The Turin University Parser at Evalita 2009

Leonardo Lesmo

Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, I-10149 Torino, Italy
lesmo@di.unito.it

Abstract. This paper describes the parser that produced the results submitted to Evalita for both the Main and the Pilot Subtasks. This is a system description, which stands in the mid-way between a scientific paper and an internal report. The aim of the author is just to provide the reader with the information needed to understand how the parser works and how the final results were obtained. Some attention will be paid to the difference in performance between the Main and the Pilot Subtasks, that led to substantially different rankings of the system in the two subtasks.

Keywords: Rule-based Parsing, Chunk Parsing, Parsing Evaluation

1 Introduction

The Turin University Parser (TUP) is a Rule-based heuristic parser that produces a dependency tree for each sentence of the input text. The parser includes a number of modules, some of which will not be presented here, since this description will be focused on the parsing proper. The morphological analyzer and the PoS Tagger, that precede the operation of the parser, are described in some detail in the paper addressing the participation of our system to the Evalita 2009 PoS tagging task [1].

TUP is a large coverage parser based on two main steps: chunking and analysis of verbal dependents. Chunk parsing was first introduced in [2] and is based on the idea of extracting relevant portions of a sentence (chunks) on the basis of highly reliable rules. Various chunk parsers have been developed in the last two decades, and the approach is still in use [3]. The goal of the second step is to collect chunks and to attach them to verbs in order to build complete connected structures. This requires a selection of clause boundaries (accomplished via heuristic rules) and the decision about the role of the dependents, made via verbal subcategorization and a flexible representation of verbal case frames.

The parser is being used in a number of projects addressing multilingual NL access to Cultural Information [4], information extraction from legal texts [5], question answering in restricted domains [6], translation from Italian to LIS (Italian Sign Language of the deaf) [7].

A scheme of the whole system is reported in fig.1. The paper is organized as follows: we first describe the resulting syntactic structures (parse trees), then we give some details about the modules composing the system. Finally, we discuss the performances of the parser in the Evalita parsing task.

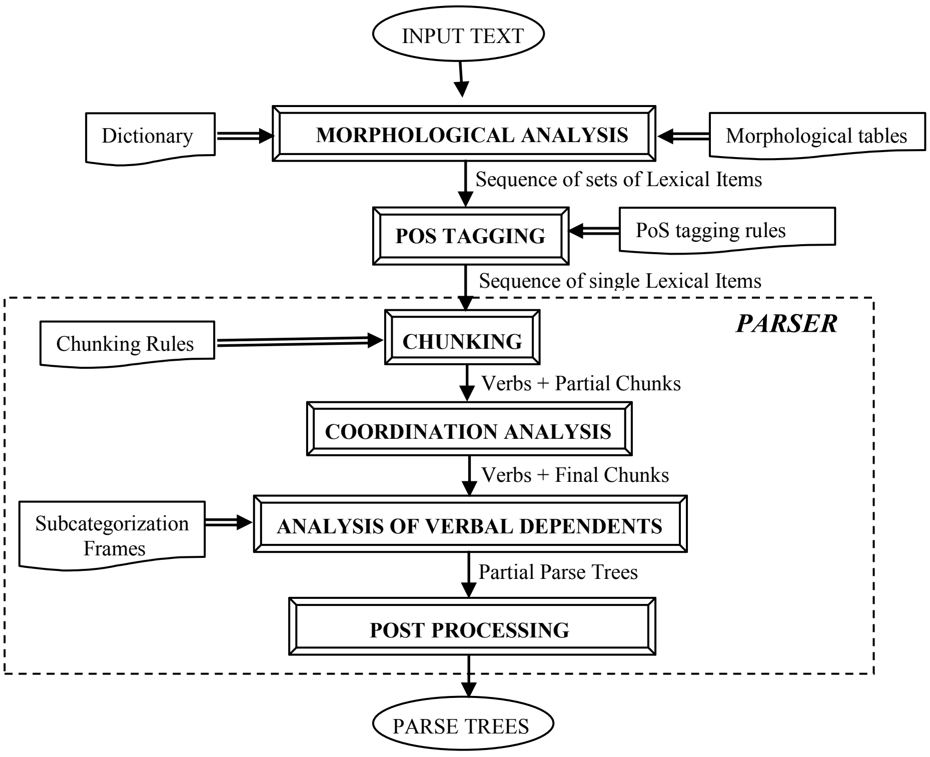

Fig. 1. Architecture of the parser (including components not described in this paper)

2 The parse tree (TUT format)

This section describes the structure of the dependency trees in the native format, i.e. the one of the Turin University Treebank (TUT). In §4, we discuss the problems associated with the conversion into the CoNLL format required for the parsing task of Evalita. We cannot describe here the details of the representation (see [5]), but we provide the reader with information needed to follow the rest of the paper.

