Journal of Machine Learning Research 2 (2002) 669–693 Submitted 9/01; Published 3/02

Learning Rules and Their Exceptions

Hervé Déjean HERVE.DEJEAN@XRCE.XEROX.COM
Xerox Research Center Europe
6, chemin de Maupertuis
38240 Meylan

Editors: James Hammerton, Miles Osborne, Susan Armstrong and Walter Daelemans

Abstract

We present in this article a top-down inductive system, ALLiS, for learning linguistic structures. Two difficulties came up during the development of the system: the presence of a significant amount of noise in the data and the presence of exceptions linguistically motivated. It is then a challenge for an inductive system to learn rules from this kind of data. This leads us to add a specific mechanism, **refinement**, which enables learning rules and their exceptions. In the first part of this article we evaluate the usefulness of this device and show that it improves results when learning linguistic structures.

In the second part, we explore how to improve the efficiency of the system by using prior knowledge. Since Natural Language is a strongly structured object, it may be important to investigate whether linguistic knowledge can help to make natural language learning more efficiently and accurately. This article presents some experiments demonstrating that linguistic knowledge improves learning. The system has been applied to the shared task of the CoNLL'00 workshop.

Keywords: Symbolic Learning, Rule Induction, Learning Exceptions, Natural Language Processing, Chunking

1. Introduction

For now more than a decade, Natural Language has been a common domain of application for machine learning algorithms. In this article is presented a top-down inductive system, ALLiS, for learning linguistic structures. Two difficulties came up during the development of the system: the presence of a significant amount of noise in the data and the presence of exceptions (linguistically motivated) in the encoding schemata. It is then a challenge for an inductive system to learn rules from this kind of data. This problem is not new and was addressed among others by Quinlan (1986) and Brunk and Pazzani (1991). Noise occurs in training data when a datum is not assigned to the correct class. But noise is not the only problem: Natural Language is an object which contains many sub-regularities and exceptions which can not be considered as noise.

This leads us to add a specific mechanism for facing such problems: *refinement*. Whenever a rule is learned, exceptions to this rule are systematically searched. The result of this algorithm is a set of rules where each of them is associated with a set of exceptions. In the first part of this article, we will evaluate the usefulness of this device and will show that it improves results when learning linguistic structures. We will also show that, with the use of refinement, some traditional problems occurring when learning set of rules such as threshold determination fall if one uses appropriate prior knowledge.

In a second part, we explore a second way for improving the efficiency of the system by using prior knowledge. Since Natural Language is a strongly structured object, it may be important to investigate whether structural linguistic knowledge can help to make natural language learning more efficiently and accurately. The utility of (prior) knowledge has been shown with inductive systems (see Pazzani and Kibler, 1992, Cardie, 1999). This article presents some experiments, trying to answer this question: What kind of linguistic knowledge can improve learning?

This article is articulated as follows: the inductive learning system ALLiS is described and a first estimation using no prior knowledge is proposed. Results of this experiment without linguistic knowledge will be used as the baseline in order to appraise the effect of the prior knowledge. This linguistic prior knowledge is then detailed, and we will discuss its (positive) effect from a computational viewpoint as well as from a qualitative viewpoint. The system has been applied to the shared task of the CoNLL'00 workshop. We then provide a quantitative and qualitative analysis of these results. Finally we compare our algorithm with related systems, especially `FOIDL`.

A description of the Upenn tagset used along the article is given Appendix 8.

2. The ALLiS Algorithm

ALLiS is a classical top-down induction system. The general algorithm is presented in Algorithm 1. Top-down induction is a generate-then-test approach: a set of potential rules is generated according to some patterns, and each of them is tested against the training data. Those being accurate enough are kept, the others deleted. Section 2.3 explains how the set of potential rules is generated. Whenever a new rule is learned, the bit of data covered by it is removed, and the process continues with the next rule. Starting with the most general definition of a rule, ALLiS constrains it by adding new constraints (hereafter called *literals*) until its accuracy reaches a given threshold. Like other induction systems, ALLiS does not guarantee that the set of rules is optimal. If a comparison is drawn between our algorithm and those presented in the literature (see Mitchell, 1997, Quinlan, 1990), the general architecture is very similar. The major modification concerns the stopping criteria: whenever a rule is learned, ALLiS tests whether *exceptions* to this rule can be learned as well. This process, which we call *refinement*, is described in detail Section 2.4. When most algorithms learn a set of (ordered) rules, ALLiS learns a *structured set of rules*. With each rule is explicitly associated a set of exceptions. As the result will show, the accuracy of the structure (rule + {exceptions}) is generally greater than the accuracy of the single rule itself (Section 2.4).

Why learning exceptions? The main motivation of the introduction of this mechanism is to deal with noise. The specific attention placed on handling noise is not only due to *real* noise in the data, but also due to the presence of exceptions in the language itself (or more exactly in linguistic theories used for annotating data) (Section 2.4). Note that other inductive systems such as FOIDL (see Mooney and Califf, 1995) or TBL (see Brill, 1995) also need to introduce such a mechanism when they are applied to linguistic data (for example, learning past tense of English verbs).

Algorithm 1 learn_set_of_rules (rule,θ,pos,neg)

```
list ← rule
while ∃ x in list do
  eval = evaluation(x,pos,neg)
  if eval> θ then
    set_of_rules ← x
    set_of_rules ← learn_set_of_rules (x,θ,neg,pos)
    clean_data(x,pos);
  else
    list ← expand_rule(x)
  end if
  delete (x,list)
end while
```

The kind of rules ALLiS learns is categorization rules: we try to categorize each element into the proper category. Problems have then to be reformulated as categorization problems. An example of a rule learned by ALLiS is:

```
<RULE NUM='98' S='I-NP' ACC='0.95' FREQ='175'>
 <N C='DT' LEFT='1'/>
 <N C='VBG' />
</RULE>
```

This rule is to be read: if a word tagged VBG (*C='VBG'*) is preceded by a word tagged DT, then its category is I-NP (inside an NP, see Section 3). The rule also contains information about itself: one unique identifier (*NUM*), its frequency (*FREQ*), its accuracy (*ACC*).

In the next sections, we detail first the data and representation used, and then each important step of the algorithm, the more important being the refinement step, the others being quite traditional.

2.1 Representation of the Data

In this paper, our experiments use the CoNLL shared task data (see Sang, 2000a). It provides a corpus annotated with chunk information. Chunk information is extracted from a manually annotated corpus, the Wall Street Journal, part of the Upenn Treebank described in Marcus et al. (1993). The corpus is tagged with the Brill tagger (see Brill, 1995), in order to make sure that the performance rates obtained with this data are realistic estimates. After this preprocessing, with each word of the data are associated its Part-of-Speech tag and its chunk category.

