Automatic assignment of grammatical relations

Leonardo Lesmo*§ and Vincenzo Lombardo§**

*Dipartimento di Informatica Università di Torino
c.so Svizzera, 185, 10149 Torino, Italy
*DISTA – Università del Piemonte Orientale "A. Avogadro"
c.so Borsalino 54, 15100 Alessandria, Italy
§Centro di Scienza Cognitiva – Università di Torino
via Lagrange 3, 10123 Torino, Italy
{lesmo/vincenzo}@di.unito.it

Abstract

This paper presents a method for the assignment of grammatical relation labels in a sentence structure. The method has been implemented in the software tool AGRA (Automatic Grammatical Relation Assigner), which is part of a project for the development of a treebank of Italian sentences, and a knowledge base of Italian subcategorization frames. The annotation schema implements a notion of underspecification, that arranges grammatical relations from generic to specific one onto a hierarchy; the software tool works with hand-coded rules, which apply heuristic knowledge (on syntactic and semantic cues) to distinguish between complements and modifiers.

1. Introduction

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).

Existing annotation schemata roughly rely on two basic syntactic paradigms: phrase structure grammars and dependency grammars. The project described in this paper adopts a dependency-based annotation schema, which immediately recalls the notion of predicate-argument structure. Dependency formats seem to be specially adequate for non-configurational languages, for example free-word order languages (cf. Skut et al. 1997). Italian can be classified as a partially configurational language, where SVO order is only the preferred arrangement when a sentence is interpreted in isolation; all the other 5 permutations can often occur in naturally occurring texts[1].

The major advantage of including predicate-argument relations into the annotation schema is the possibility of having an accurate description of the roles played by the syntactic units in the sentence structure. However, this descriptive richness can produce a large amount of ambiguity. The selection of the correct set of grammatical relations occurring in a sentence can become a relevant workload for both an automatic parser and a human annotator. Therefore, most researchers have devised software modules specialized in the assignment of grammatical relations (GRs). These modules usually rely on stochastic methods (Brants et al. 1997) and/or machine learning techniques (Buchholz et al. 1999) (Ferro et al. 1999). A common characteristics of all the approaches is to keep low the number of GR labels (around a dozen): this satisfies some trade-off between the accuracy of the syntactic description and the tractability of the assignment task.

The solution proposed in this paper is to realize this trade-off through a flexible annotation schema, where syntactic categories and grammatical relations are arranged on a hierarchy from generic to specific ones, implementing a notion of underspecification. When specialization is possible, the parser and/or the annotator descend the hierarchy to assign specific GR labels; otherwise, they stop at some higher point in the hierarchy, where relation labels cover a larger number of cases. The software module for GR assignment (called AGRA, Automatic Grammatical Relations Assigner) implements a number of heuristics that take into account the hierarchical organization. Heuristics are expressed as condition-action rules, which realize the mapping between GR labels and the syntactic structures of arguments. The rules are hand-coded, and incrementally updated through cycles of rule application and manual error analysis on a set of sentences used for training. As far as we know, this is the first attempt of building such a module for Italian, a partially configurational language where GR assignment depends on a number of factors, like word order, case suffixes and semantics.

The paper is organized as follows: in the next section we outline the annotation schema, with some examples; then we describe AGRA, with the hand-coded rules and the search engine for rule application; finally we illustrate the cycles of rule application and error analysis, with some numerical results and the comparison with other approaches.

[1] Note that we do not know how often, since this is one of the goals for constructing a treebank including predicate-argument structures.