TUT trees are projective dependency trees. They include empty categories (traces) that are used to respect the condition of projectivity in the presence, for instance, of long-distance dependencies. Another noticeable feature of the representation is that, when determiners are present, nominal groups are headed by them, instead of the associated noun. Compound words (e.g. verbs including enclitics) are split into their components: with each of them a distinct node of the parse tree is associated. The same happens for multi-word expressions: each of their components is associated with a different node, but in this case the lemma is the same for all of them.

Each node includes a number of features, including the syntactic category (PoS) of the item, a subcategory (e.g. *indefinite* for adjectives) and information about gender, number and, for verbs, tense, mood, etc. We currently use 17 syntactic categories, i.e. *noun*, *verb*, *adj* (adjectives), *adv* (adverbs), *art* (articles), *pron* (pronouns)*, num* (numbers), *prep* (prepositions), *conj* (conjunctions), *predet* (predeterminers), *punct* (punctuation marks), *date* (in standard format, as 18/12/2009), *interj* (interjections), *marker* (textual markers), *phras* (phrasals, i.e. ‘yes’, ‘no’), *special* (special symbols, as #), *index* (for chapter/paragraph numbers). If we take into account the subcategories, we get 52 different classes for TUT items.

The label of the arcs are given in terms of a hierarchy of labels: topmost labels are a form of underspecification that enables an annotator to delay the decision about a given specific label. The topmost levels are *ARGUMENT* and *MODIFIER*, while at lower levels we have, for instance, *RMOD* (restrictive modifier), *VERB-RMOD+RELCL* (verbal restrictive modifier which is a relative clause), *VERB-SUBJ* (verbal subject). Detailed information about these structures can be found in [8].

3 The parser

As it can be seen in Fig.1, the parser includes 4 main components: the chunker, the analyzer of coordination, the analyzer of verbal dependents, and a post-processor. Actually, in order to guarantee the coverage also for ill-formed inputs, and to account for some special phenomena, the number of modules is slightly larger. We first provide an overview of all modules, and then pay special attention to the main ones.

3.1 Parser submodules

Parentheses - Since the parser works on single sentences, a first step consists in the removal of all parentheses occurring within the sentence. The material in parenthesis is parsed separately, and then the result is re-inserted in the same place.
Multi-word - The second step takes care of setting the links related with multi-word expressions. They have already been identified by the PoS Tagger, and now all their components are linked via *CONTIN+LOCUT* arcs.
Chunking - Then, the first main step, i.e. the application of the chunking rules, is executed (see §3.2). The final result is a set of chunks (nominal, adjectival, adverbial, etc.) that are unattached, i.e. the head of the chunk still has an empty link.
Item lists - Now, item lists are checked. For instance, in case of a fragment as:
… tutta la famiglia (il padre, la madre, i loro genitori) …
… the whole family (the father, the mother, their parents) …
The material in parenthesis is taken as a special form of coordination. It is ‘special’, in the sense that no coordinating conjunction appears, but only commas. In this step, the sequence is recognized and the proper coordination arcs (see §4.3) are introduced.
“about which in” - This step is rather peculiar, and we mention it here just to give an idea about what we mean by ‘coverage’. Consider the following construction:
… il commento di cui al paragrafo successivo è molto importante …
(… the comment about which in the next paragraph is very important …)
This construction is rather common in legal texts. Since we assume the existence of a verbal trace (“di cui ‘si tratta’ al paragrafo successivo” – about which ‘we talk’ in the next paragraph), this step of the parser looks for the sequence “di cui a” and inserts the proper trace. Note that this trace includes as lemma the required verb (“trattare” – talk), so that the verbal analysis that follows is applied to it in the standard way.
Analysis of conjunctions - Possible conjuncts are looked for and linked to each other (see §4.4). So, larger chunks are formed, in preparation for the next step.

