

Theory Refinement and Natural Language Learning

Hervé Déjean*

Seminar für Sprachwissenschaft

Universität Tübingen

dejean@sfs.nphil.uni-tuebingen.de

Abstract

This paper presents a learning system for identifying syntactic structures. This system relies on the use of background knowledge and default values in order to build up an initial grammar and the use of theory refinement in order to improve this grammar. This combination provides a good machine learning framework for Natural Language Learning. We illustrate this point with the presentation of ALLiS, a learning system which generates a regular expression grammar of non-recursive phrases from bracketed corpora.

1 Introduction

Applying Machine Learning techniques to Natural Language Processing is a booming domain of research. One of the reasons is the development of corpora with morpho-syntactic and syntactic annotation (Marcus et al., 1993), (Sampson, 1995). One recent popular subtask is the learning of non-recursive Nouns Phrases (NP) (Ramshaw and Marcus, 1995), (Tjong Kim Sang and Veenstra, 1999), (Muñoz et al., 1999), (Group, 1998), (Cardie and Pierce, 1999), (Buchholz et al., 1999).

When other learning techniques (symbolic or statistical) are widely used in Natural Language Learning, theory refinement (Abecker and Schmid, 1996), (Mooney, 1993) seems to be ignored (except (Brunk and Pazzani, 1995)). Theory refinement consists of improving an existing knowledge base so that it better accords with data. No work using theory refinement applied to the grammar learning paradigm seems to have been developed. We would like to point out in this article the adequacy between theory refinement and Natural Language Learning.

To illustrate this claim, we present ALLiS (Architecture for Learning Linguistic Structures), a learning system which uses theory refinement in order to learn non-recursive noun phrases (also called base noun phrases) and non-recursive verbal phrases. We will show that this technique combined with the

use of default values provides a good architecture to learn natural language structures.

This article is organised as follows: Section 2 gives an overview of theory refinement. Section 3 explains the advantage of combining default values and theory refinement to build a learning system. Section 4 describes the general characteristics of ALLiS, and Section 5 explains the learning algorithm. The evaluation of ALLiS is described Section 6. The examples which illustrate this article correspond to English NPs.

2 Theory Refinement

We present here a brief introduction to theory refinement. For a more detailed presentation, we refer the reader to (Abecker and Schmid, 1996), (Brunk, 1996) or (Ourston and Mooney, 1990). (Mooney, 1993) defines it as:

Theory refinement systems developed in Machine Learning automatically modify a Knowledge Base to render it consistent with a set of classified training examples.

This technique thus consists of improving a given Knowledge Base (here a grammar) on the basis of examples (here a treebank). Some impose to modify the initial knowledge base as little as possible. Applied in conjunction with existing learning techniques (Explanation-Based Learning, Inductive Logic Programming), TR seems to achieve better results than these techniques used alone (Mooney, 1997). Theory refinement is mainly used (and has its origin) in Knowledge Based Systems (KBS) (Craw and Sleeman, 1990). It consists of two main steps:

1. Build a more or less correct grammar *on the basis of background knowledge*.
2. Refine this grammar using training examples:
 - (a) Identify the revision points
 - (b) Correct them

The first step consists of acquiring an initial grammar (or more generally a knowledge base). In this work, the initial grammar is automatically induced

* This research is funded by the TMR network Learning Computational Grammars www.lcg-www.uia.ac.be/lcg/

from a tagged and bracketed corpus. The second step (the refinement) compares the prediction of the initial grammar with the training corpus in order to firstly identify the *revision points*, i.e. points that are not correctly described by the grammar, and secondly, to correct these revision points. The error identification and refinement operations are explained Section 5.3.

The main difference between a TR system and other symbolic learning systems is that a TR system must be able to revise existing rules given to the system as background knowledge. (A system such as TBL (Brill, 1993) can not be considered as TR since it only acquires new rules). In the case of other techniques, new rules are learned in order to improve the general efficiency of the system (selection of the “best rule” according to a preference function) and not in order to correct a specific rule.

3 Theory Refinement, Default values and Natural Language Learning

This section explains how default values combined with theory refinement can provide a good machine learning framework for NLP.

