Dependency Parsing Based on Dynamic Local Optimization

Ting Liu Jinshan Ma Huijia Zhu Sheng Li
Information Retrieval Lab
Harbin Institute of Technology
Harbin, 150001, China
{tliu,mjs,hjzhu,ls}@ir.hit.edu.cn

Abstract

This paper presents a deterministic parsing algorithm for projective dependency grammar. In a bottom-up way the algorithm finds the local optimum dynamically. A constraint procedure is made to use more structure information. The algorithm parses sentences in linear time and labeling is integrated with the parsing. This parser achieves 63.29% labeled attachment score on the average in CoNLL-X Shared Task.

1 Introduction

Recently, dependency grammar has gained renewed attention in the parsing community. Good results have been achieved in some dependency parsers (Yamada and Matsumoto, 2003; Nivre et al., 2004). With the availability of many dependency treebanks (van der Beek et al., 2002; Hajič et al., 2004; Böhmová et al., 2003; Kromann, 2003; Džeroski et al., 2006) and more other treebanks which can be converted to dependency annotation (Brants et al., 2002; Nilsson et al., 2005; Chen et al., 2003; Kawata and Bartels, 2000), multi-lingual dependency parsing is proposed in CoNLL shared task (Buchholz et al., 2006).

Many previous works focus on unlabeled parsing, in which exhaustive methods are often used (Eisner, 1996). Their global searching performs well in the unlabeled dependency parsing. But with the increase of parameters, efficiency has to be considered in labeled dependency parsing. Thus deterministic parsing was proposed as a robust and efficient method in recent years. Such method breaks the construction of dependency tree into a series of actions. A classifier is often used to choose the most probable action to assemble the dependency tree. (Yamada and Matsumoto, 2003) defined three actions and used a SVM classifier to choose one of them in a bottom-up way. The algorithm in (Nivre et al., 2004) is a blend of bottom-up and top-down processing. Its classifier is trained by memory-based learning.

Deterministic parsing derives an analysis without redundancy or backtracking, and linear time can be achieved. But when searching the local optimum in the order of left-to-right, some wrong reduce may prevent next analysis with more possibility. (Jin et al., 2005) used a two-phase shift-reduce to decrease such errors, and improved the accuracy of long distance dependencies.

In this paper a deterministic parsing based on dynamic local optimization is proposed. According to the probabilities of dependency arcs, the algorithm dynamically finds the one with the highest probabilities instead of dealing with the sentence in order. A procedure of constraint which can integrate more structure information is made to check the rationality of the reduce. Finally our results and error analysis are presented.

2 Dependency Probabilities

An example of Chinese dependency tree is showed in Figure1. The tree can be represented as a directed graph with nodes representing word tokens and arcs

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 211–215, New York City, June 2006.

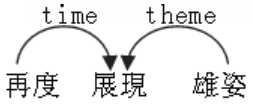

Figure 1: A Chinese dependency tree

representing dependency relations. The assumption that the arcs are independent on each other often is made so that parsing can be handled easily. On the other side the independence assumption will result in the loss of information because dependencies are interrelated on each other actually. Therefore, two kinds of probabilities are used in our parser. One is arc probabilities which are the possibility that two nodes form an arc, and the other is structure probabilities which are used to describe some specific syntactic structures.

2.1 Arc Probabilities

A dependency arc A_i can be expressed as a 4-tuple $A_i = <Node_i, Node_j, D, R>$. $Node_i$ and $Node_j$ are nodes that constitute the directed arc. D is the direction of the arc, which can be *left* or *right*. R is relation type labeled on the arc. Under the independence assumption that an arc depends on its two nodes we can calculate arc probability given two nodes. In our paper the arc probabilities are calculated as follows:

$P_1 = P(R,D|CTag_i, CTag_j, Dist)$
$P_2 = P(R,D|FTag_i, FTag_j)$
$P_3 = P(R,D|CTag_i, Word_j)$
$P_4 = P(R,D|Word_i, CTag_j)$
$P_5 = P(R,D|Word_i, CTag_i, Word_j, CTag_j)$
$P_6 = P(R,D|CTag_{i-1}, CTag_i, CTag_j, CTag_{j+1})$

Where *CTag* is coarse-grained part of speech tag and *FTag* is fine-grained tag. As to *Word* we choose its lemma if it exists. *Dist* is the distance between $Node_i$ and $Node_j$. It is divided into four parts:

$Dist = 1 \quad if \quad j\text{-}i = 1$
$Dist = 2 \quad if \quad j\text{-}i = 2$
$Dist = 3 \quad if \quad 3 \leqslant j\text{-}i \leqslant 6$
$Dist = 4 \quad if \quad j\text{-}i > 6$

All the probabilities are obtained by maximum likelihood estimation from the training data. Then interpolation smoothing is made to get the final arc probabilities.

