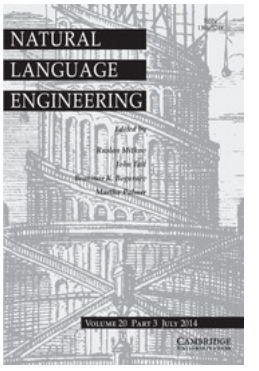

Pattern-based unsupervised parsing method

JESÚS SANTAMARÍA and LOURDES ARAUJO

Natural Language Engineering / *FirstView* Article / June 2014, pp 1 - 26
DOI: 10.1017/S1351324914000072, Published online: 04 June 2014

Natural Language Engineering: page 1 of 26. © Cambridge University Press 2014
doi:10.1017/S1351324914000072

Pattern-based unsupervised parsing method

JESÚS SANTAMARÍA and LOURDES ARAUJO
Lenguajes y Sistemas Informáticos, Universidad Nacional de Educación a Distancia (UNED), Madrid 28040, Spain
email: {jsant,lurdes}@lsi.uned.es

(*Received 19 June 2013; revised 5 May 2014; accepted 7 May 2014*)

Abstract

We have developed a heuristic method for unsupervised parsing of unrestricted text. Our method relies on detecting certain patterns of part-of-speech tag sequences of words in sentences. This detection is based on statistical data obtained from the corpus and allows us to classify part-of-speech tags into classes that play specific roles in the parse trees. These classes are then used to construct the parse tree of new sentences via a set of deterministic rules. Aiming to asses the viability of the method on different languages, we have tested it on English, Spanish, Italian, Hebrew, German, and Chinese. We have obtained a significant improvement over other unsupervised approaches for some languages, including English, and provided, as far as we know, the first results of this kind for others.

1 Introduction

A grammar induction algorithm must infer sentence structures. If the algorithm is unsupervised, this structure can be discovered by identifying patterns that occur at a higher frequency than pure chance would explain (e.g., by randomly shuffling the texts). A common procedure in unsupervised grammar induction (UGI) (Klein and Manning 2005; Bod 2006; Bod 2007) is to start off from the part-of-speech (PoS) tag sequences assigned to the words in the texts to be parsed. This amounts to preprocessing the texts for PoS tagging – something that can be achieved either in a supervised (achieving approximately 97% accuracy) or, if necessary, an unsupervised manner. We follow the same scheme in this work, so the goal of our method will be to identify statistical patterns of PoS tags that provide us with sufficient clues to infer, to a certain extent, the shape of parse trees.

Parse trees are made of *constituents*. Each node in the parse tree corresponds to a constituent. The tree leaves identify the PoS tags. Thus, constructing the parse tree of a sentence is equivalent to identifying all its constituents. Constituents corresponding to nodes from which only leaves hang define a class that plays a special role in our algorithm. Accordingly, we will refer to them as *base constituents* throughout this paper.

Our proposal for unsupervised grammar induction is heuristic. It relies upon a few empirical observations about particular patterns in parse trees which help in the task of identifying constituents. For instance., most sentences exhibit a parse

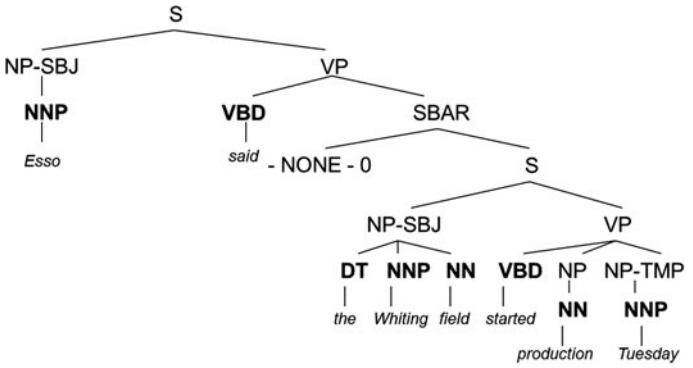

Fig. 1. Parse tree for the sentence *Esso said the Whiting field started production Tuesday* from the Penn Treebank.

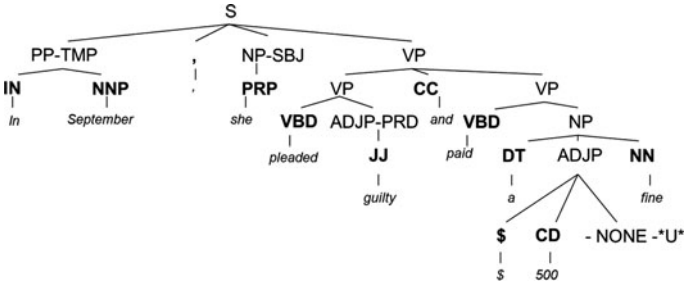

Fig. 2. Parse tree for the sentence *In September, she pleaded guilty and paid a $ 500 fine* from the Penn Treebank.

tree that splits off at the root into two main branches. Also, PoS tag patterns corresponding to constituents occur in texts at a frequency higher than they would do in a random sequence of PoS tags. Certain lexical categories of words (which depend on the language we are parsing) appear much more frequently than others at the beginning or at end of the base constituents. Other lexical categories often appear in specific patterns. For instance, two common patterns are $E\,C_1$ (or $C_1\,E$, depending on the language) or $C_1\,E\,C_2$, where E stands for a PoS tag and C_1 and C_2 for two constituents. The PoS tags E involved in these patterns will prove key to inferring the parse tree.

In order to flesh out these observations, consider the example trees shown in Figures 1–3, extracted from the *Wall Street Journal* (WSJ) section of the Penn

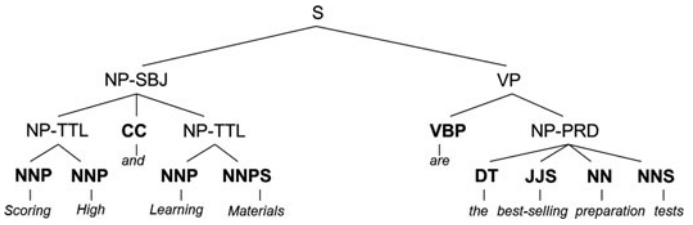

Fig. 3. Parse tree for the sentence *Scoring High and Learning Materials are the best-selling preparation tests* from the Penn Treebank.

Treebank (Tables 11 and 12 in Appendix A lists the PoS tags used to annotate this corpus and the internal tags appearing in the examples, respectively. We can observe that the pattern DT $\cdots$ NN*[1] appears in all three examples. The fact that this pattern occurs at an unusually high frequency in the corpus suggests that it may identify a constituent. Another important observation is that some PoS tags often appear inside these constituents – e.g., JJ* tends to appear between DT and NN* (see Figure 3). Also noticeable is the fact that certain PoS tags tend to divide the sentence into two main parts – e.g., VBD in Figure 1, and VBP in Figure 3. Moreover, some PoS tags – frequently including those that divide sentences – play the role of tag E in constituents with the pattern $E\,C_1$. This is the case for VBD in Figure 1 – which appears twice: in the first occurrence it precedes the sequence DT NNP NN VBD NN NNP, and in the second occurrence, the sequence NN NNP, for PRP and VBD in Figure 2, and for VBP in Figure 3. Finally, tags such as CC in Figure 3 are the E in the pattern $C_1\,E\,C_2$.

Our approach is based on an iterative procedure. We first detect some constituents by statistical analysis and then use this information to identify PoS tags that are typically found in the kinds of common patterns mentioned above. This in turn helps us detect the remainder constituents. In order to test this procedure, we have applied it to different corpora annotated with PoS tags. More precisely, and for the sake of comparison, we have adopted the Klein and Manning (2005) evaluation setting for the problem, as well as those of Bod's (2006, 2007) approach. As far as we know, these or some of their extensions using the same experimental setting, provide the best results obtained so far for unsupervised grammar induction for constituent grammars using a monolingual corpus and PoS-tagged corpora. As in these two cases, the grammar we obtain does not specify the left-hand side of the rules, only the shape of the tree (i.e., the PoS tag sequences defining constituents). Also similar to them, we have used the Penn Treebank (Marcus, Santorini and Marcinkiewicz 1994) as one of our test corpora, employing the syntactic annotations that it provides only for evaluation purposes.

An important remark is that similar patterns are common to many languages. This means that our approach should (to a large extent) be able to parse sentences irrespective of the language. Accordingly, we have tested our model with different languages belonging to different families and showing different degrees of freedom in word order – namely English, German, Spanish, Italian, Chinese, and Hebrew. The results obtained for all of these are surprisingly good, given the simplicity of the approach, and suggest that searching for statistical patterns may be a powerful tool to infer grammars when little is known about the underlying language.