3.1 The Use of Default Values

The use of default values is not new in NLP (Brill, 1993), (Vergne and Giguët, 1998). We can observe that often (but not necessarily) in a language, an element belongs to a predominant class (Vergne and Giguët, 1998). Some systems such as stochastic models use this property implicitly. Some others use it explicitly. For instance, the general principle of the Transformation-Based Learning (Brill, 1993) is to assign to each element its most frequent category, and then to learn transformation rules which correct its initial categorisation. A second example is: the parser described in (Vergne and Giguët, 1998). They first assign to each grammatical word a default category (default tag), and then might modify it thanks to local contexts and grammatical relation assignment (in order to deal with constraints due to long distance relations which can not be expressed by local contexts).

The main work is done by the lexicon and by default values (even if further operations are obviously necessary)

These approaches are thus different for the disambiguation often used in tagging. The default rules are not numerous (one per tag), easy to automatically generate but they nevertheless produce a satisfactory starting level.

3.2 The Combination of Default Values with TR

The idea on which ALLiS relies is the following: a first “naive grammar” is built up using default values, and then TR is used in order to provide a “more

realistic grammar”. This initial grammar assigns to each element its default category (the algorithm is explained in Section 5.2). The rules learned are categorisation rules: *assign a category to an element* (a tag or a word). Since an element is automatically assigned to its default category, the system has not to learn the categorisation rules for its category, and just learns categorisation rules which correspond to cases in which the element does not belong to its default category. This minimised the number of rules that have to be learned. Suppose the element e can belong to several categories (a frequent case). The first rule learned is the “default” rule: *assign(e, dc)*, where dc is the default category of e . Then ALLiS just learns rules for cases where e does not belong to its default category. The numerous rules concerning the default category are replaced by the simple default rule.

4 ALLiS

The goal of ALLiS¹ is to automatically build a regular expression grammar from a bracketed and tagged corpus². In this training data, only the structures we want to learn are marked at their boundaries by square brackets³. The following sentence shows an example of the training corpus for the NP structure (only base-NPs occur inside brackets).

```
In/IN [ early/JJ trading/NN ] in/IN [
  Hong/NNP Kong/NNP ] [ Monday/NNP
] ,/, [ gold/NN ] was/VBD quoted/VBN
at/IN [ $/$ 366.50/CD ] [ an/DT
ounce/NN ] ./.
```

ALLiS uses an internal formalism in order to represent the grammar rules. In order to parse a text, a module converts its formalism into a regular expression grammar which can be used by a parser using such representation (two modules exist: one for the CASS parser (Abney, 1996) and one for XFST (Karttunen et al., 1997)).

Following the principle of theory refinement, the learning task is composed of two steps. The first step is the generation of the *initial grammar*. This grammar is generated from examples and background knowledge (Section 5). This initial grammar provides an incomplete and/or incorrect analysis of the data. The second step is the refinement of this grammar (Section 5.3). During this step, the validity of the grammar rules is checked and the rules are improved (refined) if necessary. This improvement corresponds to find contexts in which elements which

¹<http://www.sfb441.uni-tuebingen.de/~dejean/chunker.html>.

²The WSJ corpus (Marcus et al., 1993).

³(Muñoz et al., 1999) showed that this representation tends to provide better results than the representation used in (Ramshaw and Marcus, 1995) where each word is tagged with a tag I(inside), O(outside), or B(breaker).

are considered to be members of the structure do not belong to this structure (and reciprocally).

We give here a simple example to illustrate the learning process. The first step (initial grammar generation) categorises the tag *JJ* (adjective) as belonging *by default* to the NP structure if it occurs before a noun. The second step (refinement) finds out that some adjectives do not obey to these rules⁴. The refinement is triggered in order to modify the default rule so that these exceptions can be correctly processed.

Thus, the learning algorithm simply consists of categorising the elements of the corpus (tags and words) into specific categories, and this categorisation allows the extraction of the structures we want to learn. These categories are explained in the next section.

5 The Learning System

5.1 The Background Knowledge

In order to ease the learning, the system uses background knowledge. This knowledge provides a formal and general description of the structures that ALLiS can learn. We suppose that the structures are composed of a nucleus with optional left and right adjuncts. We here give informal definitions, the formal/distributional ones are given in Section 5.2.

The **nucleus** is the head of the structure. We authorise the presence of several nuclei in the same structure.

