UnsuParse: Unsupervised Parsing with unsupervised Part of Speech tagging

Christian Hänig, Stefan Bordag, Uwe Quasthoff
University of Leipzig, Natural Language Processing Dpt.
Johannisgasse 26, D-04081 Leipzig
chr_haenig@yahoo.de, {sbordag, quasthoff}@informatik.uni-leipzig.de

Abstract

Based on simple methods such as observing word and part of speech tag co-occurrence and clustering, we generate syntactic parses of sentences in an entirely unsupervised and self-inducing manner. The parser learns the structure of the language in question based on measuring 'breaking points' within sentences. The learning process is divided into two phases, learning and application of learned knowledge. The basic learning works in an iterative manner which results in a hierarchical constituent representation of the sentence. Part-of-Speech tags are used to circumvent the data sparseness problem for rare words. The algorithm is applied on untagged data, on manually assigned tags and on tags produced by an unsupervised part of speech tagger. The results are unsurpassed by any self-induced parser and challenge the quality of trained parsers with respect to finding certain structures such as noun phrases.

1. Introduction

Recently, unsupervised (also called *knowledge-free*) methods for acquiring language specific knowledge out of a raw text corpus began to receive more attention. Examples for unsupervised algorithms include simulating semantic relatedness of words by comparing co-occurrence vectors (Curran 03, Sahlgren 06, Bordag 07), dividing word forms into morphs (Kurimo 07), word sense induction and disambiguation, or part of speech tagging (Biemann 06). Usually such algorithms do not achieve the same quality as semi-supervised machine learning algorithms trained from manually annotated data. However, in situations where precision is less important compared to the cost of producing manually annotated data or where coverage is more important than precision, unsupervised algorithms represent a viable, cheap and fast source of knowledge. In some cases they achieve similar (Kurimo 07) or even better results than traditional machine learning algorithms when used in real-world applications (Bod 07).

Currently there are several approaches to induce syntactic (and in most cases semantic) structure from a given raw corpus in an unsupervised manner (grammar inference).

One approach is to compare all sentences with each other and hypothesize matching sequences as being constituents such as in the alignment based learning (ABL) (Zaanen 01) or the syntagmatic paradigmatic model (SPM) (Dennis & Harrington 01). This approach has obvious time-complexity problems, which perhaps are solvable by use of heuristics.

Another approach is to measure the in- and outgoing path density of word (or morpheme) sequences within a set of sentences, see the ADIOS system (Solan 06), or the nearly equivalent SOG system (Schwiebert and Rolshoven 06). Here, a graph is built by taking words as nodes and connecting them if they appear in a sequence. Both systems represent an elegant combination of learning syntagmatic and paradigmatic relations in a unified way. For both, but especially for ADIOS the evaluations are more expressive, but a standardized evaluation instance such as the Morpho Challenge for the unsupervised morpheme segmentation task (Kurimo 07) is still missing. There is one independent evaluation, comparing ADIOS with Emile (Adrianns and Veervoort 02) and ABL on a small corpus containing 7k sentences (Cramer 07). According to this evaluation, all three systems are not able to infer structure and only ABL is better then the random baseline. However, this evaluation also shows that data sparseness is the main problem for these algorithms, but fails to test these algorithms on significantly larger corpora. It stands to reason, whether grammar inference is possible on a corpus as small as that.

A different approach is the Incremental Parsing (Seginer 2007). It uses common cover links similar to dependency links. It does not use POS-tags for parsing.

There is also the Constituent-Context Model (CCM) (Klein & Manning 2002), which uses the assumption that constituents appear in constituent context along with a variant that models simple head-outward dependency over word classes including valence (DMV) (Klein 05). This algorithms makes use of the fact that especially long constituents often have short equivalents (pro-forms) appearing in similar contexts. Incidentally, a very similar idea is used to compare compounds with paraphrases of these compounds (Holz & Biemann 08).

Finally, there is a simpler all-subtrees approach (Bod 06a), which is also partly based on earlier work (Klein & Manning 02). It operates by generating all possible binary trees for each encountered sentence. Parsing a new sentence consists of computing the most probable parse from the accumulated frequencies of observed subtrees with respect to the currently observed combination of words or part of speech tags. Problems with this approach again concern mainly computational complexity – but to such a degree that it appears to be impossible to extend the same approach to more than binary subtrees. This algorithm is the first to be systematically compared with traditional machine learning methods.

