The final goal of this paper is to show that this interleaving of manual and automatic steps produces good parsing results; that the annotation tools are improved while the treebank grows; that some linguistic knowledge developed during the process is valuable, in the sense that it can be reused in case more difficult tasks (e.g. full parsing) need be faced. The paper describes the whole annotation process, but, since some modules have been described elsewhere, the main topic is the chunk parser, which is new. In particular, section 2 describes the structure of the parse trees which are obtained, section 3 reviews the POS tagging process, section 4 introduces our approach to subcategorization, section 5 is devoted to the chunk parser, and section 6 presents the experimental results.

2. Syntactic Structures: Dependency

The parse trees built by the parser follow the Dependency Approach to syntactic structures. We cannot review here the presumed advantages of dependency over constituency; both theoretical work (Mel'cuk 1988) (Hudson 1990) and practical work (Hajic 1998) have been carried on within this paradigm. Also constituency-based trees usually include dependency concepts (e.g. functional structures). The goal of this section is just to introduce briefly some notation which will be used in the following.

We use 16 Part Of Speech tags (ADJ, ADV, ART, CONJ, DATE, INTERJ, MARKER, NOUN, NUM, PHRAS, PREDET, PREP, PRON, PUNCT, SPECIAL, VERB). For some of them, subcategories are defined, some of which have mainly a semantic flavour. Examples of subcategories are COMMON and PROPER (for NOUN), MAIN, MOD (modals) and AUX (for VERB), and ADFIRM, ADVERS, COMPAR, DOUBT, etc. (for ADV). 51 subcategories are currently in use. Most categories have features associated with them, as GENDER and NUMBER (for ADJ, ART, NOUN, PRON. PREDET and some VERB), MOOD, TENSE, PERSON (for VERB). So, in evaluating the performances of the POS tagger, it must be observed that not only category, but also subcategory misclassifications and errors in gender, number, person count as errors.

Each entry (words, punctuation, numbers), after being POS disambiguated, becomes a node in a Dependency Parse Tree (DPT). So, according to the dependency approach, each node in a DPT corresponds to an input word (but we augment dependency trees with trace elements, see below), and not only the leaves. The structure of the tree is given by labelled arcs, which connect pairs of nodes. An example of a DPT is reported in fig.1.

The organization of the arc labels is a practical application of the theory exposed in (Hudson 1990). Labels belong to a taxonomy, where a label L_i is a daughter of a label L_j, in case L_i is "more specific" than L_j. The root of the taxonomy is the label DEPENDENT, whose daughters are COORD, EXTRA, SEPARATOR, NONVERBAL-DEPENDENT, VERBAL-DEPENDENT. An example of a path in the taxonomy is

DEPENDENT ✍ VERBAL-DEPENDENT ✍
✍ COMPLEMENT ✍ SUBJ ✍
✍ VERBCOMPL-SUBJ ✍ INFVERBCOMPL-SUBJ

where the leaf INFVERBCOMPL-SUBJ is associated with infinite clausal subjects. "More specific" has to be interpreted in some cases in terms of syntactic realization (VERBCOMPL-SUBJ specializes in INFVERBCOMPL-SUBJ vs. FINVERBCOMPL-SUBJ) and/or semantic content (LOCCOMPL specializes in LOCCOMPL-IN, LOCCOMPL-TO, LOCCOMPL-FROM).

The taxonomy has a double function: first, it enables to express in a compact way properties of relations (obligatoriness of COMPLEMENT, agreement of SUBJ); second, and more important in the present context, it enables the annotator (manual or automatic, to introduce in the DPT underspecified labels, so that the annotation is speeded up in case dubious cases are faced; but at the same time very specific labels can be assigned in more clear-cut cases, so that the more detailed information is included in the tree (Bosco, 2000).

Finally, we must notice that a DPT can also include non-lexical nodes, i.e. traces. Traces are used to represent movement (e.g. raising), sharing (e.g. equi), deletion (Pro-Drop) and long distance dependencies. So, in the trees appearing in the treebank, we have a full representation of the verbal dependents (subjects appear also in case of Pro-Drop or for infinitives), and crossing arcs are avoided.

