中文语义依存关系分析的统计模型

李明琴[1] 李涓子[2] 王作英[1] 陆大绘[1]

[1](清华大学电子工程系 北京 100084)

[2](清华大学计算机科学与技术系 北京 100084)

摘 要 该文提出了一个统计语义分析器,它能够发现中文句子中的语义依存关系.这些语义依存关系可以用于表示句子的意义和结构.语义分析器在 1 百万词的标有语义依存关系的语料库(语义依存网络语料库,SDN)上训练并测试,文中设计、实现了多个实验以分析语义分析器的性能.实验结果表明,分析器在非限定领域中表现出了较好的性能,分析正确率与中文句法分析器基本相当.

关键词 语义分析;统计分析器;依存语法;语义依存网络

中图法分类号 TP39

A Statistical Model for Parsing Semantic Dependency Relations in a Chinese Sentence

LI Ming-Qin[1] LI Juan-Zi[2] WANG Zuo-Ying[1] LU Da-Jin[1]

[1](*Department of Electronic Engineering, Tsinghua University, Beijing* 100084)

[2](*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

Abstract This paper presents a statistical semantic parser, which could discover the semantic dependency relations in a Chinese sentence. By these relations the meaning and structure of sentence could be represented. The parser was trained and evaluated on a 1M-word-scale corpus annotated with semantic dependency, Semantic Dependency Net (SDN). Various experiments were carried out to analyze the performance of the parser, and good performance was shown in parsing domain-unrestricted text. The accuracy reached as high level as the Chinese syntactic parser.

Keywords semantic parsing; statistical parser; dependency grammar; semantic dependency net

1 Introduction

Recently many efforts have been made to research on statistical parsers, and many of them are focused on the syntactic parsing. By using statistical models trained on the Penn Treebank[1], exciting results were obtained on English parsers[2~5]. In these English parsers, two conceptions are applied: the headword of constituent and lexical information. The headword is a word in the constituent that could represent the main syntactic and semantic features of the constituent. For example, in Collins' parser[2], phrase structure rules were first converted to the head-modifier dependency relations, and then the parser was based on the bigram of these dependencies. In Charniak's parser[4], the model was based on the probabilities of rules that extend a parse tree. Furthermore, the probability of a rule was factored into three parts: the probability of the headword of the constituent, the prob-

收稿日期:2003-03-21;修改稿收到日期:2004-07-12.本课题得到国家"八六三"高技术研究发展计划项目基金(2001AA114071)资助.
李明琴,女,1977 年生,博士研究生,主要研究领域为语音识别、自然语言处理和中文语义分析.**李涓子**,女,1964 年,副教授,主要研究领域为自然语言理解、中文信息处理、网络中的知识挖掘和管理.**王作英**,男,1935 年生,教授,博士生导师,主要研究领域为语音识别与合成.**陆大绘**,男,1928 年生,教授,博士生导师,主要研究领域为信号与信息处理、语音识别.

ability of the form of the constituent structure given the headword, and the probabilities of sub-constituents. High accuracy of 90.1% precision/recall on average has been reported on the English parsers evaluated on Penn Treebank[4]. For Chinese, a few efforts were made. Zhou Qiang proposed a statistics-based Chinese parser[6], in which different kinds of statistics extracted from treebank were combined. High accuracy was achieved, but it was evaluated on a small corpus. Zhou Ming proposed a block-based dependency parser[7] for unrestricted Chinese text, which integrated phrase structure approach and dependency parsing approach successfully and achieved satisfactory initial results.

However, all above works are focused on syntax. Very limited work has been done with semantic parsing for domain-unrestricted text in all languages. For English, a large semantic annotated corpus (FrameNet)[8] was built, and some work on the corpus has been reported, while for Chinese few works was reported at present. But Semantic parsing is an important task for natural language processing (NLP), because understanding the meaning of sentences, even paragraphs and documents is one of the main goals of NLP, and this cannot accomplish without the semantic parsing. Furthermore, Chinese is a language quite different from English, for it is meaning-combined and word-order-free language. So semantic parsing is more important to Chinese NLP.