However, it is still hard to tell how the various approaches would perform when compared directly with each other.

In this work we take the same evaluation route as taken by Rens Bod and compare our algorithm to his and to those based on machine learning.

The algorithm in this work approaches the goal of learning syntactic structure from a different direction. Observations of significant co-occurrences of word forms or part of speech tags allow determining word pairs in a sentence that appear to have a constituent boundary between them or inversely appear to represent a constituent, or, in other words, belong together. Using this information, an iterative learning process combines such words pairs for further iterations until each sentence is a single constituent. This algorithm also takes non-contiguous dependencies into account. It can be applied either on the observed word forms directly, or on part of speech tags. We test the performance of the algorithm both on manually annotated part of speech tags, as well as on automatically acquired ones. We show that such algorithms are indeed easily extensible to be used with unsupervised part of speech taggers (as claimed in (Bod 06b)), but we also show that the resulting performance does not yet compete with using the same approach on manually acquired part of speech tags. We do show, however, that the existence of any part of speech tags dramatically increases the performance. This is because POS tags allow to avoid the data sparseness, or rather move it to the POS tagger (or inducer). As in ABL, we do not come up with an explicit grammar. Instead, our algorithm also produces a bracketed version of the corpus. Compared to the evaluation of ADIOS given in (Cramer 07) our algorithm significantly outperforms the random branching baseline.

The following Section 2 describes in more detail the assumptions that are made for the underlying constituent detection algorithm. Section 3 describes the iterative learning algorithm and how the resulting parser uses the acquired syntactic knowledge. Finally, in Section 4 experimental results are presented and compared to other related work.

2. Constituent detection

One assumption our constituent detection algorithm unsuParse is based on states that a word within a constituent prefers a certain position. Note that this does not state general restrictions on word order. We use two special cases where the word prefers either the first, or the last position of a sentence. These two positions are obviously constituent boundaries. For a given corpus, the variable *a* represents the statistical significance of having observed a word *A* at the end of sentences (marked with the symbol \$) n_A times,:

$$a = sig_{nA}(A,\$)$$

On the contrary, a second variable *b* represents the significance of having observed a different word *B* at the beginning of sentences marked with the symbol ^) n_B times:

$$b = sig_{nB}(\wedge, B)$$

The variables *a* and *b* are then compared with *c*, which is the statistical significance of having observed the word A and then the word B (next to each other in this order) n_{AB} times:

$$c = sig_{nAB}(A,B)$$

Significance for all variables is computed by using the log-likelihood significance measure which takes the four parameters corpus size n, frequency n_A of term A, frequency n_B of term B and frequency n_{AB} of co-occurrence and returns a value corresponding to the significance of the observed n_{AB}-fold joint co-occurrence of A and B to be not random events. In this setup, values over a threshold such as roughly 5 can be assumed to be significant with an error probability of 2.5%.

Comparing *a* and *b* with *c* is done by defining the separation value $sep(A,B)$ of any two words A and B in a sentence:

$$sep(A,B) = \frac{a \cdot b}{c^2}$$

The motivation behind this is the following. As long as we do not yet know the boundary of constituents, we can begin by assuming that sentence boundaries are constituent boundaries. The variable a is larger than c (and hence, the quotient $a/c > 1$) if the word A occurs more significantly at the end of a sentence as compared to occurring before B. Additionally, the variable *b* is larger than *c* (and correspondingly the quotient $b/c > 1$) if the word B occurs more significantly at the beginning of a sentence as compared to occurring after A. When A occurs in front of B and the product of both quotients is larger than one, then obviously this is a very atypical combination for A and B and the words A and B represent the end of an old and the beginning of a new constituent, respectively. In other words, in this case there is a constituent border between A and B.

The resulting basic algorithm for learning parse trees then proceeds by iteratively picking the smallest separation value and merging the two corresponding words into a new node and treating this node as a new constituent in the further iterations. The following example illustrates typical values and the resulting bracketing.