The formalism used by ALLiS in order to encode training data is XML. Even if other representation schemas can be equivalently used (such as database formalism), XML offers an adequate representation allowing structured (here hierarchical) data encoding. Data is then encoded in a tree structure. For each sentence a node $< S >$ is created and for each word of the sentence an empty node $< N/ >$. With each node is associated its features (in the following examples, the word itself (feature W), its POS-tag (feature C) and its category

(S)). This is a traditional attribute-value representation. More complex annotation can be used, for example, the tree of a given linguistic analysis can be kept. Table 1 shows data with a flat structure, and Table 2 with several linguistic hierarchical levels of information (chunk, clause). We use the first representation for the CoNLL 200 shared task, and the second one was used for the CoNLL2001 shared task (see Déjean, 2001).

```
<S>
<N W=During C=IN S=B-PP/>
<N W=the C=DT S=B-NP/>
<N W=Nixon C=NNP S=I-NP/>
<N W=administration C=NN S=I-NP/>
<N W=, C=, S=O/>
<N W=the C=DT S=B-NP/>
<N W=Drug C=NNP S=I-NP/>
<N W=Enforcement C=NNP S=I-NP/>
<N W=Administration C=NNP S=I-NP/>
<N W=became C=VBD S=B-VP/>
<N W=dismayed C=VBN S=I-VP/>
<N W=at C=IN S=B-PP/>
<N W=the C=DT S=B-NP/>
<N W=extent C=NN S=I-NP/>
<N W=of C=IN S=B-PP/>
<N W=the C=DT S=B-NP/>
<N W=G-2 C=NN S=I-NP/>
<N W='s C=POS S=B-NP/>
<N W=connections C=NNS S=I-NP/>
<N W=to C=TO S=B-PP/>
<N W=arrested C=VBN S=B-NP/>
<N W=drug C=NN S=I-NP/>
<N W=traffickers C=NNS S=I-NP/>
</S>
```

Table 1: Example of training corpus for chunking (flat structure).

We now explain how positive and negative examples are generated from this annotation. The division of the data into positive and negative examples is done thanks to the features. For instance, let the example shown Table 1 be the complete training data and let NP chunking be the task. The task is then to assign the correct value of the feature S to each node by using the features W and C. Each node (word) with the features *S='I-NP'* or *S='B-NP'*[1] is then a positive example, and each node without his feature is a negative example. For example, the node *<N W=dismayed C=VBN S=I-VP/>* is a negative example, when the node *<N W=arrested C=VBN S=B-NP/>* is a positive example, although both are tagged *VBN*. This example shows that the information attached to each node may be

1. Section 3 explains this 'subcategorization' of NP into B-NP and I-NP.

```
<S NUM='4563'>
<CL>
 <PHR S='NP' >
  <W BP='B' C='VBN' W='Estimated'/>
  <W BP='I' C='NN' W='volume'/>
 </PHR>
 <PHR S='VP'>
  <W BP='B' C='VBD' W='was'/>
 </PHR>
 <PHR S='NP' B='N'>
  <W BP='B' C='DT' W='a'/>
  <W BP='I' C='NN' W='light'/>
  <W BP='I' C='CD' W='2.4'/>
  <W BP='I' C='CD' W='million'/>
  <W BP='I' C='NNS' W='ounces'/>
 </PHR>
 <PHR S='O'>
   <W C='.' W='.'/>
 </PHR>
</CL>
</S>
```

Table 2: Example of training data for clausing using a hierarchical structure.

insufficient in order to correctly recognize the attribute *S*. Section 2.3 explains how the search space is extended.

2.2 Rule Generation and Evaluation

ALLiS generates a set of rules using the general-to-specific approach. It starts with the most general rule and specifies it until its accuracy is high enough. The purpose of a rule is to categorize a node with the attribute *S*, called the *current node* of the rule, the other nodes occurring in it are called the *contextual nodes*. If one still uses the preceding example (learning NP), the initial rule is:

```
<RULE S='I-NP'>
<N/>
</RULE>
```

This first initial rule tags as I-NP (feature S of the rule) a node without specificities (so all nodes). The quality of the rule is then estimated. We just use its accuracy: $\frac{pos}{pos+neg}$ where *pos* is the number of positive examples and *neg* the number of negative examples covered by the rule. Positive examples of the rule correspond to data that match the rule where the current node is tagged *S=I-NP*. Negative examples correspond to data that match the rule where the current node has not the feature *S=I-NP*. This simple measure, the accuracy, is also used in several other systems: (see Sima'an, 1997, Argamon et al., 1998). Other functions are proposed in the literature, for example the Laplacian expected error (see Soderland, 1999). If the accuracy of a rule r is greater than a given threshold called hereafter θ, r is judged reliable enough and is added into the set of rules. θ is then a core parameter of the system. Section 3 will show its influence on the system.

In the case of our initial rule, its accuracy is of course very low, since this rule is underspecified. The next step of the general-to-specific algorithm is to add it literals in order to increase its accuracy. We choose a new literal, *C*, which is added to the rule, creating new potential rules. One new rule is created for each different value of the new literal. Here are two new rules among other possibilities:

```
<RULE S='I-NP'>
<N C='NN'/>
</RULE>
<RULE S='I-NP'>
<N C='IN'/>
</RULE>
...
```

Each of these new rules is evaluated and specialized if necessary. The first rule, which tags nouns (*C='NN'*) as being part of an NP, is accurate enough (0.98), and is then validated.

```
<RULE NUM='1' ACC='0.98' FREQ='29027'>
<N C='NN'/>
</RULE>
```

The accuracy of the rule is not equal to 1 and it generates errors. Section 2.4 will explain how these errors can be reduced thanks to the notion of *refinement*.

The second rule (*C='IN'*) has to be specified, and new literals are added. This generates rules such as:

```
<RULE NUM='39' S='I-NP' ACC='1.00' FREQ='2'>
<N W='about' C='IN'/>
<N C='$' RIGHT='1'/>
</RULE>
```

The tag *RIGHT* means that this node is a right contextual node of the rule. This rule matches the sequence *about $*, and in this context, the word *about* belongs to an NP according to the training data. No refinement is required since the refinement of the rule is 1.

The next section explains how new literals are selected.

2.3 The Search Space

Since information contained in each node may not be enough in order to find the category of each node, we will use additional information, namely the context of each node, which is traditional in Natural Language Processing.

Once a rule has been evaluated, and if its accuracy is not high enough, the expansion step is set off: new literals are added to the rule in order to constrain it. By adding new literals, the purpose is to reduce the number of negative examples covered by the current rule. One main problem in this kind of approach is to determine which literal has to be added. FOIL, the system described in Quinlan (1990), uses a measure for selecting them, the *information-gain*. We do not implement this selection process and do not impose any criteria for selecting literals, which will be improved in the next versions of ALLiS.

