A Simulated Shallow Dependency Parser Based on Weighted Hierarchical Structure Learning

Zhiming Kang, Chun Chen*, Jiajun Bu, Peng Huang, and Guang Qiu

College of Computer Science, Zhejiang University, Hangzhou, China
{kzm,chenc,bjj,huangp,qiuguang}@zju.edu.cn

Abstract. In the past years much research has been done on data-driven dependency parsing and performance has increased steadily. Dependency grammar has an important inherent characteristic, that is, the nodes closer to root usually make more contribution to audiences than the others. However, that is ignored in previous research in which every node in a dependency structure is considered to play the same role. In this paper a parser based on weighted hierarchical structure learning is proposed to simulate shallow dependency parsing, which has the preference for nodes closer to root during learning. The experimental results show that the accuracies of nodes closer to root are improved at the cost of a little decrease of accuracies of nodes far from root.

1 Introduction

Recently, dependency grammar has gained renewed attention and becomes more prominent. Currently it is dominant that using data-driven approaches to learn parsers automatically from experience, such as probabilistic generative models [3], generative probabilistic parsing models [2] and deterministic discriminative model [7] and so on. Generally speaking, data-driven approaches fall into two categories, i.e. generative models and discriminative models. The latest state-of-the-art dependency parsers are discriminative which are based on classifiers trained to score trees, given a sentence, either via factored whole structure scores [5] or local parsing decision scores [6]. However, seldom work about shallow dependency parsing like shallow phrase-structure parsing has been done. In the paper, a discriminative dependency parser based on weighted hierarchical structure learning is proposed to simulate shallow dependency parsing, aiming at improving dependency parsing for nodes closer to the root node.

The remainder of this paper is organized as follows. Section 2 first makes a brief introduction to dependency grammar, and then describe dependency parsing algorithm in detail. Section 3 gives the details of adopted learning algorithm and some discussion. To demonstrate the usefulness of our algorithm, Section 4 contains the results produced by several dependency parsers. Last section contains some conclusions plus some ideas for future work.

* Corresponding author.

H. Li et al. (Eds.): AIRS 2008, LNCS 4993, pp. 484–489, 2008.

2 Dependency Parsing

2.1 Overview of Dependency Grammar

In Dependency Grammar, individual words in a sentence are considered to be linked together in dependency relations instead of being combined just mechanically. Whenever two words are linked by a dependency relation, we say that one of them is the head and the other is the dependent, and that there is an edge connecting them. In general, the dependent is the modifier or complement; the head plays the larger role in determining the behavior of the pair. The dependent presupposes the presence of the head; the head may require the presence of the dependent. The figure 1 depicts the skeleton of dependency structure of a sentence. The dashed line means the head 'Root' and the relation $<$'Root', 'had'$>$ both are dummies. Essentially, a dependency link is a directed arc pointing from head to dependent. The dependency structure is a tree with the main verb as its root (head).

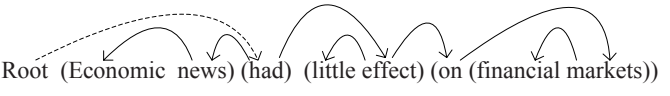

Fig. 1. An example of annotated image

Similar to shallow phrase-structure parsing, shallow dependency parsing breaks up sentence into 'spans', and then link them with directed arcs. The edges connecting different spans are named 'span-link', and the two nodes linked by 'span-link' are defined as 'span-head' with respect to corresponding span. Different from full parsing, shallow dependency parsing only focuses on 'span-head' and 'span-link', instead of nodes and edges inside spans. Currently there is no standard about what is shallow dependency parsing like shallow phrase-structure parsing, and the following gives a rough guideline: the dummy node 'Root' is the root of a dependency tree; each subtree is treated as a span in shallow parsing. Based on above analysis, it is reasonable to think that 'span-head' and 'span-link' closer to 'Root' are more important than the others in shallow parsing, such as that inside span. Thus it is feasible to improve accuracy of dependency relations closer to 'Root' to simulate shallow dependency parsing, with regular full parsing.