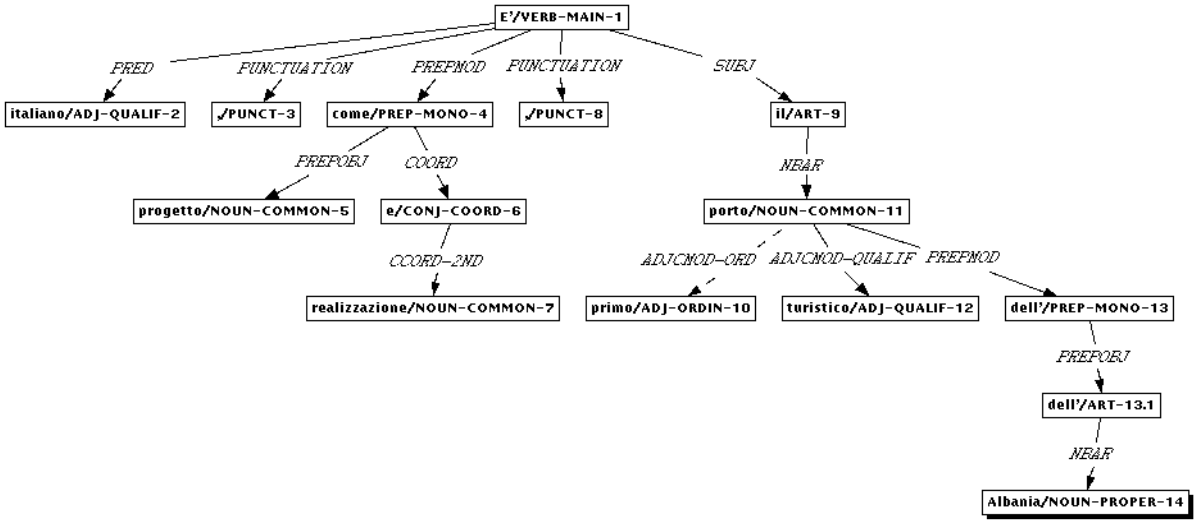

Figure 1. Dependency tree of the sentence "E' italiano, come progetto e realizzazione, il primo porto turistico dell'Albania" (*It is Italian, as for project and realization, the first tourist port in Albania*). This figure, which is in the interface format of the annotation tool, is realized with the DaVinci program.

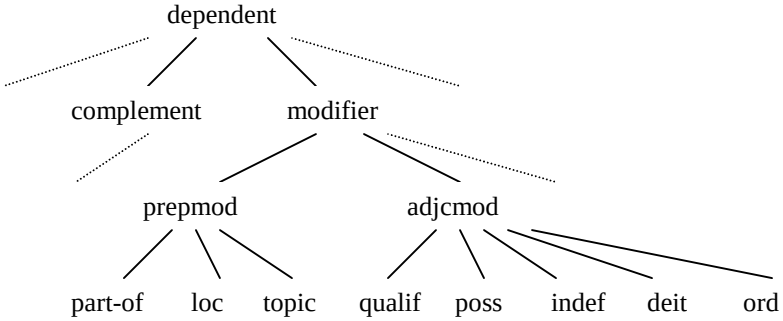

Figure 2. A partial view of the hierarchy of grammatical relations: only adjectival and prepositional modifiers.

2. An annotation schema for an Italian treebank

A dependency-based annotation schema permits a perspicuous representation of Italian sentences, which feature a non-configurationality at the verbal level (this means that the dependents of a verb are loosely constrained in the linear order). Subject-Verb-Complement (a generalization of SVO) is the most likely order, but the other five permutations may occur as well. Consider the following examples :

[Le autorità di Tirana]$_{SUBJ}$ [hanno scelto]$_{VERB}$ [il progetto....]$_{COMP}$

"The authorities of Tirana have chosen the project ... "

[E']$_{VERB}$ [italiano]$_{COMP}$, [come progetto e realizzazione]$_{MOD}$, [il primo porto turistico dell'Albania]$_{SUBJ}$

"It is Italian, as for design and realization, the first tourist port of Albania"

Anche [sull'Albania]$_{COMP}$ [soffia]$_{VERB}$ [il vento della protesta]$_{SUBJ}$.

"The wind of protest blows over Albania too"

In fig. 1 there is a dependency tree from the treebank (for the second sentence).

The characteristics of non-configurationality make Italian a language for which the assignment of grammatical relations is particularly relevant. Grammatical relations can exhibit various degrees of specificity with respect to a number of properties of the relation arguments: syntactic category, semantic type, several morpho-syntactic features. However, GR assignment becomes a very difficult task, as the variety and the specificity of GR labels increases. In order to reach a valuable trade-off between richness of description and tractability of assignment (being performed by a human annotator or an automatic assigner) the GR system proposed in this paper offers a high degree of flexibility. Flexibility is related to the fact that dependency relations are organized hierarchically (from generic to specific ones (Fig. 2)). In the dependency literature, a similar system is proposed by (Hudson, 1990), who extends the hierarchical organization to all the grammar elements. For example, the relation between a noun (head) and an adjective (dependent) is an adjectival modification (adjcmod) that can be further specified on the basis of the adjective features (qualificative, possessive, indefinite, …). When

GR assignment becomes a hard ambiguity problem, the assigner can generalize to some grammatical relation selected from the upper levels of the hierarchy.

In the adjectival part of the hierarchy, shown in figure 2, the feature information in the POS tag can help in selecting the correct grammatical relation (it depends on the type of adjective). See, for example, the POS tags of the adjectives "primo" (first) and "turistico" (tourist) in figure 1. But this is not always the case. In fact, the situation is much more intricate for prepositional modifiers. Again, in figure 1, it is hard to decide the specific grammatical relation of the prepositional modifier "come progetto e realizzazione" (as for project and realization): in this case, the annotator can select a more general (see figure 2) prepositional modifier (prepmod).