All the other elements in the structure (except the linker) are considered as **adjuncts**. They are in dependence relation with the head of the structure. The adjuncts are characterised by their position (left/right) relative to the nucleus.

A **linker** is a special element which builds an endocentric structure with two elements. It usually corresponds to coordination⁵.

An element (nucleus or adjunct) might possess the **break** property. This notion is introduced (as in (Ramshaw and Marcus, 1995), (Muñoz et al., 1999)) in order to deal with sequences where adjacent nuclei compose several structures (Section 5.2.4).

This pattern can be seen as a variant of the X-bar template (Head, Spec and Comp), which was already used in a learning system (Berwick, 1985) (although Comp is not useful for the non-recursive structures).

The possible different categories of an element are summarised in Figure 5.1.

The following sentence shows an example of categorisation (elements which do not appear in the structure (NP) are tagged O):

⁴For example: [the/DT 7/CD %/NN benchmark/NN issue/NN] **due/JJ** [October/NNP 1999/CD].

⁵Only linkers occurring between the two coordinated elements are processed.

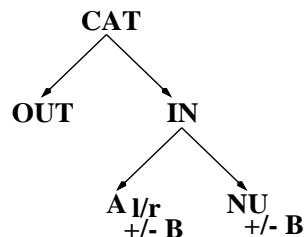


Figure 1: The different categories.

In₀ early_{AB+} trading_{NU} in₀ Hong_{NU} Kong_{NU}
 Monday_{NUB+} ,₀ gold_{NU} was₀ quoted₀ at₀ \$_A
 366.50_{NU} an_{AB+} ounce_{NU} .₀

The structure is formally defined as:

$$S \rightarrow A_{l,b+}^* [A_{l,b-}^* NU_{b-} A_{r,b-}^*]_+ A_{r,b+}^* \\ | A_{l,b+}^* A_{l,b-}^* NU_{b+} A_{r,b-}^* A_{r,b+}^*$$

The symbol * corresponds to the Kleene star. For the sake of legibility, we do not introduce linkers in this expression. But each symbol X (NU, A) can be defined by the rule⁶ $X \rightarrow X \mid X \ l \ X$ where l is the list of linkers. The symbols $B+$ and $B-$ indicate whether the element has the breaker property or not.

Since the corpus does not contain information about these distributional categories, ALLiS has to figure them out. This categorisation relies on the distributional behaviour of the elements, and can be automatically achieved.

5.2 The Initial Categorisation

The general idea to categorise elements is to use specific contexts which point out some of the distributional properties of the category. The categorisation is a sequential process. First the nuclei have to be found out. For each tag of the corpus, we apply the function f_{nu} (described below). This function selects a list of elements which are categorised as nuclei. The function f_b is applied to this list in order to figure out nuclei which are breakers. Then the adjuncts are found out, and the function f_b is also applied to them to figure out breakers.

5.2.1 Categorisation of Nuclei

The context used to find out the nuclei relies on this simple following observation: A structure requires at least one nucleus⁷. Thus, the elements that occur alone in a structure are assimilated to nucleus, since a structure requires a nucleus. For example, the tags PRP (pronouns) and NNP (proper nouns) may compose alone a structure (respectively 99%

⁶The regular expression formalism does not allow such rule, and then, this recursion is simulated.

⁷The partial structures (structures without nucleus) represent 2% of all the structures in the training corpus, and then introduce little noise.

and 48% of these tags appear alone in NP) but the tag JJ appears alone only 0.009%. We deduce that PRP and NNP belong to the nuclei and not JJ. But this criterion does not allow the identification of all the nuclei. Some often appear with adjuncts (an English noun (NN) often⁸ occurs with a determiner or an adjective and thus appears alone only 13%). The single use of this characteristic provides a continuum of values where the automatic set up of a threshold between adjuncts and nuclei is problematic and depends on the structure. To solve this problem, the idea is to decompose the categorisation of nuclei in two steps. First we identify characteristic adjuncts. The adjuncts can not appear alone since they depend on a nucleus. The function f_{char} is built so that it provides a very small value for adjuncts. If the value of an element is lower than a given threshold ($\theta_{\text{char}} = 0.05$), then it is categorised as a characteristic adjunct.