Currently, the taxonomy includes 370 labels.

3. POS tagging

The POS tagger is rule-based. A first set of rules has been developed manually, and has subsequently been extended via the application of an automatic learning tool (Boella & Lesmo, 1998). The tagger works strictly left-to-right: when it encounters a word having multiple interpretations, it collects the different possible syntactic tags for that word, and applies the rules aimed at disamb-

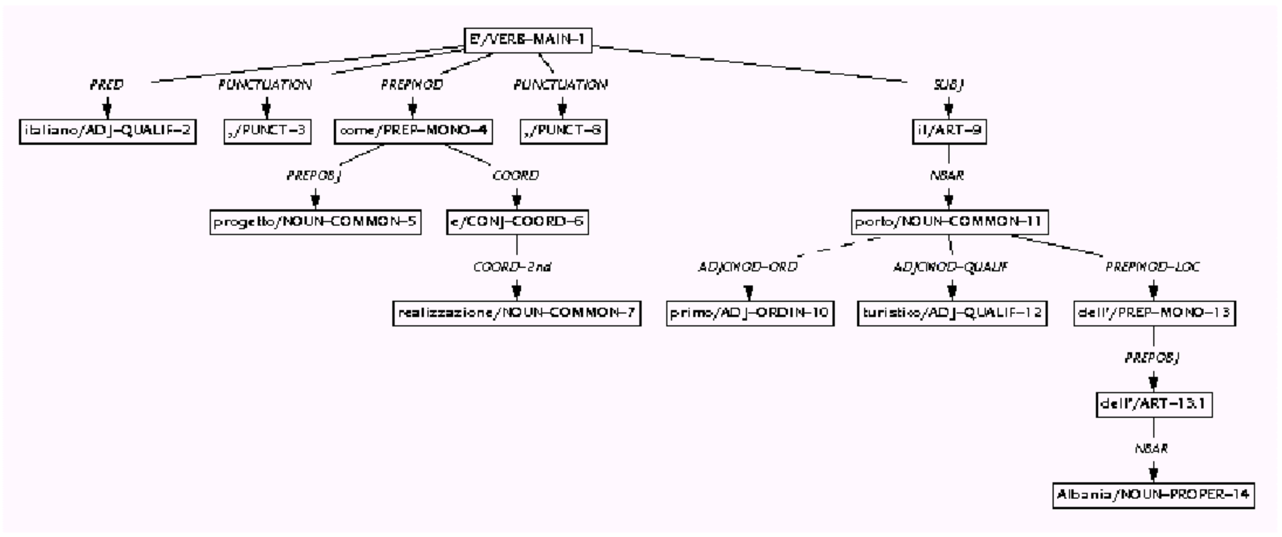

Fig.1 – The Dependency Parse Tree of the sentence "E' italiano, come progetto e realizzazione, il primo porto turistico dell'Albania"

iguating among those tags. Rules are organized in packets, each of which contains the rules that handle a given ambiguity. So, there exists a packet for NOUN-VERB ambiguity, another packet for ADJ-ADV-NOUN, and so on. Each packet also include a default tag, which is chosen in case no rule applies. Most packets aim at disambiguating among categories (e.g. NOUN and VERB), but a few packets apply to subcategories or features (e.g. for PERSON ambiguity of verbs).

Each rule has the simple form

IF condition THEN tag CF certainty-factor

where *certainty-factor* can assume the values C(ertain), A(lmost certain), U(ncertain), and is used to state the priority of application of the rules. The condition tests the neighborhood of the ambiguous word, which consists in the two preceding and the two following words. Notice that the previous words have already been disambiguated, but the next two words can still be possibly ambiguous. This makes the checks on the following context less reliable than the ones on the preceding context.

Currently, we use 136 rules, 127 of which handle inter-category ambiguities, while 9 of them apply to intra-category ambiguities.