The rest of the paper is organized as follows. In Section 2, we review some current approaches to unsupervised grammar induction. In Section 3, we propose our approach based on PoS tag patterns to identify PoS tag classes, which will later allow us to parse sentences. In Section 4, we provide the procedure for constructing a parse tree, given the PoS tag classes appearing in the sentence. In Section 5, we

[1] XX* stands for any tag comprising XX and any sequence of letters (including the empty string); e.g., NN* can be replaced by any noun (NN, NNP, NNS, NNPS).

report and discuss the results obtained for different languages and compare them to those obtained with other systems. Finally, we put our approach in context and provide directions for future work in Section 6.

2 Related works

The aim of grammar inference is to learn a grammar to model the structure of the sentences of a language. There are two main approaches to this task: supervised and unsupervised. In supervised grammar induction, the grammar and the parsing model are obtained from a treebank of syntactically annotated sentences. There are quite a few works following this approach. In recent works (Petrov 2010), the level of performance has been risen up to 91.8%.

Our work deals with unsupervised empirical grammar induction, though, so we will mainly focus on the work made within this category. An interesting system has been proposed by Carroll and Charniak (1992), who use the Inside-Outside (IO) algorithm to improve the initially calculated estimates of the probabilities of rules, rejecting those with a probability below a small threshold. The authors investigated the reasons for the limited quality of the results achieved and noted that the grammar obtained was very sensitive to the initial estimates for probabilities.

Stolke and Omohundro (1994) induced a grammar by incorporating samples as *ad hoc* rules to a working grammar. Subsequently, elements of the model are merged to achieve generalization and a more compact representation. The choice of what to merge and when to stop is governed by the Bayesian posterior probability of the grammar given the data. The authors conclude that the method requires some refinements to be successfully applied to natural language grammars.

Clark (2000) proposes an algorithm to learn a phrase-structure grammar from tagged text. It clusters sequences of tags together based on local distributional information, and selects clusters that satisfy a mutual information criterion. Sets of tag sequences can be clustered together based on the contexts they appear in. Results show that this criterion selects linguistically plausible constituents.

Klein and Manning (2005) borrow the main ideas from Carroll and Charniak's (1992) work creating a model called Constituent-Context Model (CCM). The model is intended to be applied to PoS tagged texts without considering punctuation marks, and uses the Expectation-Maximization (EM) algorithm to maximize the likelihood of contiguous subsequences of PoS tags. The system has been tested on English, German, and Chinese Treebanks, but only for sentences of up to ten words. The model relies on the idea that long constituents usually have short equivalents in the same contexts, an idea that is also a cornerstone in our proposal. The system achieves an F-measure of 71.1% for English, 64.2% for German, and 36.8% for Chinese. Recently, Golland, DeNero and Uszkoreit (2012) have proposed Log-Linear CCM (LLCCM), a log-linear variant of CCM, which aims to avoid a drastic drop in performance for long sentences.

An alternative system for unsupervised parsing is Bod's (2006) Unsupervised Data-Oriented Parsing (U-DOP). Bod (2006) points out that Klein and Manning's (2005)

approach neglects dependencies that are non-contiguous by using an 'all-substrings' approach. U-DOP directly models structural as well as lexical context without constraining any dependencies beforehand. Subtrees can model both contiguous and non-contiguous lexical dependencies. Accordingly to Bod (2006), this all-subtrees approach is a generalization of Klein and Manning's (2005) all-substrings approach. Bod's system (2006) achieves an F-measure of 82.9% for the English corpus WSJ10 used by Klein and Manning (2005). Since it generates a large amount of subtrees for each phrase, the system becomes impractical for longer sentences. As a result, Bod (2007) describes a method that generates far fewer subtrees for each phrase, but at the cost of lowering the performance. This new method achieves 77.9% on the WSJ10 corpus.

Other related works are those of Magerman and Marcus (1990) and van Zaanen (2000). These works aim to learn a representation of the constituent boundaries of the language. The work by Magerman and Marcus (1990) uses mutual information statistics, while the work by van Zaanen (2000) is based on the alignment of sentences sharing some part.

Seginer (2007a) has proposed an unsupervised parser from plain text which uses a local, greedy parsing algorithm. The parser is incremental and uses a new link representation for the syntactic structure. This representation allows a prefix of an utterance to be parsed before the full utterance has been read. This parser improves on previously published results for unsupervised parsing from plain text, achieving an F-measure of 75.9% for the English corpus WSJ10.

Recently, Spitkovsky, Alshawi and Jurafsky (2011) have achieved significant improvement over their base model for unsupervised dependency parsing by taking the advantage of punctuation marks and capitalization (Spitkovsky *et al.* 2012). These results have been taken into account in the design of our method for constituent parsing.

The approach we propose in this work differs in many aspects from the above-described works. To begin with, our algorithm is not constrained to generate binary trees. Furthermore, our method does not rely on any machine learning algorithm, but on a number of heuristics that allow us to extract general patterns present in many languages. Preliminary results of this approach were presented in Santamaría and Araujo (2010), showing the potential of the method in a restricted version which only uses two of the PoS tag classes that are identified by their role in the patterns (namely, separators and delimiters; see below). Clark's (2000) idea of contexts is close to our idea of separators, and plays a similar role, although these are identified in a different manner. Besides, the clusters of PoS tag sequences determined by a particular context are also somehow related to our idea of delimiter, since they represent the set of PoS tag sequences with the same limits. The current work extends that initial procedure by adding new PoS tag classes (such as those prone to divide sentences into two main parts, and joiners, a particular kind of separators linking two similar sub-structures). We have also enlarged the model by considering punctuation marks. Finally, we have evaluated the model for different languages in order to assess its generality.

3 The PoS tag pattern method for unsupervised parsing

We have used the empirical observations described in the Introduction to elaborate a deterministic procedure to parse sentences of a PoS-tagged text. The procedure is implemented in two algorithms: In the first one a few sets of tags are identified according to their role in different patterns; in the second one these tags are used to produce parse trees. In this section, we will describe how to construct the first algorithm (the pseudocode of which appears in Figure 4). The second algorithm is described in Section 4.

There are four relevant classes that we need to identify: delimiters, separators, joiners, and punctuation marks. Delimiters and separators are mutually exclusive. Joiners are a subset of separators, and, of course, punctuation marks do not overlap any other class. This classification is not exhaustive. There remain tags that play no special role.

We call *delimiter* any PoS tag occupying the first or last position of a base constituent. For instance, DT and NN in Figures 1 and 2, and DT and NNS in Figure 3, are delimiters.

A *separator* is any PoS tag T which appears in patterns of the form $T\,C_1$ or $C_1\,T$, with C_1 any constituent. Tags such as VBD in Figure 1 (in its two appearances), PRP and VBD in Figure 2, or VBP in Figure 3, are examples of separators. If the separator plays the role of the head of the sentence (i.e., splits the sentence into two main parts), we refer to it as *predominant separator*; e.g., VBD in Figure 1 or VBP in Figure 3 are predominant separators.

A *joiner* is any PoS tag T which appears in patterns of the form $C_1\,T\,C_2$, with C_1 and C_2 two arbitrary constituents. A joiner in the trees given in Figures 2 and 3 is CC.

Finally, we interpret any character other than alphanumeric characters as *punctuation marks*. Its identification is therefore a trivial matter.

The problem we face now is to identify tags that are likely to play the role of delimiters, separators, and joiners in the texts to be parsed.

The cornerstone of our method is to identify at least one sequence of PoS tags that forms a constituent with high probability. We achieve this by looking throughout the corpus for the most frequent pair of PoS tags, $L_{sc}\,R_{sc}$, to which we refer as the *safe constituent (sc)*. This automatically identifies L_{sc} and R_{sc} as two delimiters. This empirical assumption is based on the observation that in all corpora that we have examined, the most common sequence of length two is a noun phrase.

According to Klein and Manning (2005), long constituents often have short, common equivalents that appear in similar contexts. It is thus reasonable to assume that occurrences of these in the corpus correspond to constituents fitting the pattern $L_{sc}\cdots R_{sc}$. Certainly not all of them, but our guess is that this happens more often than not. Under this hypothesis we can identify separators and new delimiters in relation to their occurrences relative to these two delimiters.