Attachment of verbal dependents – This step (§ 4.5) links chunks to verbs, in order to build up the final representation of the sentence structure.
Possible corrections of prepositional attachments – The whole tree is inspected, to see if it is possible to change the links of prepositions, thus obtaining a better structure. This applies to examples as “schiarite durante la giornata lungo il mare” (weather improvements during the day near the sea) and what is ‘better’ is defined in a separate KB. In the example above, the KB specifies that “schiarita lungo mare” (weather improvement on see) is a better attachment than “giornata lungo mare” (day on sea), so that the PP is moved up in the tree. Note that this is just a surrogate for semantic processing; in fact an ontological KB could provide the parser with the needed information. Since such a KB is not yet in use, the required data are supplied by dedicated KB’s (which are currently very small – 25 entries).
Postprocessing – It may happen that some items are still unattached, in part due to deficiencies of the parser, in part because of specific design choices. They are connected to the parse tree in this step. See §4.6 for details.

3.2 Chunker

The chunker inspects the sentence in different passes, and tries to apply to each word any chunking rule of the packet associated with the category of the word. Currently, there are 325 chunking rules. The various passes on the sentence refer to chunks of increasing complexity, in the following order, where the items in the list specify the syntactic category of the head of the chunk (see§4.1): *SPECIAL, PUNCT, ADV, PREDET, CONJ, ADJ, NUM, NOUN, PRON, ART, PREP, VERB.*

Each chunking rule has the following format

Head-category <Position, Dependent-category, Conditions, Label>

An example is:

```
NOUN  <Precedes (ADJ qualif T),
         ADJ,
         (type INDEF) (agree) (dep-follows ART),
        ADJ+QUANTIF-RMOD>
```

The rule specifies that an indefinite *(type INDEF) ADJ*ective (e.g. ‘many’) that precedes a *NOUN*, with zero or more possible qualificative adjectives in between *(ADJ qualif T)*, which *agrees* (in gender and number) with the noun, and which follows an article *(dep-follows ART)* must be linked to that noun as a dependent via an arc labeled as *ADJ+QUANTIF-RMOD*. For instance, this rule produces the attachment of *tanti* to *amici* in a phrase as “i tanti cari amici” (the many dear friends). The last condition prevents the attachment in “sono venuti tanti cari amici” (many dear friends have come), where the indefinite adjective governs the noun as determiner.

3.3 Coordination

The module that takes care of the analysis of coordination is probably the one that deserves more attention in the future development of the parser (see §4.3). Currently, it is based on procedural knowledge that, as such, is much more error prone than declarative knowledge.

When a coordinative conjunction is met, the parser first tries to identify candidate heads for the second conjunct (sc). If the conjunction is *and* or *or* then the candidates include all unattached items that follow the conjunction until the next verb, otherwise, only the next verb is used. After that, for each candidate, the possible first conjuncts (fc) are looked for before the conjunction. This produces a set of possible pairs:

$<<fc_{11},sc_1>, <fc_{12},sc_1>, \ldots <fc_{21},sc_2>, <fc_{22},sc_2>, \ldots>$

Finally, a set of heuristics is applied to find the best pair. Among them, the closeness of the items in the pair and the preference of noun group conjuncts over prepositional conjuncts; also the lexical features of verbs are taken into account.

3.4 Verbal Dependents

As stated in the introduction, the decision about the verbal structures is made in two steps: first, clause boundaries are determined, and then all chunks inside the boundaries are attached to the (single) verb occurring there.

For boundaries, heuristics are applied. The application of verbal rules is strictly left to right, so all unattached items to the left of the verb under analysis are taken as dependents of the verb. For what concerns the unattached items that follow, all of them are taken as dependents, until another verb is found. The found verb is included in the list of dependents if it is in the infinite or gerund mood, or if the verb under analysis admits direct discourse. There are, however, other items that can block the search: they include subordinating conjunctions, question words (assumed to be attached to the next verb), relative pronouns, and so on.

The second step takes care of choosing the right label for the chunks to be attached to the verb. This involves separating arguments from adjuncts and assigning each argument the correct label; traces are inserted in place of any missing argument. This work is made on the basis of a hierarchy of verbal subcategories, that include information about the possible arguments and their admitted realizations. Each class is then 'transformed' on the basis of a set of defined transformations in order to obtain all possible surface realizations of the class [8]. For instance the *basic-trans* (basic transitives) class, which licenses a *VERB-SUBJ* and a *VERB-OBJ*, undergoes, among others, *PASSIVIZATION*; so, verbs in that class can govern, in an input sentence, a *VERB-SUBJ* and a *VERB-INDCOMPL-AGENT* (agent complement).

