Covington Variations

Svetoslav Marinov
School of Humanities and Informatics,
University College Skövde, 54128 Skövde &
GSLT, Göteborg University, 40530 Göteborg,
Sweden
Svetoslav.Marinov@his.se

Abstract

Three versions of the Covington algorithm for non-projective dependency parsing have been tested on the ten different languages for the Multilingual track of the CoNLL-X Shared Task. The results were achieved by using only information about heads and daughters as features to guide the parser which obeys strict incrementality.

1 Introduction

In this paper we focus on two things. First, we investigate the impact of using different flavours of Covington's algorithm (Covington, 2001) for non-projective dependency parsing on the ten different languages provided for CoNLL-X Shared Task (Nivre et al., 2007). Second, we test the performance of a pure grammar-based feature model in strictly incremental fashion. The grammar model relies only on the knowledge of heads and daughters of two given words, as well as the words themselves, in order to decide whether they can be linked with a certain dependency relation. In addition, none of the three parsing algorithms guarantees that the output dependency graph will be projective.

2 Covington's algorithm(s)

In his (2001) paper, Covington presents a "fundamental" algorithm for dependency parsing, which he claims has been known since the 1960s but has, up to his paper-publication, not been presented systematically in the literature. We take three of its flavours, which enforce uniqueness (a.k.a. single-headedness) but do not observe projectivity. The algorithms work one word at a time and attempt to build a connected dependency graph with only a single left-to-right pass through the input. The three flavours are: Exhaustive Search, Head First with Uniqueness (ESHU), Exhaustive Search Dependents First with Uniqueness (ESDU) and List-based search with Uniqueness (LSU).

<u>**ESHU**</u>	<u>**ESDU**</u>
for *i* = 1 **to** *n*	**for** *i* = 1 **to** *n*
for *j* = *i*-1 **downto** 0	**for** *j* = *i*-1 **downto** 0
if HEAD?(*j*,*i*)	**if HEAD?**(*i*,*j*)
LINK(*j*,*i*)	**LINK**(*i*,*j*)
if HEAD?(*i*,*j*)	**if HEAD?**(*j*,*i*)
LINK(*i*,*j*)	**LINK**(*j*,*i*)

The yes/no function **HEAD?**(*w1*,*w2*), checks whether a word *w1* can be a head of a word *w2* according to a grammar *G*. It also respects the single-head and no-cycle conditions. The **LINK**(*w1*,*w2*) procedure links word *w1* as the head of word *w2* with a dependency relation as proposed by *G*. When traversing *Headlist* and *Wordlist* we start with the last word added. (Nivre, 2007) describes an optimized version of Covington's algorithm implemented in MaltParser (Nivre, 2006) with a running time $c(\frac{n^2}{2} - \frac{n}{2})$ for an n-word sentence, where c is some constant time in which the **LINK** operation can be performed. However, due to time constraints, we will not bring this version of the algorithm into focus, but see some preliminary remarks on it with respect to our parsing model in 6.

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1144–1148,
Prague, June 2007.

```
LSU[1]
Headlist := []
Wordlist := []
while (!end-of-sentence)
  W := next input word;
  foreach D in Headlist
    if HEAD?(W,D)
      LINK(W,D);
      delete D from Headlist;
  end
  foreach H in Wordlist
    if HEAD?(H,W)
      LINK(H,W);
      terminate this foreach loop;
  end
  if no head for W was found then
    Headlist := W + Headlist;
  end
  Wordlist := W + Wordlist;
end
```

3 Classifier as an Instant Grammar

The **HEAD?** function in the algorithms presented in 2, requires an "instant grammar" (Covington, 2001) of some kind, which can tell the parser whether the two words under scrutiny can be linked and with what dependency relation. To satisfy this requirement, we use TiMBL - a Memory-based learner (Daelemans et al., 2004) - as a classifier to predict the relation (if any) holding between the two words.

