Large-Margin Training of Dependency Parsers Using Pegasos Algorithm

Changki Lee, Soojong Lim, and Myung-Gil Jang

This letter presents a modified Pegasos algorithm for graph-based dependency parsing. We show that the modified Pegasos algorithm outperforms the margin infused relaxed and passive aggressive algorithms in three languages.

Keywords: Dependency parsing, stochastic gradient descent, Pegasos.

I. Introduction

Dependency parsing has been a growing interest for applications such as relation extraction and machine translation. Dependency representations of sentences model head-dependent syntactic relations as edges in a directed graph. For example, Fig. 1 shows a dependency graph for the sentence, "The dog chased the cat."

Many dependency parsers are based on data-driven parsing models, which learn to produce dependency graphs for sentences solely from an annotated corpus, and can be easily ported to any language in which annotated resources exist. Most data-driven parsing models proposed in recent years can be described as either transition-based or graph-based [1]. In graph-based parsing, McDonald and others showed that treating dependency parsing as a search for the highest-scoring maximum spanning tree (MST) in a directed graph yields efficient algorithms for both projective and non-projective trees [2]. These models provide state-of-the-art performances across multiple languages. They extended the MST parsing framework to incorporate higher-order feature representations [3], [4]. In their works, the margin infused relaxed algorithm (MIRA) [5], an online large-margin learning algorithm inspired by the perceptron, is used to compute model parameters. Online learning algorithms provide strong theoretical guarantees and run very quickly, but their performance is typically inferior to that of batch algorithms.

An alternative approach for structured prediction is to use stochastic gradient decent (SGD) methods. SGD methods use approximate gradients estimated from subsets of the training data and update the weights of the features in an online fashion. For large-scale problems, SGD methods are claimed to be sufficiently precise while delivering the best performance versus training time trade-off [6]. Among SGD methods, the Pegasos algorithm has shown promising performance for binary classification support vector machines (SVMs) [7]. The Pegasos algorithm shares the simplicity and speed of online learning algorithms but is guaranteed to converge to an actual SVM solution.

In this letter, we extend the Pegasos algorithm for graph-based dependency parsing. We evaluate our system on three languages from the CoNLL-X shared task [8]. We show that our extended Pegasos algorithm can produce accurate models quickly.

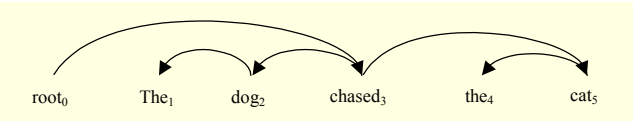

Fig. 1. Example dependency graph.

II. Graph-Based Dependency Parsing

We used the same MST unlabeled dependency parsing framework proposed in [2]. This framework leads to efficient parsing algorithms for both projective and non-projective

Manuscript received Nov. 24, 2009; revised Feb. 11, 2010; accepted Feb. 26, 2010.
Changki Lee (phone: +82 42 860 6879, email: leeck@etri.re.kr), Soojong Lim (email: lsj@etri.re.kr), and Myung-Gil Jang (email: mgjang@etri.re.kr) are with the Software Research Laboratory, ETRI, Daejeon, Rep. of Korea.
doi:10.4218/etrij.10.0209.0483

dependency trees with the Eisner and Chu-Liu-Edmonds algorithms, respectively [2], [3]. Following this approach, we define the score of an unlabeled dependency tree as

$$s(\mathbf{x},\mathbf{y}) = \sum_{(i,j)\in\mathbf{y}} s(i,j) = \sum_{(i,j)\in\mathbf{y}} \mathbf{w}^T f(i,j) = \mathbf{w}^T f(\mathbf{x},\mathbf{y}),$$

where $\mathbf{x} = x_1,\cdots, x_n$ is an input sentence, and $\mathbf{y}$ is a dependency tree for $\mathbf{x}$. We can view $\mathbf{y}$ as a set of tree edges and write $(i,j)\in\mathbf{y}$ to indicate an edge in $\mathbf{y}$ from word x_i to word x_j. Consider the example from Fig. 1. The score of this tree would be

$$s(0,3)+s(3,2)+s(2,1)+s(3,5)+s(5,4).$$

For second-order non-projective MST dependency parsing, we followed the approximate algorithm based on the exact $O(n^3)$ second-order projective Eisner algorithm proposed in [3].

III. Large-Margin Training

McDonald and others extended the MIRA to learning with dependency trees [2]. Algorithm 1 gives a pseudocode for the MIRA. The algorithm considers a single training instance at each update to $\mathbf{w}$. The final weight vector is the average of the weight vectors after each iteration.