The original tests, carried out on an eterogeneous corpus including newspaper articles, chapters of novels and of technical books, amounting to 48.000 lexical items gave a result of 3.42% wrong tags. However, we have recently started the analysis of some parts of the Italian Civil Code. Unexpectedly, the tagging errors on the sentences analyzed (13491 items, including punctuation) were 260 (1.93%). This is probably due to the more regular structure of the legal language, with respect to other texts, but it can be argued (although the new data are limited in size) that the tagging rules implicitly encode some principles of well-written Italian; this is presumably due to their having been written manually, trying to face the problems found in the corpus from a "general knowledge of language" perspective: the encoder of the rules, in fact, tried to write tagging rules reasonably general, i.e. to avoid rules applying just to peculiar cases casually occurring in the corpus. The tagger disambiguates about 800 words per second, in compiled LISP code on a Pentium III processor (including the writing of the output file).

4. Syntactic knowledge: Subcategorization

In order to describe the chunk parser, we need to introduce the approach to verbal subcategorization we have adopted. Each verbal subcategory is associated with a Subcategorization Class (SC); SC's are arranged in a taxonomy. The root of the taxonomy is the class VERBS, while other nodes are, for instance, TRANS(itive), INTRANS(itive). Other nodes refer to less standard subcategories; for instance, INTRANS-INDOBJ-PRED refers to verbs as "sembrare" (to seem), which admit a subject (which can be a subordinate), an indirect object, and a predicative complement:

Ex 1. La casa gli sembra bella
(*The house to-him seems nice*)
Ex 2. Correre nei prati mi sembra splendido
(*To-run in-the fields to-me seems beautiful*)

Some of the subcategories refer to verbal locutions, as PERDERE-DI-VISTA (*to lose sight of*)

Ex 3. Mario ha perso di vista il suo obiettivo
(*Mario has lost sight of [the] his goal*)

Each SC is associated with a subcategorization frame (SF), which includes the definition of one or more complements, given by the complement name (i.e. the label of arcs that will appear in the resulting DPT to connect a verb with that complement), and a description of the syntactic structure licensed for that complement.

Actually, the true definition of the set of complements defined for a class is obtained via an inheritance mechanism. For instance, the SC of SUBJ-VERBS includes the definition of standard 'subjects', and the class TRANS (transitives) is a daughter of SUBJ-VERBS. So, for TRANS, the syntactic structure of subjects need not be defined, since it is inherited from SUBJ-VERBS. Currently, the hierarchy includes 150 classes, 37 of which refer to verbal locutions (as "aver bisogno", to need). 361 verbs have been classified according to the classes. On the average, each verb belongs to 2.3 different classes.

Of course, the definition of the classes regards a 'base' form: the verb is active and finite, pro-drop has not been applied, no cliticization and no movement has occurred. But the verb can appear in a sentence in other forms, for which the surface realization defined in the class cannot match the input pattern. So, the base class definitions undergo a process of transformation, where all possible (according to the current set of 11 defined transformations) surface realizations are generated. The final outcome of this process is that the original (base) hierarchy (of 150 nodes) is translated into a surface hierarchy including 3507 transformed classes.

For instance, the base class TRANS gives rise to 28 derived class (among which TRANS+INFINITIVIZATION, TRANS+PASSIVIZATION, TRANS+INFINITIVIZATION+OBJECT-RAISING+IMPERSONALIZATION, etc.). In some cases, the complement label is affected by a transformation; although our approach is strictly mono-stratal, we have adopted the convention that a transformed complement name can keep trace of the transformation: so, arc labels as SUBJ/AGTCOMPL can be found in the parse trees.

5. A chunk parser

As stated in the introduction, the parser operates on the POS-tagged sentences, after they have been manually corrected. It works according to the steps shown in Fig.1.

5.1 Non-Verbal Chunking

Non-verbal chunking rules (NVCR) work in a rather standard way, starting from smaller chunks, and then trying to include them in larger chunks. It must be observed that sentential dependents (subordinates and relative clauses) are not linked to their governor in this step, but they are left unattached until the Verbal-chunking step is applied.