$$f_{\text{char}}(X) = \frac{\sum_C [X]}{\sum_C X}$$

$\sum_C P$ corresponds to the number of occurrences of the pattern P in the corpus C . For example the number of occurrences in the training corpus of the pattern $[JJ]$ is 99, and the number of occurrences of the pattern JJ (including the pattern $[JJ]$) is 11097. So $f_{\text{char}}(JJ) = 0.009$, value being low enough to consider JJ as a characteristic adjunct. The list provided by f_{char} for English NP is:

$$A_{\text{char}} = \{DT, PRP\$, POS, JJ, JJR, JJS, \text{`}, \text{'}\}$$

These elements can correspond to left or right adjuncts. All the adjuncts are not identified, but this list allows the identification of the nuclei as explained in the next paragraph.

The second step consists of introducing these elements into a new pattern used by the function f_{nu} . This pattern matches elements surrounded by these characteristic adjuncts. It thus matches nuclei which often appear with adjuncts. Since a sequence of adjuncts (as an adjunct alone) can not alone compose a complete structure, X only matches elements which correspond to a nucleus.

$$f_{\text{nu}}(X) = \frac{\sum_C [A_{\text{char}} * X * A_{\text{char}}]}{\sum_C X}$$

The function f_{nu} is a good discrimination function between nuclei and adjuncts and provides very low values for adjuncts and very high values for nuclei (table 1).

5.2.2 Categorisation of Adjuncts

Once the nuclei are identified, we can easily find out the adjuncts. They correspond to all the other elements which appear in the context: There are two

x	Freq(x)	$f_{\text{nu}}(x)$	nucleus
POS	1759	0.00	no
PRP\$	1876	0.01	no
JJ	11097	0.02	no
DT	18097	0.03	no
RB	919	0.06	no
NNP	11046	0.74	yes
NN	21240	0.87	yes
NNPS	164	0.93	yes
NNS	7774	0.95	yes
WP	527	0.97	yes
PRP	3800	0.99	yes

Table 1: Detection of some nuclei of the English NP (nouns and pronouns).

kinds of adjuncts: the left and the right adjuncts. The contexts used are:

$$\begin{aligned} [_ \text{NU}] & \quad \text{for the left adjuncts} \\ [A_1 * \text{NU} _] & \quad \text{for the right adjuncts} \end{aligned}$$

If an element appears at the position of the underscore, it is categorised as adjunct. Once the left adjuncts are found out, they can be used for the categorisation of the right adjuncts. They thus appear in the context as optional elements (this is helpful to capture circumpositions).

Since the Adjective Phrase occurring inside an NP is not marked in the Upenn treebank, we introduce the class of *adjunct of adjunct*. The contexts used to find out the adjuncts of the left adjunct are:

$$\begin{aligned} [_ A_1 \text{NU}] & \quad \text{for the left adjuncts of } A_1 \\ [a_1 * A_1 _ \text{NU}] & \quad \text{for the right adjuncts of } A_1 \end{aligned}$$

The contexts are similar for adjuncts of right adjuncts.

5.2.3 The Linkers

By definition, a linker connects two elements, and appears between them. The contexts used to find linkers are:

$$\begin{aligned} [\text{NU} _ \text{NU}] & \quad \text{linker of nuclei} \\ [A _ A \text{NU}] & \quad \text{linker of left adjuncts} \\ [\text{NU} A _ A] & \quad \text{linker of right adjuncts} \end{aligned}$$

Elements which occur in these contexts but which have already been categorised as nucleus or adjuncts are deleted from the list.

5.2.4 The Break Property

A sequence of several nuclei (and the adjuncts which depend on them) can belong to a unique structure or compose several adjacent structures. An element is a breaker if its presence introduces a break into a sequence of adjacent nuclei. For example, the presence of the tag DT in the sequence $NN \text{ } DT \text{ } JJ \text{ } NN$ introduces a break before the tag DT , although the

⁸At least, in the training corpus.

sequence NN JJ NN (without DT) can compose a single structure in the training corpus.

... [the/DT coming/VBG week/NN]
 [the/DT foreign/JJ exchange/NN mar-
 ket/NN] ...

The tag *DT* introduces a break on its left, but some tags can introduce a break on their right or on their left and right. For instance, the tag *WDT* (NU by default) introduces a break on its left and on its right. In other words, this tag can not belong to the same structure as the preceding adjacent nucleus and to the same structure as the following adjacent nucleus.