Actually, following Cohen and Smith (2009) we make a small refinement which improves performance: We extend the definition of *sc* by generalizing its pair of PoS tags to the pair of coarse-grained PoS tag categories which they belong to. For

```
1   (L_sc, R_sc) ← most freq. sequence of two PoS tags in the corpus
2   sc ← (L_sc, R_sc) // safe constituent
3   S ← ∅, D ← {L_sc, R_sc}, O ← ∅, J ← ∅, P ← ∅, DP ← ∅;
4   // Identify separators (S), delimiters (D). The remaining tags are classified as other(O)
5   for each E ∈ T do
6       if |#(E, L_sc) − #(L_sc, E)| ≥ |#(E, R_sc) − #(R_sc, E)| // L_sc: determining side
7           if 3/4 ≤ #(E, L_sc)/#(L_sc E) ≤ 4/3 then // #(E, L_sc) and #(L_sc E) are similar
8               D ← E // delimiter
9           else if (#(E, L_sc) > #(L_sc, E)) then // E tends to appear on the left of L_sc
10              S ← E // separator
11          else O ← E // other: E tends to appear on the right of L_sc
12      else                                    //R_sc: determining side
13          if 3/4 ≤ #(E, R_sc)/#(R_sc E) ≤ 4/3 then // #(E, R_sc) and #(R_sc E) are similar
14              D ← E // delimiter
15          else if (#(R_sc, E) > #(E, R_sc)) then // E tends to appear on the right of R_sc
16              S ← E // separator
17          else O ← E // other: E tends to appear on the left of R_sc
18  for each E ∈ D do // Identify the preferred direction for the delimiters
19      X: most freq. tag to the left of E, Y: most freq. tag to the right
20      if #(X, E) > #(E, Y) then preference_direction(E) ← right
21      else preference_direction(E) ← left
22  // predominant separator category (SC_pred)
23  // separator category sec: PoS tags in S sharing the same prefix
24  SC_pred ← {sec : #sec ≥ N}
25  // separator hierarchy
26  for each E ∈ SC_pred do // can be generalized if there are more levels
27      if ( | log10(#(sc, E)/#(sc, max))| > 1) then // same order of magnitude
28          first_level ← E
29      else second_level ← E
30  // Identify partners
31  for each E ∈ SC_pred do
32      Partners ← most frequent PoS tag X preceding E
33  Partners ← remove from Partners tags appearing before (but not
34                          contiguous to) the head of a sentence
35  //Identify joiners (J)
36  for each E_i ∈ S do
37      total_counter ← #different sequences E_j E_i E_k with E_j, E_k ∈ T
38      for each E_j ∈ S do
39          if the most frequent sequence E_j E_i E_k has E_j = E_k then
40              joiner_counter ← joiner_counter + 1
41      if (joiner_counter ≥ total_counter / 2 then J ← E_i
42  //Identify punctuation marks(P) and double punctuation marks(DP)
43  DP ← ∅
44  P ← PoS tags of words not made of letters and numbers
45  for each E ∈ P do
46      for each E' ∈ P do
47          if #sentences with E ≈ #sentences with E' then DP ← (E, E')
```

Fig. 4. Algorithm for the induction of PoS tag classes. The separator category *sec* is a set of PoS tags of T beginning with the same letters, if any, or a single separator otherwise. *N* is the number of sentences in the corpus. *sec* are the PoS tags in S(separators) sharing the same prefix. This optimization is only applied to those sets in which the PoS tags of each category share the first letter. *max* is the PoS tag from *SC_pred* appearing most often immediately following the safe constituent *sc*.

instance, (DT, NN) is usually found to be the most frequent pair of PoS tags in English corpora; thus, *sc* is generalized to (DT, NN*), i.e., a determiner and *any* noun. We only do this when the corpus annotations identify the PoS tag categories by assigning (as in this example) the same initial letters to all PoS tags belonging to it (most corpora use this convention). Otherwise we stick to the found pair of most frequent PoS tags.

3.1 Detecting separators and delimiters

Given a PoS tag E, we look for all occurrences of the two pairs $E\,L_{sc}$ and $L_{sc}\,E$ (likewise $E\,R_{sc}$ and $R_{sc}\,E$). Either there is a bias toward one of the two pairs or there is no bias. If $E\,L_{sc}$ (likewise $R_{sc}\,E$) appears significantly more often, then we classify E as a separator. If the difference is not significant then we classify E as a delimiter. The rationale for this is as follows. Arguably, most occurrences of L_{sc} and R_{sc} in the text will correspond to a constituent of the form $L_{sc} \ldots R_{sc}$. Hence, if $E\,L_{sc}$ (respectively $R_{sc}\,E$) appears significantly more often than $L_{sc}\,E$ (respectively $E\,R_{sc}$), this means that E tends to form the pattern $E\,C$ (with $C = L_{sc} \ldots R_{sc}$) that identifies a separator. Delimiters, on the other hand, can be found at any side of L_{sc} or R_{sc} because base constituents may be a part of or adjacent to another constituent. So if both sequences, $E\,L_{sc}$ and $L_{sc}\,E$ (respectively $R_{sc}\,E$ and $E\,R_{sc}$), appear a comparable number of times, then it is likely that E is a delimiter.

Most delimiters in English corpora are either determiners (DT) or nouns (NN*), so the *sc* almost exhausts the delimiter class. There are a few others, though, such as possessive ending (POS). As a matter of illustration, POS is often found in the Penn Treebank ending a base constituent, e.g., (NP (NN today) (POS's)), but sometimes this constituent precedes a noun, e.g., ((NP (NP (DT the) (NNS funds) (POS')) (NNS investments)), therefore neither pattern, POS NN* nor NN* POS, is expected to show up more often than the other. This identifies POS as a delimiter.

More formally, the precise procedure we have implemented to assess whether a PoS tag E is a separator or a delimiter is the following. For a given PoS tag E we compute[2] $\#(E\,L_{sc})$, $\#(L_{sc}\,E)$, $\#(E\,R_{sc})$, and $\#(R_{sc}\,E)$. We refer to $E\,L_{sc}$ and $R_{sc}\,E$ as the *outer* pairs, and $L_{sc}\,E$ and $E\,R_{sc}$ as the *inner* pairs – implying that L_{sc} and R_{sc} are, respectively, the left and right ends of a constituent. We define the *determining side* of E (denoted $ds(E)$) as the tag, L_{sc} or R_{sc}, showing the largest difference in the number of occurrences between its inner and outer pairs. Finally, the similarity is analyzed by checking whether the ratio

$$rt = \#(\text{outer pair of } ds(E))/\#(\text{inner pair of } ds(E))$$

is sufficiently close to unity (more precisely in the interval $(3/4, 4/3)$).

Now, if $3/4 < rt < 4/3$, then we consider that the 'inner-outer bias' is not significant (less than 3:4) and thus classify E as a delimiter. If $4/3 < rt$, then the bias toward the outer pair is considered significant and we classify E as a separator. Finally, if $rt < 3/4$, the bias toward the inner pair is significant, meaning that E is just a part of the constituent.

We have checked thresholds as large as 1:2 finding that the results are little affected or not at all regardless of the corpus or the language. Therefore, even though the choice 3:4 looks arbitrary, its precise value is not very relevant. This probably means that whenever there is a bias, it is strong.

[2] $\#(E_1 \cdots E_n)$ stands for the number of occurrences of the PoS tag sequence $(E_1 \cdots E_n)$.

Table 1. *Separators and delimiters obtained for the WSJ corpus. Others correspond to the remaining PoS tags, excluding punctuation marks*

Separators	Delimiters	Others
MD, PRP, IN, RB, RBR, CC, TO, VB, VBD, VBN, VBZ, VBP, VBG, EX, LS, RP, UH, WP, WRB, WDT	DT, PDT, POS, SYM, NN, NNS, NNP, NNPS	CD, FW, JJ, JJR, JJS, LS, PP$, RBS, WP$, PRP$

In the algorithm (Figure 4) we study the similarity by checking the *rt* ratio corresponding to each case. We introduce the predicate

$$sim(E_1, E_2) = \text{ true iff } \#(E_1\,E_2)/\#(E_2\,E_1) \in [3/4, 4/3]$$

which is true if the difference in the number of occurrences of the PoS tag pairs $E_1\,E_2$ and $E_2\,E_1$ is not significant. A tag E is then considered a separator if the predicate

$$sep(L_{sc}, E, R_{sc}) = ((ds(E) = L_{sc}) \wedge (\neg sim(E, L_{sc}) \wedge \#(E, L_{sc}) > \#(L_{sc}, E)))$$
$$\vee((ds(E) = R_{sc}) \wedge (\neg sim(E, R_{sc}) \wedge \#(R_{sc}, E) > \#(E, R_{sc})))$$

is true, i.e., if the determining side is L_{sc}, the number of occurrences on both sides of this boundary is not similar, and E appears more often to the left than to the right of L_{sc}, or if the determining side is R_{sc}, the number of occurrences on both sides of this boundary is not similar, and E appears more often to the right than to the left of R_{sc}.

On the other hand, E is considered a delimiter if the predicate

$$delim(L_{sc}, E, R_{sc}) = ((ds(E) = L_{sc}) \wedge sim(E, L_{sc})) \vee ((ds(E) = R_{sc}) \wedge sim(E, R_{sc}))$$

is true, i.e., if the determining side is L_{sc} and the number of occurrences of E on both sides of L_{sc} is similar, or if the determining side is R_{sc} and the number of occurrences of E on both sides of R_{sc} is similar.