- Input sentence: Ich kaufe mir das Auto (engl. I buy [me] the car)
- Separation values: Ich 0.02 kaufe 2.01 mir 1.57 das 0.04 Auto
- Iteration steps:
  - [Ich 0.02 kaufe] 2.01 mir 1.57 das 0.04 Auto
  - [Ich 0.02 kaufe] 2.01 mir 1.57 [das 0.04 Auto]
  - [Ich 0.02 kaufe] 2.01 [mir 1.57 [das 0.04 Auto]]
- Resulting Bracketing: [Ich kaufe] [mir [das Auto]]

3. Extensions

This basic separation value (in Table 1 referenced to as ‘unsuParse on words’) already detects intuitive constituent boundaries and is especially good at finding noun phrases. However, it has the following weakness: It is only reliable if both words A and B are sufficiently frequent to get reliable values for the co-occurrence measure. This is not the case in the following two cases:

1. NPs of the type Det-Adj-N with a very low frequent Adj are not recognized. The same applies more generally to very low frequent words within a constituent.
2. On the contrary but due to the same reason, some very frequent word combinations such as “and in”, belonging to different phrases, will not be separated.

Therefore several enhancements are introduced, but only briefly described at this point. The complete algorithm with all enhancements is referred to as ‘unsuParse on unsuPOS tags’ in Table 1.

3.1. Dealing with rare words: larger windows

For the first problem there are two solutions. The first solution is to skip these low frequent words, or equivalently, to enlarge window size and analyze the separation value for more distant words. For example when computing separation values between C and D in the sequence A B C D, then in the basic version only the pair C D is considered. Instead, it makes sense to consider the pairs A D and B D additionally.

In this reformulation, all pairings of words from the left and right side of a possible constituent border within a sentence are taken into account that have at least one word next to the position *i*. Given a position *i* between two words n_i and n_{i+1} within a sentence of length *m*, the new separation value *sep(i)* is than the minimum of the separation values of all pairs of words where the one word is anything from n_0 to n_i and the other word from n_{i+1} to nm:

$$sep(i) = \min\left(\min_{\substack{j=0\ldots i \\ k=i+1}} \left(sep(n_j, n_k) \right), \min_{\substack{j=i \\ k=i+1\ldots m}} \left(sep(n_j, n_k) \right) \right)$$

This allows the algorithm to cope with several of the aforementioned problems such as atypical adjectives breaking up noun phrases, because it can take long distance dependencies into account. Hence, in the example sentence ‘I want to buy a fast and costly car’ the new algorithm is still able to detect the noun phrase ‘a fast and costly car’.

3.2. Using POS tags and positional preferences

A solution that more generally takes care of the frequency distortion problems is to use POS tags. In the case of a rare word, it can be replaced by its POS tag to get more reliable statistical information. Especially with nouns and adjectives it is the case that most nouns are too infrequent for reliable statistics. However, when they are all summarized into a single tag, such as “NN”, this tag becomes a very frequent chunk whose distributional properties can then be analyzed properly in a statistically based method.

The combination of POS information with already acquired knowledge about constituents allows to compute a preference value to each POS tag. Hence, the value *pref(A)* expresses the preference of A to be the first element in a constituent:

$$pref(A) = 2^{f-l}$$

where *f* is the number of constituents with A at the first position and l the number of constituents with A at the last position. Hence, *pref(A)* becomes large if A prefers the front position of newly found constituents and small in the opposite case. Hence the reformulation of the basic separation value

$$sep(A,B) = \frac{pref(B)}{pref(A)} \frac{a \cdot b}{c^2}$$

takes the knowledge already learned at any point of the learning process into account when learning further rules.

3.3. Iterative learning in two phases

A third enhancement is to make rule learning iterative and to split learning into two phases. In each iteration the entire corpus is processed and for each sentence only the best merge of two words or phrases is accepted and treated as a new constituent in the following iteration. However, forcedly joining two constituents in a sentence where the separation value has a large value results in frequent mistakes.

Therefore we split the learning into a safe and an unsafe learning phase and into a parsing phase. In the first learning phase in each iteration a frequency ranking of hypothesized new constituents is used to cut off probably correct from probably incorrect ones. This phase ends once no more constituents can be found. This means that either the corresponding separation values are above a threshold or the frequency of the hypothesized constituents is too low. The unsafe learning then proceeds by combining all remaining constituents of each sentence hierarchically according to the separation values.

