Easy-First POS Tagging and Dependency Parsing with Beam Search

Ji Ma[†] **JingboZhu**[†] **Tong Xiao**[†] **Nan Yang**[‡]
[†]Natrual Language Processing Lab., Northeastern University, Shenyang, China
[‡]MOE-MS Key Lab of MCC, University of Science and Technology of China, Hefei, China
majineu@outlook.com
{zhujingbo, xiaotong}@mail.neu.edu.cn
nyang.ustc@gmail.com

Abstract

In this paper, we combine easy-first dependency parsing and POS tagging algorithms with beam search and structured perceptron. We propose a simple variant of "early-update" to ensure valid update in the training process. The proposed solution can also be applied to combine beam search and structured perceptron with other systems that exhibit spurious ambiguity. On CTB, we achieve 94.01% tagging accuracy and 86.33% unlabeled attachment score with a relatively small beam width. On PTB, we also achieve state-of-the-art performance.

1 Introduction

The easy-first dependency parsing algorithm (Goldberg and Elhadad, 2010) is attractive due to its good accuracy, fast speed and simplicity. The easy-first parser has been applied to many applications (Seeker et al., 2012; Søggard and Wulff, 2012). By processing the input tokens in an easy-to-hard order, the algorithm could make use of structured information on *both sides* of the hard token thus making more indicative predictions. However, rich structured information also causes exhaustive inference intractable. As an alternative, greedy search which only explores a tiny fraction of the search space is adopted (Goldberg and Elhadad, 2010).

To enlarge the search space, a natural extension to greedy search is beam search. Recent work also shows that beam search together with perceptron-based global learning (Collins, 2002) enable the use of non-local features that are helpful to improve parsing performance without overfitting (Zhang and Nivre, 2012). Due to these advantages, beam search and global learning has been applied to many NLP tasks (Collins and Roark 2004; Zhang and Clark, 2007). However, to the best of our knowledge, no work in the literature has ever applied the two techniques to easy-first dependency parsing.

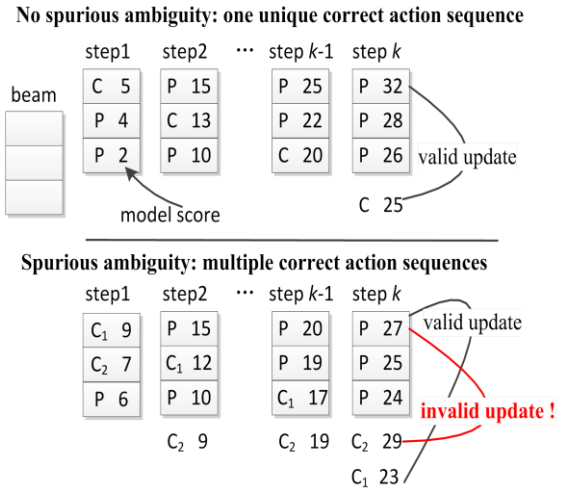

Figure 1: Example of cases without/with spurious ambiguity. The 3 × 1 table denotes a beam. "C/P" denotes correct/predicted action sequence. The numbers following C/P are model scores.

While applying beam-search is relatively straightforward, the main difficulty comes from combining easy-first dependency parsing with perceptron-based global learning. In particular, one needs to guarantee that each parameter update is *valid*, i.e., the correct action sequence has lower model score than the predicted one[1]. The difficulty in ensuring validity of parameter update for the easy-first algorithm is caused by its spurious ambiguity, i.e., the same result might be derived by more than one action sequences.

For algorithms which do not exhibit spurious ambiguity, "*early update*" (Collins and Roark 2004) is always valid: at the *k*-th step when the *single* correct action sequence falls off the beam,

[1] As shown by (Huang et al., 2012), only valid update guarantees the convergence of any perceptron-based training. Invalid update may lead to bad learning or even make the learning not converge at all.

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 110–114, Sofia, Bulgaria, August 4-9 2013.

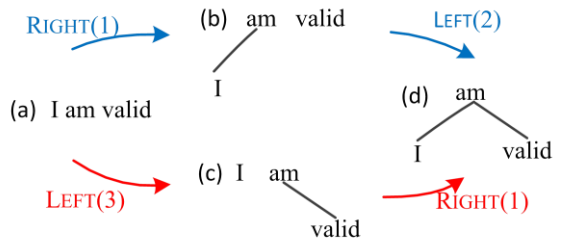

Figure 2: An example of parsing "I am valid". Spurious ambiguity: (d) can be derived by both [RIGHT(1), LEFT(2)] and [LEFT(3), RIGHT(1)].

its model score must be lower than those still in the beam (as illustrated in figure 1, also see the proof in (Huang et al., 2012)). While for easy-first dependency parsing, there could be multiple action sequences that yield the gold result (C_1 and C_2 in figure 1). When all correct sequences fall off the beam, some may indeed have higher model score than those still in the beam (C_2 in figure 1), causing invalid update.

