Online learning for Deterministic Dependency Parsing

Prashanth Reddy Mannem
Language Technologies Research Center
IIIT-Hyderabad, India
prashanth@research.iiit.ac.in

Abstract

Deterministic parsing has emerged as an effective alternative for complex parsing algorithms which search the entire search space to get the best probable parse tree. In this paper, we present an online large margin based training framework for deterministic parsing using Nivre's Shift-Reduce parsing algorithm. Online training facilitates the use of high dimensional features without creating memory bottlenecks unlike the popular SVMs. We participated in the CoNLL Shared Task-2007 and evaluated our system for ten languages. We got an average multilingual labeled attachment score of 74.54 % (with 65.50% being the average and 80.32% the highest) and an average multilingual unlabeled attachment score of 80.30% (with 71.13% being the average and 86.55% the highest).

1 Introduction

CoNLL-X had a shared task on multilingual dependency parsing (Buchholz et al., 2006) by providing treebanks for 13 languages in the same dependency format. A look at the performance sheet in the contest shows that two systems with quite different approaches (one using deterministic parsing with SVM and the other using MIRA with nondeterministic and dynamic programming based MST approach) performed with good results (McDonald et al., 2006; Nivre et al., 2006).

More recently, deterministic parsing has generated a lot of interest because of their simplicity (Nivre, 2003). One of the main advantages of deterministic parsing lies in the ability to use the subtree information in the features to decide the next step. Parsing algorithms which search the entire space (Eisner, 1996; McDonald, 2006) are restricted in the features they use to score a relation. They rely only on the context information and not the history information to score a relation. Using history information makes the search intractable. Whereas, since deterministic parsers are at worst $O(n^2)$ (Yamada and Matsumoto, 2003) (Nivre (2003) is only $O(2n)$ in the worst case), they can use the crucial history information to make parsing decisions. So, in our work Nivre's parsing algorithm has been used to arrive at the dependency parse tree.

Popular learning algorithms for deterministic parsing like Support Vector Machines (SVM) run into memory issues for large data since they are batch learning algorithms. Though more information is available in deterministic parsing in terms of subtree information, high dimensional features can't be used due to the large training times for SVMs. This is where online methods come into the picture.

Unlike batch algorithms, online algorithms consider only one training instance at a time when optimizing parameters. This restriction to single-instance optimization might be seen as a weakness, since the algorithm uses less information about the objective function and constraints than batch algorithms. However, McDonald (2006) argues that this potential weakness is balanced by the simplicity of online learning, which allows for more streamlined training methods. This work focuses purely on online learning for deterministic parsing.

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1139–1143,
Prague, June 2007.

In the remaining part of the paper, we introduce Nivre's parsing algorithm, propose a framework for online learning for deterministic parsing and present the results for all the languages with various feature models.

2 Parsing Algorithm

We used Nivre's top-down/bottom-up linear time parsing algorithm proposed in Nivre (2003). A parser configuration is represented by triples (S, I, E) where *S* is the stack (represented as a list), *I* is the list of (remaining) tokens and *E* is the set of edges for the dependency graph *D*. *S* is a list of partially processed tokens, whose subtrees are incomplete i.e tokens whose parent or children have not yet been established. *top* is the top of the stack S, $next$ is the next token in the list I.

Nivre's algorithm consists of four elementary actions $Shift$, $Left$, $Right$ and $Reduce$ to build the dependency tree from the initial configuration $(\mathbf{nil}, \mathbf{W}, \emptyset)$, where *W* is the input sentence. $Shift$ pushes $next$ onto the stack S. $Reduce$ pops the stack. $Right$ adds an arc from top to $next$ and pushes $next$ onto the stack S. $Left$ adds an arc from $next$ to top and pops the stack. The parser terminates when it reaches a configuration $(S, \mathbf{nil}, \mathbf{E})$ (for any list *S* and set of edges *E*).

The labels for each relation are determined after a new arc is formed (by $left$ and $right$ actions). The parser always constructs a dependency graph that is acyclic and projective. For non-projective parsing, we followed the pseudo projective parsing approach proposed by Nivre and Nilson (2005). In this approach, the training data is projectivized by a minimal transformation, lifting non-projective arcs one step at a time, and extending the arc label of the lifted arcs using the encoding scheme called HEAD+PATH. The non-projective arcs can be recovered by applying an inverse transformation to the output of the parser, using a left-to-right, top-down, breadth-first search, guided by the extended arc labels. This method has been used for all the languages.