Parsing sentences works in a very straightforward way by finding the most significant constituents first. Significance of constituents is derived from the learning phase in that the earlier a constituent was learned, the more significant it is.

Additionally, if a very significant constituent contains another one then that subtree is flattened into a ternary (or more) tree. For example in “We have a pretty house” normally it would first find “pretty house” due to the highly significant constituent akin to ADJ NN. Then it would find the constituent that essentially says DET NP. However, both constituent types were learned early and hence, the resulting bracketing is “We have [a pretty house]”

Finally, specific patterns are used to find phrases that belong together. For example if the analysis is “[Mr. Peters] [his [pretty pet]] …” then the first phrase should be bracketed together with the second phrase. Such cases are recognized by means of the following method. Once both noun phrases were found, the last word of the first phrase is checked, whether it occurs significantly often next to the first word of the second phrase or with the entire phrase. The same is done checking, whether the first word of the second phrase co-occurs significantly often with the last word of the first phrase. If any of these conditions is met, the two phrases are found to belong together.

Clearly, the most important enhancement is to apply the entire algorithm not on the words directly, but on their part of speech tags, instead. This allows the algorithm to have a clearer view on the structure of the sentence without being hampered too much by specific typical uses of certain words. This also circumvents the data sparseness problem, because even very rare words are subsumed under word classes such as nouns, adjectives or adverbs (assuming that the tagging is correct).

Algorithm	Precision	Recall	F
CCM	0.481	0.855	0.616
DMV	0.384	0.695	0.495
DMV+CCM	0.496	0.897	0.639
U-DOP	0.512	0.905	0.654
U-DOP*			0.638
UML-DOP			**0.652**
unsuParse on Negra tags	0.769	0.539	**0.634**
Baseline	0.279	0.496	0.357
upper bound	0.563	1.000	0.721
Incremental Parsing	0.510	0.698	0.590
unsuParse on words	0.337	0.628	0.439
unsuParse on unsuPOS tags	0.612	0.591	**0.602**

Table 1: The upper part shows grammar inference algorithms based on manually annotated POS tags, whereas the lower part shows algorithms applied on words directly or on automatically induced POS tags. CCM is the Constituent-Context Model (Klein and Manning 02), the three variants of DOP represent the all subtree approach (Bod 06a) and Incremental Parsing is the algorithm from Seginer (Seginer 07). Our algorithm unsuParse is applied either on the NEGRA tags or on induced tags using the unsuPOS algorithm (Biemann 06).

4. Evaluation and Conclusions

In order to assess the quality of the parses generated by the complete algorithm, several evaluations were run and a few examples are given.

In line with the evaluation in (Klein and Manning 2004), the algorithm was tested on a subset of the NEGRA Corpus (Skut et al. 98) containing all sentences with at most 10 words (referred to as NEGRA10). Using the same measures as in (Klein and Manning 2004), which essentially means counting brackets matching with the gold standard, allows to compare our algorithm with other existing algorithms. Other algorithms tested with this method include CCM (Klein and Manning 02), DMV (Klein 05), Incremental Parsing (Seginer 2007) and several variants of the U-DOP algorithm (Bod 06a). The results are shown in Table 1 with values for other algorithms taken from the respective publications.

The evaluated solutions are divided into two groups – first those that use manually annotated part of speech tags (and thus are not fully unsupervised) and those that are applied either on the words directly or on part of speech tags acquired from an unsupervised part of speech tagger, such as unsuPOS (Biemann 06). Additionally, a baseline is given which shows the performance of an algorithm that finds constituents randomly (i.e. the separation values are produced by a random number generator). An upper bound is provided which shows the maximum achievable performance when using binary trees (relevant for U-DOP).

Since unsuParse is not restricted to binary trees, it is unsurprising that it has the highest precision out of all compared algorithms. However, despite being able to account for distant dependencies, the restriction to contiguous constituents and other effects, such as typical prepositions reduces recall significantly. Nevertheless, the performance of this computationally less demanding algorithm compares very well with other approaches and among the fully unsupervised implementations it is currently the best.