In this paper, a novel parser based on semantic dependency is presented. The parser can discover the semantic dependency relations in a sentence, which are used to represent the meaning and structure of the sentence. A large corpus annotated with semantic dependency, Semantic Dependency Net (SDN), was built[9]. The parser is trained and evaluated on it. Various experiments are carried out to analyze the performance of the parser. Although the research of the parser is still in its early stage, it has shown a promising performance. The accuracy reached as high level as the Chinese syntactic parser[7].

This parser has three notable features. First, it parses the semantic structure of sentence directly, avoiding the stage of parsing syntactic structure. Second, it is a data-driven parser, for no hand-crafted rules are needed unlike phrase structure parsers. In our model, every word is permitted to be dependent on any other word in a sentence. Finally, the probability of relation between words is trained on annotated corpus. All knowledge used in the parser is automatically acquired from the corpus.

In the following, the annotated corpus used in training and evaluating the parser is briefly introduced in section 2. Then the statistical model and algorithm are described in section 3 and 4 in detail. Some experimental results with analysis are given in section 5, and detailed error analyses are presented in section 6. Finally, concluding remarks and future research directions are given in section 7.

2 Semantic dependency net

A large corpus annotated with semantic dependency was built[9]. The corpus consisted of about 1,000,000 words (about 1,500,000 Chinese characters). The texts were selected from the news of *People Daily*, and its domain covers the politics, economy, science, sports, etc. All words in the corpus were tagged with the semantic classes. All sentences were annotated with semantic dependency relations, which could represent Chinese syntactic and semantic structures at the same time.

2.1 Semantic classes

The definition of Semantic classes follows *Dictionary of Synonymous Words* (*Tong Yi Ci Ci Lin*)[10]. All these semantic classes are organized as a tree, which has three levels. The first level contains 18 classes, the second level contains 101 classes, and the third level contains 1434 classes. Since the project of tagging semantic classes was completed in 1998, the annotated corpus has been used for several years in the researches on automatic semantic class tagging. Now, high accuracy of 92.73% has reached[11] on tagging semantic classes.

2.2 Semantic dependency relationship

In our tagging scheme, dependency grammar is used to represent the meaning and structure of a sentence, and conceptions of headword and lexical information are stressed. Let S be the sentence composed of words tagged with semantic classes, $S=\{\langle w_1,s_1\rangle,\langle w_2,s_2\rangle,\cdots,\langle w_N,s_N\rangle\}$. Then, a list of semantic dependency relations[12] is defined as: $SRL=\{SR(1),SR(2),\cdots,SR(N)\}$, where $SR(i)=(H_i,R_i)$. SR stands for semantic relation. $SR(i)=(H_i,R_i)$ states that the H_i-th word is the headword to the i-th word with semantic relation R_i. If the word j is the sentential head, then $SR(j)=(H_j,R_j)=(-1,\text{“}Kernel\ word\text{”})$.

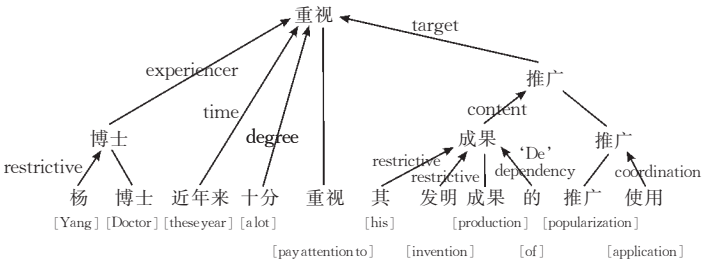
For example, a sentence is shown in Figure 1 (a). The semantic dependency relation list and semantic dependency tree are shown in Figure 1 (b) and (c) respectively.

Chinese(w_i/s_i): 杨/Dd15 博士/Ae13 近年来/Call 十分/Ka01 重视/Gb21 其/Aa04 发明/Hc05 成果/Da14 的/Kd01 推广/Ie01 使用/Hj28
English: These years, Doctor Yang pays a lot of attention to the popularization and application of his invention.

