

A Statistical Dependency Parser of Chinese under Small Training Data*

Ma Jinshan Zhang Yu Liu Ting Li Sheng

School of Computer Science and Technology

Harbin Institute of Technology

Harbin,150001

mjs@ir.hit.edu.cn

Abstract

Parsing is an important and difficult problem in natural language processing. In this paper a probabilistic parsing model is proposed in terms of dependency grammar. A small corpus annotated manually serves as training data because the large-scale Chinese Treebank is unavailable at present. Dependency relations of the part-of-speech tags are obtained from training data, and their probabilities are used as parameters of model. Then we present a dynamic programming method to find the best path. Experiments show that our method achieves a good result with precision of 80.25% on open test.

1 Introduction

Parsing has to resolve two problems: modeling and searching. Before parsing we should develop a model according to the knowledge extracted from human brain or training data, that is knowledge acquisition. The model should be made to evaluate every parsing result. Then searching algorithm will find the satisfying one from all the results within a reasonable time.

Knowledge acquisition is implemented mainly by rule-based methods and statistical methods. The rules are often written manually and sometimes represented in the form of probability. (Booth, 1973) applied rules to parse the English sentence on context-free grammar, and (Zhou, 2000) experi-

mented with this method for Chinese dependency parsing. However hand-written grammar is hard to obtain because it depends on writers' experience and linguistic knowledge. Thus the application of this method is limited.

An annotated corpus is required for statistical methods, and the knowledge is extracted from it automatically and used as parameters to build parsing model. Sufficiency of training data and accuracy of knowledge from it are crucial for this method. By statistical learning from Penn Tree-bank, (Collins, 1997; Charniak, 1997) have parsed English to an advanced level. (Bikel and Chiang, 2000) also applied two statistical models, which originally designed for English parsing, to Chinese Treebank on context-free grammar.

Many legal paths, that are parsing results, are generated when a sentence is parsed by statistical method, so it is important to find the best one quickly from all the paths. Some algorithms with different time are implemented according to different needs. (Gao, 2003) used a parsing algorithm with $O(n^2)$ time, and (Arnola, 1998)'s algorithm is only linear. But those algorithms are approximate and don't assure that the best path can be found.

In this paper we work with dependencies (as opposed to the well-know phrase-structure) as the syntactic theory of choice, and propose a Chinese statistical parsing model based on dependency relations between the part of speech (POS) tags. Then we apply a dynamic programming algorithm with $O(n^3)$ time to finding the best path.

Dependency representations are based on explicit representations of the relationship between individual words. In the literature there are many ways of defining both this relation and the properties required from the relation. (Zhou, 1994) has form a detailed scheme for Chinese dependency representations. Our dependencies are almost the same.

* This research was supported by National Natural Science Foundation (60203020) and Science Foundation of Harbin Institute of technology (hit.2002.73).

2 Probabilistic Model

A probabilistic parsing model can be viewed as a probability function over a set of objects. Under given grammar the function assigns every path a probability that is used to measure which path is the best. Dependency parsing model transforms sentence S into dependency tree t . S is the input sentence, and T is the set of all the dependency trees that the grammar G generates and P is the probability function. For any $t \in T$, function P will assigns it a probability $p(t/S, G)$, which $\sum_t p(t/S, G) = 1$.

2.1 Representations of Dependency Relation

Our task is to build a parsing mode that outputs the most probable parsing result t^* in terms of dependency grammar when the sentence $S = O_1 O_2 \dots O_n$ is input. S is a sentence with segmentation and POS tagging; O is the observation of the sentence S , which may be word, POS or word sense, etc. According to dependency grammar a sentence with N words will be transformed into a tree with $N-1$ arcs, each of which links two words. In our model we make the independence assumption that every arc is independent on others in the same tree. Hence our task can be express as follows:

$$\begin{aligned} t^* &= \operatorname{argmax}_{t \in G} P(t | S) \\ &= \operatorname{argmax}_{t \in G} P(t | O_1, O_2, \dots, O_n) \\ &= \operatorname{argmax}_{t \in G} P(L_1, L_2, \dots, L_{n-1} | O_1, O_2, \dots, O_n) \quad (1) \\ &= \operatorname{argmax}_{t \in G} \prod_{1 \leq i, j \leq n} p(L_{ij}) \quad (2) \end{aligned}$$

where L_1, L_2, \dots, L_{n-1} in (1) are links that compose a dependency tree; L_{ij} ($i < j$) in (2) is a directed arc, and the product of its probability is equal to the probability of the dependency tree. Directed arc L_{ij} can be described as 4-tuple,

$$\langle O_i, O_j, Direction, Distance_{j-i} \rangle$$

where $Direction$, the direction of arc, is defined as follows:

$$Direction = \begin{cases} 0 & \text{if } O_j \text{ depends on } O_i \\ 1 & \text{if } O_i \text{ depends on } O_j \end{cases}$$