Penn Treebank researchers adopt a similar solution in the phrase-structure format when an annotator is sure that a sequence of words is a certain major constituent but cannot decide its syntactic category. The solution is to introduce a generic constituent label X (Marcus et al. 1993). This approach corresponds to the application of a two-level underspecification mechanism: the annotator has only one possible recover for annotation uncertainty. In our approach, the annotator can use multiple degrees of abstraction.

Now we turn to the automatic GR assigner (AGRA).

3. The software tool for the assignment of grammatical relations (AGRA)

The assigner works on a data structure representing an unlabelled dependency tree. In other words, the input specifies, for each node, its syntactic dependents. So, the task of the assigner is to perform a *match* between the set of actual dependents and the possible dependents[2]. A knowledge base of rules for the assignment of GR label (described below) determines the number and the relations of the possible dependents.

The match is performed in different ways, according to the syntactic category of the governor. For all categories, except verbs, the match is performed on single links. So, in case the governor is a noun, AGRA inspects each dependent, and applies the assignment rules to the pair governor-dependent in order to find out the correct label. It is clear that this approach overgeneralizes: nothing prevents from having multiple links with the same label, without any contextual check of the mutual compatibility. However, since the assigner is run on real text, this situation can hardly arise[3].

With respect to verbal labels, the assignment is based on Verbal Subcategorization Patterns (VSPs). These can be seen as classes of verbs: for each class, the VSP lists the set of obligatory grammatical dependents (complements). Here, the matching process is more complex. In fact, it has to be checked that all complements appearing in the VSP are present without repetitions in the input pattern. This task interacts with the presence of adjuncts: both an adjunct and a required complement have the same surface realization. Consider, for instance:

- *Questa domenica l'ho passata proprio bene*
  (This sunday it [I] have spent really well:
  I have spent this sunday really well)
- *Questa domenica l'ho vista proprio bene*
  (This sunday her [I] have seen really well:
  This sunday, I saw her really well)

In this case, *This sunday* plays the role of direct object of the verb *to spend* in the first sentence (with *l'* as a reinforcement clitic), while it plays the role of a temporal adjunct (with *l'* acting as true direct object) in the second sentence.

Moreover, surface realizations could be altered in the input as a consequence of several phenomena: passives, the pro-drop facility of Italian language, movements (like raising or equi). So, the matching process is augmented with a set of transformations, which enable the assigner to obtain a set of derived VSPs from the basic VSPs (subcategorization classes).

One of the main task of this project is to devise a set of wide-coverage VSPs and transformations, so that the task of the lexicographer is reduced to find the right class(es), or VSPs, of a given verb (see section 4). In the next subsections we describe the assignment rules, split in non-verbal plus adjunct rules (operating on individual links and do not take VSPs into account) and verbal rules (matching the set of dependents of a node with some VSP in the knowledge base).

3.1 Non-verbal rules and adjuncts

These rules are implemented as condition-action rules, with the pair *<head-category, dependent-category>* acting as an access key. So, for instance, we have:

<noun, adj>: noun-adjr$_1$*, noun-adjr*$_2$*, ..., noun-adjr*$_n$

Each rule (*noun-adjr*$_i$, in the example) has the form:

if condition **then** label

A summary of the specific labels is in section 4. The condition part may take into account many pieces of information, some of which are discussed below:

a. The **syntactic subtype** (subcategory) of the head or the dependent: as said in the previous paragraph, if the adjective has subcategory *poss* (possessive), then the label will be *adjcmod-poss*.
b. The **actual words**: e.g. in case the connection is *<noun, prep>*, the governor word is *opinione* ("opinion"), and the dependent preposition is *su* ("on"), then the link is labeled as *prepmod-topic*
c. The **semantic type** of the involved elements: in case of *<noun, prep>*, the preposition is *di* ("by"), and the governor noun has the semantic category *£auth-work*[4]

[2] The unlabelled trees are the result of two tools developed previously: a POS tagger and an interactive parser. Both of them work on unrestricted texts. The tagger works left-to-right in a single step and is based on condition-action rules (Boella & Lesmo 1998). The interactive parser takes in input the POS-tagged sentence and produces a dependency parse tree, by proposing graphically tentative parse trees to the human annotator while proceeding incrementally from left to right (Lombardo et al. 1999).

[3] But it can happen that the presence of a link could help to disambiguate another link, which, in the current implementation, cannot be accomplished

[4] Throughout the paper, semantic types are strings in italics prefixed by the symbol £.