For the purpose of valid update, we present a simple solution which is based on early update. The basic idea is to use one of the correct action sequences that were pruned right at the *k*-th step (C_1 in figure 1) for parameter update.

The proposed solution is general and can also be applied to other algorithms that exhibit spurious ambiguity, such as easy-first POS tagging (Ma et al., 2012) and transition-based dependency parsing with dynamic oracle (Goldberg and Nivre, 2012). In this paper, we report experimental results on both easy-first dependency parsing and POS tagging (Ma et al., 2012). We show that both easy-first POS tagging and dependency parsing can be improved significantly from beam search and global learning. Specifically, on CTB we achieve 94.01% tagging accuracy which is the best result to date[2] for a single tagging model. With a relatively small beam, we achieve 86.33% unlabeled score (assume gold tags), better than state-of-the-art transition-based parsers (Huang and Sagae, 2010; Zhang and Nivre, 2011). On PTB, we also achieve good results that are comparable to the state-of-the-art.

2 Easy-first dependency parsing

The easy-first dependency parsing algorithm (Goldberg and Elhadad, 2010) builds a dependency tree by performing two types of actions LEFT(*i*) and RIGHT(*i*) to a list of sub-tree structures $p_1,\ldots, p_r$. p_i is initialized with the *i*-th word of the input sentence. Action LEFT(*i*)/RIGHT(*i*) attaches p_i to its left/right neighbor and then removes p_i from the sub-tree list. The algorithm proceeds until only one sub-tree left which is the dependency tree of the input sentence (see the example in figure 2). Each step, the algorithm chooses the highest score action to perform according to the linear model:

$$\text{Score}(x) = \boldsymbol{w} \cdot \boldsymbol{\varphi}(x)$$

Here, $\boldsymbol{w}$ is the weight vector and $\boldsymbol{\varphi}$ is the feature representation. In particular, $\boldsymbol{\varphi}(\text{LEFT}(i)/\text{RIGHT}(i))$ denotes features extracted from p_i.

The parsing algorithm is greedy which explores a tiny fraction of the search space. Once an incorrect action is selected, it can never yield the correct dependency tree. To enlarge the search space, we introduce the beam-search extension in the next section.

Algorithm 1: Easy-first with beam search

Input: sentence x of n words, beam width s
Output: one best dependency tree

$\text{BEST}_s(x, ß, \boldsymbol{w}) \triangleq \text{argtop}^s_{y' \in \cup_{y \in ß} \text{EXTEN}(y)} \boldsymbol{w} \cdot \boldsymbol{\varphi}(y')$
// top s extensions from the beam
1 $ß_0 \leftarrow []$ // initially, empty beam
2 **for** $k \in 1 \ldots n-1$ **do**
3 $ß_k \leftarrow \text{BEST}_s(x, ß_{k-1}, \boldsymbol{w})$
4 **return** $ß_{n-1}[0](x)$ // tree built by the best sequence

3 Easy-first with beam search

In this section, we introduce easy-first with beam search in our own notations that will be used throughout the rest of this paper.

For a sentence *x* of *n* words, let *y* be the action (sub-)sequence that can be applied, in sequence, to *x* and the result sub-tree list is denoted by $y(x)$. For example, suppose *x* is "I am valid" and *y* is [RIGHT(1)], then *y*(*x*) yields figure 2(b). Let A_l to be LEFT(*i*)/RIGHT(*i*) actions where $i \in [1, l]$. Thus, the set of all possible one-action extension of *y* is:

$$\text{EXTEN}(y) \triangleq \{y \circ a | a \in A_{|y(x)|}\}$$

Here, '$\circ$' means insert a to the end of y. Following (Huang et al., 2012), in order to formalize beam search, we also use the $\text{argtop}^s_{y \in \boldsymbol{Y}} \boldsymbol{w} \cdot \boldsymbol{\varphi}(y)$ operation which returns the top *s* action sequences in $\boldsymbol{Y}$ according to $\boldsymbol{w} \cdot \boldsymbol{\varphi}(y)$. Here, $\boldsymbol{Y}$ denotes a set of action sequences, $\boldsymbol{\varphi}(y)$ denotes the sum of feature vectors of each action in y

Pseudo-code of easy-first with beam search is shown in algorithm 1. Beam search grows *s* (beam width) action sequences in parallel using a

[2] Joint tagging-parsing models achieve higher accuracy, but those models are not directly comparable to ours.