2.2 Structure Probabilities

Structure information plays the critical role in syntactic analysis. Nevertheless the flexibility of syntactic structures and data sparseness pose obstacles to us. Especially some structures are related to specific language and cannot be employed in multilingual parsing. We have to find those language-independent features.

In valency theory "valence" represents the number of arguments that a verb is able to govern. In this paper we extend the range of verbs and arguments to all the words. We call the new "valence" *Governing Degree* (GD), which means the ability of one node governing other nodes. In Figure1, the GD of node "展現" is 2 and the GDs of two other nodes are 0. The governing degree of nodes in dependency tree often shows directionality. For example, Chinese token "的" always governs one left node. Furthermore, we subdivide the GD into *Left Governing Degree* (LGD) and *Right Governing Degree* (RGD), which are the ability of words governing their left children or right children. In Figure 1 the LGD and RGD of verb "展現" are both 1.

In the paper we use the probabilities of GD over the fine-grained tags. The probabilities of *P(LDG|FTag)* and *P(RGD|FTag)* are calculated from training data. Then we only reserve the *FTags* with large probability because their GDs are stable and helpful to syntactic analysis. Other *FTags* with small probabilities are unstable in GDs and cannot provide efficient information for syntactic analysis. If their probabilities are less than 0.65 they will be ignored in our dependency parsing.

3 Dynamic local optimization

Many previous methods are based on history-based models. Despite many obvious advantages, these methods can be awkward to encode some constrains within their framework (Collins, 2000). Classifiers are good at encoding more features in the deterministic parsing (Yamada and Matsumoto, 2003; Nivre et al., 2004). However, such algorithm often make more probable dependencies be prevented by preceding errors. An example is showed in Figure 2. Arc *a* is a frequent dependency and *b* is an arc with more probability. Arc *b* will be prevented by *a* if the reduce is carried out in order.

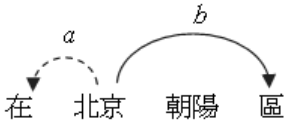

Figure 2: A common error in deterministic parsing

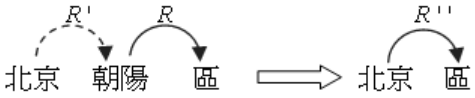

Figure 3: Adjustment

3.1 Our algorithm

Our deterministic parsing is based on dynamic local optimization. The algorithm calculates the arc probabilities of two continuous nodes, and then reduces the most probable arc. The construction of dependency tree includes four actions: **Check**, **Reduce**, **Delete**, and **Insert**. Before a node is reduced, the Check procedure is made to validate its correctness. Only if the arc passes the Check procedure it can be reduced. Otherwise the Reduce will be delayed. Delete and Insert are then carried out to adjust the changed arcs. The complete algorithm is depicted as follows:

```
Input Sentence: S = (w1, w2,    , wn)
Initialize:
for i = 1 to n
    Ri = GetArcProb(wi,wi+1);
    Push(Ri) onto Stack;
Sort(Stack);
Start:
i = 0;
While  Stack.empty = false
        R = Stack.top+i;
        if Check(R) = true
           Reduce(R);
           Delete(R');
           Insert(R");
           i = 0;
        else
           i++;
```

The algorithm has following advantages:

- Projectivity can be guaranteed. The node is only reduced with its neighboring node. If a node is reduced as a leaf it will be removed from the sentence and doesn't take part in next Reduce. So no cross arc will occur.
- After n-1 pass a projective dependency tree is complete. Algorithm is finished in linear time.
- The algorithm always reduces the node with the highest probability if it passes the Check. No any limitation on order thus the spread of errors can be mitigated effectively.
- Check is an open process. Various constrains can be encoded in this process. Structural constrains, partial parsed information or language-dependent knowledge can be added.