Another evaluation run on more complex sentences underlines this. The results of an evaluation using all sentences of the NEGRA Corpus with at most 40 words (referred to as NEGRA40) and, as above, the same measures as in (Klein and Manning 2004) are shown in Table 2. While the results of testing on NEGRA10 are closer to each other, this experiment shows a bigger difference compared to Incremental Parsing. However, the performance of both algorithms decreases significantly for long sentences.

Algorithm	Precision	Recall	F
Incremental Parsing	0.348	0.489	0.406
unsuParse on unsuPOS tags	0.476	0.435	**0.455**

Table 2: Evaluation of the Incremental Parsing (Seginer 07) and our algorithm on long sentences.

Hence it can be said that unsuParse can be used to parse longer, more natural sentences, but probably only the lower parts of the resulting syntactic tree can be assumed to be mostly correct, including specific noun, adjective and verb phrases as well as simple combinations of specific phrases. For many tasks, such as Information Extraction or Named Entity recognition this might prove to be helpful.

In order to assess the influence the various parts of the algorithm we performed several tests where specific parts of the algorithm were omitted. Specifically, we created the following versions (and tested them on the NEGRA tags):

- unsuParse : This is the full version of out algorithm for reference.
- unsuParseNB : This is based only on the initially

introduced variable *c* of the separation value. It does not take *a*, *b* and positional preferences into account and does not distinguish between safe and unsafe learning.

- unsuParse safe learning : This is the full version except that it does not apply the forced combination of all remaining constituents irrespective of the separation values (the unsafe learning step).
- unsuParseNB safe learning : This is based only on the *c* variable and the safe learning step without the unsafe learning.
- unsuParseHybrid : This version is like the full version until including the safe learning step. Afterwards it proceeds by combining all remaining constituents based only the variable *c*.

These experiments provide a number of surprising insights. Apparently the variable *c* already suffices to produce very competitive results, assuming POS tags were used. The extensions prove to be useful, but their effect is relatively small.

It is important to note that the highest F-score for a subpart of the algorithm comes at a cost in Recall. Essentially, the version "unsuParse safe learning" is unable to completely parse sentences. Only the full version "unsuParse" produces full parses.

Algorithm	Precision	Recall	F
unsuParse	0.535	0.666	0.593
unsuParseNB	0.553	0.668	**0.605**
unsuParse safe learning	0.769	0.539	**0.634**
unsuParseNB safe learning	0.679	0.558	0.612
unsuParseHybrid	0.546	0.684	**0.607**

Table 3: Tests of various versions of the algorithm where certain mechanisms were omitted.

It also seems that the influence of three variables and the preference quotient improves the results during earlier iterations. Simplifying the separation value to only variable *c* improves the results after the complete learning phase slightly. Combining those two separation values to create a hybrid algorithm which first uses all variables and changes to simple neighborhood co-occurrences after the safe learning has almost no effect.

Examples

The following examples from Spanish, German, English and French illustrate the performance of unsuParse:

Spanish:
```
[Todo ello , [de [conformidad con los]] [principios que] siempre [hemos apoyado]]

[Mi Grupo [ha hecho] importantes [enmiendas [a los]] dos informes que [se debaten] hoy]
```

German:
```
[[Die Titel] [Feldbergfestsieger [und -siegerin]] werden [[in der Dreikampf-Oberstufe] vergeben]]

[Das [von Seoul] finanzierte Projekt ist [in der Anfangsphase]]
```

English:
```
[At [the beginning] , [the Mexican attitude] was very macho]

[Barco said he will present [the [proposed treaty]] to [the lawmakers] [next week]]

[[Bondholders agreed] to reschedule [the debt payments]]
```

French:
```
[[Lionnel Luca] est député [des Alpes-Maritimes] (UMP)]

[Et , ajoute-t-on , “ [il est] [essentiel [de respecter]] [les engagements] du ministre ” ]
```

However, it should be noted that a typical scenario for such an algorithm is a language for which there is no syntactic parser yet. This usually also means the absence of a POS tagger. This is not contradiction to the assumptions given in Section 3.2. It suffices to have identical tags for words with similar syntactic features. Some tagging weaker than POS tagging is sufficient. We refer to (Biemann 06) for unsupervized POS tagging which works without any prior knowledge about the language under consideration.