Algorithm 2: Perceptron-based training over one training sample (x, t)

Input: (x, t), s, parameter $\boldsymbol{w}$
Output: new parameter $\boldsymbol{w}$

$\text{TOPC}(x, ß, \boldsymbol{w}, \boldsymbol{C}) \triangleq \text{argmax}_{y' \in \boldsymbol{C} \cap (\cup_{y \in ß} \text{EXTEN}(y))} \boldsymbol{w} \cdot \boldsymbol{\varphi}(y')$
// top *correct* extension from the beam

1 $ß_0 \leftarrow []$
2 **for** $k \in 1 \ldots n-1$ **do**
3 $\quad \hat{y} = \text{TOPC}(x, ß_{k-1}, \boldsymbol{w}, \boldsymbol{C})$
4 $\quad ß_k \leftarrow \text{BEST}_s(x, ß_{k-1}, \boldsymbol{w})$
5 $\quad$ **if** $ß_k \cap \boldsymbol{C} = \emptyset$ // all correct seq. falls off the beam
6 $\quad\quad \boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{\varphi}(\hat{y}) - \boldsymbol{\varphi}(ß_k[0])$
7 $\quad\quad$ **break**
8 **if** $ß_{n-1}[0](x) \neq t$ // full update
9 $\quad \boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{\varphi}(\hat{y}) - \boldsymbol{\varphi}(ß_{n-1}[0])$
10 **return** $\boldsymbol{w}$

beam ß, (sequences in ß are sorted in terms of model score, i.e., $\boldsymbol{w} \cdot \boldsymbol{\varphi}(ß[0]) > \boldsymbol{w} \cdot \boldsymbol{\varphi}(ß[1])$ ...). At each step, the sequences in ß are expanded in all possible ways and then ß is filled up with the top s newly expanded sequences (line 2 ~ line 3). Finally, it returns the dependency tree built by the top action sequence in $ß_{n-1}$.

4 Training

To learn the weight vector $\boldsymbol{w}$, we use the perceptron-based global learning[3] (Collins, 2002) which updates $\boldsymbol{w}$ by rewarding the feature weights fired in the correct action sequence and punish those fired in the predicted incorrect action sequence. Current work (Huang et al., 2012) rigorously explained that only valid update ensures convergence of any perceptron variants. They also justified that the popular "early update" (Collins and Roark, 2004) is valid for the systems that do not exhibit spurious ambiguity[4].

However, for the easy-first algorithm or more generally, systems that exhibit spurious ambiguity, even "early update" could fail to ensure validity of update (see the example in figure 1). For validity of update, we propose a simple solution which is based on "early update" and which can accommodate spurious ambiguity. The basic idea is to use the correct action sequence which was pruned right at the step when all correct sequence falls off the beam (as C_1 in figure 1).

Features of (Goldberg and Elhadad, 2010)	
for p in p_{i-1}, p_i, p_{i+1}	w_p-vl_p, w_p-vr_p, t_p-vl_p, t_p-vr_p, tlc_p, trc_p, wlc_p, wlc_p
for p in p_{i-2}, p_{i-1}, p_i, p_{i+1}, p_{i+2}	t_p-tlc_p, t_p-trc_p, t_p.tlc_p-trc_p
for p, q, r in (p_{i-2}, p_{i-1}, p_i), (p_{i-1}, p_{i+1}, p_i), (p_{i+1}, p_{i+2} ,p_i)	t_p-t_q-t_r, t_p-t_q-w_r
for p, q in (p_{i-1}, p_i)	t_p-tlc_p-t_q, t_p-trc_p-t_q, ,t_p-tlc_p-w_q,, t_p-trc_p-w_q, t_p-w_q-tlc_q, t_p-w_q-trc_q

Table 1: Feature templates for English dependency parsing. w_p denotes the head word of p, t_p denotes the POS tag of w_p. vl_p/vr_p denotes the number p's of left/right child. lc_p/rc_p denotes p's leftmost/rightmost child. p_i denotes partial tree being considered.