Finally, a set of verbs has been classified according to the defined classes. This set includes 435 verbs, but all verbs in the dictionary (approximately 4300) have a default class (*TRANS*, *INTRANS* or *REFL*). The main difference is that the dictionary class has been assigned without a detailed analysis of the behavior of that verb.

Currently, there are 134 verbal subcategories (30 of which are *abstract*, in the sense that no verb can be directly attached to them), and 16 transformations. The number of surface realizations obtained from transformations is 3928.

3.5 Post-Processing

The main goal of the post processing step is to set the links of punctuations and verbs (apart from auxiliaries and reduced relatives). In particular, the main verb is chosen and, if no verb is available, another root is selected. For punctuation, various rules are applied, in order to mark, when possible, parentheticals. This process takes into account the structure of chunks and verbal phrases previously established.

Finally, the remaining unattached items are processed. This covers approximately 2.9% of the total items. This figure could seem higher than expected, but many chunks of sentences without verbs remain unattached, and also some adverbs still have to find the proper attachment point.

4 Results in the Evalita Parsing Tasks

The parser described in the previous section has taken part in both parsing subtasks of Evalita 2009, with very different results, since it rank was 1st in the Main Task, and 5th (with a relevant gap in performance from the 4th) in the Pilot task. The goal of this section is to explain the (presumed) reasons of this difference.

4.1 The Main Subtask

The first relevant feature of this task is that the result had to be expressed in native TUT format (with a simple final conversion – see 4 below). This format is the same as described in §2, i.e. the output format of the parser. This means that the parser, its rules, and its architecture were tested and refined along the years to comply with this format. The main extensions that were needed to get the Evalita required results are:

1. Modifiers (RMOD), auxiliaries (AUX) and coordinations (COORD) lack, in the parser output, the pseudo-semantic component (e.g. LOC for modifiers, PASSIVE for auxiliaries, etc.), so specific disambiguation rules were added.
2. Traces, that appear in TUT, but not in Evalita, were removed. This is a rather easy task, apart from the “about which in” case mentioned above, where the entire structure had to be revised.
3. The lines were re-numbered. This is because in compound forms (e.g. prepositions with article), TUT assigns a single line number (with an index for the second component), while CoNLL assigns different line numbers.
4. The TUT format was translated into the CoNLL format (this is just an output format conversion that did not require any special processing).

All changes listed above are quite error-free, apart from the first one; but also in this case, the involved lexical items and the syntactic structures can keep the number of errors low.

4.2 The Pilot Subtask

The syntactic structures used in the Pilot Subtask differ considerably from those in the TUT Treebank. Since the sequence of steps that lead to the final result consists in a standard TUT format parsing, and then the conversion of that format into the CoNLL-ISST one, we shortly list the main changes that had to be carried out.

1. The continuations used for multi-words and names involving more than one proper name (e.g. a sequence name-surname), that in TUT have a right-branching structure, had to be modified.
2. The determiners had to be moved from the position of head of nominal structures to the position of dependent of the noun.

3. The coordinated structures were changed from the TUT right-branching organization to the balanced one of CoNLL-ISST.
4. Modals, that in TUT are autonomous verbs, were placed in a position similar to the one of auxiliaries (i.e. as dependents of the related verb).
5. The modifier labels (RMOD) had to be possibly extended with the LOC or TEMP information. Similarly, for auxiliaries and coordination.
6. Clitics got new labels.
7. TUT labels had to be changed into the corresponding CoNLL-ISST labels.
8. The TUT format was translated into the CoNLL format (see 4 in §4.1).

All of this had to be made via the implementation of suitable procedures. It is clear that this is the main disadvantage with respect to machine learning techniques, but it is also clear that this has to be done once for any new format to be covered. This work, for Evalita, required approximately 20 person-days of work, but certainly something has been left out[1], as the discussion in the following section will show.

4.3 Comparison between the two subtasks

In this section, we discuss the results on the basis of the evaluation procedures we wrote, since they enable us to make a detailed analysis of the error types. For the Pilot Subtask, the overall results differ very slightly from the official ones: there are 25 discrepancies which were considered errors in the official evaluation referring to dependency labels which do not exist in the CoNLL-ISST tagset), so that we assume that the detailed analysis reported here is mainly correct. In Table 2, we report the figures of the errors and we try to explain below the great differences in performance.