(a) The sentence tagged with semantic classes

i		SR(i)		
Modifier(i)		HeadWord(H_i)		Semantic Relation (R_i)
Index	Word	Index	Word	
1	杨/Yang	2	博士/Doctor	限定/Restrictive
2	博士/Doctor	5	重视/pay attention to	经验者/Experiencer
3	近年来/these years	5	重视/pay attention to	时间/Time
4	十分/a lot	5	重视/pay attention to	程度/Degree
5	重视/pay attention to	−1	−1	核心成分/Kernel word
6	其/his	8	成果/production	限定/Restrictive
7	发明/invention	8	成果/production	限定/Restrictive
8	成果/production	10	推广/popularization	内容/Content
9	的/of	8	成果/product	"的"字依存/'De'dependency
10	推广/popularization	5	重视/pay attention to	目标/Target
11	使用/application	10	推广/popularization	连接/Coordination

(b) The sentence annotated with semantic dependency



(c) The semantic dependency tree of the sentence, headwords are linked with bold lines, and modifier words are linked with arrow lines

Figure 1　An overview of the representation used by the model

The tag set of dependency relations contains 70 relations[9], which include 59 semantic relations, 9 syntactic relations and 2 special relations. Most of semantic relations are selected from the set of semantic relations defined in HowNet[13], for instance, *Agent*, *Patient*, *Possessor*, *Possession*, *Experiencer*, *Content*, *Time*, *TimeIni*, *TimeFin*, *Location*, *LocationIni*, *LocationFin*, *Cause*, *Purpose*, and so on. Syntactic relations are used to annotate the special structures that do not have exact sense in terms of semantics, for instance, '*De*' *dependency*, *Tense&Voice*, *Preposition*, and so on. In addition, there are two special relations: "*Kernel word*" is to indicate the headword of a sentence, and "*Failure*" is to indicate the word that cannot be annotated with dependency relations because the sentence is not completed.

3　Statistical model

3.1　Parsing procedure

How to parse a sentence with dependency grammar will be described before discussing the statistical model and algorithm. To begin parsing, pairs of neighbor words that are dependent are merged firstly to extend larger constituents, and these new constituents are labeled with the headwords and their merging operations (For convenience, they are called as the headword and the last operation of the constituents, respectively). Then the new constituents are merged with other dependent neighbor constituents or words, and so on, until a constituent spanning the whole sentence is created. The merging operation can be written in the format of (Semantic Relation, left/right), where "Semantic Relation" is the relation between the two merged words, and it can be any relation defined in section 2.2. "Left/right" indicates the origin of the headword, that is, "Left/right" means the headword comes from the left or right node. Especially, *NULL* operation is defined to describe the last operation for leaves of parse tree. For example, the merging procedure can be described as a binary tree in Figure 2. In the binary semantic dependency tree, leaf node "Fa Ming/invention" is firstly merged with leaf node "Cheng Guo/production" to create a new constituent labeled with "Cheng Guo/production" and (*Restrictive*, right). Then the node is merged with "Qi/his", and so on.

Sometimes two words are directly dependent

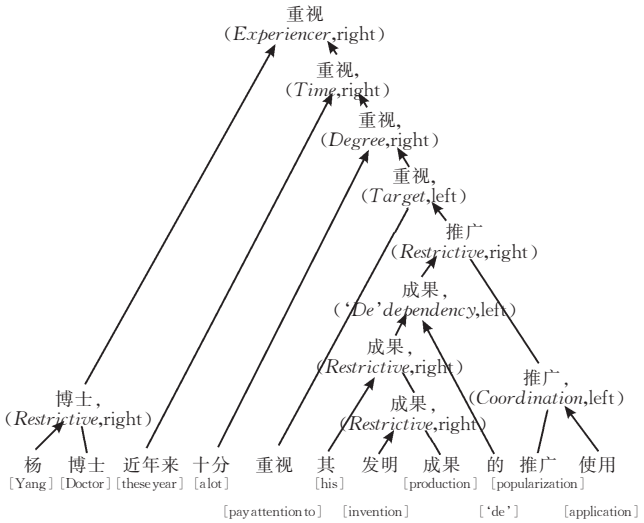


Figure 2 One of binary semantic parse trees in Example 1. Headwords are linked with bold lines, and modifier words are linked with arrow lines

on another word at the same time, that is, two words are in the same level of the semantic dependency tree and they both are adjacent to the headword. For example, "Fa Ming/invention" and "de/of" are both directly dependent on their headword "Cheng Guo/production". Their merging sequence is arbitrary, so more than one binary tree can be mapped to one semantic dependency tree. Figure 2 is only one of them. But a binary semantic dependency tree can be converted to one and only one semantic dependency tree, because a series of merging operations can build a unique semantic dependency tree.