2.2 Parsing Algorithm

The CKY algorithm is a well-known $O(n^3)$ algorithm for PCFG parsing [4]. When applied to dependency parsing, however, the CKY has the time complexity of $O(n^5)$. Eisner proposed an parsing algorithm similar to CKY that has a time complexity of $O(n^3)$ [3]. The idea is to parse the left and right dependents of a word independently and combine them at a later stage. During dependency parsing, there are many spans produced. Among them adjacent spans are possible to be combined into a longer span iteratively. At last one span including all words can be generated as output. This parsing algorithm removes the need for the additional head indices and requires only two additional binary variables that specify the direction of the item and whether an item is

complete. For space limitation the parsing algorithm is described here briefly, and for details please refer to [3,5].

3 Learning

As indicated earlier, dependency tree is built bottom-up via combining small spans iteratively. The number of generated spans, however, grows exponentially with the size of sentence length, so the learning task is to, given a sentence, find the best one from numerous candidates. In this paper we adopt a strategy similar to McDonald et al [5], that is to say, every candidate is scored and chooses the one with highest score as final dependency parsing output.

An extension of original binary perceptron for multiclass problem (MPA) is proposed by Collins [1] as follows:

$$w = w + \alpha \cdot (\phi(x_i, y_i) - \phi(x_i, z_i)) \tag{1}$$

where z_i is a prediction for instance x_i and α is a constant positive factor for promotion or demotion, and the definition of ϕ is the same as the previous representations of feature vector. Note that the parameter α is a constant, that means weights of all relations in dependency structure are updated (add or minus) with equal scalar. However, it is not always reasonable. For instance, in figure 2 there are three dependency tree candidates - a correct dependency tree (a), both incorrect dependency trees (b) and (c) (node enclosed by dashed circle has incorrect head) - for sentence "Economic news had little effect on financial markets". The shallow parsing result, "news had effect on markets", can be easily drawn from the right candidate (a) or the wrong candidate (b), except for (c). So from the viewpoint of shallow parsing, (b) is better than (c) in spite of both having one wrong relation. As discussed in subsection 2.1, we only concentrate on 'span-head' and 'span-link' in shallow dependency parsing: the nodes and edges closer to 'Root' transmit more semantic information than others. Based on analysis above, we proposed a simulated shallow dependency parser derived from a full parsing.

To differentiate nodes and edges in dependency tree we replace the scalar factor α with a diagonal matrix $A = (\partial_1, \ldots, \partial_n)$. Assuming that feature vector $\phi(x, y)$ is denoted by $(f_1, \ldots, f_n)$, we obtain $(\alpha f_1, \ldots, \alpha f_n)$ as the result of '$\alpha \cdot \phi(x, y)$', or $(\partial_1 f_1, \ldots, \partial_n f_n)$ as the result of '$A \cdot \phi(x, y)$'. Note that feature f_i is a binary value, i.e. 1 or 0. In what follows we make some assumptions for simplicity. Given a sentence x_i and corresponding dependency tree y_i, let T be the set of candidates. The inner product, $\phi(x_i, y_i) \cdot w^T$, is defined to be the score of candidate y_i of sentence x_i. The error set for instance (x_i, y_i) is defined to be the set of the index of candidates which achieve higher scores than correct dependency tree y_i:

$$E = \{r \neq y_i | r \in T,\ \phi(x_i, r) \cdot w^T > \phi(x_i, y_i) \cdot w^T\} \tag{2}$$

Comparing to original perceptron algorithm and others, the innovation of our algorithm derived from the refinement of update factor, i.e. diagonal matrix A. Given a candidate z drawn from error set E, A_z is defined as follows:

$$A_z = diag(\partial_1, ..., \partial_n),\ \partial_i = 2f_i \cdot (1 + e^{hl(f_i, z)})^{-1} \tag{3}$$

