Probabilistic Models for Action-Based Chinese Dependency Parsing

Xiangyu Duan, Jun Zhao, and Bo Xu

Institute of Automation, Chinese Academy of Sciences, Beijing, China
{xyduan,jzhao}@nlpr.ia.ac.cn, xubo@hitic.ia.ac.cn

Abstract. Action-based dependency parsing, also known as deterministic dependency parsing, has often been regarded as a time efficient parsing algorithm while its parsing accuracy is a little lower than the best results reported by more complex parsing models. In this paper, we compare action-based dependency parsers with complex parsing methods such as all-pairs parsers on Penn Chinese Treebank. For Chinese dependency parsing, action-based parsers outperform all-pairs parsers. But action-based parsers do not compute the probability of the whole dependency tree. They only determine parsing actions stepwisely by a trained classifier. To globally model parsing actions of all steps that are taken on the input sentence, we propose two kinds of probabilistic parsing action models that can compute the probability of the whole dependency tree. Results show that our probabilistic parsing action models perform better than the original action-based parsers, and our best result improves much over them.

Keywords: probabilistic, action-based, Chinese dependency parsing.

1 Introduction

Syntactic parsing is one of the most important tasks in Natural Language Processing (NLP). The mainstream of syntactic parsing is the statistical method that often focuses on generative and discriminative models. These models use different optimization objects for parameter training, and use non-deterministic parsing techniques, usually employing some kind of dynamic programming, to compute the probability of the candidate trees. The tree with the highest probability is outputted. If reranking is used, n-best trees are outputted and a different ranking scheme is adopted to rerank these trees.

All these methods perform well while the time complexity is very high due to the computation of candidate trees. Action-based parsers, also known as deterministic parsers, emerge as efficient algorithms that take parsing actions stepwisely on the input sentence, and reduce the time complexity to linear or quadratic with the sentence's length. Action-based parsers were firstly proposed for dependency parsing [1, 2, 3, 4]. Later, Sagae and Lavie [5] and Wang et al. [6] applied deterministic parsing for phrase structure parsing.

On the standard data set of Penn English Treebank, action-based parsers show great efficiency in terms of time, offering accuracy just below the state-of-the-art

J.N. Kok et al. (Eds.): ECML 2007, LNAI 4701, pp. 559–566, 2007.
© Springer-Verlag Berlin Heidelberg 2007

parsing methods. In this paper, for Chinese dependency parsing, we use action-based algorithms [2, 4] and compare them with state-of-the-art parsing methods such as a generative constituent parser [7] and a discriminative all-pairs dependency parser (MSTParser version 0.2) [8, 9] on Penn Chinese Treebank version 5.0 [10]. The comparison has never been done before. Contrary to English parsing, we get the result that action-based parsers perform much better than the generative constituent parser and the discriminative all-pairs dependency parser.

Furthermore, we observe that original action-based parsers are greedy. They do not score the entire dependency tree, and only stepwisely choose the most probable parsing action. To avoid greedy property and further enhance the performance of the original action-based parsers, we propose two kinds of probabilistic models of parsing actions at all steps. Results show that our two probabilistic models perform better than the original action-based dependency parsers. Our best dependency parser improves much over them and gets the state-of-the-art performance.

This paper is organized as follows. Section 2 introduces the action-based dependency parsers that are basic components of our models. In section 3, we present our two probabilistic models for the modeling of parsing actions. Experiments and results are presented in section 4. We get a conclusion in section 5.

2 Introduction of Action-Based Dependency Parsing

There are two representative action-based dependency parsing algorithms which are proposed respectively by Yamada and Matsumoto [2], Nivre [3]. Action-based parsing algorithms regard parsing as a sequence of parsing actions that are taken step by step on the input sentence. Parsing actions construct dependency relations between words. A classifier is trained to classify parsing actions. During testing, parsing actions are determined by the trained classifier.

Next we briefly describe Yamada and Matsumoto's method as an illustration of action-based dependency parsing. The other representative method of Nivre also parses sentences in a similar deterministic manner except different data structure and parsing actions.

Figure 1 illustrates the parsing process of Yamada and Matsumoto's method. The example sentence is "Work achieves remarkable success." There are three kinds of parsing actions used to construct the dependency relation between two focus words. In figure 1, the two focus words are in black bold boxes. Every parsing action results in a new parsing state, which includes all elements of the current partially built tree. In the training phase, features extracted from current parsing state and corresponding parsing actions compose the training data. In the testing phase, the classifier determines which parsing action should be taken based on the features. The parsing algorithm ends when there is no further dependency relation can be made on the whole sentence. The details of the three parsing actions are as follows:

***LEFT*:** it constructs the dependency that the right focus word depends on the left focus word.

***RIGHT*:** it constructs the dependency that the left focus word depends on the right focus word.

***SHIFT*:** it does not construct dependency, just moves the parsing focus.

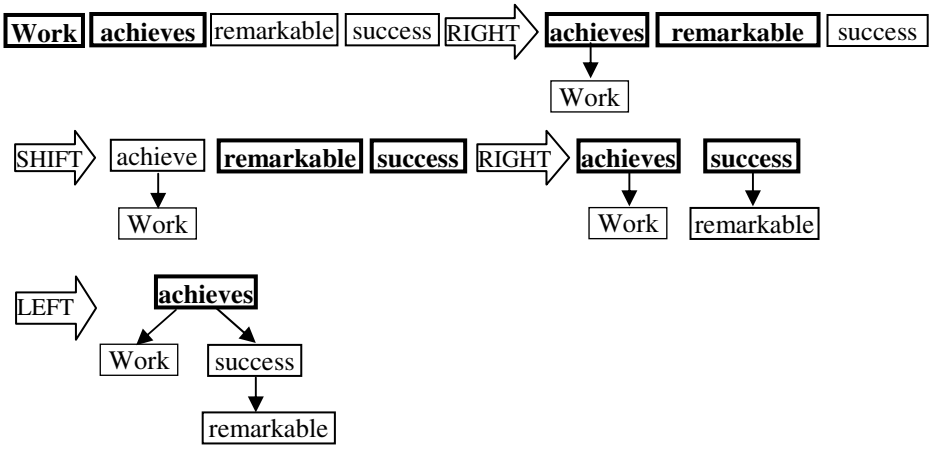

Fig. 1. The example of parsing process of the method of Yamada and Matsumoto

3 Probabilistic Models of Parsing Actions

Action-based dependency parsing introduced in section 2 is greedy. They only choose the most probable parsing action at every parsing step. To overcome this shortsightedness, we propose two kinds of probabilistic models of parsing actions to compute the probability of whole dependency tree. The two models are different from sequential and structural learning models in a way that is explained at the end of section 3.1.

3.1 Parsing Action Chain Model (PACM)

The parsing process can be viewed as a Markov Chain. At every parsing step, there are several candidate parsing actions. The object of this model is to find the right sequence of parsing actions that constructs the dependency tree. As shown in figure 1, the action sequence "***RIGHT*** -> ***SHIFT*** -> ***RIGHT*** -> ***LEFT***" is the right sequence.

Firstly, we should define the probability of the dependency tree conditioned on the input sentence.

$$P(T \mid S) = \prod_{i=1...n} P(d_i \mid d_0 ... d_{i-1}, S) . \tag{1}$$

where T denotes the dependency tree, S denotes the original input sentence, d_i denotes the parsing action at time step i. We add an artificial parsing action d_0 as initial action.

We introduce a variable $context_{d_i}$ to denote the resulting parsing state when the action d_i is taken on $context_{d_{i-1}}$. $context_{d_0}$ is the original input sentence.

Suppose $d_0...d_n$ are taken sequentially on the input sentence *S*, and result in a sequence of parsing states $context_{d_0}...context_{d_n}$, then *P*(*T*\|*S*) defined in equation (1) becomes as below:

$$\prod_{i=1...n} P(context_{d_i} \mid context_{d_0},...,context_{d_{i-1}}) \approx \prod_{i=1...n} P(context_{d_i} \mid context_{d_{i-1}}) = \prod_{i=1...n} P(d_i \mid context_{d_{i-1}}). \quad (2)$$

In equation (2), the second formula comes from the first formula by obeying the Markov assumption. Note that the third formula is about the classifier of parsing actions. It denotes the probability of the parsing action d_i given the parsing state $context_{d_{i-1}}$. If we train a classifier that can predict with probability output, then we can compute *P(T\|S)* by computing the product of the probabilities of parsing actions. The classifier we use throughout this paper is Libsvm [11], which can train multi-class classifier and support training and predicting with probability output.

For this model, the object is to choose the parsing action sequence that constructs the dependency tree with the maximal probability.

$$\max P(T \mid S) = \max_{d_1...d_n} \prod_{i=1...n} P(d_i \mid context_{d_{i-1}}). \quad (3)$$

Because this model chooses the most probable sequence, not the most probable parsing action at only one step, it avoids the greedy property of the original parsers.

Note that probabilistic models of parsing actions decompose parsing problem into actions. This is the main difference between them and traditional structural learning models, which decompose parsing problem into dependency pairs solely over graphs (dependency trees). PACM is related with Searn [14], which also decomposes structural learning into incremental decisions. But Searn uses policy iterations to find the optimal decision sequence.