Each NVCR is associated with a given category, and aims at finding links from a word to that category (the governor) to a depending word. The format of a NCVR is:

(<categ> <subcateg> <direction> <condition> <label>)

where:

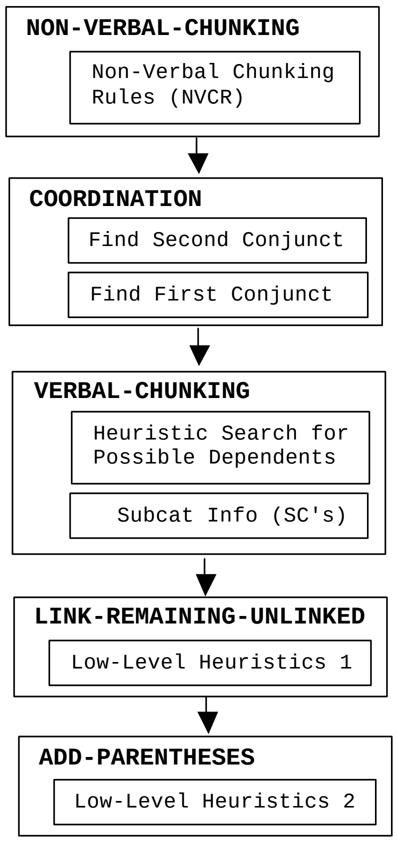

Figure 1 - The Chunk-Parser

- *<categ>* is the category of the head word
- *<subcateg>* is the subcategory of the head word
- *<direction>* specifies where the dependent can appear (see below)
- *<condition>* specifies the features of the dependent
- *<label>* is the label that must be assigned to the dependency arc connecting the head with the dependent

A NCVR is applied when a word (the potential 'head') of the given *<category>* and *<subcategory>* is found in the text, and aims at looking for surrounding words that can act as dependents for the head. Such a dependent must be found in a position specified by *<direction>*, and must have the features specified in *<condition>*. If such a word is found, then it is linked to the head, via an arc labelled as *<label>*. The major power of the NVCR comes from the flexibility of the *<direction>* specification. The simplest cases are *'before'* and *'after'*, which require that the dependent comes immediately before or after (i.e. it is adjacent to) the head (as for adverbs with respect to a governing adjective). A bit more complex are *'precedes <skip-word>*'* and *'follows <skip-word>*'*, which license zero or more occurrences of a word having the features described in <*skip-word*> to appear between the head and the dependent. Finally, *'chunk-precedes <category>'* and *'chunk-follows <category>'* enable the parser to attach to the head any word of <*category*>, which is the root of a subtree (immediately preceding or following) the head.

An example of a NVCR is the one that links an adverb to an adjective:

```
(ADJ QUALIF
        BEFORE (ADV (TYPE MANNER))
        ADVBMOD-MANNER)
```

So, if an adverb of subcategory (*TYPE*) MANNER immediately precedes a qualificative adjective, then it can depend from it via an arc labelled as ADVBMOD-MANNER. A simple example of a *'chunk-follows'* rule is:

```
(ART DEF
        CHUNK-FOLLOWS (NOUN (AGREE))
        NBAR-DEF)
```

which enables the parser to link a noun-rooted chunk to a governing definite article, as in:

Ex 4. IL mio più grande AMICO ...
(*THE my most great FRIEND …*
"My greatest friend …")

The NVCR's are applied in different cycles to the input text, according to the head categories, in the following sequence: SPECIAL, PUNCT, ADV, CONJ, ADJ, NUM, NOUN, ART, PRON, PREP, VERB (the VERB rules take care just for the attachment of the depending auxiliaries). So, larger and larger chunks are built, but some items still remain unattached; in particular, prepositions not following a noun (PP's), articles not following a preposition (NP's not part of a PP), conjunctions, punctuation marks, and verbs. It must be observed that the application of NVCR's is the most reliable step, with the exception of PP attachment, which favours the attachment to a preceding NOUN rather than to a VERB[1] (see Section 6).

5.2 Coordination

Coordinate structures probably are the most complex compounds of NL. There are very few reasonable criteria enabling a parser to find out which are the two conjuncts, and in many cases the final word can only be said by the semantics. So, the only possible approach is to use heuristic criteria. Arguably, sensible criteria can only be found by means of statistics: whenever a required knowledge source, as the meaning of words, is missing, statistics can act as a surrogate. However, we had still to face the problem of the shortage of data. So, we approached the problem by hand-writing intuitively reasonable criteria, by leaving the application of automatic learning methods to the time when the treebank has grown enough (the same method we adopted for the POS tagger).