Algorithm 1. MIRA learning algorithm.
Training data: $S = \{(\mathbf{x}_i,\mathbf{y}_i)\}_{i=1}^m$
1: $\mathbf{w}_0 = 0; \mathbf{v} = 0; t = 0$
2: For $n = 1, 2,\cdots, N$
3: For $i = 1, 2,\cdots, m$
4: $\min \|\mathbf{w}_{t+1} - \mathbf{w}_t\|$
s.t. $s(\mathbf{x}_i,\mathbf{y}_i) - s(\mathbf{x}_i,\mathbf{y}) \ge L(\mathbf{y}_i,\mathbf{y}), \forall \mathbf{y}\in\mathbf{Y}$
5: $\mathbf{v} = \mathbf{v} + \mathbf{w}_{t+1}$
6: $t = t+1$
7: $\mathbf{w}_{\text{averaged}} = \mathbf{v}/(N\times m)$
8: Output $\mathbf{w}_{t+1}$ and $\mathbf{w}_{\text{averaged}}$

We used the margin rescaling formula of structural SVMs [9] for dependency parsing as

$$\min_{w,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{m}\sum_i^m \xi_i, \text{ s.t. } \forall i, \xi_i \ge 0,$$
$$\forall i, \forall \mathbf{y}\in\mathbf{Y}\setminus\mathbf{y}_i : \mathbf{w}^T f(\mathbf{x}_i,\mathbf{y}_i) - \mathbf{w}^T f(\mathbf{x}_i,\mathbf{y}) \ge L(\mathbf{y}_i,\mathbf{y}) - \xi_i, \quad (1)$$

where $\mathbf{x}_i$ is a sentence, and $L(\mathbf{y}_i,\mathbf{y})$ is a real-valued loss for dependency tree $\mathbf{y}$ relative to the correct dependency tree $\mathbf{y}_i$. We define the loss of a dependency tree as the number of words that have an incorrect parent. We extend the Pegasos algorithm to structural SVMs for graph-based dependency parsing. We replace the objective of Pegasos with an approximate objective function of structural SVMs by setting $\lambda = 1/C$:

$$f(\mathbf{w}; A_t) = \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{k}\sum_{(\mathbf{x}_i,\mathbf{y}_i)\in A_t} l(\mathbf{w};(\mathbf{x}_i,\mathbf{y}_i)),$$
$$\text{where } l(\mathbf{w};(\mathbf{x}_i,\mathbf{y}_i)) = \max_{\mathbf{y}} \left\{ L(\mathbf{y}_i,\mathbf{y}) - \mathbf{w}^T f(\mathbf{x}_i,\mathbf{y}_i) + \mathbf{w}^T f(\mathbf{x}_i,\mathbf{y}) \right\}. \quad (2)$$

Parameter k is the number of examples used for calculating the subgradients, and A_t is the subset of training set $S = \{(\mathbf{x}_i, \mathbf{y}_i): i = 1, 2,\cdots, m\}$. The subgradient of $f(\mathbf{w}; A_t)$ is

$$\nabla f(\mathbf{w}; A_t) = \lambda\mathbf{w} - \frac{1}{|A_t|}\sum_{(\mathbf{x}_i,\mathbf{y}_i)\in A_t} \{ f(\mathbf{x}_i,\mathbf{y}_i) - f(\mathbf{x}_i,\hat{\mathbf{y}}_i) \},$$
$$\text{where } \hat{\mathbf{y}}_i = \arg\max_{\mathbf{y}} \left\{ L(\mathbf{y}_i,\mathbf{y}) - \mathbf{w}^T f(\mathbf{x}_i,\mathbf{y}_i) + \mathbf{w}^T f(\mathbf{x}_i,\mathbf{y}) \right\}. \quad (3)$$

If we choose $A_t = S$, that is, $k = m$ on each round t, then we obtain the subgradient projection method. For the case of $k = 1$, we obtain a variant of the stochastic gradient method.