At each step, although there are same candidate parsing actions, the parsing states are variant. This property makes exact inference like Viterbi unsuitable for the decoding. Best-first search is the appropriate one. Considering efficiency, we use beam search for the decoding of this model. *m* is used to denote the beam size. At every parsing step, all parsing states are ordered (or partially *m* ordered) according to their probabilities. Probability of a parsing state is determined by multiplying the probabilities of actions that generate that state. Then we choose *m* best parsing states for this step, and next parsing step only consider these *m* best parsing states. Parsing terminates when the first entire dependency tree is constructed.

3.2 Parsing Action Phrase Model (PAPM)

In the Parsing Action Chain Model (PACM), actions are competing at every parsing step. That is, only *m* best parsing states resulted by the corresponding actions are kept at every step. But for the parsing problem, it is reasonable that actions are competing for which next phrase should be built. This is the motivation of Parsing Action Phrase Model (PAPM). For dependency syntax, one phrase consists of the head word and all

its dependents. The key question is when the next phrase is built. This can be solved by dividing parsing actions into two classes: constructing action and shifting action.

If a phrase is built after an action is performed, the action is called constructing action. In Yamada and Matsumoto's algorithm, constructing actions are ***LEFT*** and ***RIGHT***. For example, if ***LEFT*** is taken, the right focus word has found all its dependents and becomes the head of this new phrase. Note that one word with no dependents can also be viewed as a phrase if its dependence on other word is constructed. Nivre's method has the similar constructing actions.

If no phrase is built after an action is performed, the action is called shifting action. Such action is ***SHIFT*** in both Yamada and Matsumoto's method and Nivre's method.

We introduce a new concept: parsing action phrase. It is denoted by A_i, the ith parsing action phrase. It can be expanded as $A_i \rightarrow b_{j-k}...b_{j-1}a_j$, where a is constructing action and b is shifting action, j indexes the time step. That is, parsing action phrase A_i is a sequence of parsing actions, which consists a constructing action at last step and all its preceding shifting actions. It is this action sequence that constructs the next syntactic phrase.

For example, consider the parsing process in figure 1, A_1 is "***RIGHT***", A_2 is "***SHIFT***, ***RIGHT***", A_3 is "***LEFT***".

The probability of the dependency tree given the input sentence is redefined as:

$$P(T \mid S) = \prod_{i=1...n} P(A_i \mid A_1...A_{i-1}, S) = \prod_{i=1...n} P(A_i \mid context_{A_{i-1}}) . \quad (4)$$

where $context_{A_i}$ is the parsing state resulted by a sequence of actions taken on $context_{A_{i-1}}$. The object in this model is to find the most probable sequence of parsing action phrases.

Similar with Parsing Action Chain Model (PACM), we use beam search for the decoding of Parsing Action Phrase Model (PAPM). The difference is that PAPM do not keep *m* best parsing states at every parsing step. Instead, PAPM keep *m* best states which are corresponding to *m* best current parsing action phrases (several steps of ***SHIFT*** and the last step of a constructing action).

Table 1. The division of CHTB data set.

	CHTB files	word num
Train set	001-815, 1001-1136	434,936
Development set	886-931, 1148-1151	21,595
Test set	816-885, 1137-1147	50,319

4 Experiments and Results

4.1 Experimental Setup

The data set for the experiments is taken from Penn Chinese Treebank (CHTB) version 5.0 [10], consisting of 500k words mostly from different resources of Xinhua

newswire, Sinorama news magazine and Hongkong news. To balance each resource in train set, development set and test set, we split the data set as in table 1. We use head rules reported in Sun and Jurafsky's work [12] to convert constituent structure to dependency structure

We implement Yamada and Matsumo's method, and the optimal size of the feature context window is six, which consists of left two sub trees, two focus words and right two sub trees. Nivre's method is also implemented, and the optimal feature template is the same with the work [4].

The following metrics are used for evaluation:

Dependency accuracy (**DA**): The proportion of non-root words (excluding punctuations) that are assigned the correct head.

Root accuracy (**RA**): The proportion of root words that are correctly found.

Complete match (**CM**): The proportion of sentences whose dependency structures are completely correct.

4.2 Comparison of Action-Based Parsers with Generative Constituent Parser and Discriminative All-Pairs Parser

For the comparison, we use dbparser, a generative constituent parser implemented by Daniel M. Bikel [7], and MSTParser version 0.2, a discriminative dependency parser implemented by Ryan Mcdonald [8, 9]. The dbparser is an emulating version of Collins parser [13]. We use the same head rules as that used in this paper for both training and testing of dbparser. We present both first-order ($MSTParser_1$) and second-order ($MSTParser_2$) performances of MSTParser. The comparison of these parsers is presented in part of table 2.