At this level of development, coordination is handled by first determining a plausible second conjunct, and then finding out a syntactically homogeneous first conjunct. The second conjunct is:

- ✍ if the conjunction is coordinative or disjunctive (basically 'and' and 'or'), then the head of the second conjunct is the word immediately following the conjunction
- ✍ in other cases (e.g. adversatives, 'ma': 'but') the second conjunct is the first 'free' verb after a conjunction, where a 'free' verb is a verb that is not preceded by a subordinative conjunction or by a relative pronoun (if such a verb cannot be found, suitable 'escape' rules are applied).

Of course, it is easy to find cases where these two criteria fail (and there are plenty of them in our small corpus), but these simple rules work rather well (see, again, section 6).

So far as the first conjunct is concerned, it must be of the same category of the second conjunct, and, in case it is a verb, it must also be of the same mood (remember, that, by 'conjunct', we mean 'head of the conjunct', because of the dependency approach).

5.3 Verbal Chunking

The SC’s described in section 4 put at disposal to the parser the information about the complements licensed by a verb belonging to the class. This information is obtained by first retrieving the SC's to which a verb belongs (there may be more than one class, because of different meanings of the verb), then by selecting a subset of the possible transformations, and finally by retrieving the SC definitions (complement names and surface realizations). The second step aims at filtering out the transformation not compatible with the surface form of the verb (e.g. passivization, in case the verb is not passive), and at forcing the transformation required by the surface form (e.g. passivization, if the verb is passive). At this stage, all possible sets of complements are available.

The second thing to do is to find out the dependents of the verb appearing in the input sentence. This is achieved by determining some heuristic boundaries for the verbal caseframe and by extracting out of that interval all unlinked elements. At this point, the transformed class definitions can be matched against the set of unlinked elements, in order to determine their grammatical function (expressed via the link label).

Unfortunately, even if all necessary information is available, the problem is complex (Buchholz et al. 1999) (Lesmo & Lombardo, 2000). For instance, in

Ex 5. la legge che applica il giudice in questi casi …
(*the law that applies the judge in these cases …*
"the law that the judge applies in these cases ")

[1] Actually, the PP's headed by 'di' (of) are preferrably linked to a preceding noun, while other prepositions are assumed to head verbal dependents, unless they are explicitly specified as possible nominal arguments.

Italian allows for the subject to follow the verb, so that the assignment of Subject and Direct Object can only be made on the basis of semantic knowledge. So we introduced some heuristics to choose the best assignment (heuristics that would fail, for instance, in the example above). In sum, there are three possible sources of errors: lack of information about the subcategorization frame of a given verb, failure in determining the right set of dependents, and failures in case assignment (section 6).

5.4 Unlinked elements and parentheses

After the Verbal Chunking step, some elements still remain unlinked to any parent. In particular, the root of the tree, the verbal heads of relative clauses, verbs governed by subordinating conjunctions, punctuation. Moreover, we approached the problem of parenthesized material by analyzing the contents of each parenthesis as a single sentence, so the parenthesis as a whole has still to be inserted in the whole tree. The last two steps of the parser aims at completing the construction of the parse tree. The unattached verbs are inspected to find the most promising attachment point (a noun preceding a relative pronoun or a subordinating conjunction); punctuation is attached to the lowest common ancestor of the surrounding words. For parentheses, the situation is a bit more complex, since the choice of the attachment point depends on the head word inside the parenthesis (which could be a conjunction, a preposition, a noun, etc.). The results reported in section 6 show that in the attachment of these elements is much more domain dependent than for the elements handled in the previous steps. In other words, it is hard to devise heuristics of wide application.

6. Experimental results

The experiments have been carried out on 645 sentences, for a total of 17.397 items. The sentences are divided in four groups: 437 sentences from the Italian Civil Code, used to refine the parse rules (LEARN); 100 sentences from the same corpus (TEST1), 58 sentences from a corpus semantically related to the learn set (legal sentencing: TEST2), and 50 sentences from an heterogeneous corpus extracted from newspaper articles (TEST3).

