Using ALLiS for Clausing

Hervé Déjean
Seminar für Sprachwissenschaft
Universität Tübingen
dejean@sfs.nphil.uni-tuebingen.de

Abstract

We present the result of a symbolic machine learning system, ALLiS 2.0 for the CoNLL-2001 shared task. ALLiS 2.0 is a theory refinement system using hierarchical data. Results are F=89.04 for subtask 1, F=68.02 for subtask 2 and F=67.70 for subtask 3 (development test). Adding manual rules improves considerably results specially for task 2 (F=79.44). For the test data, results are slightly worst (F=62.27 for subtask 3).

1 Introduction

ALLiS (Architecture for Learning Linguistic Structure) (Déjean, 2000a), (Déjean, 2000b) is a symbolic machine learning system. The learning system is based on theory refinement. It tries to refine (to improve) an existing imperfect grammar using operators such as contextualization and lexicalization. ALLiS separates the task of the generation of rules and the task of the use of these rules (task of parsing). First symbolic rules are learned and saved using an own formalism, and in a second time, these rules are converted into a proper formalism used by a specific rule-based parser. ALLiS uses XML formalism for learning as well as for parsing. The following XML Components are:

- LT TTT, a text tokenization system and toolset which enables users to produce a swift and individually-tailored tokenization of text.
- LT XML , an integrated set of XML tools and a developers tool-kit, including a C-based API.
- XMLQUERY, (Mckelvie, 2000), an extension of the LTXML query language to allow more complex queries including operators for finding sequences of XML elements.

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 (Mooney, 1993). (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). 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 in acquiring an initial grammar (or more generally a knowledge base). In this work, the initial grammar is automatically induced 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.

3 Hierarchical Data

The difference between ALLiS 1.0 and ALLiS 2.0 relies on the use of hierarchical structure. The alternate usual solution is to replace a segment (chunk for example) by a unique element (a word, namely the head of the structure). The advantage of this solution is to reduce the search space, the drawback to delete some potential useful information.

3.1 Data representation

Data provided by (Tjong Kim Sang and Déjean, 2001) are first converted into ALLiS' input formalism. The three hierarchical levels are S (sentence), PHR (phrase), and W (word). Here is the DTD used for training data:

```
<!ELEMENT WSJ (S)*>
<!ATTLIST WSJ S CDATA "0">

<!ELEMENT S (S|PHR|W)*>
<!ATTLIST S C CDATA ""
            NUM CDATA "">
<!ELEMENT PHR (W)* >
<!ATTLIST PHR CAT CDATA "0"
              B CDATA "">

<!ELEMENT W EMPTY>
<!ATTLIST W W CDATA ""
            BP CDATA ""
            B CDATA ""
            C CDATA "">
```

An example of data:

```
<S NUM='38'>
 <PHR CAT='A'><W CAT='A'/></PHR>
 <PHR CAT='NP' B='Y'>
  <W BP='B' C='VBN' W='Estimated'/>
  <W BP='I' C='NN' W='volume'/>
 </PHR>
 <PHR CAT='VP' B='N'>
  <W BP='B' C='VBD' W='was'/>
 </PHR>
 <PHR CAT='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 CAT='WORD' B='N'>
   <W C='.' W='.'/>
 </PHR>
 <PHR CAT='E'> <W S='E'/></PHR>
</S>
```

Two PHR elements are added, marking up beginning and end of sentence. Information provided by the original corpus (word, POS-tag, chunk-tag) are integrated in the XML corpus by the way of attributes. The attribute B carried on by PHR element corresponds to the boundary clause we are looking for. Values are Y or N.