A pseudocode of the modified Pegasos is depicted in algorithm 2. The algorithm receives as input two parameters: T, the number of iterations to perform; and k, the number of examples to use for calculating the subgradients. Initially, we set $\mathbf{w}_1$ to any vector whose norm is at most $1/\sqrt{\lambda}$. On iteration t, the algorithm first chooses set A_t of size k in S (line 4), finds the "most violated" dependency parse tree $\hat{\mathbf{y}}_i$ for $(\mathbf{x}_i, \mathbf{y}_i)$ in A_t (line 5), sets the learning rate (line 6), calculates $\mathbf{w}_{t+1/2}$ (line 7), and sets $\mathbf{w}_{t+1}$ to be the projection of $\mathbf{w}_{t+1/2}$ onto the set $\{\mathbf{w} : \|\mathbf{w}\| \le 1/\sqrt{\lambda}\}$ (line 8). The final weight vector can be the average of the weight vectors after each iteration (lines 9 to 10). In line 5, $\hat{\mathbf{y}}_i$ can be found using the same inference algorithms, for example the Eisner or CLE algorithm, as $\mathbf{y}^* = \arg\max_{\mathbf{y}} \mathbf{w}^T f(\mathbf{x}_i,\mathbf{y})$ because $L(\mathbf{y}_i, \mathbf{y})$ decomposes over variable subsets no larger than the subsets in the decomposition of $f(\mathbf{x}_i, \mathbf{y})$. We adapted the Eisner algorithm to solve the arg max in line 5.

Algorithm 2. Modified Pegasos algorithm for dependency parsing (Pegasos-DP).
1: Input: S, λ, T, k
2: Initialize: Choose $\mathbf{w}_1$ s.t. $\|\mathbf{w}_1\| \le 1/\sqrt{\lambda}$, $\mathbf{v} = 0$
3: For $t = 1, 2,\cdots, T$
4: Choose $A_t \subseteq S$, where $|A_t| = k$
5: $\forall (\mathbf{x}_i,\mathbf{y}_i)\in A_t : \hat{\mathbf{y}}_i = \arg\max_{\mathbf{y}} \left\{ L(\mathbf{y}_i,\mathbf{y}) + \mathbf{w}^T f(\mathbf{x}_i,\mathbf{y}) \right\}$
6: $\eta_t = 1/\lambda t$
7: $\mathbf{w}_{t+1/2} = (1-\eta_t\lambda)\mathbf{w}_t + \frac{\eta_t}{k}\sum_{(\mathbf{x}_i,\mathbf{y}_i)\in A_t} \{ f(\mathbf{x}_i,\mathbf{y}_i) - f(\mathbf{x}_i,\hat{\mathbf{y}}_i) \}$
8: $\mathbf{w}_{t+1} = \min\left\{1, \frac{1/\sqrt{\lambda}}{\|\mathbf{w}_{t+1/2}\|}\right\}\mathbf{w}_{t+1/2}$
9: $\mathbf{v} = \mathbf{v} + \mathbf{w}_{t+1}$
10: $\mathbf{w}_{\text{averaged}} = \mathbf{v}/T$
11: Output: $\mathbf{w}_{T+1}$ and $\mathbf{w}_{\text{averaged}}$

IV. Experiments

We demonstrated the effectiveness of our modified Pegasos for dependency parsing with experiments on three languages from the CoNLL-X shared task: Dutch, Swedish, and Danish [8]. All experiments are evaluated using an unlabeled attachment score (UAS), utilizing the default settings. We used the approximate second-order non-projective dependency parsing algorithm and similar features as in [3] and [4].

We implemented the modified Pegasos algorithm for the training of unlabeled dependency parsing in C++. For comparison, we also implemented the MIRA, the passive-aggressive (PA) algorithm [5], and the 1-slack structural SVM [10], [11]. In the experiments, we set k=10. Regularization constant λ from $\{10^{-2}, 3.3\times10^{-2}, 10^{-3}, 3.3\times10^{-3}, 10^{-4}\}$ was chosen by optimization on the test set for all experiments. All our experiments were conducted on an Intel Core i5 CPU PC with 2.67 GHz and 8 GB of RAM.

Table 1 shows the performances of the compared systems. M&P2006 is a graph-based dependency-parsing model using the averaged MIRA proposed in [4]. Nivre2006 is a transition model presented in [12]. N&M2008 is a hybrid model that combines transition-based and graph-based parsers [1]. M&P2006 and Nivre2006 are the two best performing systems in the CoNLL-X shared task. MST+MIRA, MST+PA, MST+Pegasos-DP, and MST+1-slack S-SVM are our dependency parsers using the MIRA, PA, Pegasos-DP, and 1-slack structural SVM, respectively. When comparing our dependency parsers, the Pegasos-DP and 1-slack S-SVM have similar performance and outperform the MIRA and the PA algorithm. Compared with other systems, Pegasos-DP outperforms M&P2006 and Nivre2006, and the performance of Pegasos-DP is similar to N&M2008, which is obtained by combining M&P2006 and Nivre2006.

Table 1. UAS for three languages.

