In: *Proceedings of CoNLL-2000 and LLL-2000*, pages 133–135, Lisbon, Portugal, 2000.

# Learning Syntactic Structures with XML

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

## 1 Introduction

We present the result of a symbolic machine learning system for the CoNLL-2000 shared task. This system, ALLiS, is based on theory refinement. In this paper we want to show that XML format not only offers a good framework to annotate texts, but also provides a good formalism and tools in order to learn (syntactic) structures.

## 2 ALLiS

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 contextualisation and lexicalisation (Section 4). 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. The rules generated by ALLiS contain enough contextual information so that their conversions into other formalisms is possible (Déjean, 2000b).

We present here the three tools we tested in order to parse texts.

**CASS** : the CASS system (Abney, 1996) provides a very fast parser which uses Regular Expression Grammar. The inconvenient of this system is twofold: first CASS only handles tags. Some pre- and post-processings are needed. Second, it is impossible to use contextual information. Using CASS, it is thus difficult to implement rules generated by the two refinement operators used in ALLiS: contextualisation (introduction of contexts) and lexicalisation (use of the word level).

**XFST** : the Xerox finite State Tool (Karttunen et al., 1997) offers a rich finite state formalism (operator of contextualisation, replace operator). If the formalism is powerful enough, the main problem with XFST is the number of transducers generated by ALLiS. For each context and each word occurring in the rules learned by ALLiS corresponds a transducer. The size of the final Regular Expression Grammar is quite big and the compilation and processing time keeps low. A phase of optimisation would be required.

**LT TTT** : the last tool tried, LT TTT (Grover et al., 1999), is a text tokenisation system and toolset. One of the tools, *fsgmatch* reads a grammar (also written in XML format) which is used to annotate/parse XML texts. Using XML properties, the grammar has easily access to all the levels of the document (word, tag, phrase, and higher structures). The manipulation/annotation of several structures (as in the CoNLL-2000 shared task) is very easy. LT TTT is a good trade-off between the rapidity of CASS and the rich formalism of XFST.

We now explain how the LT TTT is used during the learning and the parsing task.

## 3 Data and Learning Algorithm

The training corpus is an XML document where each structure is marked up (Table 1). Training corpus can contain other structures or information. Using XML tools, we can choose which information we want to use. The DTD used by ALLiS (Table 2) describes a document composed of words (W) which compose phrases (PHR), and a sequence of phrases composes a sentence (S). This DTD should be updated in order to take into account intermediate levels

```
<S>
<PHR C='NP'>
<W C='NNP'>Mr.</W>
<W C='NNP'>Percival</W>
</PHR>
<PHR C='VP>
<W C='VBD'>declined</W>
<W C='TO'>to</W>
<W C='VB'>comment</W>
</PHR>
<W C='.'>.</W>
</S>
```

Table 1: Example of training corpus.

```
<!ELEMENT DOCS      (#PCDATA|TEXT)* >
<!ELEMENT TEXT      (S|PHR|W)+ >
<!ELEMENT S         (PHR|W)+ >
<!ELEMENT PHR       (W)+ >
<!ATTLIST PHR       C     CDATA  "" >
<!ELEMENT W         (#PCDATA)>
<!ATTLIST W         C     CDATA  ""
                    S     CDATA  ""
                    BK    CDATA  ""
                    CAT   CDATA  "" >
```

Table 2: DTD used for the training corpus.

between the phrase and the sentence.

The learning method consists of finding contexts in which an element (tag or word) can be associated to a specific category with high confidence. Some elements do not need contexts and are themselves confident. In the case an element requires contexts, these contexts are computed by the use of queries which examine training corpus. XML Path Language provides an easy way of addressing nodes of an XML document. For example, the query `/TEXT/S/PHR[C='NP']/W` determines elements W directly under an entity PHR with the attribute C='NP', under the entities S and TEXT. A query returns the indicated items one by one until the set denoted by the query is exhausted. The above query returns all the elements occurring in a phrase marked NP.

The next section provides some examples of queries used during the learning task. (Grover et al., 1999) describes the syntax of the LT XML query language.

## 4 Examples of LT XML Queries

The default category of a tag is given by the ration between its number of occurrences in the structure we want to recognise and the number of occurrences in the training corpus. The two following queries compute this ration for the tag *VBG* and the NP structure:

```
.*/(PHR[C='NP']/W[C='VBG'])
.*/W[C='VBG']
```

The first query can be read as follows: element W with the attribute `C='VBG'` occurring in an entity PHR with the attribute `C='NP'`, this last structure occurring anywhere in the document. The second query counts all the occurrences of the entity W with the attribute `C='VBG'` anywhere in the document. If the ration is higher than a threshold $\theta$, then the tag is considered as belonging to the structure. If not, we have to use operators of specialisation.