Building heavily on the ideas of History-based parsing (Black et al., 1993; Nivre, 2006), training the parser means essentially running the parsing algorithms in a learning mode on the data in order to gather training instances for the memory-based learner. In a learning mode, the **HEAD?** function has access to a fully parsed dependency graph. In the parsing mode, the **HEAD?** function in the algorithms issues a call to the classifier using features from the parsing history (i.e. a partially built dependency graph PG).

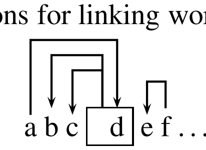
Given words i and j to be linked, and a PG, the call to the classifier is a feature vector $\Phi(i,j,PG) = (\phi_1,\ldots,\phi_m)$ (cf. (Nivre, 2006; Nivre, 2007)). The classifier then attempts to map this feature vector to any of predefined classes. These are all the dependency relations, as defined by the treebank and the class "NO" in the cases where no link between the two words is possible.

4 The Grammar model

The features used in our history-based model are restricted only to the partially built graph PG. We call this model a pure grammar-based model since the only information the parsing algorithms have at their disposal is extracted from the graph, such as the head and daughters of the current word. Preceding words not included in the PG as well as words following the current word are not available to the algorithm. In this respect such a model is very restrictive and suffers from the pitfalls of the incremental processing (Nivre, 2004).

The motivation for the chosen model, was to approximate a Data Oriented Parsing (DOP) model (e.g. (Bod et al., 2003)) for Dependency Grammar. Under DOP, analyses of new sentences are produced by combining previously seen tree fragments. However, the tree fragments under the original DOP model are static, i.e. we have a corpus of all possible subtrees derived from a treebank. Under our approach, these tree fragments are built dynamically, as we try to parse the sentence. Because of the chosen DOP approximation, we have not included information about the preceding and following words of the two words to be linked in our feature model.

To exemplify our approach, (1) shows a partially build graph and all the words encountered so far and Fig. 1 shows two examples of the tree-building operations for linking words f and d, and f and a.

(1) 

Given two words i and j to be linked with a dependency relation, such that word j precedes word i, the following features describe the models on which the algorithms have been trained and tested:

Word form: i, j, $ds(i)$, $ds(j)$, $h(j/i)$, $h(h(j/i))$
Lemma (if available): i, j, $ds(i)$, $ds(j)$, $h(j/i)$, $h(h(j/i))$

[1] Covington adds *W* to the *Wordlist* as soon as it has been seen, however we have chosen to wait until after all tests have been completed.

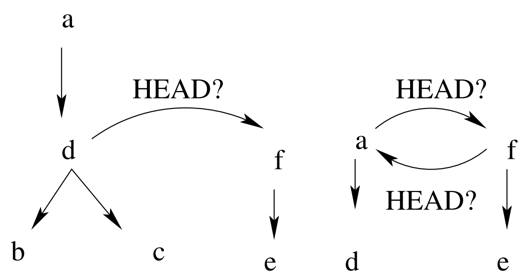


Figure 1: Application of the **HEAD?** function on an input from the PG in (1)

Part-of-Speech: $i, j, ds(i), ds(j), h(j/i), h(h(j/i))$
Dependency type: i, j, $ds(i)$, $ds(j)$, $h(j/i)$, $h(h(j/i))$
Features (if available): i, j, $ds(i)$, $ds(j)$, $h(j/i)$, $h(h(j/i))$

$ds(i)$ means any two daughters (if available) of word i, $h(i/j)$ refers to the head of word i or word j, depending on the direction of applying the **HEAD?** function (see Fig 1) and $h(h(i/j))$ stands for the head of the head of word i or word j.

The basic model, which was used for the largest training data sets of Czech and Chinese, includes only the first four features in every category. A larger model used for the datasets of Catalan and Hungarian adds the $h(j/i)$ feature from every category. The enhanced model used for Arabic, Basque, English, Greek, Italian and Turkish uses the full set of features. This tripartite division of models was motivated only by time- and resource-constraints. The simplest model is for Chinese and uses only 5 features while the enhanced model for Arabic for example uses a total of 39 features.

