Incremental Dependency Parsing Using Online Learning

Richard Johansson and **Pierre Nugues**
Department of Computer Science, Lund University, Sweden
{richard, pierre}@cs.lth.se

Abstract

We describe an incremental parser that was trained to minimize cost over sentences rather than over individual parsing actions. This is an attempt to use the advantages of the two top-scoring systems in the CoNLL-X shared task.

In the evaluation, we present the performance of the parser in the Multilingual task, as well as an evaluation of the contribution of bidirectional parsing and beam search to the parsing performance.

1 Introduction

The two best-performing systems in the CoNLL-X shared task (Buchholz and Marsi, 2006) can be classified along two lines depending on the method they used to train the parsing models. Although the parsers are quite different, their creators could report near-tie scores. The approach of the top system (McDonald et al., 2006) was to fit the model to minimize cost over sentences, while the second-best system (Nivre et al., 2006) trained the model to maximize performance over individual decisions in an incremental algorithm. This difference is a natural consequence of their respective parsing strategies: CKY-style maximization of link score and incremental parsing.

In this paper, we describe an attempt to unify the two approaches: an incremental parsing strategy that is trained to maximize performance over sentences rather than over individual parsing actions.

2 Parsing Method

2.1 Nivre's Parser

We used Nivre's algorithm (Nivre et al., 2006), which is a variant of the shift–reduce parser. Like the regular shift–reduce, it uses a stack S and a list of input words W, and builds the parse tree incrementally using a set of parsing actions (see Table 1). It can be shown that Nivre's parser creates projective and acyclic graphs and that every projective dependency graph can be produced by a sequence of parser actions. In addition, the worst-case number of actions is linear with respect to the number of words in the sentence.

2.2 Handling Nonprojective Parse Trees

While the parsing algorithm produces projective trees only, nonprojective arcs can be handled using a preprocessing step before training the model and a postprocessing step after parsing the sentences.

The projectivization algorithm (Nivre and Nilsson, 2005) iteratively moves each nonprojective arc upward in the tree until the whole tree is projective. To be able to recover the nonprojective arcs after parsing, the projectivization operation replaces the labels of the arcs it modifies with traces indicating which links should be moved and where attach to attach them (the "Head+Path" encoding). The model is trained with these new labels that makes it possible to carry out the reverse operation and produce nonprojective structures.

2.3 Bidirectional Parsing

Shift-reduce is by construction a directional parser, typically applied from left to right. To make better use of the training set, we applied the algorithm in both directions as Johansson and Nugues (2006) and Sagae and Lavie (2006) for all languages except Catalan and Hungarian. This, we believe, also has the advantage of making the parser less sensitive to whether the language is head-initial or head-final.

We trained the model on projectivized graphs from left to right and right to left and used a voting strategy based on link scores. Each link was assigned a score (simply by using the score of the `la` or `ra` actions for each link). To resolve the conflicts

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1134–1138,
Prague, June 2007.

Table 1: Nivre's parser transitions where W is the initial word list; I, the current input word list; A, the graph of dependencies; and S, the stack. (n', n) denotes a dependency relations between n' and n, where n' is the head and n the dependent.

Actions	**Parser actions**	**Conditions**
Initialize	$\langle nil, W, \emptyset \rangle$	
Terminate	$\langle S, nil, A \rangle$	
Left-arc	$\langle n\|S, n'\|I, A \rangle \rightarrow \langle S, n'\|I, A \cup \{(n', n)\} \rangle$	$\neg\exists n''(n'', n) \in A$
Right-arc	$\langle n\|S, n'\|I, A \rangle \rightarrow \langle n'\|n\|S, I, A \cup \{(n, n')\} \rangle$	$\neg\exists n''(n'', n') \in A$
Reduce	$\langle n\|S, I, A \rangle \rightarrow \langle S, I, A \rangle$	$\exists n'(n', n) \in A$
Shift	$\langle S, n\|I, A \rangle \rightarrow \langle n\|S, I, A \rangle$	

between the two parses in a manner that makes the tree projective, single-head, rooted, and cycle-free, we applied the Eisner algorithm (Eisner, 1996).

2.4 Beam Search

As in our previous parser (Johansson and Nugues, 2006), we used a beam-search extension to Nivre's original algorithm (which is greedy in its original formulation). Each parsing action was assigned a score, and the beam search allows us to find a better overall score of the sequence of actions. In this work, we used a beam width of 8 for Catalan, Chinese, Czech, and English and 16 for the other languages.

3 Learning Method

3.1 Overview

We model the parsing problem for a sentence $\boldsymbol{x}$ as finding the parse $\hat{y} = \arg\max_y F(\boldsymbol{x}, y)$ that maximizes a discriminant function F. In this work, we consider linear discriminants of the following form:

$$F(\boldsymbol{x}, y) = \boldsymbol{w} \cdot \boldsymbol{\Psi}(\boldsymbol{x}, y)$$

where $\boldsymbol{\Psi}(\boldsymbol{x}, y)$ is a numeric feature representation of the pair $(\boldsymbol{x}, y)$ and $\boldsymbol{w}$ a vector of feature weights. Learning F in this case comes down to assigning good weights in the vector $\boldsymbol{w}$.