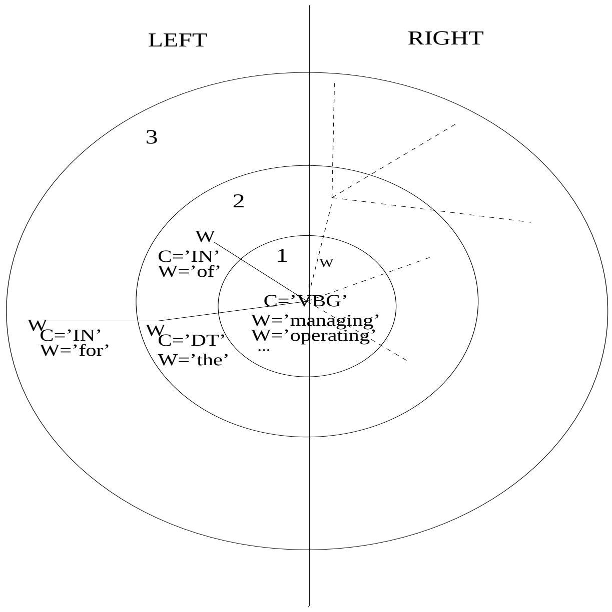

Figure 1: Organization of the search space. First, local information is used, and the space is then expanded.

We partially order the way the search space is explored (Figure 1). First, local information is used: the search space is reduced to the node itself. If the accuracy is still not high enough, the immediate adjacent nodes of the current node are added into the space, until the maximum number of neighbors authorized is reached or the accuracy is high enough. Concerning the feature of each node, the more general features are first added (C, the POS tag of the word, then W the word itself). This requires then knowledge about the feature used, knowledge which is not always available in a more general case.

2.4 Refinement

The main difference between ALLiS and classical inductive systems relies on the explicit account of noise by using refinement. The training data is a portion of the Wall Street Journal corpus tagged with the Brill tagger. This tagger has an average error rate of 5%, and during the extraction of the data from the original corpus, errors of bracketing are also introduced (chunks are not explicitly bracketed in the original corpus, and a non trivial processing is required for extracting them). Besides, since some very frequent words are mainly correctly tagged (such as *the*, *to*), it means that some other words are tagged with an error rate largely greater than 5% (especially nouns and verbs). It is then sensible to consider that, for some rules, an accuracy greater than 0.80 or 0.90 is not reasonable. The second reason for introducing this mechanism of refinement is linked to a property of Natural Languages: *they contain also many sub-regularities and (pocket of) exceptions* (see Daelemans et al., 1999). These two points (noise and exceptions) lead us not except finding general rules (rules with a high coverage) with an accuracy equal to 1.

Often, systems only consider noiseless data and try to learn "perfect" rules (with an accuracy of 1). In order to deal with noise, some integrate mechanisms for stopping the expansion (see Pazzani and Kibler (1992)) or pruning mechanisms (for example, *reduced error pruning* in Brunk and Pazzani (1991)). As stopping mechanism, FOIL uses encoded-length heuristic indicating that the rule is becoming too complex. Another simple way is to set up a threshold lower than 1 as in Soderland (1999). This is the same solution we have chosen to adopt. But what about the few percents wrongly categorized? Should we ignore them? The refinement process tries to learn these cases, considered as exceptions of the rule just learned. Let us take a simple example in order to illustrate this mechanism. Consider the rule:

```
<RULE NUM='1' ACC='0.98' FREQ='29027'>
<N C='NN'/>
</RULE>
```

Its accuracy is 0.98. Let the threshold θ be 0.8. The accuracy is high enough and the rule is learned but will maybe generate 2% of errors. In order to fix them, we just learn rules that correct them, relying on the fact that some kinds of errors are regular enough and can be explained. The initial rule is the rule itself and in order to learn exceptions, we just *switch positive and negative examples*, and apply the same algorithm (see line 6 of Algorithm 1). The goal is now to categorize elements matching the context described by the rule *1* which do not belong to NP. The positive examples of the rule *1* become then the negative examples of these exceptions. The positive examples are now composed of contexts where nodes with the feature *C='NN'* have not the feature *S='I-NP'*. There is a recursive call of the learning procedure with this inversion in the positive and negative data (Algorithm 1). The following rules are then learned:

```
<RULE EXCEP='1' NUM='3' ACC='1.00' FREQ='17'>
<N W='have' C='NN'/>
</RULE>

<RULE EXCEP='1' NUM='98' ACC='0.90' FREQ='9'>
<N C='IN' LEFT='1'/>
<N W='order' C='NN'/>
<N C='TO' RIGHT='1'/>
</RULE>
```

Such exceptions are called *refinement rules*. The feature *EXCEP* points to the rule (its identifier) to which it is an exception. Both exceptions are due to two different explanations.

The first refinement rule illustrates well the interest of refinement for noisy data. In the training data, some occurrences of the word *have* are wrongly tagged as NN (singular noun) instead of verb (remember that this tagging is done automatically).

The second rule illustrates another kind of "noise", or more appropriately, illustrates a linguistic exception in the encoding schemata. It is specified in the guidelines of the Upenn Treebank, from where our data is extracted, that in the context *in order to*, the word *order* has to be tagged *NN* but does not belong to an NP. One can find other exceptions similar to this one in the guidelines (*subject to* for instance). These exceptions, which are not noise, since linguistically motivated, can be easily encapsulated thanks to refinement.

ALLiS can also learn exceptions of exceptions (and so on). This happens rarely when θ is high enough, but frequently in other cases. The next rule could be an exception to the rule *98*, but it is not kept since its frequency is to small.

```
<RULE EXCEP='98' NUM='99' ACC='1.00' FREQ='1'>
<N W='of' C='IN' LEFT='1'/>
<N W='order' C='NN'/>
<N C='TO' RIGHT='1'/>
</RULE>
```

Each refinement rule is in fact a more specific rule describing a context in which the "mother" rule[2] (this referred by the feature *EXCEP*) is no longer validated.

2.5 Parsing

Once a set of rules is learned, it has to be applied to data. For this purpose, several tools have been used: CASS (see Abney, 1996), XFST (see Karttunen et al., 1997), fsgmatch (see Grover et al., 1999), and XIP (see Aït-Mokhtar et al., 2001). XIP offers the most appropriate formalism and a good parsing speed. The results presented in this paper have been obtained using XIP. ALLiS rules have to be converted into the adequate (XIP) format. Here are the rules we discussed in the preceding version.

```
1> ?[i-np=+, C:NN ].
2> ?[C:IN ], ?[i-np=~, W:order ,C:NN ], ?[C:TO ].
2> ?[i-np=~, W:have ,C:NN ].
```

XIP allows the ordering rules. A straightforward way to implement exceptions is to apply them after their "mother" rule. The first rule adds the feature *i-np* and is applied first (level *1*). The other rules, exceptions of the first, are triggered after (level *2*), and delete the feature *i-np* thanks to the operator `=~`.