5 Results and Setup

Table 1 summarizes the results of testing the three algorithms on the ten different languages.

The parser was written in C#. Training and testing were performed on a MacOSX 10.4.9 with 2GHz Intel Core2Duo processor and 1GB memory, and a Dell Dimension with 2.80GHz Pentium 4 processor and 1GB memory running Mepis Linux. TiMBL was run in client-server mode with default settings (IB1 learning algorithm, extrapolation from the most similar example, i.e. $k = 1$, initiated with the command "Timbl -S <portnumber> -f <training_file>"). Additionally, we attempted to use Support Vector Machines (SVM) as an alternative classifier. However, due to the long training time, results from using SVM were not included but training an SVM classifier for some of the languages has started.

	ESHU	ESDU	LSU
Arabic	LA: 53.72 UA: 63.58	LA: **54.00** UA: 63.76	LA: 53.86 UA: **63.78**
Basque	LA: 49.52 UA: 56.83	LA: 50.20 UA: 57.81	LA: **51.24** UA: **58.53**
Catalan	LA: 69.56 UA: 74.32	LA: **69.80** UA: **74.46**	LA: 69.42 UA: 74.22
Chinese	LA: 47.57 UA: 53.46	LA: **50.61** UA: **56.75**	LA: 49.82 UA: 56.02
Czech	LA: 44.41 UA: 49.20	LA: **53.66** UA: **60.01**	LA: 53.47 UA: 59.55
English	LA: 51.05 UA: 53.41	LA: 51.35 UA: 53.65	LA: **52.11** UA: **54.33**
Greek	LA: 54.68 UA: 61.55	LA: 54.62 UA: 61.45	LA: **55.02** UA: **61.80**
Hungarian	LA: 44.34 UA: 50.12	LA: **45.11** UA: **50.78**	LA: 44.57 UA: 50.46
Italian	LA: **61.60** UA: **67.01**	LA: 60.95 UA: 66.25	LA: 61.52 UA: 66.39
Turkish	LA: 55.57 UA: 62.13	LA: **57.01** UA: **63.77**	LA: 56.59 UA: 63.17

Table 1: Test results for the 10 languages. LA is the Labelled Attachment Score and UA is the Unlabelled Attachment Score

6 Discussion

Before we attempt a discussion on the results presented in Table 1, we give a short summary of the basic word order typology of these languages according to (Greenberg, 1963). Table 2 shows whether the languages are SVO (subject-verb-object) or SOV (subject-object-verb), or VSO (verb-subject-object); contain Pr (prepositions) or Po (postpositions); NG (noun precedes genitive) or GN (genitive precedes noun); AN (adjective precedes noun) or NA (noun precedes adjective).

[2]Greenberg had give *varying* for the word-order typology of English. However, we trusted our own intuition as well as the hint of one of the reviewers.

Arabic	VSO	Pr	NG	NA
Basque	SOV	Po	GN	NA
Catalan	SVO	Pr	NG	NA
Chinese	SVO	Po	GN	AN
Czech	SVO	Pr	NG	AN
English[2]	SVO	Pr	GN	AN
Greek	SVO	Pr	NG	AN
Hungarian	SOV	Po	GN	AN
Italian	SVO	Pr	NG	NA
Turkish	SOV	Po	?	AN

Table 2: Basic word order typology of the ten languages following Greenberg's Universals

Looking at the data in Table 1, several observations can be made. One is the different performance of languages from the same language family, i.e. Italian, Greek and Catalan. However, the head-first (ESHU) algorithm presented better than the dependents-first (ESDU) one in all of these languages. The SOV languages like Hungarian, Basque and Turkish had preference for the dependent's first algorithms (ESDU and LSU). The ESDU algorithm also fared better with the SVO languages, except for Italian.

However, the Greenberg's basic word order typology cannot shed enough light into the performance of the three parsing algorithms. One question that pops up immediately is whether a different feature-model using the same parsing algorithms would achieve similar results. Can the different performance be attributed to the treebank annotation? Would another classifier fare better than the Memory-based one? These questions remain for future research though.