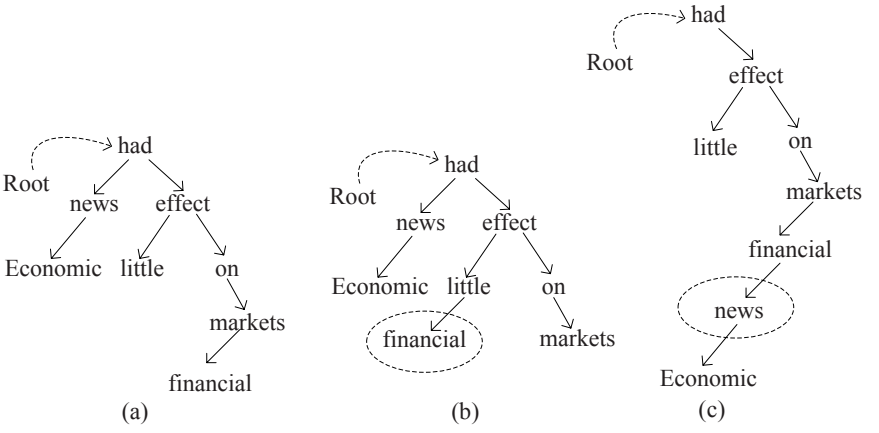

Fig. 2. Dependency tree examples: one correct (a) and two incorrect (b) and (c)

where $hl(f_i, z)$ is a function named 'hierarchical length function' and defined as: (1). The height of a relation $< i, j >$ is the number of passed edges from 'Root' to node j, for example the height of <'had', 'news'> in figure 2(a) is 2, while that of <'on', 'markets'> is 4; (2). If feature f_i is derived purely from a relation r, then $hl(f_i, z)$ is the height of r; (3). If feature f_i is a mixture of features derived from more than one relations $r_1, r_2, \ldots, r_m$, then $hl(f_i, z)$ is the minimum relation height among $r_1, \ldots, r_m$.

Then we can rewrite formula 1 by substituting 'A' for 'α' and obtain formula 4:

$$w = w + A_{y_i} \cdot \phi(x_i, y_i) - \frac{1}{k} \sum_{i=1}^{k} A_{z_i} \cdot \phi(x_i, z_i),\ z_i \in E \tag{4}$$

where E is the error set of size k and z_i is some candidate belonging to error set E. It is clear that diagonal element ∂_i is zero when corresponding feature f_i is zero, therefore, we can ignore 'zero' features in implementation for simplifying the computation. The definition of A in equation 3 implies that the closer to 'root' nodes and relations are, the more aggressive the update to them are during learning. Consequently, we make a trade-off between relations closer to 'root' and relations far from 'root': improve the learning of the former at the cost of decrease of the latter. We have drawn a conclusion in subsection 2.1 that nodes and edges close to 'root' are more semantic and important than those far from 'root'. Likewise, shallow dependency parsing concentrates on nodes and edges which are close to 'root'. So it is feasible to simulate shallow dependency parser via the proposed approach.

4 Experimental Study

4.1 Data and Task Definition

The original data consisting of 10,000 Chinese instances are provided by Information Retrieval Lab of Harbin Institute of Technology. These instances have been labeled

manually with POS, relation and relation type in advance. The total data in the experiments were randomly divided into two groups: one for training with size of 9000 and one for test with size of 1000.

The task in the experiment is to assign labeled dependency edges to Chinese sentences. Each sentence was represented as a sequence of tokens plus POS. For each token, the parser must output its head and corresponding dependency relation. The metrics applying to evaluate parsers are defined as follows (Specially, excluding punctuation from scoring):

- LAS (labeled attachment score). It is the proportion of "scoring" tokens that are assigned both the correct head and the correct dependency relation label
- UAS (unlabeled attachment score). It is the proportion of "scoring" tokens that are assigned the correct head (regardless of the dependency relation label)
- LS (label attachment score). It is the proportion of "scoring" tokens that are assigned the correct dependency relation label (regardless of the head)
- LAS-*n* (labeled attachment score with height n). The height of a token is the number of passed edges from the dummy node 'Root' of dependency tree to itself. Specially, the height of one token t is equal to the height of one relation whose child is just t. Arabic numeral n indicates a concrete height. For example, in figure 2 (a) the height of "financial" is 5, "news" is 2, and "Economic" is 3. LAS-*n* is the proportion of "scoring" tokens with height n in dependency tree that are assigned both the correct head and the correct dependency relation label. In this experiment dependency structure are divided into 5 partitions: LAS-1, LAS-2, LAS-3, LAS-4, and LAS-5_+. Specially, LAS-5_+ includes nodes whose height is larger than 5 (including 5). If there are 100 tokens with height 1, and 80 tokens are assigned both the correct head and the correct label, then LVS-1 is 80%.