3. First Experiment

We now present a first evaluation of ALLiS using no prior knowledge. This evaluation will be used as baseline when integrating prior knowledge.

2. We call *mother* rule of a rule r, the rule to which r is an exception.

Presentation of the task and evaluation: In order to evaluate the impact of some parameters used by ALLiS, we conducted some experiments. The task here is the extraction of non-overlapping Noun Phrases (hereafter NP). This problem can be viewed as a problem of categorization using three categories: B-NP, I-NP and O. I-NP means that the element belongs to an NP, a B-NP is used when the current element belongs to an NP and the preceding belongs to another NP. We refer the reader to Ramshaw and Marcus (1995) for more information. Here is an example of such a segmentation with these three categories:

In$_{\texttt{O}}$ early$_{\texttt{I-NP}}$ trading$_{\texttt{I-NP}}$ in$_{\texttt{O}}$ Hong$_{\texttt{I-NP}}$ Kong$_{\texttt{I-NP}}$ Monday$_{\texttt{B-NP}}$,$_{\texttt{O}}$ gold$_{\texttt{I-NP}}$ was$_{\texttt{O}}$ quoted$_{\texttt{O}}$ at$_{\texttt{O}}$ \$$_{\texttt{I-NP}}$ 366.50$_{\texttt{I-NP}}$ an$_{\texttt{B-NP}}$ ounce$_{\texttt{I-NP}}$.$_{\texttt{O}}$

In order to evaluate the system, we use two measures: precision and recall. The F1-score presented in van Rijsbergen (1979) offers a simple way to combine both (in our evaluation, we give the same weight to recall and precision: $\beta = 1$).

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

$$F_{\beta} = \frac{(\beta^2+1)*R*P}{\beta^2*R+P}$$

Evaluation: Table 3 shows the result for several values of the principal parameters. The baseline, (assign an element its most frequent category: O, I-NP or B-NP) provides an F1-score of 80.13. These results were obtained without using lexicalized information (**W** feature in the example Figure 1).

θ	0.9	0.85	0.80	0.75	0.75	0.5	0.5
length of the context	2	2	2	2	1	2	1
nb rules	1144	930	884	811	361	1122	592
nb exceptions	192	230	287	297	134	669	357
F w/o excep	88.79	90.19	89.12	89.28	88.08	88.21	87.61
F with excep	88.93	90.31	90.47	90.62	89.26	90.70	90.07

Table 3: Influence of some parameters.

The effect of refinement: We can immediately see the positive effect of the refinement mechanism. The last two rows provide the F1-score of the system for different parameters values without and with the use of exception rules. Refinement improves all results, the F1-score gaining between 1 and 2.5%, which is a very good increase for this task. More interestingly, refinement is especially efficient for $\theta = 0.50$ since the gain is +2.49 and then provides the best F1-score, outperforming all the other results without refinement. Results deteriorate for higher values of θ (0.90). This may be due to the fact that θ is too close or greater than the noise level. One other interesting result is the flattening of the F1-scores. Without refinement rules, the variation of F1-score is much significant and more correlated to θ. The use of refinement lessens these variations. This result meets our expectation. We can then conclude that refinement is very helpful since it improves results uniformly and gives θ less importance. This is an interesting issue since such a threshold is a major parameter of inductive systems and has a significant impact on results.

In traditional top-down induction systems, a high value of θ provides a high precision, but can provide a poor recall. Here, precision and recall are always close one another, even for "extreme" values such as 0.5 (P:90.72, R:90.69) or 0.9 (P:88.67, R:89.20). Without refinement the difference between precision and recall increases: P:87.50, R:88.94 for $\theta = 0.5$. To increase the size of the context (3 elements) marginally improves results.

The value θ=0.75 provides the best tradeoff between accuracy and the number of rules (and also the best result without using refinement). When θ is low (for example 0.5), exceptions of exceptions appear and even exceptions of exceptions of exceptions. We have never observed a recursion level greater than 3. Of course, the lower the value of θ, the higher the number of exceptions. When $\theta = 0.90$ exceptions represent 16% of the rules, when $\theta = 0.50$, they represent 60%.

Small Disjuncts are beneficial: As Table 4 shows, a majority of rules have a small coverage relatively to the size of the training data. The rules with a coverage lesser or equal to 5 represent 63% of the rules. But rules covering only a few cases of the training data can perform relatively badly since they have inadequate statistical support. This problem has been referred to in the literature as the small disjuncts problem by Holte et al. (1989). Several results, (see Holte et al., 1989, Holder, 1992, Provost and Kolluri, 1997, Ali and Pazzani, 1995) show that accuracy decreases with the addition of small disjuncts.

Coverage	nb rules	Precision	Recall	F1
<3	174(13%)	+0.47%	+0.28%	+0.37
<6	880(63%)	+1.17%	+0.75%	+0.96

Table 4: Evolution of F1-score using low frequency rules ($\theta = 0.75$,lg=2).

But Table 4 indicates that, in our experiment, the rules with low frequency (lesser or equal to 5) provide a substantial improvement of the F1-score. Even rules with a frequency of 2, which can be considered as the least reliable, improve the F1-score. Whatever the value of the other parameters, low frequency rules have to be learned for this data.

In fact, our result is not in contradiction with the previous ones. Holte et al. (1989) conclude that if the small disjuncts are specific enough, their error rate decreases considerably. This is in fact a specificity of our rules with low frequency, which are very specific, almost describing a lexicalized context. Note that the coverage of a rule has to be nevertheless greater than 1.

Comparison with TBL: Transformation-based learning (see Brill, 1995) is a now popular symbolic machine learning approach. Since it has been applied to the same task (see Ramshaw and Marcus, 1995), it is interesting to compare both algorithms. Furthermore, TBL principles are very similar in some ways: learning transformations can be seen as learning our refinement rules (or chronologically the other way around). Two main differences have to be mentioned. First, rules are not learned in the same order. In TBL, a rule is learned because its application over the data introduces the best improvement. A rule which fixes a preceding rule (equivalent to our refinement rule) will be learned after any number of intermediate rules. In ALLiS, refinement rules are learned immediately after the "mother rule", and both rules are explicitly linked.

A second major difference is the kind of information used by TBL. TBL first tags data using the most frequent category of the elements. The system can then use the category of adjacent elements (left and right), information very useful that ALLiS does not use. ALLiS could use the category of the preceding elements, since this information will be available during parsing, but we do not incorporate this information into the features[3]. Both systems do not use the same information, but they provide comparable results for this task (NP extraction) (Table 5). Besides, ALLiS only generates 1369 rules whereas TBL stops after 2000 rules. The quality of the rules seems then to be better. Table 5 also shows the improvement due to the use of the lexicalized information (W feature).