Table 2. Performances of different Parsers

	DA	RA	CM
dbparser	79.84	69.03	27.56
$MSTParser_1$	80.83	68.20	25.72
$MSTParser_2$	82.26	69.36	28.23
Nivre	82.52	68.19	29.82
Yamada	82.82	70.13	30.39
PACM	84.05	73.49	32.34
PAPM	**84.36**	**73.70**	**32.70**

From table 2, we can see that action-based parsers perform better than both dbparser and MSTParser. It is interesting that Wang etc. [6] got the similar results of Chinese constituent parsing on Penn Chinese Treebank. Their experiments showed that action-based parser outperform state-of-the-art parsers. This observation is contrary as to English parsing.

4.3 The Performances of Parsers Based on Action Modeling

Due to the simplicity and comparable performance of Yamada and Matsumoto's method, only their method is adopted in parsers which are based on action modeling.

The performances of these parsers are presented in part of table 2. We can see that the two probabilistic models (PACM and PAPM) all perform much better than the original action-based parsers. Parsing action phrase model (PAPM) gets the highest performance with 9% error reduction over Yamada and Matsumoto's method considering DA, 12% error reduction over MSTParser's second-order model, 22% error reduction over dbparser. It shows the great effectiveness in avoiding greediness when we save *m*-best parses at each parsing step, and the promising tree can possibly rank first at the end of parsing.

We also do the experiments of outputting n-best parses by using Parsing Action Chain Model (PACM) and Parsing Action Phrase Model (PAPM) respectively. The performance is listed in table 3, which presents the performance of a "perfect" parser that magically picks the best parse tree from the top *n* trees. The best parse tree has the highest average accuracy when compared to the Treebank. In terms of outputting n-best parses, PAPM is superior to PACM by a large margin and shows its propriety for reranking research which appears to be a promising work to improve the performance.

Table 3. The performance of a "perfect" parser that picks best among 20-best trees

	DA	RA	CM
PACM	88.64	84.78	46.35
PAPM	**91.30**	**86.88**	**54.59**

5 Conclusion

This paper compares the original action-based dependency parsers with state-of-the-art parsing methods such as a generative constituent parser and a discriminative all-pairs dependency parser for Chinese dependency parsing. The results show that original action-based dependency parsers perform best. Based on the observation that original action-based parsers are greedy, we propose two kinds of probabilistic models of the parsing actions for Chinese dependency parsing. The results show that our probabilistic parsing action models improve much over original action-based parsers.

Acknowledgements

This work was supported by Hi-tech Research and Development Program of China under grant No. 2006AA01Z144, the Natural Sciences Foundation of China under grant No. 60673042, and the Natural Science Foundation of Beijing under grant No. 4052027, 4073043.

References

1. Kudo, T., Matsumoto, Y.: Japanese dependency analysis using cascaded chunking. In: Proceedings of the Sixth Conference on ComputationalLanguage Learning (CoNLL) (2002)
2. Yamada, H., Matsumoto, Y.: Statistical dependency analysis with support vector machines. In: Proceedings of the 8th International Workshop on Parsing Technologies (IWPT) (2003)
3. Nivre, J.: An efficient algorithm for projective dependency parsing. In: Proceedings of the 8th International Workshop on Parsing Technologies (IWPT) (2003)
4. Nivre, J., Scholz, M.: Deterministic dependency parsing of English text. In: Proceedings of the 20th International Conference on Computational Linguistics (COLING) (2004)
5. Sagae, K., Lavie, A.: A classifier-based parser with linear run-time complexity. In: Proceedings of the 9th International Workshop on Parsing Technologies (IWPT) (2005)
6. Wang, M., Sagae, K., Mitamura, T.: A fast, accurate deterministic parser for Chinese. In: Proceedings of the 44th AnnualMeeting of the Association for ComputationalLinguistics (ACL) (2006)
7. Daniel, M.: Bikel, On the Parameter Space of Generative Lexicalized Statistical Parsing Models. Ph.D. thesis, University of Pennsylvania (2004)
8. McDonald, R., Crammer, K., Pereira, F.: Online Large-margin Training of Dependency Parsers. In: Proceedings of the 43rd AnnualMeeting of the Association for Computational Linguistics (ACL) (2005)
9. McDonald, R., Pereira, F.: Online Learning of Approximate Dependency Parsing Algorithms. In: Proceedings of the 11th European Chapter of the Association for Computational Linguistics (EACL) (2006)
10. Xue, N., Xia, F., Chiou, F.-D., Palmer, M.: The Penn Chinese Treebank: Phrase structure annotation of a large corpus. Natural Language Engineering (2005)
11. Chang, C.-C., Lin, C.-J.: LIBSVM: A library for support vector machines (2005)
12. Sun, H., Jurafsky, D.: Shallow semantic parsing of Chinese. In: Proceedings of the HLT/NAACL '04 (2004)
13. Collins, M.: Head-Driven Statistical Models for Natural Language Parsing. Ph.D. thesis, University of Pennsylvania (1999)
14. Daum'e III, H., Langford, I.J., Marcu, D.: Search-based structured prediction, 2006 (Submission)