Algorithm 2 shows the pseudo-code of the training procedure over one training sample (x, t), a sentence-tree pair. Here we assume $\boldsymbol{C}$ to be the set of all correct action sequences/subsequences. At step k, the algorithm constructs a correct action sequence $\hat{y}$ of length k by extending those in $ß_{k-1}$ (line 3). It also checks whether $ß_k$ no longer contains any correct sequence. If so, $\hat{y}$ together with $ß_k[0]$ are used for parameter update (line 5 ~ line 6). It can be easily verified that each update in line 6 is valid. Note that both 'TOPC' and the operation in line 5 use $\boldsymbol{C}$ to check whether an action sequence y is correct or not. This can be efficiently implemented (without explicitly enumerating $\boldsymbol{C}$) by checking if each LEFT(i)/RIGHT(i) in y are compatible with (x, t): p_i already collected all its dependents according to t; p_i is attached to the correct neighbor suggested by t.

5 Experiments

For English, we use PTB as our data set. We use the standard split for dependency parsing and the split used by (Ratnaparkhi, 1996) for POS tagging. Penn2Malt[5] is used to convert the bracketed structure into dependencies. For dependency parsing, POS tags of the training set are generated using 10-fold jack-knifing.

For Chinese, we use CTB 5.1 and the split suggested by (Duan et al., 2007) for both tagging and dependency parsing. We also use Penn2Malt and the head-finding rules of (Zhang and Clark 2008) to convert constituency trees into dependencies. For dependency parsing, we assume gold segmentation and POS tags for the input.

[3] Following (Zhang and Nivre, 2012), we say the training algorithm is global if it optimizes the score of an entire action sequence. A local learner trains a classifier which distinguishes between single actions.

[4] As shown in (Goldberg and Nivre 2012), most transition-based dependency parsers (Nivre et al., 2003; Huang and Sagae 2010;Zhang and Clark 2008) ignores spurious ambiguity by using a static oracle which maps a dependency tree to a single action sequence.

[5] http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html

Features used in English dependency parsing are listed in table 1. Besides the features in (Goldberg and Elhadad, 2010), we also include some trigram features and valency features which are useful for transition-based dependency parsing (Zhang and Nivre, 2011). For English POS tagging, we use the same features as in (Shen et al., 2007). For Chinese POS tagging and dependency parsing, we use the same features as (Ma et al., 2012). All of our experiments are conducted on a Core i7 (2.93GHz) machine, both the tagger and parser are implemented using C++.

5.1 Effect of beam width

Tagging/parsing performances with different beam widths on the development set are listed in table 2 and table 3. We can see that Chinese POS tagging, dependency parsing as well as English dependency parsing greatly benefit from beam search. While tagging accuracy on English only slightly improved. This may because that the accuracy of the greedy baseline tagger is already very high and it is hard to get further improvement. Table 2 and table 3 also show that the speed of both tagging and dependency parsing drops linearly with the growth of beam width.

5.2 Final results

Tagging results on the test set together with some previous results are listed in table 4. Dependency parsing results on CTB and PTB are listed in table 5 and table 6, respectively.

On CTB, tagging accuracy of our greedy baseline is already comparable to the state-of-the-art. As the beam size grows to 5, tagging accuracy increases to 94.01% which is 2.3% error reduction. This is also the best tagging accuracy comparing with previous single tagging models (For limited space, we do not list the performance of joint tagging-parsing models).

Parsing performances on both PTB and CTB are significantly improved with a relatively small beam width (s = 8). In particular, we achieve 86.33% uas on CTB which is 1.54% uas improvement over the greedy baseline parser. Moreover, the performance is better than the best transition-based parser (Zhang and Nivre, 2011) which adopts a much larger beam width (s = 64).

6 Conclusion and related work

This work directly extends (Goldberg and Elhadad, 2010) with beam search and global learning. We show that both the easy-first POS tagger and dependency parser can be significantly improved using beam search and global learning.

This work can also be considered as applying (Huang et al., 2012) to the systems that exhibit spurious ambiguity. One future direction might be to apply the training method to transition-based parsers with dynamic oracle (Goldberg and Nivre, 2012) and potentially further advance performances of state-of-the-art transition-based parsers.

s	PTB	CTB	speed
1	97.17	93.91	1350
3	97.20	94.15	560
5	97.22	94.17	385

Table 2: Tagging accuracy vs beam width vs. Speed is evaluated using the number of sentences that can be processed in one second

s	PTB		CTB		speed
	uas	compl	uas	compl	
1	91.77	45.29	84.54	33.75	221
2	92.29	46.28	85.11	34.62	124
4	92.50	46.82	85.62	37.11	71
8	92.74	48.12	86.00	35.87	39

Table 3: Parsing accuracy vs beam width. 'uas' and 'compl' denote unlabeled score and complete match rate respectively (all excluding punctuations).