Repeating this procedure with every PoS tag we have obtained the sets of separators and delimiters for WSJ. They are shown in Table 1. In this case, determiners and nouns are classified as delimiters, and prepositions and verbs are classified as separators. Although this is common to most tested corpora, we have found some differences. For instance, some verbs (VAIMP, VMINF, VMPP) are classified as delimiters in the German corpus NEGRA. On the other hand, other PoS tags are more heterogeneous; e.g., adverbs are in the separator class in the Penn Treebank, but appear distributed between the two classes in the Spanish UAM Treebank.

Delimiters can be either right or left delimiters. Left delimiters tend to appear at the first position, whereas right delimiters tend to appear at the last position. To determine the type of delimiter L, we find a tag E such that EL is the most frequent combination and a tag E' such that LE' is the most frequent combination. If $\#(EL) > \#(LE')$, then L is a right delimiter; otherwise it is a left delimiter. Table 2

Table 2. *Preferred direction in which each delimiter clusters in WSJ10. The first column corresponds to the delimiter, the second column to the most frequent sequence comprising the delimiter and a tag on its right, the third column to the frequency of this sequence, the fourth column to the most frequent sequence of the delimiter and a tag on its left, the fifth column to its corresponding frequency, and the last column to the resulting direction. For NNP, for which the frequency of the most frequent tag to the right and to the left are the same, we have searched for the second most frequent sequence to choose the grouping direction: (NNP VBZ)(454) and (IN NNP)(540). Accordingly, the direction of preference for NNP is the right*

Delimiter	Most freq. left	Freq.	Most freq. right	Freq.	D
DT	(DT, NN)	2222	(IN, DT)	894	L
PDT	(PDT, DT)	28	(NN, PDT)	14	L
POS	(POS, NN)	169	(NNP, POS)	223	R
SYM	(SYM, IN)	11	(NN, SYM)	4	L
NN	(NN, IN)	892	(DT, NN)	2,222	R
NNS	(NNS, VBP)	591	(JJ, NNS)	797	R
NNP	(NNP, NNP)	2127	(NNP, NNP)	2,127	R
NNPS	(NNPS, NNP)	42	(NNP, NNPS)	82	R

shows the results obtained for WSJ10, the type of the delimiter appearing in the last column.

The process to identify delimiters and separators appears at the beginning of the induction algorithm in Figure 4 (lines 4–21).

3.2 Identifying the main parts in sentences

In general, the head of the sentence is a separator because in most cases it triggers new levels of the tree. A sentence may, however, contain more than one separator, so a procedure is needed to discriminate which one acts as head of the sentence. To this end we have introduced a hierarchy among the separators that allows us to decide which one is the main separator of a sentence.

We first define the predominant separator class, SC_pred, as the class of separators (excluding punctuation marks) whose number of occurrences in the corpus is larger than N – the number of sentences in the corpus. Actually, we do not consider separators independently, but group those belonging to the same lexical category (e.g., VB* if we are considering verbs). The number of occurrences of a separator category *sec* is then counted as

$$\#sec = \sum_{s \in sec} \#s.$$

Grouping by lexical category is only done for corpora that annotate PoS tags using the same initials if they belong to the same category (most corpora are annotated this way). Otherwise each PoS tag is considered separately. Thus, the predominant separator class is defined as the set $SC\_pred = \{sec : \#sec \geq N\}$.

As expected, several languages and corpora used in this work do have a predominant separator category. In the English corpus WSJ10, 9,114 out of the 9,799 (93%) occurrences of the category VERB (VB, VBD, VBG, VBN, VBP,

```
1   for each E ∈ SC_pred do
2       if #(X E) = max{#(Y E), Y ∈ T}, then
3           Partners ← X
4   for each P ∈ Partners do
5       for each E ∈ SC_pred do
6           if there is any sequence P ... T E do
7               Partners ← Partners − P
```

Fig. 5. Identifying partners. T stands for the set of PoS tags.

and VBZ) are separators. Analogously, in the Spanish corpus UAM10, 466 out of the 497 (93.76%) occurrences of the VERB category (VPAST, VPRES, VPART, VINFINITE, VIMPERFECT, VFUT, VCOND, and VNO) are separators. The same is found in the Italian and the German corpora. On the contrary, for the Hebrew and the Chinese corpora *SC_pred* is empty, hence this technique cannot be used to identify the head of sentences.

For corpora with a nonempty *SC_pred* and sentences with only one separator in this class, the head of the sentence is immediately identified as this separator. However, many sentences have more than one predominant separator, so establishing a hierarchy within *SC_pred* will help decide which one should be identified as the head of the sentence. Again we resort to *sc* to build this hierarchy.

It is an empirical observation that holds in all analyzed corpora that *sc* tends to appear before the predominant separators. This makes sense for the English corpus because *sc* is a noun phrase, but is also valid for other languages. The hierarchy is therefore determined by the frequency of appearance of the elements of *SC_pred* after *sc* in the sentences of the corpus – the more frequent a predominant separator, the higher in the hierarchy. Thus, if a sentence contains more than one separator in the *SC_pred* class, the head of the sentence is that higher in the hierarchy.

There is a small refinement that we still need to deal with. Separators in the *SC_pred* class may have special PoS tags associated to them. This is the case of auxiliary verbs, for instance. We call these tags *partners* of the predominant separators. These are detected in two steps. The pseudocode in Figure 5 represents the proccess for identifying partners. In the first step, we find the PoS tags that most frequently appear preceding each separator of *SC_pred* (lines 1–3). In the second step, we remove from the partners those tags whose appearance may be due just to pure chance. This is ascertained by checking whether the tag also appears before – but not immediately before – the head[3] of the sentence (lines 4–7).

In the English corpus, EX, MD, PRP, TO, WDT, and WP are the PoS tags that most often precede a predominant separator. Except for TO and MD, these are also found preceding the head of the sentence; therefore the only partners of WSJ are TO and MD.

[3] At that point, the algorithm has already identified the predominant separator class. Therefore, we consider that the head of the sentence is the tag (if any) belonging to the predominant separator class (the highest in the hierarchy if there are more than one, and the leftmost one if they are at the same level).

Table 3. *Hierarchy of predominant separators used to determine the tag that divides sentences. Vx stands for other PoS tags beginning with V at the same level*

Level	WSJ
1	VBD, VBZ, VBP, TO Vx, MD Vx
2	VB, VBG, VBN

Partners are included in the hierarchy, along with the predominant separator they accompany, using the same criterion as with the ordinary elements of the *SC_pred* class – counting the number of occurrences after *sc*. Table 3 shows the hierarchy obtained for WSJ. Levels in this hierarchy are determined by the order of magnitude of the number of occurrences of the tag after the safe constituent. Level 1 in Table 3 corresponds to hundreds of occurrences, whereas level 2 corresponds to tens.

The process to identify the hierarchy of predominant separators appears in the induction algorithm (Figure 4, lines 22–34).

3.3 Detecting joiners

Joiners are a special type of separators that not only trigger a new level in the tree but also join pieces of information at the same level. This is the case of CC in Figures 2 and 3. We expect to find similar patterns on both sides of a joiner, and this feature identifies them.

We assume that a particular PoS tag is a joiner if it tends to be preceded and succeeded by the same tag. In particular, the algorithm for determining whether tag X is a joiner searches for sequences of three tags of the form YXZ in the corpus and checks whether those with the form YXY are more than 50%. This test appears in the induction algorithm in Figure 4 (lines 35–41).

The set of joiners detected for the English corpus WSJ comprises only *CC*.

3.4 Punctuation marks

The presence of punctuation marks in sentences can provide important clues to the underlying structure. In fact, Spitkovsky *et al.* (2011) have observed a strong connection between English punctuation and phrase boundaries in the Penn Treebank. They used this link between punctuation and constituent boundaries to approximate parsing by treating inter-punctuation fragments independently, obtaining an important improvement in unsupervised dependency parsing. We have also considered punctuation in our patterns definition.

We distinguish two different groups: Single punctuation marks – such as the comma, semicolon, and colon, and double punctuation marks – such as quotations, parentheses, and brackets. Again, we perform an automatic search for this class of PoS tags. We check for non-alphanumeric symbols appearing in the corpus and mark the positions of the sentence where they appear. For any identified punctuation mark

we check whether it has a complement, i.e., another punctuation mark appearing in a similar number of sentences.[4]

The process to identify punctuation marks appears at the end of the induction algorithm (Figure 4, lines 42–47).

The set of punctuation marks detected for the English corpus WSJ is ” “ ’ ‘ # $. , :.