3.2 Statistical model

The aim of our parser is to take a sentence tagged with semantic classes as its input (for example Figure 1(a)) and to produce a semantic dependency tree as its output (Figure 1(c)). This parser is based upon a statistical model, which assigns a probability to every candidate semantic dependency tree DT of a sentence S. The parser searches for the semantic dependency tree that maximizes the probability.

The probability of semantic dependency tree DT of sentence S can be defined as

$$P(DT|S)=\arg\max_{\Theta_{DT}} P(T|S) \quad (1)$$

where Θ_{DT} is the set of all binary semantic dependency trees that can be converted to dependency tree DT. Because one binary tree can be mapped only one dependency tree, to search the dependency tree with the maximum probability is equivalent to search the binary tree with the maximum probability. The optimal binary tree T^* corresponds to the optimal semantic dependency tree DT^*.

$$P(DT^*|S)=P(T^*|S)=\max_{\Omega_D} P(DT|S)=\max_{\Omega} P(T|S) \quad (2)$$

where Ω_D, Ω are the set of all possible semantic dependency trees and the set of all possible binary semantic dependency trees, respectively. The probability of binary tree T can be defined as the product of the probabilities of all operations that extend the tree. For an N-word sentence, N operations are needed to build the semantic dependency tree.

The process of building a semantic dependency tree can be described as Figure 3. q_k is the k-th operation to be taken by the parser, and T_{k-1} is the prefix structure that is built by operations $q_1, q_2, \cdots, q_{k-1}$. Especially, T_0 is a tree that has a node $\langle w_1, s_1\rangle$. The probability of operation q_k is only related to prefix structure T_{k-1} and sentence S, that is, $P(q_k|T_{k-1},S)$. So the probability of binary semantic dependency tree is

$$P(T|S)=\prod_{k=1}^{N}[P(q_k|T_{k-1},S)] \quad (3)$$

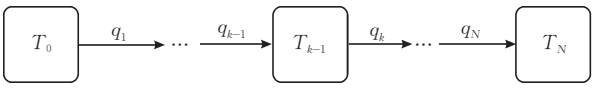


Figure 3 Building a semantic dependency tree

where $q_k \in \mathcal{R}$, $\mathcal{R}$ is the operation set, and it includes all merging operation and "*NULL*" operation("*NULL*" operation means we do nothing.). In building the tree in Figure 2, for example, operation sequence is (*Restriction*, right) between first and second word, (*Restriction*, right) between seventh and eighth word, ..., (*Experiencer*, right) between node "博士, (*Restriction*, right)" and node "重视, (*Time*, right)".

Here, we assume the operation is only determined by the two nodes that are to be merged, and two kinds of information will help to predict the next operation: the headwords $P(q_k|\langle w_{l_k}, s_{l_k}\rangle, \langle w_{g_k}, s_{g_k}\rangle)$ and last operations $P(q_k|r_{l_k}, r_{g_k})$ of the two nodes. l_k, g_k are the left node and right node to be operated, w_i, s_i are the word and semantic class of the headword of the node i, and r_i is last operation of node i, the operation by which the node i is created.

In our parser, the two kind of statistical models are combined, and the score of semantic dependency tree is:

$$Score(T|S)=\sum_{k=1}^{N}[(1-\lambda)\log P(q_k|\langle w_{l_k}, s_{l_k}\rangle, \langle w_{g_k}, s_{g_k}\rangle)+\lambda\log P(q_k|r_{l_k}, r_{g_k})] \quad (4)$$

where λ is a interpolation weight, and in the following experiments $\lambda=0.3$.

Here $P(q_k \mid \langle w_{l_k}, s_{l_k}\rangle, \langle w_{g_k}, s_{g_k}\rangle)$ is called as dependency bigram, and $P(q_k \mid r_{l_k}, r_{g_k})$ as relation bigram. Although there are two conditions in the above two conditional functions, they are still called 'bigram', because if operation decisions are viewed as Markov chain, and current decision is at the state t, then both conditions have been determined at the preceding state $t-1$.