... [railroads/NNS and/CC trucking/NN
 companies/NNS] [**that/WDT**] be-
 gan/VBD in/IN [1980/CD] ...

... in/IN [**which/WDT**] [people/NNS]
 generally/RB are/VBP ...

In order to detect which tag has the break property, we build up two functions $f_{b \text{ left}}$ and $f_{b \text{ right}}$:

$$f_{b \text{ left}}(X) = \frac{\sum_{C_{wb}} \frac{NU}{X} | [X]}{\sum_{C_{wb}} \frac{NU}{X}}$$

$$f_{b \text{ right}}(X) = \frac{\sum_{C_{wb}} \frac{X}{NU} | [NU]}{\sum_{C_{wb}} \frac{X}{NU}} \quad NU : \{nuclei\}$$

C_{wb} : corpus without brackets

These functions are used to compute the break property for nuclei, but also for adjuncts (In this case, the pattern X is completed by adding the element NU to the left or to the right of X (the potential adjunct) according to the kind of adjunct (left or right adjunct)). The table 2 shows some values for some tags. An element can be a left breaker (DT), a right breaker (no example for English NP at the tag level), or both (PRP). The break property is generally well-marked and the threshold is easy to set up (0.66 in practice).

TAG	$f_{b \text{ left}}$	$f_{b \text{ right}}$
DT	0.97 (yes)	0.00(no)
PRP	0.97 (yes)	0.68(yes)
POS	0.95 (yes)	0.00(no)
PRP\$	0.94 (yes)	0.00(no)
JJ	0.44 (no)	0.00(no)
NN	0.04 (no)	0.11(no)
NNS	0.03 (no)	0.14(no)

Table 2: Breaker determination. Values of the functions $f_{b \text{ left}}$ and $f_{b \text{ right}}$ for some elements.

In the refinement step (Section 5.3), the breaker property can be extended to words. Thus, the word

yesterday is considered as a right breaker, although its tag (NN) is not.

5.3 The Refinement Step

5.3.1 The notion of reliability

The preceding functions identify the category of an element *when it occurs in the structure*. But an element can occur in the structure as well as out of the structure. For example, the tag *VBG* is only considered as adjunct when it occurs in the structure. Nevertheless, it mainly occurs out of the structure (84% of its occurrences). If an element mainly⁹ occurs out of the structure, it is considered as *non-reliable* and its default category is OUT. *For each element occurring inside the structure, its reliability is tested*. The initial grammar corresponds to the grammar which only contains the reliable elements. Its precision and its recall are around 86%.

How is determined the reliability of an element? This notion of reliable element is contextual and depends on the category of the element.

For the nuclei, the context is empty. We just compute the ratio between the number of occurrences in the structure over the number of occurrences occurring outside of the structure.

For the adjuncts, the context includes an adjacent nucleus (on the right for left adjuncts or on the left for right adjuncts). For instance, the tag *JJ* is categorised as left adjunct for the English NP. It appears 9617 times before a nucleus and 9489 times in the structure. It is thus considered as reliable, and its default category is left adjunct. In the case where the tag *JJ* occurs without nucleus on its right (a predicative use), it is not considered as adjunct and this kind of occurrences is not used to determine the reliability of the element. On the contrary, the tag *VBG* appears 468 times before a nucleus, but, in this context, it occurs only 138 times in the structure. This is not enough (29%) to consider the element as a reliable left adjunct, and thus its default category is OUT. For the adjunct of adjunct, the context includes adjunct and nucleus.

5.3.2 Detection of errors

Once the initial grammar is built up, its errors have to be corrected. The detection of the errors corresponds to a miscategorisation of a tag. An automatic error done by the initial grammar is to wrongly analyse the structures composed with non-reliable elements (false negative examples). Each time that a non-reliable element occurs in the structure corresponds to an error. For instance, the initial grammar can not correctly recognise the following sequence as an NP, the default category of the tag *VBG* being OUT (outside the structure):

... [the/DT coming/**VBG** week/NN] ...

⁹The threshold used is of 50%.

The second kind of errors corresponds to sequences wrongly recognised as structures (false positive examples). This kind of error is generated by reliable elements which exceptionally do not occur in the structure. In the following example, *order/NN* occurs outside of the structure, although the default category of the tag *NN* is *NU* (nucleus), and thus the initial grammar recognises an NP.