(θ, lg)	w/o words	with words
(0.50,1)	90.70	92.18
TBL	90.60	92.03

Table 5: Comparison between TBL and ALLiS.

Another interesting point in TBL is the absence of threshold, since, at each iteration, the best rule is chosen. No threshold equivalent to our θ is needed. The stopping criterion is determined by the maximum number of rules the system learns (after n rules, it stops). Increasing the number of rules generally improves results.

After this first experiment using no prior knowledge, we now present the linguistic knowledge we will used in order to improve ALLiS.

4. Prior Knowledge

In order to facilitate the learning process, we introduce prior knowledge into the system. This knowledge provides a formal and general description of the linguistic structures that ALLiS can learn. The essential kind of knowledge we introduce consists on more appropriate and distributional categories than the ones previously used. We show that if a larger set of categories is used and if these categories are more suitable, the learner's task is eased (the processing time is reduced, as well as the number of rules) and the quality of the rules is better. The following sections describe in detail the different types of prior knowledge we use.

4.1 Refining the Initial Categorization

We replace the categories provided by the data (I, B, O) with more distributional categories. The category I is divided into three distributional subcategories.

$$S = AL^* \ NU \ AR^*$$

We suppose that structures are composed of a nucleus (NU) with optional left (AL) and right (AR) adjuncts. We here give informal definitions, the formal/distributional ones are given in the next sections. The **nucleus** is the main element (head) of a given structure: the noun for Noun Phrase, the verb for VP, VP for clause. The adjuncts correspond to auxiliary elements, for instance adjectives for a noun. They are in a dependence relation with the

3. This would imply modifications of the system we have not done yet for lack of time.

head of the structure. The adjuncts are characterized by their formal position (left/right) relative to the nucleus. Adjuncts are optional (a nucleus may occur alone). Apart from these two main categories, we need to introduce an additional element: the *linker*. The **linker** is a special element which builds an endocentric structure with two elements. It usually corresponds to coordination.

An element of the structure (nucleus or adjunct) might possess the **break** property. This notion is introduced in order to manage sequences where adjacent nuclei belong to different structures. When an element has the break property, its presence closes the preceding structure (if any), and opens a new one. The following sentence shows an example of the new categorization (compare with example Section 3):

> In$_{\texttt{O}}$ early$_{\texttt{AB+}}$ trading$_{\texttt{NU}}$ in$_{\texttt{O}}$ Hong$_{\texttt{NU}}$ Kong$_{\texttt{NU}}$ Monday$_{\texttt{NUB+}}$,$_{\texttt{O}}$ gold$_{\texttt{NU}}$ was$_{\texttt{O}}$ quoted$_{\texttt{O}}$ at$_{\texttt{O}}$ \$$_{\texttt{A}}$ 366.50$_{\texttt{NU}}$ an$_{\texttt{AB+}}$ounce$_{\texttt{NU}}$.$_{\texttt{O}}$

Advantage of the new categories: Is such knowledge interesting or useful? One simple experiment can be carried out, using this information in order to compute a new baseline. As for the preceding baseline, each word is categorized into its most frequent category. The baseline reaches now a F1-score of 86%, an improvement of 6%.

Table 6 well illustrates the advantages of the new subcategories. Let us take the element JJ. Its default category is I (84%) in the old categorization. In the new categorization, its default category is left adjunct (AL) with a very high accuracy (99%). It is then really easier to categorize it. In other words, an adjective belongs to an NP if it occurs before a noun (with 1% of exceptions).

TAG	O	B-NP	I-NP	I				
				NU	AL	AR	O	B+
PRP	0%	9%	91%	100%	-		0%	100%
NN	2%	17%	81%	98%	-		2%	3%
JJ	15%	1%	84%	-	99%	-	1%	24%
VBG	87%	6%	10%	-	22%	-	78%	3%

Table 6: Distribution of some elements in the old and new categories.

Another advantage of this categorization concerns the breaker problem. In the new categorization, the break is a property that each category can have, and *it is not a competing categorization*. This corresponds to the following view: One problem is the membership of an element to a given structure (categorization into NU or A). The second problem is to determine whether an element is a breaker or not. The two facts are not competing as they are in the IOB categorization: if an element is a breaker (tagged B in the old categorization), it has to belong to the structure. The tag PRP offers a good example. In the old categorization, it is mostly tagged I (91%), and sometimes B. In the new, it is always tagged N, and always has the break property. The 9% of the tag B correspond to the 9% where the preceding element is an NP. Two rules are then sufficient where a dozen were required with the old categorization.

4.2 The Categorization Process

The general idea for categorizing elements is to use specific contexts which point out some of the distributional properties of the category. The categorization is a sequential process. First the nuclei have to be found out. For each tag of the corpus, we apply the function f_{nu} (equation 2). This function selects a list of elements which are categorized as nuclei. The function f_{b} is then applied to this list in order to figure out nuclei which are breakers. Then the adjuncts are found out, and the function f_{b} (equation 3 and 4) is also applied to them to figure out breakers.

Since the corpus does not contain information about these distributional categories, ALLiS has to figure them out. We do not want to introduce them into data since we still want to compare our approach with others using the same data. This categorization relies on the distributional behavior of the elements, and can be automatically achieved using unsupervised learning. We now explain how elements are categorized into these new sub-categories.

4.2.1 Categorization of Nuclei

The context used to find out the nuclei relies on this simple following observation: A structure requires at least one nucleus[4]. The pattern used in order to determine nuclei is:

$$f_{\text{char}}(X) = \frac{\sum_C [X]}{\sum_C X} \tag{1}$$

$\sum_C P$ corresponds to the number of occurrences of the pattern P in the corpus C. The brackets are used in order to delimit the structure (NP for example). [PRP] means that the structure is composed of only one word tagged PRP. Elements that occur alone in a structure are assimilated to nucleus, since a structure requires a nucleus. For example, the tags PRP (pronoun) and NNP (proper noun) may compose alone a structure (respectively 99% and 48% of these tags appear alone in NP) but the tag JJ appears rarely alone (0.009%). A threshold tuned empirically is used in order to discriminate nuclei from adjuncts. 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) often[5] 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 categorization of nuclei is decomposed into two steps. First we identify characteristic adjuncts, i.e. adjuncts which can not appear alone since they depend on a nucleus. An effect of the function f_{char} is to provide 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 categorized as a characteristic adjunct. The same function is used for determining nuclei and adjuncts. In the first case, the function has to be greater than a given threshold, in the second case, lesser than a given second threshold. 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

4. The partial structures (structures without nucleus) represent 2% of all the structures in the training corpus, and then introduce a little noise.
5. At least, in the training corpus.

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:

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

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

The second step consists in 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.

$$f_{\text{nu}}(X) = \frac{\sum_C [\ A_{\text{char}} *\ \ X\ A_{\text{char}} *\ ]}{\sum_C X} \tag{2}$$

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 7).