3.3 Parameter estimation

3.3.1 Dependency bigram

Let V be the vocabulary of all words seen in training data, and T be the tag set of semantic classes. Any word pair $\langle w_i, s_i\rangle\langle w_j, s_j\rangle$, $w_i, w_j \in V, s_i, s_j \in T$ in the vocabulary can be merged by any operation $q_k \in \mathcal{R}$, and the probability is estimated in the same way as [8].

$$P(q_k \mid \langle w_i, s_i\rangle\langle w_j, s_j\rangle) = \frac{C(q_k, \langle w_i, s_i\rangle, \langle w_j, s_j\rangle)}{C(\langle w_i, s_i\rangle, \langle w_j, s_j\rangle)} \tag{5}$$

where $C(\langle w_i, s_i\rangle, \langle w_j, s_j\rangle)$ is the number of times that $\langle w_i, s_i\rangle$ and $\langle w_j, s_j\rangle$ are seen in the same sentence in training data. $C(q_k, \langle w_i, s_i\rangle, \langle w_j, s_j\rangle)$ is the number of times that $\langle w_i, s_i\rangle$ and $\langle w_j, s_j\rangle$ are seen in the same sentence and merged by the operation q_k.

In a sentence, some words may be not merged directly, for example "bo shi/doctor" and "jin nian lai/these years" in Figure 1 (c). The equation (5) implies that the probability of no-merge operation is taken into account by the model, although it is not directly multiplied to the equation (4). The sum of the probabilities of all merging operations and that of no-merge operation is 1.

3.3.2 Relation bigram

Relation bigram describes the operations sequence during building binary semantic dependency tree, and it is related to merging procedure. However, one semantic dependency tree can be mapped to more than one binary semantic dependency tree, so some additional rules must be defined to select a unique binary tree in training. Considering the characteristics of Chinese language, a simple binarization scheme (shown in Figure 4) is taken. When more than one modifier words are in the same level of the dependency tree, the words following the headword are firstly merged to the headword, and then the preceding words. This simple binarization scheme may be not proper in representing the sentence structure in some cases, for example, the binary tree converted from the dependency tree in Figure 1 according to this scheme is not the same with that one shown in Figure 2, which we preferred to. But the binarization scheme is simple and effective, and experiment 3 (in section 5.3) showed that relation bigram trained by the method improves the performance significantly.

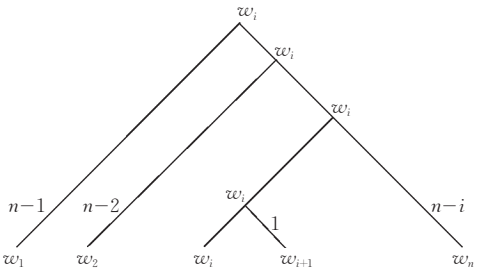


Figure 4　Binarization scheme. w_i is the headword to the constituent, and others are modifier words. The merging sequence is labeled on the edge

The operation can be described by relation bigram:

$$P(q_k \mid r_i, r_j) = \frac{C(q_k, r_i, r_j)}{\sum_{q \in \mathcal{R}} C(q, r_i, r_j)} \tag{6}$$

where $q_k, q, r_i, r_j \in \mathcal{R}$, and $C(q_k, r_i, r_j)$ is the number of the times that the operation q_k takes place between two nodes, whose last relations are respectively r_i and r_j. $\sum_{q \in \mathcal{R}} C(q, r_i, r_j)$ is the summation of all kinds of operations seen in the training data that take place between the those two nodes.

3.4 Sparse data

3.4.1 In dependency bigram

Data Sparseness is a serious problem in the domain-unrestricted statistical parsers. In the word-based dependency bigram, there are about 1,800,000,000,000 parameters to be estimated, while 30,000 pairs of $(q_k, \langle w_i, s_i\rangle, \langle w_j, s_j\rangle)$ 3-tuple are seen twice, and 200,000 pairs are seen only once. In our parser, deleted-interpolation strategy is used to smooth the parameter. When parameter is estimated, not only the lexical information but also semantic class information is concerned. In the corpus, words are tagged with hierarchical semantic classes that have three levels. Therefore, word-based parameters can be interpolated with the parameters based on three levels semantic class respectively. By the interpolation, we make full use of the semantic class information, which is also proved to be effective and robust on predicting the dependency relations in the following experiments. Firstly, four functions are defined:

$$E_1 = \frac{\eta_1}{\delta_1}, \quad E_2 = \frac{\eta_2}{\delta_2}, \quad E_3 = \frac{\eta_3}{\delta_3}, \quad E_4 = \frac{\eta_4}{\delta_4} \tag{7}$$

where:

$$\delta_1 = C(w_i, w_j), \quad \delta_2 = C(s_i^3, s_j^3),$$
$$\delta_3 = C(s_i^2, s_j^2), \quad \delta_4 = C(s_i^1, s_j^1),$$
$$\eta_1 = C(q_k, w_i, w_j), \quad \eta_2 = C(q_k, s_i^3, s_j^3),$$
$$\eta_3 = C(q_k, s_i^2, s_j^2), \quad \eta_4 = C(q_k, s_i^1, s_j^1),$$

and s^1, s^2, s^3 stand for the first-level, second-level, and third-level semantic classes, respectively. As we have described in section 2.1, the first level contains 18 classes, the second level 101 classes, and the third level 1434 classes, so E_2, E_3, E_4 will be trained more sufficiently. We smooth the dependency probability using interpolation with the formula:

$$P(q_k|\langle w_i, s_i\rangle, \langle w_j, s_j\rangle) = \lambda_1 \cdot E_1 + (1-\lambda_1)(\lambda_2 \cdot E_2 + (1-\lambda_2)(\lambda_3 \cdot E_3 + (1-\lambda_3) \cdot E_4)) \quad (8)$$

Parameters $\lambda_1, \lambda_2, \lambda_3$ are computed with a simpler approach, which is reported to perform well in [14].

$$\lambda_n = \frac{\delta_n}{\delta_n + C \cdot \sigma_n + B}, \quad n = \begin{cases} 1, & \text{at word level} \\ 2, & \text{at the third semantic level} \\ 3, & \text{at the second semantic level} \end{cases} \quad (9)$$

where $\sigma_n = \sum_{q_k \in \mathcal{R}} h(\eta_n)$, $h(\eta_n)$ is an indicator function, and $h(\eta_n) = \begin{cases} 1, & \eta_n = 1 \\ 0, & \eta_n \neq 1 \end{cases}$. σ_n represents the diversity of the distribution of E_n. The larger σ_n is in equation (9), the smaller the λ_n is in equation (8). B and C are constants. In our parser, constants B and C are both equal to 1.

3.4.2 In Relation bigram

Like dependency bigram, parameters in relation bigram model are smoothed with the similar method, while equation (7)—(9) is replaced by (10)—(12):

$$E'_1 = \frac{\eta'_1}{\delta'_1}, \quad E'_{23} = \frac{\eta'_2 + \eta'_3}{\delta'_2 + \delta'_3}, \quad E'_4 = \frac{\eta'_4}{\delta'_4} \quad (10)$$

where:

$$\delta'_1 = C(r_i, r_j), \quad \delta'_2 = C(r_i), \quad \delta'_3 = C(r_j), \quad \delta'_4 = \sum_{q_k} C(q_k),$$

$$\eta'_1 = C(q_k, r_i, r_j), \quad \eta'_2 = C(q_k, r_i), \quad \eta'_3 = C(q_k, r_j), \quad \eta'_4 = C(q_k).$$

$$P(q_k|r_i, r_j) = \lambda'_1 \cdot E'_1 + (1-\lambda'_1)(\lambda'_{23} \cdot E'_{23} + (1-\lambda'_{23}) \cdot E'_4) \quad (11)$$

$$\lambda'_n = \begin{cases} \dfrac{\delta'_n}{\delta'_n + C' \cdot \sigma'_n + B'}, & n = 1 \\ \dfrac{\delta_2 + \delta_3}{\delta'_2 + \delta'_3 + C' \cdot (\sigma'_2 + \sigma'_3) + B'}, & n = 23 \end{cases} \quad (12)$$

4 Parsing algorithm

4.1 The dynamic programming algorithm

In operation set, $NULL \in \mathcal{R}$ is the starting operation for a sentence; and $Kernel\ word \in \mathcal{R}$ is the ending operation. The central data structure of the algorithm is a dynamic programming array: $\pi[i, j, q, hw]$ holds the maximum score for a constituent, which is merged by operation q, span words $i \cdots j$, and have headword hw. The goal of the search is to find $\pi[1, n, \text{“}Kernel\ word\text{”}, hw]$. Back-pointers in the dynamic programming array can be used to store the path leading to the highest score tree.