Machine learning research for similar problems have generally used margin-based formulations. These include global batch methods such as *SVM*^{struct} (Tsochantaridis et al., 2005) as well as online methods such as the Online Passive-Aggressive Algorithm (OPA) (Crammer et al., 2006). Although the batch methods are formulated very elegantly, they do not seem to scale well to the large training sets prevalent in NLP contexts – we briefly considered using *SVM*^{struct} but training was too time-consuming. The online methods on the other hand, although less theoretically appealing, can handle realistically sized data sets and have successfully been applied in dependency parsing (McDonald et al., 2006). Because of this, we used the OPA algorithm throughout this work.

3.2 Implementation

In the online learning framework, the weight vector is constructed incrementally. At each step, it computes an update to the weight vector based on the current example. The resulting weight vector is frequently overfit to the last examples. One way to reduce overfitting is to use the average of all successive weight vectors as the result of the training (Freund and Schapire, 1999).

Algorithm 1 shows the algorithm. It uses an "aggressiveness" parameter C to reduce overfitting, analogous to the C parameter in SVMs. The algorithm also needs a cost function ρ, which describes how much a parse tree deviates from the gold standard. In this work, we defined ρ as the sum of link costs, where the link cost was 0 for a correct dependency link with a correct label, 0.5 for a correct link with an incorrect label, and 1 for an incorrect link. The number of iterations was 5 for all languages.

For a sentence $\boldsymbol{x}$ and a parse tree y, we defined the feature representation by finding the sequence $\langle\langle S_1, I_1\rangle, a_1\rangle, \langle\langle S_2, I_2\rangle, a_2\rangle \ldots$ of states and their corresponding actions, and creating a feature vector for each state/action pair. The discriminant function was thus written

$$\boldsymbol{\Psi}(\boldsymbol{x}, y) \cdot \boldsymbol{w} = \sum_i \psi(\langle S_i, I_i\rangle, a_i) \cdot \boldsymbol{w}$$

where ψ is a feature function that assigns a feature

Algorithm 1 The Online PA Algorithm

input Training set $\mathcal{T} = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^{T}$
    Number of iterations N
    Regularization parameter C
    Cost function ρ
Initialize w to zeros
repeat N times
  for $(\boldsymbol{x}_t, y_t)$ in $\mathcal{T}$
    let $\tilde{y}_t = \arg\max_y F(\boldsymbol{x}_t, y) + \sqrt{\rho(y_t, y)}$
    let $\tau_t = \min\left(C, \frac{F(\boldsymbol{x}_t,\tilde{y}_t)-F(\boldsymbol{x}_t,y_t)+\sqrt{\rho(y_t,\tilde{y}_t)}}{\|\boldsymbol{\Psi}(\boldsymbol{x},y_t)-\boldsymbol{\Psi}(\boldsymbol{x},\tilde{y}_t)\|^2}\right)$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \tau_t(\boldsymbol{\Psi}(\boldsymbol{x}, y_t) - \boldsymbol{\Psi}(\boldsymbol{x}, \tilde{y}_t))$
return $\boldsymbol{w}_{\text{average}}$

vector to a state $\langle S_i, I_i\rangle$ and the action a_i taken in that state. Table 2 shows the feature sets used in ψ for all languages. In principle, a kernel could also be used, but that would degrade performance severely. Instead, we formed a new vector by combining features pairwisely – this is equivalent to using a quadratic kernel.

Since the history-based feature set used in the parsing algorithm makes it impossible to use independence to factorize the scoring function, an exact search to find the best-scoring action sequence ($\arg\max_y$ in Algorithm 1) is not possible. However, the beam search allows us to find a reasonable approximation.

4 Results

Table 3 shows the results of our system in the Multilingual task.

4.1 Compared to SVM-based Local Classifiers

We compared the performance of the parser with a parser based on local SVM classifiers (Johansson and Nugues, 2006). Table 4 shows the performance of both parsers on the Basque test set. We see that what is gained by using a global method such as OPA is lost by sacrificing the excellent classification performance of the SVM. Possibly, better performance could be achieved by using a large-margin batch method such as *SVM^{struct}*.