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

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

[_ NU]	for the left adjuncts
[A_l* NU _]	for the right adjuncts

If an element appears in place of the underscore, and is not already categorized as nucleus, then it is categorized as adjunct. Once the left adjuncts are found out, they can be used for the categorization of the right adjuncts. They thus appear in the context as optional elements (this is helpful to capture circumpositions).

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

[_ A_l NU]	for the left adjuncts of A_l
[a_l* A_l _ NU]	for the right adjuncts of A_l

The contexts are similar for adjuncts of right adjuncts.

4.2.3 Linkers

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

[NU _ NU]	linker of nuclei
[A _ A NU]	linker of left adjuncts
[NU A _ A]	linker of right adjuncts

Elements which occur in these contexts but which have already been categorized as nucleus or adjuncts are deleted from the list. Linkers found in the training data are the tags *CC*, *TO*, and *,* (comma).

4.2.4 Break

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 DT JJ NN introduces a break before the tag DT, although the 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 market/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]** began/VBD in/IN [1980/CD] ...

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

In order to detect which tags have the break property, we build up two functions $f_{\text{b left}}$ and $f_{\text{b right}}$ described equations (3) and (4) (NU:{nuclei}, C_{wb}: corpus without brackets).

$$f_{\text{b left}}(X) = \frac{\sum_C NU\ ]\ [\ X}{\sum_{C_{\text{wb}}} NU\ X} \tag{3}$$

$$f_{\text{b right}}(X) = \frac{\sum_{C} X\ ]\ [\ NU}{\sum_{C_{\text{wb}}} X\ NU} \tag{4}$$

These functions compute the break property for the element X. Table 8 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	$f_{\text{b left}}$	$f_{\text{b right}}$
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 8: Values of the functions for some elements.

5. Second Experiment: Using Prior Knowledge

We now present new results using this prior knowledge. Table 9 shows that the use of this prior knowledge improves results (+0.41). The most significant point is that the system using knowledge and a context of one element outperforms all previous results even with a context composed of two elements. A longer context does not improve results.

	K	TBL	w/o K	best old
θ	0.50		0.5	0.50
Context	1	2	1	2
wo words	91.10	90.60	90.07	90.70
with words	92.59	92.03		92.18

Table 9: Results using prior knowledge (K).

One other advantage is the reduction of the learning time. Using this knowledge, which constrains strongly the search space, the learning process is 10 times faster.

In order to illustrate the advantage of the new categorization, we take a few examples of errors the TBL system produces and ALLiS does not.

1 are still {developed/JJ} but/CC

2 {the/DT buy-out/NN just/RB $/$ 15/CD millions/CD }

3 {late/JJ} {last/JJ year/NN}

4 {creditors/NNS early/JJ} {next/JJ month/NN}

The error (1) is due to the fact that the TBL system has to learn each context in which the element JJ is not tagged I (I being the default category automatically assigned to JJ). The problem would have been similar for ALLiS without using the new categorization: in this context, ALLiS can not assign the AL category since no nucleus occurs after the adjective. The errors (2) and (3) illustrate cases where a break is missing or wrong. The separation of the problem inside/break seems to provide better results, most of the errors made by TBL and not by ALLiS being of this kind. The error (4) is also due to a problem of breaker. Suppose ALLiS does also recognize *next* as breaker after *early*. The second chunk would be still wrong, but the first one (*creditors*) would be right, since the word *early/JJ* would be categorize as left adjunct. And since a left adjunct can not itself compose a chunk, no NP will be recognized with *early*. ALLiS' output would be: {creditors/NNS} early/JJ {next/JJ month/NN}.

The general conclusion we can draw is that a more precise distributional categorization provides better results than a general one (using only I and B) for this task.

6. Shared Task CoNLL'00

Table 10 presents the complete results of the shared task of CoNLL'00 (chunking). The task itself and the data are described in Sang (2000a). ALLiS provides state-of-the art results. All the systems do not use exactly the same features, in particular the size of the left context varies from three words (ALLiS), five in Sang (2000b), six in Koeling (2000), until eight in some cases in van Halteren (2000). A comparison using exactly the same context would discriminate more properly the different approaches. We can note that, even with the smallest context (three word), ALLiS performs better than some systems with larger contexts like Maximum Entropy used in Koeling (2000).

Note also that three out of four systems performing better than ALLiS use a system combination or voting approach. The only "simple" system performing better than ALLiS is the one using the Support Vector Machine (see Kudoh and Matsumoto, 2000). They use a context of two words after and before the current word, a context larger than this used by ALLiS, but they also add in the left context information about the chunk tag. This information is dynamically computed during the tagging step since the chunk tags are not given by the test data. Such information can not be currently used by ALLiS, and its integration would require a modification not implemented for the time being, but scheduled. Experiments done by others participants show that this information improves results.

Nerbonne et al. (2001) presents other systems that were applied on based-NP chunking. We can note that ALLiS outperforms decision tree, well known for handling noisy data (see Quinlan, 1986).

6.1 Error Analysis

The errors are the same as those found in the others systems: mainly coordination and noise. The ten first errors for NP are: (the correct NP is underlined)

- <u>fixed/VBG [leading/edges]</u>
- <u>[an/DT aerospace/NN],/, [electronics/NNS] ,/, automotive/JJ and/CC [graphics/NNS concern/VBP[6]]</u>

test data	precision	recall	$F_{\beta=1}$
Kudoh and Matsumoto	93.45	93 51	93.48
Van Halteren	93.13	93.51	93.32
Tjong Kim Sang	94.04	91.00	92.50
Zhou, Tey and Su	91.99	92.25	92.12
ALLiS	91.87	92.31	92.09
Koeling	92.08	91.86	91.97
Osborne	91.65	92.33	91.64
Veenstra and Van den Bosch	91.05	92.03	91.54
Pla, Molina and Prieto	90.63	88.25	85.76
Johansson	86.24	88.25	87.23
Vilain and Day	88.82	82.91	85.76
baseline	72.58	82.14	77.07

Table 10: CoNLL'00 shared task: chunking

- [SHEARSON/NNP LEHMAN/NNP HUTTON/NNP Inc/NNP] ./.
- [development/NN and/CC property/NN management/NN]
- [its/PRP$ total/JJ loan/NN] and/CC[real/JJ estate/NN reserves/NNS]
- [the/DT pilots/NNS] and/CC [management-led/JJ buy-out/NN]
- [Republic/NNP Airlines/NNPS and/CC Flying/NNP Tiger/NNP]
- [US/PRP] [Facilities/NNP Corp./NNP] (3 times)

These examples illustrate well the two main problems: error of tagging and the coordination structure as Table 11 confirms it. The main source of errors is due to errors of tagging or errors of bracketing (the structure extraction from the original corpus is not trivial). Among the other errors, we find the traditional structures that are problematic (gerund, apposition). For these errors, the length of the context plays a small importance and its increase provides no improvement.