The basis of the algorithm is an iterative procedure:

1. Initial step:

$\pi[i, i, NULL, w_i] = 0$, for $i = 1, 2, \cdots, n$, where w_i is the i-the word in the sentence;

2. Iterative step:

$$\pi[i, j, q, hw] = \max_{i \le m \le j, r_l, r_r \in \mathcal{R}} \{\pi[i, m, r_l, hw_l] + \pi[m+1, j, r_r, hw_r] + (1-\lambda)\log P(q|hw_l, hw_r) + \lambda \log P(q|r_l, r_r)\},$$

for $1 \le i \le j \le n$, where hw_l, hw_r are the headwords of two merged nodes, r_l, r_r are their last relations.

In iterative step, if two proposed constituents span the same set of words, have the same headword and the same last operation, the lower score constituent can be safely discarded. The dynamic programming algorithm is simpler than chart parser, which is widely used in [3], [7], etc.

4.2 Pruning

To speed up parsing procedure, pruning strategy is taken into account. All $\pi[i, j, q, hw]$ with the same i, j are stored in the same stacks and ranked by the score. If the score of first constituent in the stack is P_h, all other constituents in the same stack must have score greater than P_h/β for some constant β. The constituent whose score is lower than the threshold will be discarded. At the same time, if the rank of a constituent in the stack is lower than some threshold, the maximum stack length, it is also discarded. In our parser, β is equal to $e^3 = 20$, and maximum stack length is 20.

5 The experiment

Semantic Dependency Net (SDN) corpus was used in training and testing the parser. About 110,000 sentences (approximately 830,000 words) were used in training, and about 22,300 sentences (approximately 170,000 words) were used in testing. In following experiments, sentences with manually tagged semantic classes were taken as input.

Measures are defined to evaluate the performance of our model.

$$Relation\ Precision(RP) = \frac{number\ of\ correct\ semantic\ dependency\ 3\text{-}tuples}{number\ of\ all\ semantic\ dependency\ 3\text{-}tuples} \quad (13)$$

$$Crossing\ Brackets = number\ of\ semantic\ dependency\ 3\text{-}tuples\ in\ which\ headword\ is\ not\ correct\ given\ the\ modifier\ word \quad (14)$$

where Semantic dependency 3-tuple is (Modifier, Headword, Semantic Relation). Then, other measures are defined: *CB* is average number of crossing brackets per sentence, 0 *CB* is the percentage of sentences with zero crossing brackets, and 2 *CB* is the percentage of sentences with ≤ 2 crossing brackets.

5.1 Experiment 1: Unsmoothed parser

First, simple tests on training and testing data are carried out. In the experiment, Equation (5) is used to estimate the dependency bigram based on word, semantic class in various levels, separately. That is, only one λ in equation (8) is equal to 1, and the others are 0. Equation (6) is used to estimate the relation bigram. Table 1 shows the performance of word based, third class based, second class based, and first-class-based parsers.

Table 1 Results of unsmoothed parsers on training and testing data

	RP	
	training data(%)	testing data(%)
Word based ($\lambda_1=1$)	93.64	56.07
Third class based ($\lambda_2=1$)	77.76	60.41
Second class based ($\lambda_3=1$)	47.24	46.48
First class based ($\lambda_4=1$)	30.64	29.67

On training data, word based parser performs perfect, which proves the correctness and effective of our model. However, the performance of semantic class based parsers reduced dramatically. One reason for these is that lexical information is more predictable than semantic class. Another reason is that the definitions of semantic classes and semantic dependency relation may be not consistent, because their definitions come from different theories of semantics, but the degree how it influences on the performance has to be tested in further experiments.

On testing data, the performance of unsmoothed parsers is rather poor. The performance of word-based parser is worse than that of third-class-based one, and both of them are worse than those on trainging data, but first-class and second-class-based parsers performed similarly on training and testing data. Sparse data is the major reason for the differences, because there are too many parameters to be estimated in word-based model.

5.2 Experiment 2: Smoothed parser

In this experiment, parameters are smoothed with equation (8), (11) and the interpolation parameters are computed with equation (9), (12). The semantic class information was added to equation (8) step by step from the third semantic class to the first class. All these experiments are carried on testing data.