More examples can be easily generated since the data format used is the same as in the Leipzig Corpora Collection which offers over a dozen different languages (Biemann et al. 07). The entire Leipzig corpora collection will soon be made available with such unsupervised parses included. Finally, this is a very simple and easily extensible approach and it provides insights into how fully unsupervised methods for parsing can be further developed.

5. References

Adriaans, Pieter W. and Mark R. Vervoort (2002): The EMILE 4.1 grammar induction toolbox. In Proceedings of the 6th International Colloquium on Grammar Induction (ICGI), pages 293–295, Amsterdam, the Netherlands

Biemann Chris (2006): Unsupervised Part-of-Speech Tagging Employing Efficient Graph Clustering. In: Proceedings of the COLING/ACL-06 Student Research Workshop 2006, Sydney, Australia

Biemann, Chris, Gerhard Heyer, Uwe Quasthoff and Matthias Richter (2007): The Leipzig Corpora Collection – Monolingual corpora of standard size. In: Proceedings of Corpus Linguistics 2007, Birmingham, UK

Bod, Rens (2006a): An All-Subtrees Approach to Unsupervised Parsing. Proceedings ACL-COLING 2006, Sydney, Australia

Bod, Rens (2006b): Unsupervised Parsing with U-DOP. In: Proceedings CoNLL 2006, New York, USA

Bod, Rens (2007): Is the End of Supervised Parsing in Sight? In: Proceedings of the ACL 2007, Prague, Czech Republic

Bordag, Stefan (2007): Elements of Knowledge-free and Unsupervised Lexical Acquisition. Ph.D. thesis, Natural Language Processing Department, University of Leipzig, Germany

Cramer, Bart (2007): Limitations of Current Grammar Induction Algorithms. In: Proceedings of the ACL 2007 Student Research Workshop, pages 43-48.

Curran, James Richard (2003): From Distributional to Semantic Similarity. Ph.D. thesis, Institute for Communicating and Collaborative Systems, School of Informatics. University of Edinburgh, Edinburgh, UK.

Dennis, Simon and Michael Harrington (2001): The Syntagmatic Paradigmatic Model: An distributed instance-based model of sentence processing. The Second Workshop on Natural Language Processing and Neural Networks, Tokyo, Japan

Holz, Florian and Chris Biemann (2008): Unsupervised and Knowledge-Free Learning of Compound Splits and Periphrases. Proceedings of CicLING-08, Haifa, Israel

Klein, Dan and Christopher Manning (2002): A Generative Constituent-Context Model for Improved Grammar Induction, In: Proceedings of the ACL 2002. Philadelphia, USA

Klein, Dan and Chris Manning (2004): Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency, In Proceedings of the ACL 2004, Barcelona, Spain

Klein, Dan. 2005. The Unsupervised Learning of Language Structure. Ph.D. thesis, Stanford University, Stanford, CA, USA

Kurimo, Mikko, Mathias Creutz and Ville Turunen (2007): Overview of Morpho Challenge in CLEF 2007. In: Working Notes for the CLEF 2007 Workshop. Budapest, Hungary

Sahlgren, Magnus (2006). The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in highdimensional vector spaces. Ph.D. thesis, Swedish Intitute of Computer Science, Stockholm, Sweden

Schwiebert, Stephan und Jürgen Rolshoven (2006): SOG: Ein selbstorganisierender Graph zur Bildung von Paradigmen. In: Rapp, Reinhard, Sedlmeier & Zunker-Rapp: Perspectives on Cognition. A Festschrift for Manfred Wettler. Lengerich: Pabst Science Publishers

Seginer, Yoav (2007): Fast Unsupervised Incremental Parsing. In: Proceedings of the ACL 2007, Prague, Czech Republic

Skut, Wojciech, Thorsten Brants, Brigitte Krenn and Hans Uszkoreit (1998): A linguistically interpreted corpus of German newspaper text. In ESSLI 1998, Workshop on Recent Advances in Corpus Annotation. Saarbrücken, Germany

Solan Zach (2006): Unsupervised Learning of Natural Languages, PhD thesis, School of Physics and Astronomy, Tel-Aviv University, Israel

van Zaanen, Menno (2001): Bootstrapping Structure into Language: Alignment-Based Learning, PhD thesis, School of Computing, University of Leeds, UK