It has been considered as an error:

- The wrong choice of the parent
- The wrong arc label

The first type of errors are Attachment Errors, while the second type concerns mainly the selection of arguments for the verbs.

The errors have been subdivided according to the step of the parser where the error has been made, so there are errors in the application of Non-verbal Chunking Rules (NVCR), errors in the treatment of coordination (COORD), errors in matching the case frames of verbs (Verbal Rules: VR), errors in the attachment of the final unlinked items and in the placement of parentheses (UNL+PAR). The total figures are reported in Table 1.

The first step takes care of 50.5% of all arcs appearing in the DPT's (8786 applications of NVCR rules), and is the most reliable. Although there is a remarkable difference between the three sets of sentences, the heuristics adopted (see Footnote 1) gives good results. The difference between LEARN and TEST2-TEST3 is due to the introduction of some specific rules (e.g. for interpreting dates in the form 27-2-1998) which are useful in the Civil code, but do not apply commonly outside it. The better result of LEARN with respect to TEST1 is due to rules which, although they represent structurally valid principles, are applied to instances appearing just in LEARN. Finally, the residual errors in LEARN depend on the impossibility (or inability of the human rule writer) to devise disambiguating principles in cases where the correct choice seems to be possible just on the basis of semantic preferences.

The COORD step seems to work rather well, but it must be observed that the errors reported in the table concern only the choice of the two conjuncts, while they do not cover unrecognized coordination. This happens in sequences as "John, two girls, and an old man", where the first comma acts as a coordination, but it is not recognized as such by the parser. If we include these cases, in the analyzed corpus, we find 777 coordinating conjunctions (i.e. 1554 items that should be linked by the COORD rules, instead of the reported 1210); so, the actual figure should be 538 (194 detected + 344 undetected) errors in 1454 items (i.e. 37.00%). The 344 errors not appearing in this column occur in the table mainly as UNL+PAR errors.

The VR step shows very large differences in the corpus. Although they cover just 21.58% of the links, the presence of subcategorization data enables the parser to perform somewhat better. In the LEARN set, we have all verbs correctly classified, and the 16.84% of errors is due to wrong matches between subcategorization pattern and input pattern (e.g. Direct Object and Subject exchanged; see Ex.5 above), and to errors in the initial hypothesis of attachment of a dependent to the verb (Heuristic Search for Possible Dependents in Fig.1). In TEST1, some of the verbs appear also in LEARN, but some unseen verbs appear, so there is a limited decrease in performances. In

		NVCR			COORD			VR			UNL + PAR			TOTAL		
	Items	Tot	Err	%	Tot	Err	%	Tot	Err	%	Tot	Err	%	Tot	Err	%
LEARN	10476	5257	69	1.05	844	109	12.50	2142	406	16.84	2233	382	17.11	10476	966	9.22

TEST1	2901	1494	51	3.41	179	27	15.08	630	171	27.14	598	183	30.60	2901	432	14.89
TEST2	2166	1169	48	4.11	93	33	35.48	493	182	36.92	411	184	44.77	2166	447	20.64
TEST3	1854	866	36	4.16	94	25	26.60	490	185	37.76	404	204	50.50	1854	450	24.27
Total	17397	8786	204	2.32	1210	194	16.03	3755	944	25.27	3646	953	26.14	17397	2295	13.19

Table 1. Errors made by the parser in its different steps

TEST3, where most of the verbs lack accurate subcategorization infos, the performance drops to a 37.76% of errors. However, the results on TEST2 are somewhat contradictory, since the semantic domain is the same as TEST1 (and LEARN). On a closer inspection, however, we see that the average sentence length of TEST2 (37.3) is greater than in the other sets, with the exception of TEST3 (one sentence in TEST2 included 188 items, the longest sentence we ever found in our corpora), so, in this case, the low performance of the VR rules are probably due to the difficulties encountered by the heuristcs which determine the possible dependents (but a more in-depth analysis is required here).

Finally, in the last two steps, it is apparent that the small size of the LEARN group has strongly affected the generality of the heuristics written by the human expert, so that the uniformity of TEST1 with respect to LEARN still preserves some reasonable figures, but on the other data they affect very negatively the overall performances. It can be argued that the rules used in the first three steps arose as a combination of data inspection and human writer intuitions about language, while in these last two steps, the 'intuition' component is largely absent (in particular because of the rather irregular use of punctuation, and of the unconstrained application of parentheses).