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 11: Typology of the 200 first errors.

6.2 What Kind of Rules?

We here describe several peculiar kinds of rules the system generated. The first kind is what we called *repair rules*: rules which fix errors, generally errors of tagging.

```
<RULE S='I-NP' ACC='1.00' FREQ='17'>
 <W C='IN' W='the'/>
</RULE>

<RULE S='I-NP' ACC='0.89' FREQ='8'>
 <W C='NNS' LEFT='1'/>
 <W C='VBP' />
 <W C='VBD' RIGHT='1'/>
</RULE>
```

The first rule is a trivial example of wrong rule learned by the tagger. The second rule deals with the problematic of the noun/verb distinction. In this context, a finite verb (VBP) belongs to an NP. This corresponds to an error of tagging: the word is tagged VBP but has to be tagged NN, as in the following example:

> [...] a/DT federal/JJ appeals/NNS court/VBP vacated/VBD an/DT earlier/JJR summary/NN judgment/NN [...]

The frequency of these rules is generally low (usually lesser than 5). This explains why it is important to learn rules with low frequency (Section 3). They *repair* noise.

It is then very important to use the same tagger used for tagging training data since a different tagger would not generate exactly the same errors. This might be a way to improve a specific tagger by learning its errors. We estimate that around 10% of the errors can be learned and fixed.

Second kind of rules: the lexicalized rules. As said in Section 3, the majority of these rules have a low frequency. They generally include the feature W (the word itself), as the following rule shows.

```
<RULE S='I-NP' FREQ='12'>
 <W C='VBG' W='operating'/>
 <W C='NN' RIGHT='1'/>
</RULE>
```

This rule is learned because of the presence of frequent terms such as *operating system* and *chief operating officer*. It seems then that the utility of these lexicalized rules strongly depends on the domain of the corpus used as training data, such rules having little possibility of generalization. Rules without lexicalized items depend less on the corpus.

7. Related Work

We can find in the literature two systems using explicitly or not this notion of exceptions: TBL, already presented Section 3 and FOIDL presented in Mooney and Califf (1995). We present now FOIDL.

FOIDL is based on FOIL, (see Quinlan, 1990), and differs from it in two points: it employs intentional background knowledge, and avoids the need for explicit negative examples, using the output completeness hypothesis. The system outputs a decision list in the reverse order. But the most interesting point for us is the way this decision list is used in order to learn exceptions. The argument in favor of this reverse order is that the first rules learned tend to be more general and can be considered as defaults rules, and can then be positioned towards the end of the list (see Webb and Brkic, 1993). FOIDL uses this mechanism in order to learn exceptions.

The algorithm generating exceptions is quite different to ALLiS' one. Whenever a learned rule (or *clause* in the FOIDL terminology) does not meet the minimum clause-accuracy threshold (similar to our θ), the positive examples covered by the rule are prepended in the decision list. It is unclear whether the frequency of these examples has to be greater than 1, the minimum number of examples that a rule must cover[7]. If the minimum clause-accuracy threshold is met, exceptions can be learned using this algorithm: if the negatives covered by the rule are all examples that are covered by previously learned rules, FOIDL returns them to the set of positives examples to be covered by subsequently learned rules. As well as ALLiS, exceptions to rules are processed through a simple modification of the set of positive and negative examples, even if the modifications are not comparable. But The mechanism used in FOIDL postpones the learning of the exceptions to the rule, and does not explicitly link both as ALLiS does.

Note that the threshold called the minimum clause-accuracy threshold, which corresponds to our θ, is also set to 0.50, but without further explanations. We empirically showed (Section 3) that the best result is obtained with this value. FOIDL also includes a parameter for the minimum number of examples that a rule must cover, also set to 2. As ALLiS, the algorithm can also learn exception to the exception to the rule. Unlike ALLiS there is no explicit relation between a rule and its exceptions.

Due to computational reasons, we were unfortunately unable to train FOIDL with our training data, which is too large. Mooney and Califf (1995) already mentioned memory limitations for some experiments, and used only 500 examples as training data, when our training data contains 500,000 words. This problem seems to be mainly due to the mechanism used for handling output completeness hypothesis. Remember that FOIDL learns first-order decision lists and uses Inductive Logic Programming methods, when ALLiS 'only' learns propositional rules (attribute-value formalism). ILP formalism is far more powerful than propositional one.

If we go back to TBL, it can also be viewed as handling exceptions in a similar way FOIDL does. Once a rule is learned, this rule is applied to the training data, potentially generating errors. Then subsequent rules can transform (correct) these errors. These subsequent rules can be considered as exceptions to the rule generating the errors.

We think it is important to stress that, among many differences, these three systems, ALLiS, FOIDL and TBL share a common point at high-level: working on Natural Languages, where noise and/or exceptions in data are quite frequent, they had to develop an exception mechanism for handling these phenomena.

7. In the specific application described in Mooney and Califf (1995), learning past-tense of English verbs, this minimum threshold is validated, but for the chunking task, exceptions occurring only once in the data degrade results.

8. Conclusion and Further Work

We presented in this article ALLiS, a rule induction system. Its main specificity is refinement which allows the handling of noise and exceptions in data.

We clearly show firstly that refinement improves rules induction, and second, that the use of prior knowledge improves the accuracy of the rules in the case of linguistic structures. This remark may seem trivial, but since, in practice, such knowledge is not used (or so rarely), it may be useful to point it out.

One question that comes up is whether the use of this background knowledge is also useful with other systems or not, especially with systems that perform better than ALLiS (see Sang, 2000a). Does improvements due to the linguistic knowledge only concern "easy" structures that ALLiS can not learn and other can? A comparison with other results shows that all the systems have problems with similar structures (noise and coordination). We can point out that some of these systems generally use a combination of several representation schemas, and not only the simple I, O, B categories. We may wonder whether this representation introduces information that can be redundant with the knowledge ALLiS uses.

Except for this refinement mechanism, ALLiS is a basic inductive system with large room for improvements, in particular during the selection of literals (Section 2.3). In CoNLL'00 data, the number of possible literals is small (only POS tag and words) and does not require any sophisticated selection algorithm. In some applications where the number of literals is larger, a selection algorithm would have to be integrated into ALLiS.

Another improvement directly concerns the refinement mechanism. Instead of using a threshold for the mother rule and then learning exceptions, it would be useful to estimate the accuracy of the set {r + exceptions}. This set will be kept if and only if its accuracy is greater than a threshold. Otherwise, the rule *r* would not be kept, its exceptions being judged too difficult to learn.

Acknowledgments

This research was funded by the TMR network Learning Computational Grammars (`http://www.lcg-www.uia.ac.be/lcg/`).