Adjustment is illustrated in Figure 3, where "朝陽" is reduced and arc *R'* is deleted. Then the algorithm computes the arc probability of *R"* and inserts it to the Stack.

3.2 Checking

The information in parsing falls into two kinds: static and dynamic. The arc probabilities in 2.1 describe the static information which is not changed in parsing. They are obtained from the training data in advance. The structure probabilities in 2.2 describe the dynamic information which varies in the process of parsing. The use of dynamic information often depends on what current dependency tree is.

Besides the governing degree, Check procedure also uses another dynamic information–*Sequential Dependency*. Whether current arc can be reduced is relating to previous arc. In Figure 3 the reduce of the arc *R* depends on the arc *R'*. If *R'* has been delayed or its probability is little less than that of *R*, arc *R* will be delayed.

If the arc doesn't pass the Check it will be delayed. The delayed time ranges from 1 to *Length* which is the length of sentence. If the arc is delayed *Length* times it will be blocked. The Reduce will be delayed in the following cases:

- $\widehat{GD}(Node_i) > 0$ and its probability is *P*. If GD($Node_i$) = 0 and $Node_i$ is made as child in the Reduce, the $Node_i$ will be delayed *Length*P* times.
- $\widehat{GD}(Node_i) \leqslant m$ ($m > 0$) and its probability is *P*. If GD($Node_i$) = *m* and $Node_i$ is made as parent in the Reduce, the $Node_i$ will be delayed *Length*P* times.

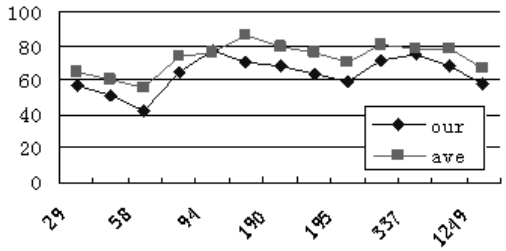

Figure 4: Token score with size of training data

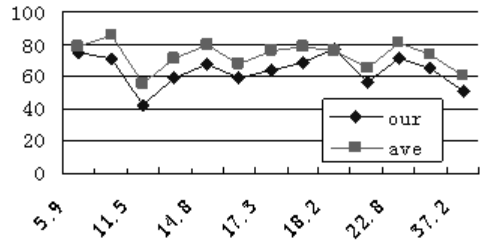

Figure 5: Token score with sentence length

- $P(R') > \lambda P(R)$, the current arc R will be delayed *Length*(P(R')/P(R))* times. *R'* is the preceding arc and $\lambda = 0.60$.
- If arc *R'* is blocking, the arc *R* will be delayed.

$\widehat{GD}$ is empirical value and GD is current value.

4 Experiments and analysis

Our parsing results and average results are listed in the Table 1. It can be seen that the attachment scores vary greatly with different languages. A general analysis and a specific analysis are made respectively in this section.

4.1 General analysis

We try to find the properties that make the difference to parsing results in multi-lingual parsing. The properties of all the training data can be found in (Buchholz et al., 2006). Intuitively the size of training data and average length of per sentence would be influential on dependency parsing. The relation of these properties and scores are showed in the Figure 4 and 5.

From the charts we cannot assuredly find the properties that are proportional to score. Whether Czech language with the largest size of training data or Chinese with the shortest sentence length, don't achieve the best results. It seems that no any factor is determining to parsing results but all the properties exert influence on the dependency parsing together.

Another factor that maybe explain the difference of scores in multi-lingual parsing is the characteristics of language. For example, the number of tokens with HEAD=0 in a sentence is not one for some languages. Table 1 shows the range of governing degree of head. This statistics is somewhat different with that from organizers because we don't distinguish the scoring tokens and non-scoring tokens.

Another characteristic is the directionality of dependency relations. As Table 1 showed, many relations in treebanks are bi-directional, which increases the number of the relation actually. Furthermore, the flexibility of some grammatical structures poses difficulties to language model. For instance, subject can appear in both sides of the predicates in some treebanks which tends to cause the confusion with the object (Kromann, 2003; Afonso et al., 2002; Civit Torruella and Martí Antonín, 2002; Oflazer et al., 2003; Atalay et al., 2003).