(e.g. *sinfonia*, *romanzo*: "symphony", "novel"), then the assigned label will be *prepmod-author*. Notice that it is not part of the present project to develop a taxonomy of semantic classes. However, the use of semantic classes stresses the fact the no proper labeling can be obtained without such knowledge.

d. **Complex downward paths**. This enables us to treat some flexible types of locutions. In the example *di corsa* ("of run" – "quickly"), we can have a specification where a verb (remember that these rules are used also for verbal adjuncts) governs *di* ("of": preposition), which, in turn, governs the word *corsa* ("run"). The locution reading is unaffected by the presence of a modifier (e.g., an adjective, *di gran corsa* – "of great run" – "very quickly").
e. **Deverbalized nouns** as governors. In these cases (the *fall* of, the *recognition* of), a subject or object label (called, distinctively, *n-subject* and *n-object*) can be assigned on the basis of the transitivity of the original verb.

Three rules are reported below. The first one is a rather standard rule. It specifies that if an adjectival dependent of a noun is of *ordinal* type, then the label must be *adjcmod-ordin* (it applies to "for the fourth time" – the two words underlined are the arguments of the relation). The second rule involves a semtype and a path of length 2: if the preposition governed by a noun is *di* ("of"), and the word governed by *di* is of semantic type *£time*, then the noun-prep label must be *prepmod-time* (this applies to *l'intervista di sabato* – "the saturday interview"). The third rule refers to a multi-word locution, *a causa di* ("because of"). If these three words form a chain of nodes in the dependency tree (a path of length 3), then the topmost link (linking the preposition *a*) must be labeled as *adjunct-reason-cause* (the other links are labeled as *locution-cont*, i.e. continuation of a locution, via other rules).

```
(noun-adj7     ((down (type ordin))) adjcmod-ordin)

(noun-prep-di20
          ((down (word di)
             (down (semtype £time))))    prepmod-time)

(verb-prep-a3
         ((down (word a)
                (down (word causa)
                      (down (word di)))))
          adjunct-reason-cause)
```

Box 1 - Examples of non-verbal rules

3.2 Verbal rules and transformations

A basic verbal subcat pattern (VSP) has the following form (three components):

(subcat-pattern-name
surface-realization
labels)

The *subcat-pattern-name* is any string: *trans*, *intrans*, *trans-dest* are examples of names.

Surface-realization and labels are two lists of the same length, where there is a correspondence between items in the same position. Items of the list *labels* are GR labels; items in the list *surface-realization* constrain the syntactic structure of the dependent fulfilling the grammatical relation which occupies the same position in the list *labels*. *Surface-realization* is described as follows:

(surf-descr$_1$ surf-descr$_2$ ... surf-descr$_m$)

In turn, each *surf-descr$_i$* has either the form

<category>

or the form

(<category> restrictions)

where the restrictions can be used to enforce any of the constraints described above for non-verbal rules (e.g. actual words or semtypes). An example of a VSP is reported in Box 2.

```
(trans-dest
        (( (noun (agree))
           (art (agree))
           (adj (semtype £geogr) (number pl) (agree))
           (pron (case nom) (agree)))
         ( noun
           art
           (pron (not (type refl-impers)) (case acc))
           (adj (semtype £geogr) (number pl)))
        ( (prep (word a) (down (semtype £city)))
          (prep (word in) (down (semtype £place)
             (not (semtype £city)))))
       (subject object to-loc))
```

Box 2 - Examples of a VSP

It specifies that the class *trans-dest* applies to verbs requiring three dependents, which will be labelled as *subject*, *object*, and *to-loc* (see the *labels* component in the last line). Moreover, the subject can be a noun, an article (remember that determiners govern nouns), an adjective (geographical, as *Italians*) or a pronoun; all of them, in order to act as subject must agree with the governor. Moreover, in case the subject is a pronoun, it must appear in the nominative (*nom*) case. The object is similar (without agreement). The destination must be realized as the preposition *a* (to) governing a word of *£city* semtype or as the preposition *in* (in) governing any place other than a £city. The pattern applies to verbs as *portare* (to bring), *inviare* (to send), etc.