Appendix A

Table 12 gives the list of the Upenn POS tags used along this article.

References

Steven Abney. Parsing via finite-state cascades. In *Proceedings of the ESSLLI'96 Robust Parsing Workshop*, 1996.

Aït-Mokhtar, Jean-Pierre Chanod, and Claude Roux. A multi-input dependency parser. In *Seventh International Workshop on Parsing Technologies*, Beijing, 2001.

#	pound sign (British currency)	PDT	predeterminer (e.g. half the, all the)
$	dollar sign	POS	's
"	close double quote	PRP	personal pronoun
(	open paren	PRP$	possessive personal pronoun
)	close paren	RB	adverb
,	comma	RBR	comparative adverb
.	full stop	RBS	superlative adverb
:	colon, dash	RP	particle
CC	coordinator	SYM	miscellaneous symbol
CD	number	TO	infinitival 'to'
DT	determiner	UH	interjection
EX	there in *there is*	VB	verb, uninflected (after modal, 'to')
FW	foreign word	VBD	verb, past-tense form
IN	preposition, complementizer	VBG	verb, present participle form
JJ	adjective	VBN	verb, past participle form
JJR	comparative adjective	VBP	verb, present plural
JJS	superlative adjective	VBZ	verb, present 3
LS	list item enumerator	WDT	wh determiner
MD	modal	WP	wh pronoun
NN	singular common noun	WP$	wh possessive pronoun
NNP	singular proper noun	WRB	wh adverb
NNPS	plural proper noun		
NNS	plural common noun		

Table 12: Upenn Treebank tagset used in this article

Kamal M. Ali and Michael J. Pazzani. Reducing the small disjuncts problem by learning probabilistic concept descriptions. In *Computational Learning Theory and Natural Learning Systems*, volume III: Selecting Good Models, pages 183–199. MIT Press, 1995.

Shlomo Argamon, Ido Dagan, and Yuval Krymolowski. A memory-based approach to learning shallow natural language patterns. In *COLING'98, Montréal*, 1998.

Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565, 1995.

C.A. Brunk and M.J. Pazzani. An investigation of noise-tolerant relational concept learning algorithms. In L. Birnbaum and G. Collins, editors, *Proceedings of the 8th International Workshop on Machine Learning*, pages 389–393. Morgan Kaufmann, 1991.

Claire Cardie. Integrating case-based learning and cognitive biases for machine learning of natural language. *Journal of Experimental Theoretical Artificial Intelligence*, 11(3): 297–337, 1999.

Walter Daelemans, Antal van den Bosch, and Jakub Zavrel. Forgetting exceptions is harmful in language learning. *Machine Learning*, 34(1–3):11–41, 1999.

Hervé Déjean. Using ALLiS for clausing. In Walter Daelemans and Rémi Zajac, editors, *Proceedings of CoNLL-2001*, pages 64–66. Toulouse, France, 2001.

Claire Grover, Andrei Mikheev, and Colin Matheson. *LT TTT version 1.0: Text Tokenisation Software*, 1999. http://www.ltg.ed.ac.uk/software/ttt/.

L. Holder. Unifying empirical and explanation-based learning by modeling the utility of learned knowledge. In *Proceedings of the ML92 Workshop on Compilation and Speedup Learning*, 1992.

Robert Holte, Liane Acker, and Bruce Porter. Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.

Lauri Karttunen, Tamás Gaál, and André Kempe. Xerox finite-state tool. Technical report, Xerox Research Centre, Grenoble, 1997.

Rob Koeling. Chunking with maximum entropy models. In Claire Cardie, Walter Daelemans, Claire Nedellec, and Erik Tjong Kim Sang, editors, *Proceedings of CoNLL-2000 and LLL-2000*, pages 139–141. Lisbon, Portugal, 2000.

Taku Kudoh and Yuji Matsumoto. Use of support vector learning for chunk identification. In Claire Cardie, Walter Daelemans, Claire Nedellec, and Erik Tjong Kim Sang, editors, *Proceedings of CoNLL-2000 and LLL-2000*, pages 142–144. Lisbon, Portugal, 2000.

M. P. Marcus, B. Santorini, and M. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

Raymond J. Mooney and Mary Elaine Califf. Induction of first-order decision lists: Results on learning the past tense of english verbs. *Journal of Artificial Intelligence Research*, 3: 1–24, 1995.

John Nerbonne, Anja Belz, Nicola Cancedda, Hervé Déjean, James Hammerton, Rob Koeling, Stasinos Konstantopoulos, Miles Osborne, Franck Thollard, and Erik F. Tjong Kim Sang. Learning computational grammars. In Walter Daelemans and Rémi Zajac, editors, *Proceedings of CoNLL-2001*, pages 97–104. Toulouse, France, 2001.

Michael Pazzani and Dennis Kibler. The role of prior knowledge in inductive learning. *Machine Learning*, 9:54–97, 1992.

F. Provost and V. Kolluri. A survey of methods for scaling up inductive learning algorithms. In *Third International Conference on Knowledge Discovery and Data Mining*, 1997.

J. R. Quinlan. The effect of noise on concept learning. In J. G. Carbonell and T. M. Mitchell, editors, *Machine learning: An artificial intelligence approach*, volume 2. Morgan Kaufmann:San Mateo CA, 1986.

J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

Lance A. Ramshaw and Mitchell P. Marcus. Text chunking using tranformation-based learning. In *ACL Third Workshop on Very Large Corpora*, pages 82–94, 1995.

Erik F. Tjong Kim Sang. Introduction to the CoNLL-2000 shared task: Chunking. In Claire Cardie, Walter Daelemans, Claire Nedellec, and Erik Tjong Kim Sang, editors, *Proceedings of workshop on Computational Natural Language Learning, CoNLL-2000 and LLL-2000*, pages 127–132. Lisbon, Portugal, 2000a.

Erik F. Tjong Kim Sang. Text chunking by system combination. In Claire Cardie, Walter Daelemans, Claire Nedellec, and Erik Tjong Kim Sang, editors, *Proceedings of CoNLL 2000 and LLL-2000*, pages 151–153. Lisbon, Portugal, 2000b.

Khalil Sima'an. Explanation-based learning of data oriented parsing. In *Computational Natural Language Learning (CoNLL), ACL/EACL-97, Madrid*, 1997.

Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1–3):233–272, 1999.

Hans van Halteren. Chunking with WPDV models. In Claire Cardie, Walter Daelemans, Claire Nedellec, and Erik Tjong Kim Sang, editors, *Proceedings of CoNLL-2000 and LLL-2000*, pages 154–156. Lisbon, Portugal, 2000.

C.J. van Rijsbergen. *Information Retrieval*. Buttersworth, 1979.

G. Webb and N. Brkic. Learning decision lists by prepending inferred rules. In *Proceedings of the Australian Workshop on Machine Learning and Hybrid Systems*, 1993.