4 Parsing sentences with PoS tag classes

Once the PoS tags have been classified, they can be used to parse sentences. Parsing a sentence starts with a preprocessing step which chops it into segments. To do this we first check whether the sentence has a head. It does if it contains at least one predominant separator (i.e., a separator in the *SC_pred* class; see Section 3.2). If there are more than one, the head is the one with the highest rank in the hierarchy defined within the *SC_pred* class. If there is a head, the part of the sentence from the beginning to the head (excluded) is identified as a segment, and the part from the head (included) to the end as another segment. If there is not a head, this division of the sentence is not applied. On top of that, any sequence of PoS tags appearing between pairs of punctuation marks is also identified as a segment.[5] Segments are our first constituents, and the parsing proceeds segment-wise.

The parsing algorithm is detailed in Figure 6. We first look for predominant separators (separators in the *SC_pred* class), including possible partners. If there is one, we identify it as the head of the sentence (lines 1–7). If there is more than one, we resort to the *SC_pred* hierarchy and identify the top one as the head of the sentence. If the sentence has a head, we divide it into two segments as described above.

In parsing sentences we assume that any PoS tag sequence inside a double punctuation mark is another phrase. In this case we apply the same procedure as for any other sentence. Otherwise, the punctuation marks are ignored. The first two commas, if any, are also considered double punctuation marks, provided the head of the sentence does not appear in between. Accordingly, we check for the presence of paired punctuation marks. Any pair of these identifies a new segment. Segments are parsed separately.

Next, we use the separators to identify further constituents (lines 9–12), which are then refined by identifying the delimiters with left and right preferred directions (lines 13–15). Finally, we identify the joiners and, if necessary, adapt the tree structure to them (lines 16–28).

For each sentence, the algorithm described in Section 3 runs along the sequence of its n PoS tags to detect separators, delimiters, etc. This process requires $O(n \times |T|)$

[4] Strictly speaking, the complement should appear exactly the same number of times, but we allow for a small difference, less than 10%, to account for errors in the PoS tagging of the corpus.

[5] During the parsing process, the sequences of tags between double punctuation marks are considered a segment and parsed. Besides, the sequence of tags between two consecutive single punctuation marks is also considered a segment.

```
1   if the language has a predominant separator then
2       Identify the PoS tags in SC_pred class
3       Select the higher one (which can include a partner) or the leftmost one if there
4           is a tie, HSC_pred, in the SC_pred hierarchy to divide the sentence
5       Split the sentence in segments according to HSC_pred and paired punct. marks
            (the first two commas, if any, are also considered paired if HSC_pred does
            not appear in between)
6   else
7       Split the sentence in segments according to the leftmost separator and
                                    paired punct. marks
8   for each segment in the sentence do
9       for each PoS tag S ∈ Separators do
10          Form a constituent C with the PoS tag sequence from the tag right after S
              to the next separator
11          Form a constituent from S to the end of the segment
12          Form a constituent from C to the end of the segment
13      for each PoS tag D ∈ Delimiters do
14          if grouping_direction(D) = right and D ∉ sc then
15              Form a constituent from the beginning of the segment up to D
16      for each PoS tag J ∈ Joiners do
17          Select the constituents C1 and C2 at both sides of J
18              if the tag S after J ∈ Separators then
19                  if S appears on the left of J then
20                      choose C1 to begin at the closest S on the left of J
21                  else if any other separator S′ appears on the left of J then
22                      choose C1 to begin at the closest S′ on the left of J
23                  else choose C1 as the longest constituent before J
24              else
25                  choose C1 as the longest constituent on the left of J
26                  choose C2 as the longest constituent on the right of J
27          Delete the constituents going beyond C2
28          Form a constituent with C1JC2
```

Fig. 6. Parsing algorithm based on the PoS tag classes, assuming a right-branching structure (separators are expected to appear on the left of the constituents). The approach is also valid if we assume left-branching structure. Segments are sequences of PoS tags that are parsed separately.

steps, $|T|$ being the size of the PoS tag set. The constituent identification algorithm described in this section takes $O(n^2)$ steps, because for each position in the sentence the algorithm must find the end of the corresponding constituent. The scope of joiners is decided after this constituent identification phase by searching for similar constituents at both sides of the joiner, deleting some previous constituents and forming a new one when necessary. The computational cost of this is $O(3n^2)$. The whole algorithm described in this section is thus $O(n^2)$. Since sentences are usually short (n hardly exceeds 40), the algorithm turns out to be very efficient.

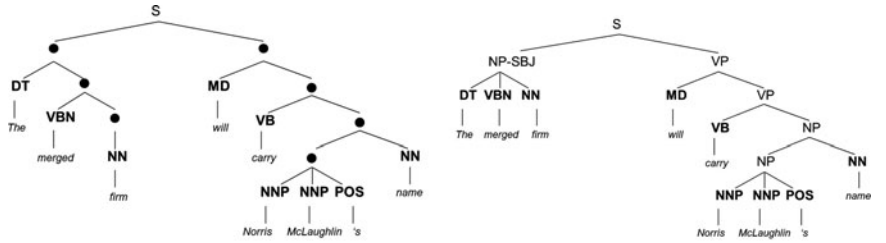

Fig. 7. Parse tree for the sentence *The merged firm will carry Norris McLaughlin's name*, according to (a) our proposal, and (b) from the Penn Treebank.

As of space requirements, the algorithm only needs to store the PoS tag sequence, the role of each PoS tag, and the identified constituents. This simply requires an $n \times n$ table.

A running example will illustrate how the parsing algorithm works. Consider the sequence of PoS tags corresponding to the WSJ10 sentence *The merged firm will carry Norris McLaughlin's name*, namely:

DT **VBN** NN **MD VB** NNP NNP POS NN

Predominant separators appear in boldface. The *SC_pred* hierarchy determines that MD Vx – which includes a partner – is above VBN; accordingly, MD is the head of the sentence, and hence the dividing tag:

[DT **VBN** NN] [**MD VB** NNP NNP POS NN]

The next step is to find additional separators not in the *SC_pred* class, if any, and use all separators to further divide the sentence. For instance, VBN yields two new constituents: the sequence from VBN (included) to the end of the segment [VBN NN], and the sequence in between VBN and MD – the next separator – [NN]. The same is done for each separator, so we end up with:

[DT [**VBN** [NN]]] [**MD** [**VB** [NNP NNP POS NN]]]

We next seek the delimiters (underlined in the sentence):

[DT [**VBN** [NN]]] [**MD** [**VB** [NNP NNP POS NN]]]

In this case, all the delimiters of the last constituent have preferred direction right, according to Table 2. Since NNP belongs to *sc*, [NNP NNP POS] is identified as a constituent and [NN] as another one. This leads to:

[DT [VBN [NN]]] [MD [VB [[NNP NNP POS] [NN]]]]

Figure 7 compares the tree resulting from applying our algorithm to this sentence with that provided by the Penn Treebank. We can see that both trees are almost identical, with the only exception that our algorithm considers that both *The merged firm* and *merged firm* are constituents, while the tree in the Penn Treebank only considers that *The merged firm* is a constituent.

5 Experimental results

The system – the code is publicly available at http://nlp.uned.es/~jsant/ – has been evaluated against the gold-standard trees provided by the treebanks. Our constituents are not assigned any class name (e.g., noun phrase, verb phrase, etc.) as in the treebanks. The comparison therefore ignores class labels, considering only groups of tags. The results provided by Klein and Manning (2005), Bod (2006), and Golland *et al.* (2012) have been our reference, as they are also designed for constituent grammars using PoS-tagged corpora. For the sake of comparison, we have evaluated our algorithm with the same corpus, the Penn Treebank, so we can readily test our results against the measures reported in those works.

The most common measures (Abney *et al.* 1991) used to evaluate the quality of an induced grammar or a parsing – and those we have employed in this work – are *precision*, *recall*, and their harmonic mean *(F-measure)*. Note that in our case these measures deliberately ignore the labels assigned to the parse tree (which are not a product of our algorithm).

5.1 Evaluation for different languages

For English language, we have used the Wall Street Journal section of the Penn Treebank, containing 49,208 sentences, but we have also applied our algorithm to other corpora in different languages. To be precise, we have tested it with the following:

- The UAM Spanish (Romance language) Treebank (Moreno *et al.* 2000), containing 1,501 syntactically annotated Spanish sentences collected from the online edition of Spanish newspapers.
- The German (Germanic language) corpus NEGRA (Skut *et al.* 1997), with 20,602 sentences from German newspaper texts.
- The Italian (Romance language) corpus TUT (Lesmo, Lombardo and Bosco 2002), containing 2,860 Italian sentences collected from different sources.
- The HTB corpus (Sima'an *et al.* 2001), a Modern Hebrew (Semitic language) Treebank with 6,501 sentences from Hebrew press sources.
- The Chinese Treebank 7.0 (Xia *et al.* 2000) in Mandarin Chinese (Chinese language family), with 51,447 sentences from Chinese newswires, magazine news, various broadcast news and broadcast conversation programs, web newsgroups, and weblogs.