	Dutch	Swedish	Danish
MST+MIRA (averaged) -baseline	83.9	89.2	90.2
MST+PA (averaged)	84.3 (+0.4)	89.6 (+0.4)	90.6 (+0.4)
MST+Pegasos-DP	**85.0** (+1.1)	**90.1** (+0.9)	**90.8** (+0.6)
MST+Pegasos-DP (averaged)	**85.1** (+1.2)	**90.1** (+0.9)	**91.0** (+0.8)
MST+1-slack S-SVM	**85.0** (+1.1)	**90.1** (+0.9)	**90.9** (+0.7)
M&P2006	83.6 (−0.3)	88.9 (−0.3)	90.6 (+0.4)
Nivre2006	81.4 (−2.5)	89.5 (+0.3)	89.8 (−0.4)
N&M2008	84.8 (+0.9)	90.5 (+1.3)	91.3 (+1.1)

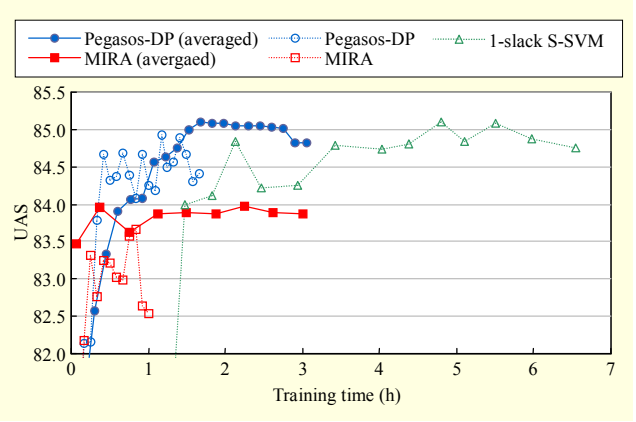

Fig. 2. Performances of Pegasos-DP, 1-slack S-SVM, and MIRA versus training time.

Figure 2 compares the performances of MST+Pegasos-DP, MST+MIRA, and MST+1-slack S-SVM in the Dutch data set. The Pegasos-DP outperforms the MIRA, but has unstable performance. Averaging helps the performance of the MIRA. In the Pegasos-DP, averaging does not significantly help the performance, but it does smooth the convergence curve. The Pegasos-DP and 1-slack S-SVM have similar performance, but the Pegasos-DP is much faster than the 1-slack S-SVM. We think that this is because the Pegasos-DP shares the speed of online learning algorithms but is guaranteed to converge to the actual structural SVM solution.

V. Conclusion

We proposed a modified Pegasos algorithm for graph-based dependency parsing. We showed that the modified Pegasos algorithm (Pegasos-DP) outperforms the MIRA and the averaged Pegasos-DP has a more stable performance than the non-averaged Pegasos-DP. We also showed that the Pegasos-DP is faster than 1-slack structural SVM.

References

[1] J. Nivre and R. McDonald, "Integrating Graph-Based and Transition-Based Dependency Parsers," *Proc. ACL-HLT*, 2008, pp. 950-958.

[2] R. McDonald and F. Pereira, "Non-Projective Dependency Parsing Using Spanning Tree Algorithms," *Proc. HLT-EMNLP*, 2005, pp. 523-530.

[3] R. McDonald and F. Pereira, "Online Learning of Approximate Dependency Parsing Algorithms," *Proc. EACL*, 2006, pp. 81-88.

[4] R. McDonald et al., "Multilingual Dependency Analysis with a Two-Stage Discriminative Parser," *Proc. CoNLL*, 2006, pp. 216-220.

[5] K. Crammer et al., "Online Passive-Aggressive Algorithms," *J.*

Machine Learning Research, vol. 7, 2006, pp. 551-585.

[6] L. Bottou and O. Bousquet, "The Tradeoffs of Large Scale Learning," *NIPS*, vol. 20, 2008, pp. 161-168.

[7] S. Shalev-Shwartz et al., "Pegasos: Primal Estimated Sub-GrAdient SOlver for SVM," *Proc. ICML*, 2007, pp. 807-814.

[8] S. Buchholz and E. Marsi, "Conll-x Shared Task on Multilingual Dependency Parsing," *Proc.CoNLL*, 2006, pp. 149-164.

[9] I. Tsochantaridis et al., "Support Vector Machine Learning for Interdependent and Structured Output Spaces," *Proc. ICML*, 2004, p. 104.

[10] C. Lee and M. Jang, "A Modified Fixed-Threshold SMO for 1-Slack Structural SVMs," *ETRI J.*, vol. 32, no. 1, Feb. 2010, pp. 120-128.

[11] C. Lee and M. Jang, "Fast Training of Structured SVM Using Fixed-Threshold Sequential Minimal Optimization," *ETRI J.*, vol. 31, no. 2, 2009, pp. 121-128.

[12] J. Nivre et al., "Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines," *Proc. CoNLL*, 2006, pp. 221-225.