Transformations are rules converting a VSP into another VSP. The resulting VSP is called *derived VSP*. The input VSP can be either a *basic* (not transformed) or a derived VSP. Some transformations can be applied to all patterns (e.g. pro-drop), while others apply only to a given subset of them (e.g. passivization applies only to the various transitive VSP). The possibility of applying a transformation to a derived VSP enables the system to obtain patterns such as, for example, passive infinitivized, which applies to *di essere visto da lui* (to be seen by him), where the subject is absent (infinitivization) and an agent complement can be present (passivization). After the application of the transformations, it is assumed that it is available a complete specification of all sets of dependents (complements) which can appear below a given verb. For instance, if the original definition of the verb *amare* (to

love) includes only the *trans* VSP, after the application of transformations, it includes *trans*, *trans+ passivization*, *trans+pro-drop*, *trans+infinitivization*, *trans+ infinitivization+passivization*, etc. (a total of 16 patterns derived from *trans*). Now, when an occurrence of the verb *amare* is found in a sentence, the matching process is applied to all derived patterns. Again, adjuncts may cause problems, but the match is performed on the derived patterns exactly as on the basic patterns. Of course, all patterns include some applicability conditions (depending by the applied transformations), so that for instance, *trans+infinitivization* only applies to verbs in the infinite mood (conversely, the basic *trans* does not match a verb in the infinite mood).

The matching process is based on the VSP of the verb. In Box 3 below, we show the basic matching procedure.

```
procedure VSPMatch (Verb, InputDep)
begin
 /* Extract from dictionary all basic VSP's of 'Verb' */
   VC := VerbClasses (Verb);
 /* Extract from DB the definitions of all VSP's (basic and
   derived) associated with VC. This also checks the applic-
   ability conditions (e.g. infinite form of 'Verb' for derived
   VSP's  obtained via infinitivization) */
   VSP := GetVSPComp (VC);
   Found := false;
   lev := 0;
/* External loop for all levels (number of applied transform-
    ations). MaxLevel currently set to 5. Do not inspect level
   i+1 if solution found at level i */
   while not Found and lev <= MaxLevel do
    begin
      NextVSP := first (VSP, lev);
      BestLabels := empty;
/* Internal loop for all VSP's at same level. The loop is not
   exited when a solution is found: all VSP's are tried in order
   to find out the one with less adjuncts */
      while NextVSP is not null do
        begin
          CaseLabels := SingleMatch (NextVSP, InputDep);
          if CaseLabels <> empty then
            Found := true;
            if Adjuncts (CaseLabels) < Adjuncts (BestLabels)
            then BestLabels := CaseLabels;
            nextVSP := next (VSP, lev)
        end;
     lev := lev + 1;
end;
```

Box 3 - The procedure which, given a Verb and its dependents, finds a match with the contents of the KB.

First, all the basic subcategorization patterns (VSP names) of the verb are retrieved, and then the components of the VSP (both basic and derived) are extracted. The VSP's are sorted according to how basic they are: basic VSP's get a level of 0, VSP's obtained via a single transformation the level 1, VSP's obtained via 2 transformations the level 2, and so on. The VSP's with a lower level are tried first: this corresponds to adopting a preference criterion where more basic VSP's are preferred. In case a VSP of level k matches the input case frame, then all other VSP's at the same level are tried, in order to find the one that matches with a minimum number of adjuncts; this is a weaker preference: at the same level (i.e. the same number of transformations applied) complements are preferred over adjuncts. In case no match is found, all links concerning dependents of the verb are labelled as *Unknown*.

In Box 4, we report the function for matching a single VSP against the actual dependents; this is carried out by first looking for any match between all surface realizations of the cases in the VSP and one of the actual dependents. The function is implemented as a depth-first search, where each match between a complement and an actual dependent is a choice point. This implies that:

- All complements are obligatory
- No order is implicit in the VSP (but some order constraints can be expressed explicitly, if any)

In case all the required complements are found, the remaining dependents are interpreted by using the rules for adjuncts. For each of them, at least one interpretation as an adjunct must be found, otherwise the whole match fails.

```
/* InputDep is a Marked list of actual dependents; the marked
    dependents are the ones which have already been matched
    with a complement at some higher level of recursion */
function SingleMatch (Verb, VSP, InputDep)
begin
  if VSP = empty
/* if all complements have been found, then try to match the
   remaining actual dependents with adjuncts */
  then SingleMatch:= CompleteWithAdjuncts(Verb, InputDep)
  else
    begin
      Found := false;
      NextCompl := first (VSP);
      NextInpDep := first-unmarked (InputDep);
/* take the next element of the VSP (the first one, at this level
   of recursion); initialize the loop on actual dependents */
      while not Found and not null NextInpDep do
      begin
/* if the actual dependent satisfies the surface description of
   the first element of the VSP, then ElemMatch succeeds */
        firstMatch := ElemMatch (NextCompl, NextInpDep)
        if firstMatch <> fail then
          begin
/* and the recursion proceeds on the remaining complements
   (after having marked the Input Dependent to signal that it
    has already been used in the match)  */
            result := SingleMatch (rest (VSP),
                                   Mark (InputDep, NextInpDep))
            if result <> fail then
/* if the match succeeds on the remaining portion of the VSP,
   then the task is accomplished */
              begin
                Found := true;
                SingleMatch := append (firstMatch, result)
              end
/* the first element matches, but the remaining ones do not; try
   another possibility with the next Input Dep */
          else NextInpDep := Next (InputDep)
        end   /* firstMatch <> fail */
/* the first element does not match: try another possibility
   with the next Input Dep */
       else NextInpDep := Next (InputDep)
     end   /* while */
  end;
```

Box 4 - The function for matching a single VSP against a set of input dependents.