4.2 Experimental Results and Analysis

In this paper we proposed a variation of multiclass perceptron algorithm (VMPA), using update rule in formula 4, for simulated shallow dependency parsing. To verify the effectiveness of the proposed approach, we carried out the experiment in which two parsers were built from the training dataset of size 9000 using VMPA and original multiclass perceptron algorithm (MPA) described in formula 1, and then applied them to the test dataset of size 1000 respectively. To make a comparison, we evaluated additional two state-of-the-art dependency parsers: MSTParser[1] and MaltParser[2], which were developed by McDonald [5] and Nivre [6] respectively. Note that MaltParser used embedded SVM learner and predefined feature model for Chinese. MSTParser used its default setting in the experiment. All results were reported in table 1. The results of VMPA showed that the accuracies of tokens with lower height, comparing to that of MPA, had some improvement. LAS-1 and LAS-2 increased 5.86% and 2.02% respectively. Of course, the improvement was at the cost of decrease of accuracies of nodes far from 'Root'. At the same time, the results of both VMPA and MPA were a little worst than two state-of-the-art parsers, i.e. MSTParser and MaltParser. We believed that may be due to the difference of learning algorithms.

[1] http://sourceforge.net/projects/mstparser

[2] http://w3.msi.vxu.se/ nivre/research/MaltParser.html

Table 1. Results of three dependency parsers. (MST:MSTParser, Malt:MaltParser)

Algos	LAS	UAS	LS	LAS-1	LAS-2	LAS-3	LAS-4	LAS-5_+
VMPA	77.16%	79.01%	80.23%	76.87%	75.67%	78.69%	79.01	75.83%
MPA	77.73%	80.20%	83.06%	71.01%	73.65%	80.95%	79.93%	76.45%
Malt	81.16%	83.20%	87.06%	75.13%	77.65%	78.15%	84.93%	84.45%
MST	81.51%	83.21%	86.19%	77.93%	84.16%	83.21%	80.63%	78.01%

5 Conclusions and Future Work

In this paper we focus on shallow dependency parsing and propose a discriminative dependency parsing algorithm based on weighted hierarchical structure learning to simulate it. The results demonstrated that accuracies of nodes closer to 'Root' increased at the cost of some decrease in nodes far from 'root'. This improvement, however, is somewhat limited because the learning is based on instance one by one thus could not make overall trade-off over all training instances. Some improvements to the proposed approach may be brought through additional research. First, the definition for shallow dependency parsing in this paper is still rough and simple. Secondly, the trade-off is based on single example one by one instead of the whole examples due to the online framework of the learning algorithm. In future work we may consider applying batch learning algorithm, such as SVM, with trade-off strategy for shallow parsing.

References

1. Collins, M.: Head-driven statistical models for natural language parsing. Computational Linguistics 29(4), 589–637 (2003)
2. Collins, M., Ramshaw, L., Haji, Ccirc, J., Tillmann, C.: A statistical parser for czech. In: Proceedings of the 37th conference on Association for Computational Linguistics, pp. 505–512 (1999)
3. Eisner, J.: Three new probabilistic models for dependency parsing: An exploration. In: Proceedings of the 16th International Conference on Computational Linguistics (COLING 1996), pp. 340–345 (1996)
4. Jurafsky, D., Martin, J.H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. MIT Press, Cambridge (2000)
5. McDonald, R., Crammer, K., Pereira, F.: Online large-margin training of dependency parsers. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 91–98 (2005)
6. Nivre, J., Hall, J., Nilsson, J.: Maltparser: A data-driven parser-generator for dependency parsing. In: Proc. of LREC 2006 (2006)
7. Yamada, H., Matsumoto, Y.: Statistical dependency analysis with support vector machines. In: Proc. IWPT (2003)