...in/*IN* **order**/*NN* to/*TO* pay/*VB* ...

5.3.3 Correction

In both kinds of errors, the same technique is used to correct them. For this purpose, ALLiS disposes of two operators, the contextualisation and the lexicalisation.

The *contextualisation* consists of finding out contexts in order to fix the errors. The idea is to add constraints for recategorising non-reliable elements as reliable¹⁰. The presence of some specific elements can completely change the behaviour of a tag. The table 3 shows the list of contexts where the tag *VBN* is not categorised as *OUT* but as left Adjunct.

PRP\$	VBG	NU
IN [VBG	NU
DT	VBG	NU
JJ	VBG	NU
POS	VBG	NU
VBG [VBG	NU

Table 3: Some contexts where the non-reliable element *VBN* becomes reliable.

For each tag occurring in the structure, all the possible contexts¹¹ are generated. For the non-reliable tags (first kind of error), we evaluate the reliability of them contextually, and we delete the contexts in which the tag is still non-reliable (the list of contexts can be empty, and in this case the error can not be fixed). For the reliable tags (second kind of error), we keep the contexts in which the tag is categorised *OUT*.

The *lexicalisation* consists of introducing lexical information: the word level. Some words can have a specific behaviour which does not appear at the Part-Of-Speech (POS) level. For instance, the word *yesterday* is a left breaker, behaviour which can not be figured out at the POS level (Table 4). The introduction of the lexicalisation improves the result by 2% (Section 6).

The lexicalisation and the contextualisation can be combined when both separately are not powerful enough to fix the error. For example, the word *about* tagged *RB* (default category of *RB*: *OUT*) followed

¹⁰The same technique is used in (Sima'an, 1997).

¹¹The contexts depend on the category of the tag, but are just composed of one element.

word(context)	default cat.	new cat.
about/ <i>RB</i> (_ <i>CD</i>)	<i>OUT</i>	<i>A₁,B₊</i>
order (<i>IN</i> _ <i>TO</i>)	<i>NU</i>	<i>OUT</i>
yesterday/ <i>NN</i>	<i>NU_{B-}</i>	<i>NU_{B+}</i>
operating/ <i>VBG</i>	<i>OUT</i>	<i>A₁,B-</i>
last/ <i>JJ</i>	<i>A₁,B-</i>	<i>A₁, B+</i>

Table 4: Some specific lexical behaviours.

by the tag *CD* is recategorised as left adjunct and left breaker (Table 4).

6 Evaluation

We now show some results and give some comparisons with other works (Table 5). The results are quite similar to other approaches. Two rates are measured: precision and recall.

$$R = \frac{\text{Number of correct proposed patterns}}{\text{Number of correct patterns}}$$

$$P = \frac{\text{Number of correct proposed patterns}}{\text{Number of proposed patterns}}$$

The training data are composed of the sections 15-18 of the Wall Street Journal Corpus (Marcus et al., 1993), and we use the section 20 for the test corpus¹². The data is tagged with the Brill tagger. The works generating symbolic rules like ALLiS are (Ramshaw and Marcus, 1995) (Transformation-Based learning) and (Cardie and Pierce, 1998) (error-driven pruning of treebank grammars). ALLiS provides better results than them. (Argamon et al., 1998) use a Memory-Based Shallow Learning system, (Tjong Kim Sang and Veenstra, 1999) the Memory-Based Learning method and (Muñoz et al., 1999) uses a network of linear functions. The latter work seems to integrate better lexical information since ALLiS gets better results with POS only.

		POS only	with words
NP	MPRZ99	90.3/90.9	92.40/93.10
	ALLiS	91.0/91.2	92.56/92.36
	TV99		92.50/92.25
	ADK98	91.6/91.6	
	RM95	90.7/90.5	92.3/91.8
	CP98	91.1/90.7	
VP	ALLiS	91.39/90.52	92.15/91.95

Table 5: Results for NP and VP structures (precision/recall).