With respect to time, the parser, which, as we said above, takes as input a POS-disambiguated file, produces the resulting parse tree at a rate of about 700 words per second (compiled Lisp code on a Pentium III PC, under Linux).

7. Conclusions

This paper has presented an approach to treebank development supported by manually developed annotation tools. In particular, the focus of the paper has been on a robust chunk parser, and on its exploitation of knowledge about verbal subcategorization frames. The experimental results have shown that the parser performs much better in case such knowledge is available.

One of the sources of efficiency in the time performances mentioned above is that the parser is strictly deterministic: no label assigned in one of the steps appearing in Fig.1 is changed by a following step. However, the lack of interaction among the different module is also a primary source of errors. In particular, for PP attachment, it seems that the only way to choose the best solution is to make preferences contrast different possibilities. However, the NVCR have to take a decision before the VR enter into play. We are currently investigating the effect of interleaving in some steps of the parser.

In a similar way, inside the Verbal Chunking phase, the possible dependents of a verb are hypothesized before the knowledge about subcategorization is applied. Again, the possibility of accepting or rejecting a dependent seem to be useful to improve the correctness of the resulting structure, but this has a non-trivial impact on the strategy for matching the subcategorization frames against the input.

8. References

ATALA (1999). Proceedings of Treebanks workshop – Journées ATALA sur les corpus annotés pour la syntaxe, 18-19 juin 1999, Paris, http://talana.linguist.jussieu.fr/treebanks99/.

Boella G., Lesmo L. (1998): Automatic refinement of Linguistic rules for tagging, Proc. 1st Int. Conf. on Language Resources and Evaluation, LREC 1998, Granada, 923-929

Bosco C. (2000): A richer annotation schema for an Italian treebank. In C. Pilière, editor, Proc. of ESSLLI-2000 Student Session, Birmingham, 22-33.

Bosco C., Lombardo V., Vassallo D., and Lesmo L. (2000): Building a treebank for Italian: a data-driven annotation schema. In Proc. 2nd Int. Conf. on Language Resources and Evaluation, LREC 2000, Athens, 99-105.

Brants T., Skut W., Uszkoreit H. (1999): Syntactic annotation of a German newspaper corpus. In Proc. of Treebanks workshop – Journées ATALA sur les corpus annotés pour la syntaxe, Paris, 69-76.

Buchholz S., Veenstra J., Daelemans W. (1999): Cascaded grammatical relation assignment. EMNLP/VLC-99, University of Maryland, USA. CL/9906004.

Collins M. (1997): Three generative, lexicalized models for statistical parsing. In Proc. of the 35th Meeting of the ACL , 16-23.

Hajic J. (1998): Building a syntactically annotated corpus: the Prague Dependency Treebank In Issues of Valency and Meaning, Karolinum, Praha, 106-132.

Hudson R. (1990): English word grammar. Basil Blackwell, Oxford and Cambridge, MA.

Lesmo L., Lombardo V. (2000): Automatic assignment of grammatical relations. In Proc. 2nd Int. Conf. on Language Resources and Evaluation, LREC 2000, Athens, 475-482.

Mel'cuk I. (1988): Dependency syntax: theory and practice, SUNY University Press.

Marcus M.P., Santorini B., Marcinkiewicz M.A. (1993): Building a large annotated corpus of English: The Penn Treebank, Computational Linguistics 19, 313-330.

Marcus M.P., Kim G., Marcinkiewicz M.A., et al. (1994): The Penn Treebank: Annotating predicate argument structure. In Proc. of The Human Language Technology Workshop, San Francisco, Morgan-Kaufmann.

Skut W., Krenn B., Brants T., Uszkoreit H. (1997): An annotation scheme for free word order languages. In Proc. of the Fifth Conference on Applied Natural Language Processing (ANLP), Washington, D.C.

Vilain M., (1999). Inferential Information Extraction, in M. T. Pazienza (ed.), Information Extraction, LNAI 1714, Springer Verlag, 95-119.