4. Incremental methodology and preliminary results

The methodology for the development of the AGRA program is a learning cycle that alternates automatic assignments based on the current KB and manual updates of the current KB. It must be clear that, differently from most existing approaches, the various KB's are built *manually*. The role of the automatic assigner, in fact, is to test the coverage of the KB's. The (manual) *learning* cycle is carried out as follows:

1. Apply the AGRA to a corpus of unlabelled trees (*learning set*) using a *bootstrapping* KB.
2. Inspect the result of the assignment and detect the assignment errors.
3. Update manually the KB, in order to correct (most of) the detected errors. This update can produce the introduction of new non-verbal rules, new basic VSP's, new assignments of verbs to a VSP, and (rarely) new transformations.
4. Apply the updated KB to the initial corpus in order to check the correctness of the newly introduced rules, and to verify which of the original errors have been corrected.
5. Apply the rules to a set of sentences (*test set*) other than the original ones, in order to evaluate the coverage of the KB on unseen data.
6. Extend the learning set (new sentences are added) and go to step 2.

It is usually claimed that manual approaches are labour-intensive and error-prone. So, most of the methods described in the literature involve learning programs applied to large corpora (e.g., the Penn Treebank). Although we agree on the big effort required to apply an approach as the one described above, we must note that any automatic method must exploit a large corpus, a resource which is currently missing for Italian. On the contrary, our method starts from a small hand-built unlabeled treebank, and the result (instead of a starting point) is a (small) labeled treebank for Italian.

In the bootstrapping phase, we have manually built an initial kernel of the three components (excluding semtypes) from general linguistic knowledge. This initial KB included 10 subcat frames, 3 transformations (passivization, pro-drop rule, infinitivization), 30 modifier relations.

At the current stage of development, we have run three learning cycles; the error rates are reported in Table 1, while the size of the KB is described in Table 2. It has just to be noted that the 192 labels used after cycle 3 are not the same as in the previous row. Some new labels have been introduced, but a more rational scheme has been devised, so that the total number remained unchanged. To provide the reader with a feeling of the label types, we report in Table 3 the 28 complement labels (which would require some discussion, that we omit for space reasons). The remaining 164 labels (192 minus 28) are adjuncts and non-verbal GRs.

Learning cycle	Training Set Sentences (words)	Test Set Sentences (words)	ERRORS Training Set	ERRORS Test Set
1	200 (4610)	50 (1063)	-	252 (23.66%)
2	500 (11809)	50 (1063)	1376 (11.65%)	187 (17.59%)
3	800 (18677)	100 (2378)	1693 (9.07%)	374 (15.73%)

Table 1 - Experimental results: Errors in label assignment in the three cycles of development. The errors on the training set after the first cycle were not evaluated.

Learning Cycle	Non-verbal + adjuncts labels	Non-verbal + adjuncts rules	Basic VSP labels	Basic VSP classes	Basic VSP verbs	Transformations	Semtype infos
1	30	30	12	10	42	3	0
2	192	264	20	44	122	11	274
3	192	389	28	64	167	12	360

Table 2 - Size of the various components of the KB after the different learning cycles

Standard	Predicative	Sentential	Miscellaneous	To be	Locutions
subject	pred-compl	scompl-obj	topic	time-spec	vadv-neg
object		scompl-ynobj	theme	pmod-possess	locut-prep
ind-obj		scompl-qobj	result	compl-compar	locut-pred
second-obj		scompl-cause	to-loc	num-eval	locut-object
empty-rel		modal-sub	in-loc	manner-spec	locut-adv
			su-loc	manner	

Table 3 - Complement labels after cycle 3

4.1. Error Analysis

There appears to be a rather high number of errors within the learning set. This is due to:

1. Difficulty to decide the correct (semantic) label for some relations (as for *con* in *in contraddizione con* (in contradiction with), or *di* in *periodo di attesa* (period of latency)
2. Difficulty to decide the correct criteria enabling the system to choose the (known) correct label (as in *gruppi di livello adeguato* - groups of adequate level - where *gruppi di* - groups of - should be labeled as *prepmod-eval*).
3. Difficulty to decide the correct criteria for assigning verbal labels (*rivoluzioni che [object] non si [impers-subject] sono mai viste* - revolutions which nobody has ever seen - vs. *ragazze che [subject] non si [object] sono mai viste* - girls who never saw each other)
4. Difficulties for handling ellipses, which currently receive a very poor treatment.