PTB		CTB	
(Collins, 2002)	97.11	(Hatori et al., 2012)	93.82
(Shen et al., 2007)	97.33	(Li et al., 2012)	93.88
(Huang et al., 2012)	97.35	(Ma et al., 2012)	93.84
this work $s = 1$	97.22	this work $s = 1$	93.87
this work $s = 4$	97.28	this work $s = 5$	94.01†

Table 4: Tagging results on the test set. '†' denotes statistically significant over the greedy baseline by McNemar's test ($p < 0.05$)

Systems	s	uas	compl
(Huang and Sagae, 2010)	8	85.20	33.72
(Zhang and Nivre, 2011)	64	86.00	36.90
(Li et al., 2012)	—	86.55	—
this work	1	84.79	32.98
this work	8	86.33†	36.13

Table 5: Parsing results on CTB test set.

Systems	s	uas	compl
(Huang and Sagae, 2010)	8	92.10	—
(Zhang and Nivre, 2011)	64	92.90	48.50
(Koo and Collins, 2010)	—	93.04	—
this work	1	91.72	44.04
this work	8	92.47†	46.07

Table 6: Parsing results on PTB test set.

Shen et al., (2007) and (Shen and Joshi, 2008) also proposed bi-directional sequential classification with beam search for POS tagging and LTAG dependency parsing, respectively. The main difference is that their training method aims to learn a classifier which distinguishes between each local action while our training method aims to distinguish between action sequences. Our method can also be applied to their framework.

Acknowledgments

We would like to thank Yue Zhang, Yoav Goldberg and Zhenghua Li for discussions and suggestions on earlier drift of this paper. We would also like to thank the three anonymous reviewers for their suggestions. This work was supported in part by the National Science Foundation of China (61073140; 61272376), Specialized Research Fund for the Doctoral Program of Higher Education (20100042110031) and the Fundamental Research Funds for the Central Universities (N100204002).

References

Collins, M. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Duan, X., Zhao, J., , and Xu, B. 2007. Probabilistic models for action-based Chinese dependency parsing. In *Proceedings of ECML/ECPPKDD*.

Goldberg, Y. and Elhadad, M. 2010 An Efficient Algorithm for Eash-First Non-Directional Dependency Parsing. In *Proceedings of NAACL*

Huang, L. and Sagae, K. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL*.

Huang, L. Fayong, S. and Guo, Y. 2012. Structured Perceptron with Inexact Search. In *Proceedings of NAACL*.

Koo, T. and Collins, M. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*.

Li, Z., Zhang, M., Che, W., Liu, T. and Chen, W. 2012. A Separately Passive-Aggressive Training Algorithm for Joint POS Tagging and Dependency Parsing. In *Proceedings of COLING*

Ma, J., Xiao, T., Zhu, J. and Ren, F. 2012. Easy-First Chinese POS Tagging and Dependency Parsing. In *Proceedings of COLING*

Rataparkhi, A. (1996) A Maximum Entropy Part-Of-Speech Tagger. In *Proceedings of EMNLP*

Shen, L., Satt, G. and Joshi, A. K. (2007) Guided Learning for Bidirectional Sequence Classification. In *Proceedings of ACL*.

Shen, L. and Josh, A. K. 2008. LTAG Dependency Parsing with Bidirectional Incremental Construction. In *Proceedings of* EMNLP.

Seeker, W., Farkas, R. and Bohnet, B. 2012 Data-driven Dependency Parsing With Empty Heads. In *Proceedings of COLING*

Søggard, A. and Wulff, J. 2012. An Empirical Study of Non-lexical Extensions to Delexicalized Transfer. In *Proceedings of COLING*

Yue Zhang and Stephen Clark. 2007 Chinese Segmentation Using a Word-based Perceptron Algorithm. In *Proceedings of* ACL.

Zhang, Y. and Clark, S. 2008. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL*.

Zhang, Y. and Nivre, J. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of ACL*.

Zhang, Y. and Nivre, J. 2012. Analyzing the Effect of Global Learning and Beam-Search for Transition-Based Dependency Parsing. In *Proceedings of COLING*.