These languages not only belong to four different families but also present different types of morphology and degrees of freedom in the word order.

Regarding morphology, although all languages are mixed types, some of these fit best into one category than into another. Most Indo-European languages, including Spanish, Italian, and German, are considered synthetic, as they form words by affixing a given number of dependent morphemes to a root morpheme. Chinese is the usual model of analytic language, as it tends to be uninflected. Hebrew grammar is partly analytic, expressing such forms as dative, ablative, and accusative using prepositional particles rather than grammatical cases. However, inflection plays a

decisive role in the formation of verbs and nouns. English is moderately analytic (more analytic than other Indo-European languages).

Spanish and Italian word order is much more flexible than that of English. In Spanish, changes in the subject-verb-object order are very frequent and common in every day writing. In German, word order is also less rigid than in English. In main clauses the inflected verb has position 2. In subordinate clauses the verb appears at the very end. The Mandarin Chinese object has a great deal of flexibility. It usually appears after the verb, but other frequent possibilities are before the verb, before the subject, or is even omitted. A specific feature of Chinese is that verbs can become nouns without undergoing any sort of change. In most cases, the order of the words in Hebrew does not affect the meaning. Hebrew sentences do not have to include verbs, and the verb *to be* in present tense is omitted. Unlike the verb *to have* in English, none of the possession terms in Hebrew is a verb.

Full corpora were used to extract the statistical patterns for classifying PoS tags. To test the parsing algorithm, though, some experiments were performed on restricted sets containing sentences up to ten words long for the sake of comparison with previous works (Klein and Manning 2005; Bod 2006; Bod 2007) with a similar experimental setting, i.e., constituent grammars and PoS-tagged corpora. We will refer to these test sets as WSJ10, UAM10, NEGRA10, TUT10, HTB10, and CTB10, respectively. In addition, we also experimented with longer sentences (see Section 5.2). We have followed the convention from other unsupervised systems (Bod 2006; Seginer 2007a) of using a test set that is a subset of the training set.

The performance of our model on a given language mainly depends on two factors: One is, of course, how well our underlying assumptions capture real linguistic in the language; but not less important is the particular annotation scheme followed in the analyzed treebanks. Our current model produces trees similar to those in the Penn Treebank (in their size and number of levels), while other treebanks use different conventions.[6] To be fair, a degrading in precision and recall due to annotation differences cannot be considered a drawback of the model. To circumvent this problem we have resorted to another measure: the average number of crossing brackets (CB) (Abney *et al.* 1991). It counts the number of constituents that violate the boundaries of a constituent in the gold standard. Obtaining a low CB along with a low F-measure is an indication that the annotation scheme differs significantly from ours.

Table 4 shows the results obtained with our algorithm for the subsets of sentences having up to ten words using different corpora. The low CB obtained in all cases indicates a high performance of the proposed approach (on average, more than a half of the constituents do not have any crossing brackets). We observe that with the exception of Spanish, Hebrew, and Chinese, CB rates are below 0.5. In Spanish, the CB rate is slightly higher than 0.5, possibly due to a small corpus size. Still the system obtains good results even for the smallest corpora: Spanish (396) and

[6] NEGRA presents very flat phrases: pre- and post-modifiers are attached directly to the phrase, nominal subjects are attached directly to the sentence, and nominal material within PPs does not project to NPs (Maier 2006).

Table 4. *Performance results, in terms of crossing brackets (CB), unlabeled precision (UP), unlabeled recall (UR), and unlabeled F-measure (UF) obtained for different languages, considering sentences of up to ten words. The last two columns correspond to the average number of constituents per sentence in each data set, according to the gold standard (#C-GS) and as obtained with our method (#C-SEP). The languages are English (WSJ10), German (NEGRA10(1) and NEGRA10(2)), Spanish (UAM10), Italian (TUT10), Hebrew (HTB10), and Mandarin Chinese (CTB10). The second column shows the size of each corpus. In all cases we have considered only sentences with up to ten words from the corresponding corpus. NEGRA10(2) provides the results for NEGRA10 annotating the verbal group separately. Corpora sizes are numbers of kept sentences*

Corpus	Size	CB	UP (%)	UR (%)	UF (%)	#C-GS	#C-SEP
WSJ10	7,422	0.30	86.24	90.33	88.23	5.05	5.28
NEGRA10(1)	7,537	0.49	50.75	81.97	62.69	3.21	5.19
NEGRA10(2)	7,537	0.42	59.51	81.31	68.72	3.21	4.39
UAM10	396	0.63	70.52	87.67	78.16	5.40	6.72
TUT10	563	0.45	73.86	87.19	79.97	4.13	4.87
HTB10	1,039	1.11	59.84	73.77	66.08	4.48	5.53
CTB10	15,138	1.11	59.23	61.14	60.17	3.80	3.92

Italian (563). A slightly lower level of precision for the German corpus (NEGRA) along with a low CB rate hints at differences in the annotation schemes. The second row for this corpus, NEGRA10(2), confirms this fact. It corresponds to the results annotating the verbal group separately in the trees produced by our system: the verb is not included in the verb constituent. We can see that this minor change improves the results significantly.

Apart from German, which uses an annotation scheme rather different from ours, the poorest results are obtained for Hebrew and Chinese. Higher CBs obtained for these corpora suggest that some of the patterns that we have considered may not be meaningful for these languages. We have delved into these corpora to identify the reasons for these lower results. The first difference that we have found is the number of verbs that have been identified as separators in each corpus. In Hebrew and Chinese this number is smaller than the number of sentences, whereas in the other corpora it is larger. Therefore, the patterns we rely upon to identify heads of sentences are invalid for Hebrew and Chinese.

Another difference lies in the contribution of different patterns, shown in Table 5. In order to study the contribution of each pattern we have evaluated the performance of the system introducing these patterns one by one: first, only separators and delimiters (first column);[7] punctuation marks are then processed as special tags in addition to separators and delimiters (second column); next we add special processing for joiners (third column); and finally all patterns are introduced, including the predominant separator hierarchy (fourth column). The results for English, German, Spanish, and Italian improve significantly with the introduction of each new pattern – the only exception being the hierarchy of the predominant

[7] Punctuation marks are processed as any other tag, and the first separator which is a verb is used to divide the sentence into two parts.

Table 5. *Contribution to the F-measure of different patterns. The languages are English (WSJ10), German (NEGRA10(1) and NEGRA10(2)), Spanish (UAM10), Italian (TUT10), Hebrew (HTB10), and Chinese (CTB10). For the Hebrew and the Chinese corpora SC_pred is empty, hence this pattern does not apply. The numbers in parentheses indicate the decrease/increase percentage with respect to the original performance (1st column)*

Corpus	Sep. & Delimit.	Punc. marks	Joiners	Pred. sep. hie.
WSJ10	81.33	85.61 (+5.26%)	86.81 (+6.73%)	88.23 (+8.48%)
NEGRA10(1)	61.23	61.44 (+0.34%)	62.22 (+1.61%)	62.69 (+2.38%)
NEGRA10(2)	67.10	67.37 (+0.40%)	68.21 (+1.65%)	68.72 (+2.41%)
UAM10	74.05	74.78 (+0.98%)	76.25 (+2.97%)	78.16 (+5.55%)
TUT10	76.89	77.59 (+0.91%)	79.97 (+4.00%)	77.39 (–0.65%)
HTB10	64.17	64.47 (+0.46%)	66.08 (+2.97%)	–
CTB10	59.48	60.03 (+0.92%)	60.17 (+1.16%)	–

separator category for Italian. As a matter of illustration, the pattern related to punctuation marks improves the results for WSJ10 from 81.33% to 85.61% , and the pattern for joiners improves the results for TUT10 from 77.59% to 79.97% . These effects are much less noticeable for Hebrew and Chinese. We also observe that the predominant separator hierarchy pattern is not detected for Hebrew and Chinese. This result agrees with the presence of constructions in these languages in which the verb can be omitted.

All in all, even for the languages with the lowest performance (Hebrew and Chinese), the results improve over those obtained by other models, as we will show below.

5.2 Comparison with other systems

We have compared our results with those obtained for the same corpus by Klein and Manning (2005) (constituent-context model), KM, Bod's (2006, 2007) U-DOP and U-DOP* (a modification of U-DOP that can manage larger subtrees), and Seginer's (2007a) incremental parser. Table 6 shows the comparison for WSJ10. Our results attain a higher value for the F-measure as well as more balanced recall and precision values.

We have also evaluated our system on longer sentences. Table 7 compares the unlabeled F-measure obtained using our algorithm with those of Bod's (2006, 2007), Seginer (2007a) , and Golland *et al.* (2012) for sentences of up to forty words long. We can see that the longer the sentences, the worse the methods perform, although our method consistently outperforms the others for all lengths.