In any case, most of the errors are due to the goal of assigning semantic labels. In case this is not required (so that, for instance, a *prepmod* label is always taken as correct, independently of its semantic interpretation) the error rate drops considerably.

Learning Cycle	Words	Total errors	Overspec errors	VSP match errors	Ellipsis errors	Others
3	2378	374 (15.73%)	228 (9.59%)	94 (3.95%)	17 (0.72%)	35 (1.47%)

Table 4 - Distribution of error types

As can be seen in table 4, among the 374 errors in the test set of the last stage, 228 were due to the (semantic) over-specification of non-verbal labels (i.e. 61% of the total number of errors). This means that, if we disregard labels as *prepmod-origin*, or *adjunct-time*, by labeling the arcs simply as *prepmod* or *adjunct*, the labeling errors are just 146 (i.e. 6.14% of the total).

Among them, most errors are due to the match of verbal case frames (3.95%). There are two major sources of these errors. The first one is the presence of particular sets of dependents for which rather specific (and complex) rules have to be devised. Some examples of this type of situation were reported earlier (see section 3 and the four points above); three more are presented below:

a) *Il 95% dei bambini vedono la luce …*
The$_{sing}$ 95% of the babies see$_{pl}$ the light (i.e. were born) …
(here, the agreement constraints between the actual subject and the verb are not respected)

b) *I tiranni noi li vogliamo vedere …*
The tyrants we them want to see
(here, there is a raising of *li* (them) from the governed 'to see' to the governing modal 'want'. This is done for the true object 'the tyrants', but not for the enforcing clitic 'them')

c) *In molti scommettono sulla vittoria …*
In many$_{subj}$ bet on the victory
(here, the surface form of the subject - i.e. a PP governed by 'in' - is not admitted in the VSP).

Of course, the VSP's can be extended suitably, but this cannot be done without verifying that the noise introduced by such extensions does not affect negatively the overall error ratio.

The second source of errors are relative clauses, for which we currently lack both agreement with the noun governing the clause (i.e. agreement is enforced with the relative pronoun *che* - who, which, that -, which does not carry any number or gender information), and ordering constraints (so that if the relative pronoun is followed by another NP - i.e. a nominal dependent - the pronoun is taken as subject and the other NP is taken as object, which is usually wrong). Again, some of these errors can be avoided by introducing suitable tools (feature migration and more ordering constraints).

4.2. Related work

It seems reasonable to state that, given the above observations, the final results are comparable with the ones reported in the literature, although comparisons are hard because of the difference in methodology and goals. For instance, (Buchholz 1998) reports a success rate between 91.6% and 94.8%. But her task is rather different: she aims at classifying as complement or adjunct a dependent according to the surrounding context. So, for instance, a *to-loc* (i.e. a complement) classified as an *adjunct-loc-to* (i.e. an adjunct) counts as an error both for her and for us, but an exchange between *subject* and *object* is an error in our case, but it can hardly arise in Buchholz's system, where labels are already provided by the Penn treebank annotation scheme. On the contrary, we assume the existence of VSP, whereas subcategorization infos can be absent in Buchholz's experiments (without them, she gets 91.6% of correct labels, while we could not obtain anything).

In a subsequent experiment, Buchholz, Veenstra, and Daelemans (Buchholz et al. 1999) tried to determine the impact of enriched input on the performances of a Grammatical Relation Assigner. Although in this case the task is much more similar to our experiments, it has to be noted that their results refer just to Verbal Dependents (i.e. to the more difficult part of the task). Our figures should be compared with the ones reported in the sixth row of table 5 in (Buchholz et al. 1999), labelled as "*GR with all chunks, without ADVFUNC label on perfect test data*". But the availability of chunks does not mean that the connection with the verb is available (our unlabelled arcs), so we have more information available. On the other hand, the number of complement labels on which they run the test is much smaller than ours, and, again, they have the Penn treebank as a reference in their supervised learning task. Their results are 77.2% correct assignments; in our

case, we must refer just to verbs. In the test set, there are 355 main verbs (excluding auxiliaries), and the 94 VSP match errors correspond to 68 wrong VSP identifications (if a VSP involves more than one dependent, more than one labelling error is induced by a single error in VSP identification). This corresponds to 19.1% of errors on VSP analysis, which approaches Buchholz et al.'s results.