Table 2. Results on the Parsing Task

	Main Subtask	Pilot Subtask
LAS	596 (11.27%)	1306 (26.06%)
UAS	408 (7.72%)	938 (18.72%)
LS	371 (7.02%)	725 (14.47%)

Punctuation. The most noticeable difference concerns the behavior with punctuation. In the Main Subtask, there were 111 punctuation errors (19.34% of all errors), while in the Pilot Subtask they were 363 (27.80%). Apart from the percentage, the absolute values are relevant, since the total LAS errors, apart from punctuation, are reduced to 485 (Main) and 943 (Pilot). The reason for this probably stands in the manual development of rules. The rules the govern punctuation are less linguistically settled than for other items, so that they are much more system-dependent, in the sense that they are based on local decisions. So, the Turin parser “knows” the TUT rules very well, while the manual development of the corresponding rules for the ISST data would require much more time and effort than the one available. In this case, machine learning procedures are more suitable, since they always learn the “local” rules.

Coordination. Similar considerations can be made for coordination. In this case, we have 125 errors in Main, and 230 errors in Pilot. Although the punctuation and

[1] For instance, in the delivered results, there still are 17 occurrences of the labels RMOD and PRON-RMOD, that do not exist in the required annotation format.

coordination errors overlap considerably (31 common errors in Main, and 83 in Pilot), if we disregard both coordination and punctuation, 391 errors are left in the Main Subtask and 796 in the Pilot Subtask.
Gold data errors. We found a number of errors in the Gold Pilot dataset. We reviewed around one half of the items, and we estimate that there are approximately 120 errors due to problems in the Gold dataset, and 90 cases of different constructions, where we prefer the Turin parser choice. An example of error is an occurrence of the verb "to have" not marked as an auxiliary. (this is a single Gold error, but it produces 4 parsing errors because of wrong attachments). An example of disagreement is "ed è pronta a segnalare" (and is ready to signal), where the Gold links "a segnalare" to the verb, while the parser links it to "pronta". This error analysis reduces the number of 'residual' errors to around 590.
Colons. There are 20 cases of colons and semicolons that do not separate autonomous sentences, but are taken as separators inside a sentence. On the contrary, the parser assumes that they always identify the end of a sentence
Semantic specifications. Various complements (COMP) and modifiers (MOD) include the specification TEMP or LOC. The parser makes approximately 120 errors on them, this clearly depends on lack of knowledge about these items, so that some effort is required to extend them (e.g. the one concerning geographic data).

5 Conclusions

We have described here the main features of the rule-based dependency parser of the University of Torino. We believe that the results obtained both in the Main and in the Pilot task are encouraging. In particular, we think that the Pilot task provided us with suggestions about the main modules that need be improved (e.g. coordination).

Although this is not the place for a comparison between rule-based (manually developed) systems and systems based on algorithms that learn from corpora, we believe that Evalita 2009 has shown that they can have similar performances. Still, we think that manual rules can show some advantages with respect to 'hidden' features, as the occurrence of traces, that can have an impact on the subsequent stages of linguistic processing, as semantic interpretation.

References

1. Lesmo, L.: The Turin PoS Tagger at Evalita 2009. In: Proceedings of EVALITA 2009 (2009)
2. Abney, S.P.: Parsing by Chunks. In: Berwick, R.C., Abney, S.P., Tenny, C. (eds.) Principle-Based Parsing: Computation and Psycholinguistics. Kluwer, Dordrecht (1991)
3. Tsuruoka Y., Jun'ichi Tsujii J.: Chunk Parsing Revisited. In: Proceedings of Ninth International Workshop on Parsing Technologies, pp. 133--140 (2005)
4. HOPS Project homepage, http://www.bcn.es/hops/it/index.htm
5. ICT4LAW project homepage, http://www.ict4law.org/
6. TOCAI project homepage, http://www.dis.uniroma1.it/~tocai/
7. ATLAS project homepage, http://www.atlas.polito.it
8. Turin University Treebank (TUT) homepage, http://www.di.unito.it/~tutreeb/
9. Lesmo L., Lombardo V.: Transformed Subcategorization Frames in Chunk Parsing. In: Proceedings of 3rd International Conference on Language Resources and Evaluation (LREC), pp. 512--519 (2002)