3 Online Learning

McDonald (2005) applied online learning by scoring edges in a connected graph and finding the Maximum Spanning Tree (MST) of the graph. McDonald et al. (2005) used *Edge Based Factorization* , where the score of a dependency tree is factored as the sum of scores of all edges in the tree. Let, $\mathbf{x} = x_1 \cdots x_n$ represents a generic input sentence , and **y** represents a generic dependency tree for sentence **x**. $(i, j) \in \mathbf{y}$ denotes the presence of a dependency relation in **y** from word x_i (parent) to word x_j (child).

In Nivre's parsing algorithm the dependency graph can be viewed as a graph resulting from a set of parsing decisions (in this case *Shift, Reduce , Left & Right*) made, starting with the initial configuration $(\mathbf{nil}, \mathbf{W}, \emptyset)$. We define this sequence of parsing decisions as $\mathbf{d} = d_1 \cdots d_m$. So, **d** is the sequence of parsing decisions made by the parser to obtain a dependency tree **y**, from an input sentence **x**. Lets also define $\mathbf{c} = c_1 \cdots c_m$ to be the configuration sequence starting from initial configuration $(\mathbf{nil}, \mathbf{W}, \emptyset)$ to the final configuration $(S, \mathbf{nil}, \mathbf{E})$.

We define the score of a parsing decision for a particular configuration to be the dot product between a high dimensional feature vector (based on both the decision and the configuration) and a weight vector. So,

$$s(d_i, c_i) = \mathbf{w} \,.\, \mathbf{f}(d_i, c_i)$$

where c_i is the configuration at the i^{th} instance and d_i is any one of the four actions {Shift, Reduce, Left, Right} .

The Margin Infused Relaxed Algorithm (MIRA) proposed by Crammer et al. (2003) attempts to keep the norm of the change to the parameter vector as small as possible, subject to correctly classifying the instance under consideration with a margin at least as large as the loss of the incorrect classifications. McDonald et al. (2005) defines the loss of a dependency tree inferred by finding the Maximum Spanning Tree(MST), as the number of words that have incorrect parent (i.e the no. of edges that have gone wrong). This satisfies the global constraint that the correct set of edges will have the highest weight. However, in Nivre's algorithm, as there is no one to one correspondence between parsing decisions and the graph edges, the number of errors in the edges can't be used as a loss function as it won't reflect the exact loss in the parsing decisions. In this method of calculating the loss function based on edges, we first get the series of decisions through inference on the training data, then concat their feature vectors and finally run the normal updates with the edge based loss (since the resulting decisions will produce a parse tree). This method gave very poor results.

So we do a factored MIRA for Nivre's algorithm by factoring the output by decisions to obtain the following constraints:

$$min\|\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}\|$$
$$s.t s(d_i, c_i) - s(d'_i, c_i) \geq 1$$
$$\forall \mathbf{c}_i \in dt(\mathbf{c}) \textit{ and}$$
$$(d_i, d'_i) \in (Shift, Left, Right, Reduce)$$

where d_i represents the correct decision and d'_i represents all the other decisions for the same configuration c_i. This states that the weight of the correct decision for a particular configuration and the weight of all other decisions must be separated by a margin of 1. For every sentence in the training data, starting with the initial configuration $(\mathbf{nil}, \mathbf{W}, \emptyset)$, weights are adjusted to satisfy the above constraints before proceeding to the next correct configuration. This process is repeated till we reach the final configuration $(\mathbf{S}, \mathbf{nil}, \mathbf{E})$.

4 Features

The two central elements in any configuration are the token on the top of the stack (*t*) and the next input token (*n*),the tokens which may be connected by a dependency relation in the current configuration. We categorize our features into $basic$, $context$, $history$ and $in-between$ feature sets. The *basic* feature set contains information about these two tokens t and n. This includes unigram, bigram combinations of the word forms (FORM), root word (LEMMA), features (FEATS) and the part-of-speech tags (both CPOS and POS) of these words. The coarse POS tag (CPOS) is useful and helps solve data sparseness to some extent.

The existence or non-existence of a relation between two words is heavily dependent on the words surrounding t and n which is the contextual information. The *context* feature set has the information about the surrounding words $t-1, t+1, n-1, n+1, n+2, n+3$. Unigram and trigram combinations (with t and n) of the lexical items, POS tags, CPOS tags of these words are part of this context feature set. We also included the second topmost element in the stack ($st-1$) word too.

The third feature set, which is the *history* feature set contains the info about the subtree at a particular parser state. One of the advantages of using deterministic parsing algorithm over nondeterministic algorithm is that history can be used as features. History features have information about the *Parent* (par), *Left Sibling* (ls) and *Right Sibling* (rs) of t. Unigram and trigram combinations (with t and n) of POS, CPOS, DEPREL tags of these words are included in the $History$ Features.