As to our parsing results, which are lower than all the average results except for Danish. That can be explained from the following aspects:

(1) Our parser uses a projective parsing algorithm and cannot deal with the non-projective tokens, which exist in all the languages except for Chinese.

(2) The information provided by training data is not fully employed. Only POS and lemma are used. The morphological and syntactic features may be helpful to parsing.

(3) We haven't explored syntactic structures in depth for multi-lingual parsing and more structural features need to be used in the Check procedure.

4.2 Specific analysis

Specifically we make error analysis to Chinese and Turkish. In Chinese result we found many errors occurred near the auxiliary word "的"(DE). We call the noun phrases with "的" *DE Structure*. The word "的" appears 355 times in the all 4970 dependencies of the test data. In Table 2 the second row shows the frequencies of "DE" as the parent of dependencies. The third row shows the frequencies while it is as child. Its error rate is 33.1% and 43.4% in our results respectively. Furthermore, each head error will result in more than one errors, so the errors from *DE Structures* are nearly 9% in our results.

	Ar	Ch	Cz	Da	Du	Ge	Ja	Po	Sl	Sp	Sw	Tu
our	50.74	75.29	58.52	77.70	59.36	68.11	70.84	71.13	57.21	65.08	63.83	41.72
ave	59.94	78.32	67.17	76.16	70.73	78.58	85.86	80.63	65.16	73.52	76.44	55.95
NH	17	1	28	4	9	1	14	1	11	1	1	5
BD	27/24	78/55	82/72	54/24	26/17	46/40	7/2	55/40	26/23	21/19	64/54	26/23

Table 1: The second and third rows are our scores and average scores. The fourth row lists the maximal number of tokens with HEAD=0 in a sentence. The last row lists the number of relations/the number of bi-directional relations of them (Our statistics are slightly different from that of organizers).

	gold	system	error	headerr
parent	320	354	106	106
child	355	355	154	74

Table 2: Chinese DE Structure Errors

	total	obj	sub	mod	D.A	L.A
Noun-V	1300	494	319	156	102	78
Pron-V	215	91	60	9	37	3

Table 3: The distribution of some common relations

The high error rate is due to the flexibility of *DE Structure*. The children of DE can be nouns and verbs, thus the ambiguities will occur. For example, the sequence "V N1 DE N2" is a common ambiguious structure in Chinese. It needs to be solved with semantic knowledge to some extent. The errors of DE being child are mostly from noun compounds. For example, the string "週詳 的 作品 維護" results in the error: "DE" as the child of "作品". It will be better that noun compounds are processed specially.

Our results and average results achieve the lowest score on Turkish. We try to find some reasons through the following analysis. Turkish is a typical head-final language and 81.1% of dependencies are right-headed. The monotone of directionality increases the difficulties of identification. Another difficulty is the diversity of the same pair. Taking noun and pronoun as example, which only achieve the accuracy of 25% and 28% in our results, there are 14 relations in the noun-verb pairs and 11 relations in the pronoun-verb pairs. Table 3 illustrates the distribution of some common relations in the test data.

The similarity of these dependencies makes our parser only recognize 23.3% noun-verb structures and 21.8% pronoun-verb structures. The syntactic or semantic knowledge maybe helpful to distinguish these similar structures.

5 Conclusion

This paper has applied a deterministic algorithm based on dynamic local optimization to multi-lingual dependency parsing. Through the error analysis for some languages, we think that the difference between languages is a main obstacle posed on multi-lingual dependency parsing. Adopting different learners according to the type of languages may be helpful to multi-lingual dependency parsing.

Acknowledgement This work was supported by the National Natural Science Foundation of China under Grant No. 60435020，60575042 and 60503072.

References

M. Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. of ICML*.

M.X. Jin, M.Y. Kim, and J.H. Lee. 2005. Two-phase shift-reduce deterministic dependency parser of chinese. In *Proc. of IJCNLP: Companion Volume including Posters/Demos and tutorial abstracts*.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proc. of the Eighth Conf. on Computational Natural Language Learning (CoNLL)*, pages 49–56.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of the 16th Intern. Conf. on Computational Linguistics (COLING)*, pages 340–345.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of the 8th Intern. Workshop on Parsing Technologies (IWPT)*.