The main errors done by ALLiS are due to errors of tagging (the corpus is tagged with the Brill tagger) or errors in the bracketing of the training corpus. Then, the second type of errors concerns

¹²This data set is available via <ftp://ftp.cis.upenn.edu/pub/chunker/>.

the coordinated structures. These two types errors correspond to 51% of the overall errors. We can find the same typology in other works (Ramshaw and Marcus, 1995), (Cardie and Pierce, 1998). We did some tries in order to manually improve the final grammar, but the only type of errors which can be manually improved concerns the problem of the quotation marks (the improvement is about of 0.2% in precision and recall).

Error types	#	%
tagging/bracketing errors	57	28.5%
coordination	45	22.5%
gerund	15	7.5%
adverb	13	6.5%
appositives	13	6.5%
quotation marks, punctuation	10	5 %
past participle	9	4.5%
that (IN)	6	3%

Table 6: Typology of the 200 first errors.

References

- Andreas Abecker and Klaus Schmid. 1996. From theory refinement to kb maintenance: a position statement. In *ECAI'96*, Budapest, Hungary.
- Steven Abney. 1996. Partial parsing via finite-state cascades. In *Proceedings of the ESSLLI '96 Robust Parsing Workshop*.
- Shlomo Argamon, Ido Dagan, and Yuval Krymolowski. 1998. A memory-based approach to learning shallow natural language patterns. In *COLING'98*, Montréal.
- Robert C. Berwick. 1985. *The acquisition of syntactic knowledge*. MIT press, Cambridge.
- Eric Brill. 1993. *A Corpus-Based Approach to Language Learning*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.
- Clifford Alan Brunk and Michael Pazzani. 1995. A lexically based semantics bias for theory revision. In Morgan Kauffman, editor, *Twelfth International Conference on Machine Learning*, pages 81–89.
- Clifford Alan Brunk. 1996. *An investigation of Knowledge Intensive Approaches to Concept Learning and Theory Refinement*. Ph.D. thesis, University of California, Irvine.
- Sabine Buchholz, Jorn Veenstra, and Walter Daelemans. 1999. Cascaded grammatical relation assignment. In *Proceedings of EMNLP/VLC-99*, pages pp. 239–246, University of Maryland, USA.
- Claire Cardie and David Pierce. 1998. Error-driven pruning of treebank grammars for base noun phrase identification. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING-ACL '98)*.
- Claire Cardie and David Pierce. 1999. The role of lexicalization and pruning for base noun phrase grammars. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*.
- Susan Crow and D. Sleeman. 1990. Automating the refinement of knowledge-based systems. In *Proceedings of the ECAI'90 Conference*, pages 167–172.
- The XTAG Research Group. 1998. A lexicalized tree adjoining grammar for english. Technical Report IRCS 98-18, University of Pennsylvania.
- Lauri Karttunen, Tamás Gal, and André Kempe. 1997. Xerox finite-state tool. Technical report, Xerox Research Centre Europe, Grenoble.
- Mitchell Marcus, Béatrice Santorini, and Marc Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.
- Raymond J. Mooney. 1993. Induction over the unexplained: Using overly-general domain theories to aid concept learning. *Machine Learning*, 10:79.
- Raymond J. Mooney. 1997. Inductive logic programming for natural language processing. In *Sixth International Inductive Logic Programming Workshop*, pages 205–224, Stockholm, Sweden.
- Marcia Muñoz, Vasin Punyakanok, Dan Roth, and Dav Zimak. 1999. A learning approach to shallow parsing. In *Proceedings of EMNLP-WVLC'99*.
- Dirk Ourston and Raymond Mooney. 1990. Changing the rules: A comprehensive approach to theory refinement. In *Proceedings of the Eight National Conference on Artificial Intelligence*, pages 815–820.
- Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *ACL Third Workshop on Very Large Corpora*, pages 82–94.
- Geoffrey Sampson. 1995. *English for the Computer. The SUSANNE Corpus and Analytic Scheme*. Oxford: Clarendon Press.
- Khalil Sima'an. 1997. Explanation-based learning of data oriented parsing. In T. Mark Ellison, editor, *Computational Natural Language Learning (CoNLL), ACL/EACL-97*, Madrid.
- Erik Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of EACL'99, Association for Computational Linguistics*, Bergen.
- Jacques Vergne and Emmanuel Giguet. 1998. Regards théoriques sur le "tagging". In *proceedings of Traitement Automatique des Langues Naturelles (TALN 1998)*, Paris.