$Distance_{j-i}$ is the distance between O_i and O_j , it is defined as follows:

$$Distance_{j-i} = \begin{cases} 1 & \text{if } j-i=1 \\ 2 & \text{if } j-i=2 \\ 3 & \text{if } j-i>2 \end{cases}$$

The distance factor is introduced in order to describe dependency analysis accurately. It is impossible to define $Distance$ according to actual distance between two observations. Collocations with long distance are often intervened with modifiers such as attribute or adverbial, and the distance between them is various but their dependency relation is still stable. So $Distance$ is taken as 3 for distance with 3 and greater than 3.

The parameter $p(L_{ij})$ is used to describe the dependency relation between O_i and O_j , so it is important to choose appropriate observation O . In a sentence words contain more information than any other observations. Using lexical head-driven method (Collins, etc.) have achieved a good result on English parse. But building tree-bank is an arduous task. For Chinese such work is lagging behind and knowledge acquisition for parsing is confronted with great difficulties.

Now we have annotated a corpus with 7300 sentences, and chose POS as observations to describe the dependency relations. When training data is sufficient the observations may be replaced by others such as words or word senses. So this model is flexible and easy to be extended.

2.2 Parameter Estimation

The parameter $p(L_{ij})$ is acquired by maximum likelihood estimation:

$$p(L_{ij}) = P(O_i, O_j, Direction, Distance_{j-i}) \quad (3)$$

$$= P(Tag_i, Tag_j, Direction, Distance_{j-i}) \quad (4)$$

$$= \frac{C(Tag_i, Tag_j, Direction, Distance_{j-i})}{C(Tag_i, Tag_j, Distance_{j-i})} \quad (5)$$

in which we make another assumption that each directed arc only depends on two POS tags it links. So the observations in (3) are represented by POS Tag_i and Tag_j in (4), and formula (5) is the probability estimation to the dependency relation,

where numerator is the number of times directed arc L_{ij} has occurred in training data, and the denominator is the number of times Tag_i and Tag_j are seen with $Distance_{j,i}$ in the same training data, including that Tag_i and Tag_j have dependency relation and have not.

Statistical information is sufficient for POS, however some correct collocations have not occurred in the training data. To deal with the data sparseness problem of MLE, we use interpolation strategy to smooth the data. Probability of L_{ij} is denoted as P_k after smoothing, in which $k = j-i$ and $k \in \{1, 2, 3\}$, P_k may be respectively expressed as:

$$P_1 = \lambda_1 \frac{\eta_1}{\delta_1} + \lambda_2 \frac{\eta_2}{\delta_1 + \delta_2} + \lambda_3 \frac{\eta_3}{\delta_1 + \delta_3} + \lambda_4 \xi$$

$$P_2 = \lambda_1 \frac{\eta_2}{\delta_2} + \lambda_2 \frac{\eta_1}{\delta_2 + \delta_1} + \lambda_3 \frac{\eta_3}{\delta_2 + \delta_3} + \lambda_4 \xi$$

$$P_3 = \lambda_1 \frac{\eta_3}{\delta_3} + \lambda_2 \frac{\eta_1}{\delta_3 + \delta_1} + \lambda_3 \frac{\eta_2}{\delta_3 + \delta_2} + \lambda_4 \xi$$

where,

$$\eta_k = C(Tag_i, Tag_j, Direction, Distance_k)$$

$$\delta_k = C(Tag_i, Tag_j, Distance_k)$$

$$= C(Tag_i, Tag_j, 0, Distance_k)$$

$$+ C(Tag_i, Tag_j, 1, Distance_k)$$

$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$, ξ is an exponential constant, which taken as 0.0001 here.

Data sparseness problem can be solved effectively through interpolation smoothing.