Finally, for the Basque data we attempted to test the optimized version of the Covington algorithm (Nivre, 2007) against the three other versions discussed here. Additionally, since our feature vectors differed from those described in (Nivre, 2007), head-dependent-features vs. j-i-features, we changed them so that all the four algorithms send a similar feature vector, j-i-features, to the classifier. The preliminary result was that Nivre's version was the fastest, with fewer calls to the **LINK** procedure and with the smallest training data-set. However, all the four algorithms showed about 20% decrease in LA/UA scores.

Our first intuition about the results from the tests done on all the 10 languages was that the classification task suffered from a highly skewed class distribution since the training instances that correspond to a dependency relation are largely outnumbered by the "NO" class (Canisius et al., 2006). The recall was low and we expected the classifier to be able to predict more of the required links. However, the results we got from additional optimizations we performed on Hungarian, following recommendation from the anonymous reviewers, may lead to a different conclusion. The chosen grammar model, relying only on connecting dynamically built partial dependency graphs, is insufficient to take us over a certain threshold.

7 Conclusion

In this paper we showed the performance of three flavours of Covington's algorithm for non-projective dependency parsing on the ten languages provided for the CoNLL-X Shared Task (Nivre et al., 2007). The experiment showed that given the grammar model we have adopted it does matter which version of the algorithm one uses. The chosen model, however, showed a poor performance and suffered from two major flaws - the use of only partially built graphs and the pure incremental processing. It remains to be seen how these parsing algorithms will perform in a parser, with a much richer feature model and whether it is worth using different flavours when parsing different languages or the differences among them are insignificant.

Acknowledgements

We would like to thank the two anonymous reviewers for their valuable comments. We are grateful to Joakim Nivre for discussion on the Covington algorithm, Bertjan Busser for help with TiMBL, Antal van den Bosch for help with paramsearch, Matthew Johnson for providing the necessary functionality to his .NET implementation of SVM and Patrycja Jabłońska for discussion on the Greenberg's Universals.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

Ezra Black, Frederick Jelinek, John D. Lafferty, David M. Magerman, Robert L. Mercer, and Salim Roukos. 1993. Towards history-based grammars: Using richer models for probabilistic parsing. In *Meeting of the Association for Computational Linguistics*, pages 31–37.

R. Bod, R. Scha, and K. Sima'an, editors. 2003. *Data Oriented Parsing*. CSLI Publications, Stanford University, Stanford, CA, USA.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7, pages 103–127.

Sander Canisius, Toine Bogers, Antal van den Bosch, Jeroen Geertzen, and Erik Tjong Kim Sang. 2006. Dependency Parsing by Inference over High-recall Dependency Predictions. In *CoNLL-X Shared Task on Multitlingual Dependency Parsing*.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

Michael A. Covington. 2001. A Fundamental Algorithm for Dependency Parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102, Athens, Georgia, USA.

D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2004. Timbl: Tilburg memory based learner, version 5.1, reference guide. Technical report, ILK Technical Report 04-02, available from http://ilk.uvt.nl/downloads/pub/papers/ilk0402.pdf.

Joseph H. Greenberg. 1963. *Universals of Language*. London: MIT Press.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez, and M. Bertran. 2007. CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (Abeillé, 2003), chapter 11, pages 189–210.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the CoNLL 2007 Shared Task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Joakim Nivre. 2004. Incrementality in Deterministic Dependency Parsing. In *Incremental Parsing: Bringing Engineering and Cognition Together*, Workshop at ACL-2004, pages 50–57, Barcelona, Spain, July, 25.

Joakim Nivre. 2006. *Inductive Dependency Parsing*. Springer.

Joakim Nivre. 2007. Incremental Non-Projective Dependency Parsing. In *Proceedings of NAACL-HLT 2007*, Rochester, NY, USA, April 22–27.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.