The principle consists of enriching the query so that the ratio $\theta$ is higher than a given threshold. Contextualisation introduces left and/or right elements in the query. Lexicalisation uses the word value and not only the tag. Since the tag VBG is not reliable (it mainly occurs in an VP structure), we have to look for larger contexts. The following query returns the list of the elements W occurring on the left of an element `W[C='VBG']` in the same phrase NP (`PHR[C='NP']`).

```
PHR[C='NP']/(W!,W[C='VBG'])
```

The tag DT is one the elements of this list. We now try to find negative examples. The second query looks for all the occurrences of the entity `W[C='VBG']` which occur outside the NP structure in the same context: a tag DT occurring before the tag VBG.

```
PHR[C='NP']/W[C='DT'],W[C='VBG'])
```

Using the result of these two queries, a new ratio is computed to determine whether the category of VBG is reliable in this context (the answer is affirmative). For some tags, the ratio still remains below the threshold.

The last query shows that we can also have access to the word itself: it looks for the word *operating* tagged VBG inside a phrase categorised as NP:

```
PHR[C='NP']/W[C='VBG']/#='operating'
```

Contexts in which the word occurs outside the structure are also built, and the ratio is computed in the same was as above.

## 5 Parsing with fsgmatch

Once rules are learned, they are converted into the formalism of the parser. Table 3 provides an example of rules used by *fsgmatch* (Grover et al., 1999). The rule consists of adding the attributes `CAT='AL' S='NP'` (left adjunct of a Noun Phrase(NP)) to each word with the attribute `C='VBG'` which occurs after a word with the attribute `C='DT'`.

```
<RULE name="AL" targ_sg="@[CAT='AL'
                              S='NP']">
<REL match="W[C='DT'
              m_mod='TEST'
              S='NP']"> </REL>
<REL match="W[C='VBG']"></REL>
</RULE>
```

Table 3: An example of *fsgmatch* rule.

```
<PHR C='PP'>
  <W CAT='N' C='IN'>Under</W>
</PHR>
<PHR C='NP'>
<W S='NP' BK='L' CAT='AL' C='DT'>the</W>
<W S='NP' CAT='AL' C='VBG'>existing</W>
<W S= 'NP' CAT='N' C='NN'>contract</W>
</PHR>
<W C=','>,</W>
```

Table 4: Example of output.

For each structure, words are marked up with their categories, and then a rule inserts an entity `PHR` with an attribute `C` which corresponds to the name of the structure. The entity `PHR` is inserted by a rule which recognises sequences of entities `W` corresponding to a structure. Table 4 shows an output where two phrases were added: a prepositional one (`<PHR C='PP'>`), and a noun phrase (`<PHR C='NP'>`).

## 6 Ordering Structures

The different structures are learned sequentially. If, in principle, the order used for learning structures is not important, the practice shows that it is better to begin with structures which contain other structures. Once a structure is learned, it is not taken into account during the learning of the next structures. This avoids generating of complementary contexts already learned at the preceding level. For instance, ALLiS first learns NP structure. Once categorisation rules for the tag VBG are learned, ALLiS does not take into account VBG occurring in a NP when it learns following structures. At the VP level, it is thus useless of learn contexts in which VBG does not occur in a VP (cases which mainly correspond to occurrences of VBG in NP). The parsing phase has of course to use the same learning order. The (partial) order used is: NP, VP, ADJP, ADVP, PP, PRT, CONJP, SBAR, INTJ, LST. Table 5 shows evaluation of ALLiS for the CoNLL-2000 shared task.

| test data | precision | recall | $F_{\beta=1}$ |
|---|---|---|---|
| ADJP | 74.26% | 68.49% | 71.26 |
| ADVP | 72.10% | 80.25% | 75.96 |
| CONJP | 100.00% | 55.56% | 71.43 |
| INTJ | 100.00% | 50.00% | 66.67 |
| LST | 0.00% | 0.00% | 0.00 |
| NP | 92.38% | 92.71% | 92.54 |
| PP | 95.57% | 97.86% | 96.70 |
| PRT | 81.43% | 53.77% | 64.77 |
| SBAR | 90.47% | 76.26% | 82.76 |
| VP | 92.48% | 92.92% | 92.70 |
| all | 91.87% | 92.31% | 92.09 |

Table 5: ALLiS results

## References

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

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

Hervé Déjean. 2000b. A Symbolic system for natural language learning. In *CONLL'2000*.

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

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