3 Searching algorithm

Statistical model could generate hundreds of results that conform to the given grammar when parsing a sentence, and the aim of searching algorithm is to find the best one from those results. Such question in this paper may be induced to find the best tree consistent with dependency grammar in a digraph, and we implements a dynamic programming method that is similar to the one in (Eisner, 1996) to find the best path from all the parsing results within $O(n^3)$ time. The algorithm is described below.

3.1 Some Definitions

Sub-tree. A dependency tree composed of some contiguous nodes in a sentence. Every word in sentence can be viewed as a node of tree. The length of sentence is N and the length of sub-tree may be from 1 to N . We call sub-tree with i nodes *i-subtree*. Two examples of sub-tree are as follows in Figure 1.

The node that has no parent is called *root*. Every sub-tree has one and only root. For example, the first node in Figure 1 is the root of corresponding sub-tree. Every sub-tree has its probability, which is called sub-tree probability.

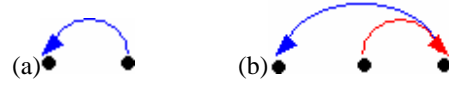


Figure 1: Two examples of sub-trees, in which each dot represents a node of sub-tree and directed arcs represent the dependency relation between two nodes. (a) is a 2-subtree and (b) is a 3-subtree

Sibling-tree. A sequence of nodes will compose more than one sub-tree, in which we call sub-trees that have same root *sibling-tree*. According to definition i nodes will compose i sibling-trees. For example, three nodes may compose seven sub-trees, and every node functions as root in turn these sub-trees may be divided into three sibling-trees as Figure 2:

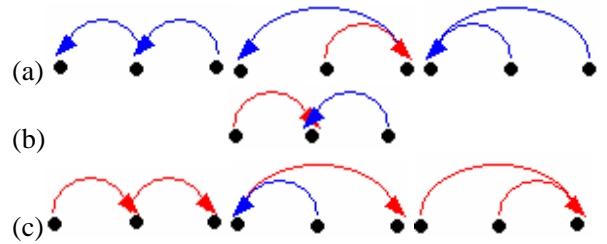


Figure 2: Three sibling-trees composed of three nodes. The sub-trees in a sibling-tree have the same root. In (a) first node is root, in (b) second is root and in (c) third is root.

There are two types of operation when searching for the best tree:

Concatenation. Combining two adjacent sub-trees into one sub-tree. This operation will concatenate the roots of two sub-trees through a new dependency arc. In Figure 3, a 5-subtree is formed after

concatenating the two sub-trees in Figure 1 through arc a .



Figure 3: Directed arc a concatenates two roots and two sub-trees form a 5-subtree. The first node is root of the new sub-tree.

Deletion. Discarding all sub-trees in a sibling-tree but the one with highest probability. The sub-trees in a sibling-tree are indistinguishable in their ability to combine with other sub-trees, so we can safely discard the lower probability sub-trees. In Figure 2 if *deletion* are carried out respectively for three sibling-trees, only three sub-trees, in which every sub-tree probability is highest in its sibling-tree, will be remained. *Deletion* is actually to prune temporary results and keep the highest one.

3.2 Searching

The algorithm will generate all the i -subtrees ($1 \leq i \leq N$) from bottom to top, and every i -subtree is generated based on all previous sub-trees by *concatenation*. After *deletion* for i sibling-trees i subtrees are remained. Because *deletion* keeps the highest one from all the previous results, the highest probability i -subtrees must be available. In turn $(i+1)$ -subtrees, $(i+2)$ -subtrees and N -subtrees are generated, finally we get N sub-trees with *root* being from 1 to N , in which the highest one is the best path. The algorithm is described in Figure 4.

```

Dependency_Parsing(S)
  for length  $\leftarrow$  2 to N
    for begin  $\leftarrow$  0 to N - length
      {
        end = begin + length - 1;
        for i  $\leftarrow$  begin to end
          {
            Concatenate(begin, i, end);
            Delete(begin,i,end);
          }
      }
  Max_tree  $\leftarrow$  Find maximal one in N trees;
  Output(Max_tree);

```

Figure 4: Searching algorithm of parsing

In Figure 4 *Concatenate(begin,i,end)* functions as combining nodes from *begin* to *end* into sub-trees, and generating a sibling-tree with i being *root*; *Delete(begin,i,end)* functions as deleting the lower probability sub-trees in this sibling-tree and remaining the highest one with i being *root*.