Table 2: Feature sets.

	ar	ca	cs	el	en	eu	hu	it	tr	zh
Fine POS top	•	•	•	•	•	•	•	•	•	•
Fine POS top-1	•	•	•	•		•	•	•		
Fine POS list	•	•	•	•	•	•	•	•	•	•
Fine POS list-1	•	•	•	•	•	•	•	•	•	•
Fine POS list+1	•	•	•	•	•	•	•	•	•	•
Fine POS list+2	•	•	•	•	•	•	•	•	•	•
Fine POS list+3		•	•			•	•	•	•	
POS top	•	•	•	•	•	•	•	•	•	•
POS top-1								•		
POS list	•	•	•	•	•	•	•	•	•	•
POS list-1		•		•	•	•			•	•
POS list+1	•	•	•	•	•	•	•	•	•	•
POS list+2		•	•	•		•	•	•	•	•
POS list+3	•	•		•	•	•	•	•	•	
Features top	•	•	•	•		•	•	•	•	
Features list	•	•	•	•		•	•	•	•	
Features list-1			•	•		•		•	•	
Features list+1	•	•	•	•		•		•	•	
Features list+2	•	•		•			•	•		
Word top	•	•	•	•	•		•	•	•	•
Word top-1		•			•					
Word list	•	•	•	•	•	•	•	•	•	•
Word list-1		•	•	•	•				•	•
Word list+1	•	•				•		•		
Lemma top	•	•	•	•		•		•		
Lemma list	•	•				•		•	•	
Lemma list-1						•	•			
Relation top	•	•								
Relation top left	•	•		•				•	•	
Relation top right	•	•	•	•				•		
Relation list right						•				
Word top left	•									
Word top right	•									
Word list left						•				
POS top left				•				•		
POS top right	•			•	•					
POS list left	•	•		•	•	•	•	•	•	
Features top right						•				
Features first left						•		•		

Table 3: Summary of results.

Languages	Unlabeled	Labeled
Arabic	80.91	71.76
Basque	80.41	75.08
Catalan	88.34	83.33
Chinese	81.30	76.30
Czech	77.39	70.98
English	81.43	80.29
Greek	79.58	72.77
Hungarian	75.53	71.31
Italian	81.55	77.55
Turkish	84.80	78.46
Average result	81.12	75.78

Table 4: Accuracy by learning method.

Learning Method	Accuracy
OPA	75.08
SVM	75.53

4.2 Beam Width

To investigate the influence of the beam width on the performance, we measured the accuracy of a left-to-right parser on a development set for Basque (15% of the training data) as a function of the width. Table 5 shows the result. We see clearly that widening the beam considerably improves the figures, especially in the lower ranges.

Table 5: Accuracy by beam width.

Width	Accuracy
2	72.01
4	74.18
6	75.05
8	75.30
12	75.49

4.3 Direction

We also investigated the contribution of the bidirectional parsing. Table 6 shows the result of this experiment on the Basque development set (the same 15% as in 4.2). The beam width was 2 in this experiment.

Table 6: Accuracy by parsing direction.

Direction	Accuracy
Left to right	72.01
Right to left	71.02
Bidirectional	74.48

Time did not allow a full-scale experiment, but for all languages except Catalan and Hungarian, the bidirectional parsing method outperformed the unidirectional methods when trained on a 20,000-word subset. However, the gain of using bidirectional parsing may be more obvious when the treebank is small. For all languages except Czech, left-to-right outperformed right-to-left parsing.

5 Discussion

The paper describes an incremental parser that we trained to minimize the cost over sentences, rather than over parsing actions as is usually done. It was trained using the Online Passive-Aggressive method, a cost-sensitive online margin-based learning method, and shows reasonable performance and received above-average scores for most languages.

The performance of the parser (relative the other teams) was best for Basque and Turkish, which were two of the smallest treebanks. Since we found that the optimal number of iterations was 5 for Basque (the smallest treebank), we used this number for all languages since we did not have time to investigate this parameter for the other languages. This may have had a detrimental effect for some languages. We think that some of the figures might be squeezed slightly higher by optimizing learning parameters and feature sets.

This work shows that it was possible to combine approaches used by Nivre's and McDonald's parsers in a single system. While the parser is outperformed by a system based on local classifiers, we still hope that the parsing and training combination described here opens new ways in parser design and eventually leads to the improvement of parsing performance.

Acknowledgements

This work was made possible because of the annotated corpora that were kindly provided to us: (Hajič et al., 2004; Aduriz et al., 2003; Martí et al., 2007; Chen et al., 2003; Böhmová et al., 2003; Marcus et al., 1993; Johansson and Nugues, 2007; Prokopidis et al., 2005; Csendes et al., 2005; Montemagni et al., 2003; Oflazer et al., 2003)

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.

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.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL-X*.

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.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Schwartz, and Y. Singer. 2006. Online passive-aggressive algorithms. *JMLR*, 2006(7):551–585.

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

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of ICCL*.

Y. Freund and R. E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

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. 2006. Investigating multilingual dependency parsing. In *CoNLL-X*.

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/.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency parsing with a two-stage discriminative parser. In *CoNLL-X*.

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 and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of ACL-05*.

J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *CoNLL-X*.

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.

K. Sagae and A. Lavie. 2006. Parser combination by reparsing. In *Proceedings of the HLT-NAACL*.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484.