Seginer (2007a, 2007b) built the Common Cover Links (CCL) parser, an implementation of his proposal for unsupervised incremental parsing based on the common cover link representation of syntactic structure. Its being publicly available has allowed us to compare our results with those obtained with the CCL parser for all corpora that we have considered. Table 8 shows this comparison. For all corpora, our results improve over those of CCL. Furthermore, the improvement persists with

Table 6. *Results (unlabeled recall, precision, and F-measure) obtained using our separator approach (first row), compared with those of Klein and Manning's (2005) (second row), Bod's (2006, 2007) (third and fourth row), and Seginer (2007a) (fifth row), for the WSJ10 corpus*

	UR (%)	UP (%)	UF (%)
Our approach	90.33	86.24	88.23
Constituent-Context Model (CCM)	80.2	63.8	71.1
Unsupervised Data-Oriented parsing (U-DOP) 2006	70.8	88.2	78.5
Unsupervised Data-Oriented parsing (U-DOP*) 2007	–	–	77.9
Common Cover Links parser (CCL)	75.6	76.2	75.9

Table 7. *Unlabeled F-measure obtained for the WSJ corpus as test corpus for different sentence lengths: up to 10, 20, 30, and 40 words. The first row corresponds to our separator approach, the second row corresponds to Bod's (2007), the third row to Seginer's (2007a), and the fourth row to Golland et al.'s (2012)*

	10	20	30	40
Our approach	88.23%	79.67%	74.29%	72.10%
Unsupervised Data-Oriented parsing (U-DOP*)	77.9%	–	–	64.2%
Common Cover Links parser (CCL)	75.9%	64.7%	59.91%	57.4%
Log-Linear CCM (LLCCM)	72.0%	60.0%	50.3%	47.6%

increased sentence length. One should realize, though, that the CCL system parses plain text, while our system requires PoS-tagged text. According to Seginer's (2007a) data, the CCL parser outperforms any other system that also parses plain text and achieves results close to those obtained by systems using PoS-tagged text, including U-DOP and DMV+CCM(PoS)[8] (Klein and Manning 2004) when parsing WSJ10 and NEGRA10.

Our system, implemented in Java on a PC Intel Core 2, parses the sentences efficiently, as shown in Table 9. The table also shows the execution times on the same computer corresponding to the CCL system (Fast Unsupervised Incremental parsing) by Seginer (2007b), which performs a particularly efficient greedy parsing. We can observe that, although the CCL times are better – about an order of magnitude smaller – the execution time required by our system is still small enough to make the system useful. Our system is capable of parsing the largest corpus in a few minutes. For instance, the complete Chinese corpus CTB is parsed in six minutes and the English corpus WSJ in less than eight minutes.

5.3 Analyzing the effect of the PoS tag set

The granularity of the PoS tag set affects the results. To investigate its influence, we have performed some experiments with other tag sets as well.

[8] Constituent-Context Model plus Dependency Model with Valence using PoS tags.

Table 8. *F-measure results obtained using Seginer's (2007a) CCL model and the algorithm described in this paper (Sep.). Results are obtained for different languages – English (WSJ), German (NEGRA), Spanish (UAM), Italian (TUT), Hebrew (HTB), and Chinese (CTB) – and considering a test corpus containing sentences of up to 10, 20, 30, and 40 words long, as well as the whole corpus. The PoS tag classes used to construct our system are extracted from the whole corpus. These classes are then used to parse the sentences of all subsets of the corpus regardless of their length. Figures with only one significant digit have been taken from the Seginer's paper*

	10		20		30		40		Whole	
Corpus	CCL	Sep.	CCL	Sep.	CCL	Sep.	CCL	Sep.	CCL	Sep.
WSJ	75.9	88.23	64.7	79.67	59.91	74.29	57.4	72.10	56.5	70.97
NEGRA	59.0	62.69	45.7	49.81	41.49	45.56	40.6	44.23	40.1	43.65
UAM	67.19	78.16	60.85	69.54	59.77	67.12	59.69	66.02	59.69	65.44
TUT	61.52	79.97	51.18	70.91	46.30	65.25	43.76	62.42	40.88	59.38
HTB	61.88	66.08	53.96	57.80	49.74	54.40	48.40	52.64	46.99	50.70
CTB	44.98	60.17	32.71	47.25	28.50	42.91	26.68	41.01	24.70	39.16

Table 9. *Execution times for our system (Sep.) and CCL (Seginer 2007a). First column shows the considered corpus, and the number of sentences it is composed of; second column shows our system (Sep.) execution time in seconds; third column shows the number of sentences processed per second in our system; fourth column shows the CCL execution time; and the fifth column shows the CCL number of sentences processed per second*

Corpus	Sep. Ex. T.(s.)	Sep. sent/s.	CCL Ex. T.(s.)	CCL sent/s.
WSJ (49,208 sent.)	468	105	38	1,294
NEGRA (20,597 sent.)	91	226	11	1,872
CTB (51,298 sent.)	360	142	47	1,091
UAM (1,501 sent.)	20	75	1	1,501
TUT (2,859 sent.)	32	89	2	1,429
HTB (6,218 sent.)	99	62	4	1,554

Table 10 compares the results of using both, a smaller PoS tag set and a set of PoS tags obtained in an unsupervised manner. For the smaller set of PoS tags, we have considered the coarse-grain tag set proposed by Petrov *et al.* (Das and Petrov 2011; Petrov, Das and McDonald 2012) comprising only twelve universal PoS tags. The corresponding results appear in the central three columns of the table. The global F-measure is not very different from the one obtained with the original PoS tag set of each corpus. In some cases, such as English, German, and Spanish, results are slightly worse. In others, such as Italian, results remain nearly the same. Finally, in some cases, such as Chinese and Hebrew, results improve by increasing the PoS tag granularity.

In addition, we evaluated the system using an unsupervised Hidden Markov Model (HMM) PoS tagging, estimated by means of a Gibbs sampling (Gao and

Table 10. *Performance results, in terms of unlabeled precision (UP), unlabeled recall (UR), and unlabeled F-measure (UF) obtained for different sets of PoS tags, considering sentences of up to ten words. The languages are English (WSJ10), German (NEGRA10), Spanish (UAM10), Italian (TUT10), Hebrew (HTB10), and Mandarin Chinese (CTB10). The first three columns show the results for the original PoS tag set of each corpus. The second three columns show the results using Petrov et al.'s (2012) coarse grain PoS tag set. The last three columns show the results for the PoS tag set provided by an unsupervised PoS tagger based on Gibbs (Gao and Johnson 2008) sampling. In all cases we have considered only sentences with up to ten words from the corresponding corpus*

	Original PoS tag set			Petrov PoS tag set			Unsup. PoS tag set		
Corpus	UP	UR	UF	UP	UR	UF	UP	UR	UF
WSJ10	86.24	90.33	88.23	77.56	88.55	82.69	73.10	78.66	75.78
NEGRA10	50.75	81.97	62.69	48.68	75.33	59.14	46.61	69.03	55.64
UAM10	70.52	87.67	78.16	67.26	82.87	74.26	63.51	79.88	70.76
TUT10	73.86	87.19	79.97	73.35	86.46	79.36	67.52	79.71	73.11
HTB10	59.84	73.77	66.08	62.68	76.85	69.04	59.49	70.43	64.50
CTB10	59.23	61.14	60.17	66.94	72.71	69.71	47.61	50.50	49.01

Johnson 2008), which is publicly available.[9] The corresponding results appear in the last three columns of the table. We can see that, *albeit* the tagging precision has been significantly lowered because we have performed an unsupervised tagging of the sentences, the results are still valuable and similar to those obtained using Seginer's (2007a, 2007b) approach, which was specifically designed for parsing plain text. In some cases, such as WSJ10 and NEGRA10, results are slightly worse than Seginer's (2007a, 2007b), but they are clearly better for the rest of the corpora. These results should improve with the tagging precision.

6 Conclusions and future work

The proposed systems have achieved a competitive performance for different corpus. The upside of our proposal is that it can be applied to any language for which we have a PoS tagger. Another advantage is its efficiency in parsing sentences: Once we have obtained the PoS tag classes from a corpus, the procedure for parsing new sentences only requires identifying the different PoS tag classes that they contain in order to group them into constituents.

There are many directions in which we are planning to extend the present work. We are studying the way of adapting the algorithm to perform dependency parsing. We think that some of the PoS tag classes identified, such as the separators, can be an important clue for doing the transformation. We also consider the way of adapting the behavior of the algorithm to other families of languages. Besides, we are planning to check the results obtained when testing different languages in more similar conditions. This will imply a fully unsupervised configuration,

[9] `http://research.microsoft.com/en-us/downloads/25e1ecf0-8cfa-4106-ba25-51b0d501017d/default.aspx`

including tagging, and using parallel corpora of the same size for training in all languages, or at least comparable corpora. Other aspect to study is the introduction of indeterminism in the method by considering probabilities in the selection of PoS tag classes. In this line, we could explore possible ways of taking advantage of small treebanks if available. We are also looking for a way to combine our system with the Expectation-Maximization approach to see whether Expectation-Maximization can improve the search of the less likely structures. Another point that can lead to some performance improvements is the introduction of some partial lexicalization, for instance, considering separately each element of some lexical categories, such as prepositions.