4 Experiments

Our tree-bank consists of 7,300 sentences and the length of sentence is 9 (words) on the average. The parser is trained on one part of tree-bank (5,300 sentences) and tested on the other part (2,000 sentences). Close test also is made for contrast. We use such measures as precision of dependency arcs, precision of dependency trees and precision of head words to evaluate our method.

The results in proposed parse are called S_1 , and corresponding results in tree-bank parse are called S_2 . The precision of dependency arcs is defined as the ratio of all correct dependency arcs in S_1 to all dependency arcs in S_2 . The precision of dependency trees is defined as the ratio of the sentences with every arc being correct in S_1 to the sentences in S_2 . The precision of head words is defined as the ratio of the sentences with head words being correct to the sentences in S_2 . Table 1 shows our experimental results.

Precision	Close Test	Open Test
Dependency Arcs	80.38%	80.25%
Head Words	81.61%	83.22%
Dependency Trees	40.27%	40.20%

Table 1: Experimental results

Experimental results show that open test has little difference with close test, and prove that the dependency relation between the POS is insensitive to data and able to represent the fundamental rules of grammar. In addition the experimental results also explain that statistical information from 5300 sentences training data is sufficient enough to dependency relation of POS. That provides a good solution to the problem of the shortage of Chinese tree-bank.

However there are some problems that are not able to be solved by the method. The relation of POS is a little coarse and cannot describe language in detail though its information is sufficient. Some specific syntactic structure cannot be recognized

through it. For example, $\langle vg, ng \rangle$ is a common collocation, in which POS “ng” (general noun) mostly depends on POS “vg” (general verb), but for Chinese there are some structures such as “运动/vg 速度/ng” (speed of motion), whose dependency relations are “vg” depends on “ng”. It is difficult for statistical information of POS to recognize them, which account for about 4.7% in our test data and have a serious effect on experimental results. This problem can be solved effectively only through applying lexical information or other.

That our model is developed in the statistical framework lays the foundation for above problem, because our method can be extended easily. When large-scale tree-bank is available the dependency relation between the words would be extracted. The observations in (3) may be replaced with $\langle Tag_i, Word_j \rangle$, $\langle Word_i, Word_j \rangle$ or other particular information, and the framework remains unchanged. Parse precision will improve with parameters of model being updated.

5 Conclusion

In recent years, the context-free grammar (CFG) has been the mainstream in the parsing community. However, dependency grammar is attracting more and more attention because of its characteristics: no link labels, no grammar, and no fuss to understand (Eisner, 1996). Especially for oriental languages dependency grammar is applied widely.

In this paper statistical information of POS is acquired from a small training data and statistical model is developed based on the information. Then we present a dynamic programming algorithm to search for the best path within $O(n^3)$ time. The experimental results are satisfying and show that statistical method has great promise in Chinese dependency parsing.

We have tried to apply lexical information from our small training data to dependency parsing. But the problem of data sparseness results in overfitting and lower precision. In next work we will enlarge the scale of tree-bank and introduce lexical information to statistical model, solving the problem of coarseness of the POS information.

References

- T. L. Booth and R. A. Thompson. 1973. Applying Probability Measures to Abstract Languages. *IEEE Transactions on Computers*.
- Zhou Ming. 2000. A Block-based Dependency Parser for Unrestricted Chinese Text. *ACL-2000 2nd Chinese Lang Processing Workshop*.
- M. Collins. 1997. Three Generative, Lexicalized Models for Statistical Parsing. In *Proceedings of the 35th annual meeting of the association for computational linguistics*.
- E. Charniak. 1997. Statistical Techniques for Natural Language Parsing. *AI Magazine*.
- Daniel M. Bikel and David Chiang. 2000. Two Statistical Parsing Models Applied to the Chinese Treebank. *Proceedings of the Second Chinese Language Workshop, ACL 2000*.
- Gao Jianfeng, Hisami Suzuki. 2003. Unsupervised Learning of Dependency Structure for Language Modeling. In *41th ACL*.
- Arnola Harri. 1998. On parsing binary dependency structures deterministically in linear time. in Kahane & Polguere (eds), *Workshop on dependency-based grammars, COLING-ACL'98, Montreal*.
- Jason M. Eisner. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*.
- Zhou Ming, Huang Changning. 1994. Approach to the Chinese Dependency Formalism For the Tagging of Corpus. *Journal of Chinese Information Processing*.