Also (Brants & Skut 1997) have faced the problem of grammatical relation assignment in the context of interactive text annotation. In particular, what they call "degree 1 of automation" corresponds rather closely to AGRA, since in level 1 "the user determines phrase boundaries and syntactic categories (S, NP, VP, …). The program automatically assigns grammatical functions" (Brants & Skut 1997). Their approach to KB development is based on statistical learning of a Markov trigram model. The assignment results are similar to ours, since the authors report an overall success rate of 94.2% (table 1), which should be compared with 6.14% of errors we found with the reduced tagset. Notice, in fact, that the complete tagset adopted in (Brants & Skut 1997) includes just 17 tags.

5. Conclusions

This paper has presented an approach the assignment of grammatical relations. This work is part of a project for a treebank development through the introduction of a flexible annotation schema, where the software tool for GR assignment is a specialized module.

The annotation schema is centered upon predicate-argument structures expressed by subcategorization frames, that are essentially sets of grammatical relations (also called VSP). The reference syntactic paradigm is dependency grammar. Grammatical relations are arranged on a hierarchy from generic to specific ones: parsers and human annotators can employ underspecified elements when there is uncertainty upon specific grammatical relations.

The assignment of grammatical relations is realized by the module AGRA, that works with hand-coded heuristic rules, that apply syntactic and semantic preference constraints.

Preliminary results are comparable with other approaches, when considered in face of the high number of GR labels, which allow a very rich corpus annotation.

6. References

Boella G., Lesmo L., (1998) Automatic Refinement of Linguistic Rules for Tagging. In Proceedings of 1st International Conference on Language Resources and Evaluation (LREC 98), Granada, pp.923-930.

Brants, T., Skut, W., Uszkoreit, H., (1999) Syntactic annotation of a German newspaper corpus. In Proceedings of Treebanks workshop - Journées ATALA sur les corpus annotés pour la syntaxe, 18-19 juin 1999, Paris, pp.69-76.

Brants T., Skut W., (1998) Automation of Treebank Annotation. In Proceedings of the Conference on Methods in Language Processing (NeMLaP-3) January 14-17, 1998, Sidney, Australia, pp.49-58.

Brants T., Skut W., Krenn B., (1997) Tagging Grammatical Functions. In Proceedings of EMNLP-97, 1997, Providence, RI, USA, pp.64-74.

Buchholz S., (1998) Distinguishing Complements from Adjuncts using Memory-based Learning. Proc. ESSLLI-98 Workshop on Automated Acquisition of Syntax and Parsing, Saarbruecken, Germany.

Buchholz S., Veenstra J., Daelemans W.: Cascaded Grammatical Relation Assignment. EMNLP/VLC-99, University of Maryland, USA, June 21-22, 1999. CL/9906004.

Collins M.: Three Generative, Lexicalized Models for Statistical Parsing. In Proceedings of the 35th Meeting of the Association for Computational Linguistics (ACL 97) (1997), 16-23.

Ferro L., Vilain M., Yeh A.: Learning Transformation Rules to find Grammatical Relations. Workshop on Computational Natural Language Learning (GNLL-99) (1999), 43-52.

Hajic J., (1998) Building a Syntactically Annotated Corpus: the Prague Dependency Treebank In Issues of Valency and Meaning, Karolinum, Praha, pp.106-132.

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

Lombardo V., Lesmo L., (in press) A formal theory of dependency syntax with non-lexical units. To appear in Traitement Automatique des Langues.

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

Marcus M.P., Santorini B., Marcinkiewicz M.A., (1993) Building a Large Annotated Corpus of English: The Penn Treebank, Computational Linguistics, 19, pp.313-330.

Marcus M.P., Kim G., Marcinkiewicz M.A., et al., (1994) The Penn Treebank: Annotating Predicate Argument Structure. In Proceedings of The Human Language Technology Workshop, San Francisco, Morgan-Kaufmann.

Moreno, A., Lopez, S., (1999) Developing a Spanish treebank. In Proceedings of Treebanks workshop - Journées ATALA sur les corpus annotés pour la syntaxe, 18-19 juin 1999, Paris, pp.51-58.

Skut W., Krenn B., Brants T., Uszkoreit H., (1997) An Annotation Scheme for Free Word Order Languages. In Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP), Washington, D.C..

Skut W., Brants T., Krenn B., Uszkoreit H., (1998) A Linguistically Interpreted Corpus of German in Newspaper Texts. In Proceedings of 1st International Conference on Language Resources and Evolution (LREC 98), Granada, pp.705-713.

Stock, O., (1989) Parsing with Flexibility, Dynamic Strategies, and Idioms in Mind. Computational Linguistics, Vol.15, Num.1, March 1989, pp.1-17.

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