4 Learning

The learning method consists in finding contexts in which an element can be associated to a specific category with a high confidence. The following query returns the list of the elements PHR with the category NP which are clause beginning (B='Y').

```
WSJ/S/PHR[CAT='NP', B='Y']
```

We now try to find negative examples, and evaluate the ratio between positive and negative examples. Extending a query consists in extending a tree using a breath-first path. First neighbors are added (left and right), and then attributes are added to each new element.

```
WSJ/S/(PHR[CAT='PP'],PHR[CAT='NP'])
```

The extension lasts until the accuracy of the rule is high enough or until the frequency of the sequence is high enough. Here is one of the possible final rules:

```
<query FREQ='11' ACC='0.92'>
<save>
<PHR CAT='PP'>
<W W='for' C='IN'/>
</PHR>
<PHR B='Y' CAT='NP'>
<W C='NNP' CUR='N'/>
</PHR>
<PHR CAT='VP'>
<W BP='B' W='to' C='TO'/>
</PHR>
</save>
</query>
```

In a more readable formalism:

```
if left elt  = PP/IN/for
   right elt = VP/TO/to
   [np NNP] -> S
```

If a NP with an NNP occurs between a PP containing IN/for and a VP containg TO/to, it is a start of clause.

5 Rules

For data *test1*, ALLiS has learned 149 rules, the coverage of the learning test being 82% (82% of the positive cases are explained). As usually, few very frequent rules insure the main part of the coverage.

Concerning the second data, rules are more difficult to learn. ALLiS only generates 25 rules and the recall is very low.

5.1 Which precision?

Another important parameter is the threshold θ which determines when the accuracy of a rule is high enough. We tried two values: 0.8 and 0.9. For the chunking task (last year shared-task), the best value was 0.9. In this task, the best one is 0.8. We think that the explanation is partially due to a more important quatity of noise in data. For the chunking task, the noise was due to POS tagging. In this data, it is due to errors of tagging and also errors of chunking. The F-score being around 92% for chunking, it is obvious that the error rate for some tags is higher than 10%. We can note that, even if the threshold is set up at 0.8, the overall precision stays high enough (94.08% against 95.76% for θ=0.9). Furthermore, setting θ at 0.8 allows a better recall (84.59% against 78.58%). *It then seems that θ's value should be correlated with the estimation of the noise in data.*

6 Result

For CoNLL'01 shared-task 1, ALLiS offers a good precision but a lower recall. Using a window of one element before and after seems to be sufficient. Concerning subtask 2, the result is unfortunately very close to the baseline. It is clear that this task does not need local information, but information about the existence of opened clauses. In order to validate this affirmation, we add a new rule which consists in adding a tag `B='E'` at the last word of the sentence when two clauses are open. This improves greatly recall (72.76) and precision keeps high enough (87.48) to improve the overall result (F=79.44).

Results of the task 3 are provided by combining task1 and 2 and using a Perl script provided by Erik Tjong Kim Sang.

developement	precision	recall	$F_{\beta=1}$
part 1	94.08%	84.59%	89.08
part 2	99.28%	51.73%	68.02
part 3	73.93%	62.44%	67.70

test	precision	recall	$F_{\beta=1}$
part 1	93.76%	81.90%	87.43
part 2	99.04%	48.90%	65.47
part 3	72.56%	54.55%	62.27

Table 1: ALLiS results

References

Andreas Abecker and Klaus Schmid. 1996. From theory refinement to kb maintenance: a position statement. In *ECAI'96*, Budapest, Hungary.

Clifford Alan Brunk. 1996. *An investigation of Knowledge Intensive Approaches to Concept Learning and Theory Refinement.* Ph.D. thesis, University of California, Irvine.

Hervé Déjean. 2000a. Theory refinement and natural language learning. In *COLING'2000*, Saarbrücken.

Hervé Déjean. 2000b. A use of xml for machine learning. In *Proceeding of the workshop on Computational Natural Language Learning, CONLL'2000.*

David Mckelvie, 2000. *XML QUERY 2.0.* Edinburgh. http://www.ltg.ed.ac.uk/software/ttt/.

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.

Erik F. Tjong Kim Sang and Hervé Déjean. 2001. Introduction to the conll-2001 shared task: Clause identification. In *Proceedings of CoNLL, shared task.*