The features in the $in-between$ feature set take the form of POS and CPOS trigrams: the POS/CPOS of t, that of the word in between and that of n.

All the features in these feature sets are conjoined with distance between t & n and the parsing decision. We experimented with a combination of these feature sets in our training. We define feature models ϕ_1, ϕ_2 and ϕ_3 for our experiments. ϕ_1 is a combination on $basic$ and $context$ feature sets. ϕ_2 is a mixture of $basic$, $context$ and $in-between$ feature sets whereas ϕ_3 contains $basic$, $context$ and $history$ feature sets. The feature models ϕ_{1-3} are the same for all the languages.

5 Results and Discussion

The system with online learning and Nivre's parsing algorithm was trained on the data released by CoNLL Shared Task Organizers for all the ten languages (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). We evaluated our system using the standard evaluation script provided by the organizers (Nivre et al., 2007). The evaluation metrics are Unlabeled Attachment Score(UAS) and Labeled Attachment Score(LAS).

The results of our system with various feature models are listed in Table 1[1]. The history information in ϕ_3 contributed to a marginal improvement in accuracy of Hungarian, Italian and Turkish. Whereas, Arabic, Catalan, Czech, English, Greek

[1]Results aren't available for the models with a '-' mark.

Language	ϕ_1		ϕ_2		ϕ_3	
	LAS	UAS	LAS	UAS	LAS	UAS
$Arabic$	71.55	81.56	**72.05**	**81.93**	71.66	81.30
$Basque$	**66.35**	**73.71**	65.64	72.86	64.56	71.69
$Catalan$	84.45	89.65	**84.47**	**89.81**	-	-
$Chinese$	**74.06**	**79.09**	73.76	78.84	72.93	77.52
$Czech$	70.49	77.05	**70.68**	**77.20**	-	-
$English$	81.19	82.41	**81.55**	**82.81**	-	-
$Greek$	71.52	78.77	**71.69**	**78.89**	71.46	78.48
$Hungarian$	70.42	75.01	70.94	75.39	**71.05**	**75.54**
$Italian$	78.30	82.54	78.67	82.91	**79.18**	**83.38**
$Turkish$	76.42	82.74	76.48	82.85	**77.29**	**83.58**

Table 1: Results of Online learning with Nivre's parsing algorithm for feature models ϕ_1, ϕ_2, ϕ_3

got their highest accuracies with feature model ϕ_2 containing $basic$, $context$ and $in-between$ feature sets. The rest of the languages, Basque and Chinese achieved highest accuracies with ϕ_1. But, a careful look at the results table shows that there isn't any significant difference in the accuracies of the system across different feature models. This is true for all the languages. The feature models ϕ_2 and ϕ_3 did not show any significant difference in accuracies even though they contain more information. Feature model ϕ_1 with $basic$ and $context$ feature sets has achieved good accuracies.

5.1 K-Best Deterministic Parsing

The deterministic parsing algorithm does not handle ambiguity. By choosing a single parser action at each opportunity, the input string is parsed deterministically and a single dependency tree is built during the parsing process from beginning to end (no other trees are even considered). A simple extension to this idea is to eliminate determinism by allowing the parser to choose several actions at each opportunity, creating different action sequences that lead to different parse trees. Since a score is assigned to every parser action, the score of a parse tree can be computed simply as the average of the scores of all actions that resulted in that parse tree (the derivation of the tree). We performed a beam search by carrying out a K-best search through the set of possible sequences of actions as proposed by Johansson and Nugues (2006). However, this did not increase the accuracy. Moreover, with larger values of K, there was a decrease in the parsing accuracy. The best-first search proposed by Sagae and Lavie (2006) was also tried out but there was similar drop in accuracy.

6 Conclusion

The evaluation shows that the labeled pseudo projective deterministic parsing with online learning gives competitive parsing accuracy for most of the languages involved in the shared task. The level of accuracy varies considerably between the languages. Analyzing the results and the effects of various features with online learning will be an important research goal in the future.

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, E. Marsi, A. Dubey, and Y. Krymolowski. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

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, S. Shalev-Shwartz, and Y. Singer. 2003. Online passive aggressive algorithms. In *Proceedings of Neural Information Processing Systems (NIPS)*.

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 the 16th International Conference on Computational Linguistics (COLING)*, pages 340–345.

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 *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 206–210.

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. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

R. McDonald. 2006. *Discriminative learning and spanning tree algorithms for dependency parsing*. Ph.D. thesis, University of Pennsylvania.

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 the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99–106.

J. Nivre, J. Hall, J. Nilson, G. Eryigit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

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 Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

J. Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of International Workshop on Parsing Technologies*, pages 149–160.

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. A best-first probabilistic shift-reduce parser. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT-2003*.