Acknowledgments

We would like to thank Prof. Yoav Seginer for making his parsing system publicly available and for supporting us in its use. We want to thank Prof. José A. Cuesta for his assistance with the writing of the manuscript. This work has been supported by the Spanish MICINN project Holopedia (Spanish Ministerio de Ciencia e Innovación (TIN2010-21128-C02) and the Universidad Nacional de Educación a Distancia (UNED) (2013-025-UNED-PROY). We also want to thank UNED's Vicerrectorado de Investigación for the editing service of the final version of the paper.

References

Abney, S., Flickenger, S., Gdaniec, C., Grishman, C., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B, and Strzalkowski, T. 1991. Procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of Human Language Technologies*, North American Chapter of the ACL, pp. 306–11. Stroudsburg, PA: Association for Computational Linguistics.

Bod, R. 2006. Unsupervised parsing with u-dop. In *Proceedings of the Conference on Computational Natural Language Learning*, pp 85–92. Stroudsburg, PA: Association for Computational Linguistics.

Bod, R. 2007. Is the end of supervised parsing in sight. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 400–7. Stroudsburg, PA: Association for Computational Linguistics.

Carroll, G., and Charniak, E. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *Working Notes of the Workshop Statistically-Based NLP Techniques*, pp. 1–13. Palo Alto, CA: Association for the Advancement of Artificial Intelligence.

Clark, A. 2000. Inducing syntactic categories by context distribution clustering. In *Proceedigns of the Workshop on Learning Language in Logic and the Conference on Computational Natural Language Learning* (vol. 7), pp. 91–4. Stroudsburg, PA: Association for Computational Linguistics.

Cohen, S. B., and Smith, N. A. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of Human Language Technologies*, North American Chapter of the ACL, pp. 74–82. Stroudsburg, PA: Association for Computational Linguistics.

Das, D., and Petrov, S. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 600–9. Stroudsburg, PA: Association for Computational Linguistics.

Gao, J., and Johnson, M. 2008. A comparison of Bayesian estimators for unsupervised hidden Markov model POS taggers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 344–52. Stroudsburg, PA: Association for Computational Linguistics.

Golland, D., DeNero, J., and Uszkoreit, J. 2012. A feature-rich constituent context model for grammar induction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (vol. 2), pp. 17–22. Stroudsburg, PA: Association for Computational Linguistics.

Klein, D., and Manning, C. D. 2004. Corpus-based induction of syntactic structure: models of dependency and constituency. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 478–85. Stroudsburg, PA: Association for Computational Linguistics.

Klein, D., and Manning, C. D. 2005. Natural language grammar induction with a generative constituent-context model. *Pattern Recognition* **38**(9), 1407–19.

Lesmo, L., Lombardo, V., and Bosco, C. 2002. Treebank development: the TUT approach. In *Proceedings of the International Conference on Natural Language Processing*, pp. 61–70. Noida, India: Vikas.

Magerman, D. M., and Marcus, M. P. 1990. Parsing a natural language using mutual information statistics. In *Proceedings of the National Conference on Artificial Intelligence* (vol. 2), pp. 984–9. Palo Alto, CA: Association for the Advancement of Artificial Intelligence.

Maier, W. 2006. Annotation schemes and their influence on parsing results. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pp. 19–24. Stroudsburg, PA: Association for Computational Linguistics.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. 1994. Building a large annotated corpus of English: the PENN treebank. *Computational Linguistics* **19**(2): 313–30.

Moreno, A., Grishman, R., Lopez, S., Sanchez, F., and Sekine, S. 2000. A treebank of Spanish and its application to parsing. In *Proceedings of the International Conference on Language Resources & Evaluation*, pp. 107–11. Paris, France: European Language Resources Association.

Petrov, S. 2010. Products of random latent variable grammars. In *Proceedings of North American Chapter of the Association for Computational Linguistics*, pp. 19–27. Stroudsburg, PA: Association for Computational Linguistics.

Petrov, S., Das, D., and McDonald, R. 2012. A universal part-of-speech tagset. In *Proceedings of the International Conference on Language Resources & Evaluation*, pp. 2089–96. Paris, France: European Language Resources Association.

Santamaría, J., and Araujo L. 2010. Identifying patterns for unsupervised grammar induction. In *Proceedings of the Conference on Computational Natural Language Learning*, pp. 38–45. Stroudsburg, PA: Association for Computational Linguistics.

Seginer, Y. 2007a. Fast unsupervised incremental parsing. In *Proceedings of the Meeting of the Association for Computational Linguistics (ACL)*, pp. 384–91. Stroudsburg, PA: Association for Computational Linguistics.

Seginer, Y. 2007b. *Learning Syntactic Structure*. PhD thesis, Faculty of Science, University of Amsterdam, The Netherlands.

Sima'an, K., Itai, A., Winter, Y., Altman, A. and Nativ, N. 2001. Building a tree-bank of modern Hebrew text. *Traitment Automatique des Langues* **42**: 346–80.

Skut, W., Krenn, B., Brants, T., and Uszkoreit, H. 1997. An annotation scheme for free word order languages. In *Proceedings of the Conference on Applied Natural Language Processing*, pp. 88–95. Stroudsburg, PA: Association for Computational Linguistics.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. 2011. Punctuation: making a point in unsupervised dependency parsing. In *Proceedings of the Conference on Computational*

Natural Language Learning, pp. 19–28. Stroudsburg, PA: Association for Computational Linguistics.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. 2012. Capitalization cues improve dependency grammar induction. In *NAACL-HLT: Workshop on Inducing Linguistic Structure (WILS 2012)*, pp. 16–22. Stroudsburg, PA: Association for Computational Linguistics.

Stolcke, A. and Omohundro, S. M. 1994. Inducing probabilistic grammars by Bayesian model merging. In *Proceedings of the International Colloquium on Grammatical Inference and Applications (ICGI)*, pp. 106–18. London: Springer-Verlag.

van Zaanen, M. 2000. Abl: alignment-based learning. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pp. 961–7. Stroudsburg, PA: Association for Computational Linguistics.

Xia, F., Palmer, M., Xue, N., Okurowski, M. E., Kovarik, J., dong Chiou, F., Huang, S., Kroch, T., and Marcus, M. 2000. Developing guidelines and ensuring consistency for Chinese text annotation. In *Proceedings of the International Conference on Language Resources & Evaluation*, pp. 1–8. Paris, France: European Language Resources Association.

Appendix Tags used in the Penn Treebank examples

Table 11. *Alphabetical list of part-of-speech tags used in the Penn Treebank, used in our experiments*

CC	Coordinating conjunction	TO	To
CD	Cardinal number	UH	Interjection
DT	Determiner	VB	Verb, base form
EX	Existential there	VBD	Verb, past tense
FW	Foreign word	VBG	Verb, gerund/present participle
IN	Preposition/subordinating conjunction	VBN	Verb, past participle
JJ	Adjective	VBP	Verb, non-3rd ps. sing. present
JJR	Adjective, comparative	VBZ	Verb, 3rd ps. sing. present
JJS	Adjective, superlative	WDT	wh-determiner
LS	List item marker	WP	wh-pronoun
MD	Modal	WP$	Possessive wh-pronoun
NN	Noun, singular or mass	WRB	wh-adverb
NNS	Noun, plural	#	Pound sign
NNP	Proper noun, singular	$	Dollar sign
NNPS	Proper noun, plural	.	Sentence – final punctuation
PDT	Predeterminer	,	Comma
POS	Possessive ending	:	Colon, semi-colon
PRP	Personal pronoun	(	Left bracket character
PP$	Possessive pronoun	)	Right bracket character
RB	Adverb	”	Straight double quote
RBR	Adverb, comparative	‘	Left open single quote
RBS	Adverb, superlative	“	Left open double quote
RP	Particle	’	Right close single quote
SYM	Symbol (mathematical or scientific)	”	Right close double quote

Table 12. *Internal tags used in the Penn Treebank examples*

S	Simple declarative clause
NP	Noun phrase
VP	Verb phrase
ADJP	Adjective phrase
SBAR	Clause introduced by a – possibly empty – subordinating conjunction
-SBJ	Marks the structural surface subject
-TMP	Marks temporal or aspectual adverbials
-PRD	Marks purpose or reason clauses and PPs
-TTL	Attached to the top node of a title