Table 2 Results of smoothed persers different method

Configuration	*RP*(%)	*CB*	0 *CB*(%)	2 *CB*(%)
Word based, unsmoothed	56.07	2.20	35.50	71.27
Smoothed by third class	64.66	1.82	43.47	73.07
Smoothed by third and second class	67.51	1.60	47.60	73.63
Smoothed by third, second, and first class	67.56	1.60	47.62	73.53

The results show that the more information is added to the model, the higher performance was achieved. The performance of the word-based parser smoothed by three levels semantic classes is similar to that of the syntactic parser[7]. Zhou Ming's parser is evaluated on the 1,400 sentences from *China Daily*, and its analysis precision is 67.7%. However, these two results are not comparable. The tagging scheme and tag set are quite different between Zhou Ming's parser and our parser. The former is of Syntax, and they only defined 11 kinds of blocks and 17 kinds of dependency relations; the latter is of Semantics, and we defined 70 kinds of dependency relations, so our task is much more difficult. On the other hand, the part-of-speech labels are automatically tagged in Zhou Ming's parser, while the semantic classes are tagged by manual in our parser.

5.3 Experiment 3: Contribution of relation bigram

The contribution of relation bigram to performance is tested in this experiment. When performance only using dependency bigram is tested, we let λ be 1 in equation (4).

Table 3 The contribution of various components of the model

Configuation	*RP*(%)	*CB*	0 *CB*(%)	2 *CB*(%)
Dependency bigram	64.21	1.83	41.53	71.26
Dependency bigram and relation bigram	67.56	1.60	47.62	73.53

Table 3 shows that relation bigram improves the performance. We think the reason is that last relations can represent some semantic and syntactic functions of the constituent, which are helpful in predicting the next operation.

6 Error analysis

There are two kinds of sources of error. One

is from the mismatch between training data and testing data. Some sentence patterns, frames, and collocations in testing data are few or never seen in training data. Generally, this kind of errors can be reduced and solved by adding more training data. The other is from model, that is, the model cannot distinguish different sentence structures due to its simplification. In the parser, we assume the merging operation is only related to the two nodes that are to be merged, which must lead to errors to some extent. Three major error types of this kind are given in Figures 5～7.

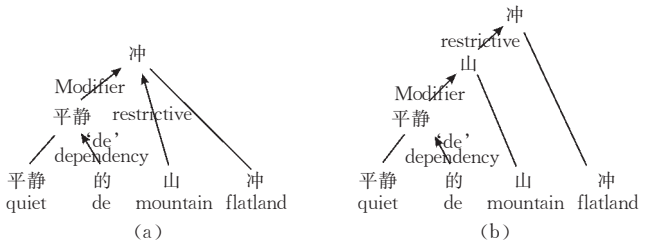


Figure 5 NP-NP modification error. Starred example is correct, and alternative is parse error

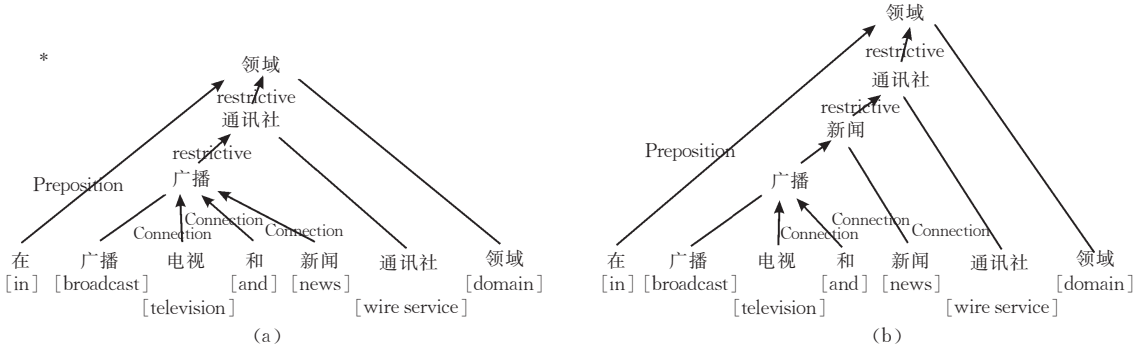


Figure 6 Coordination attachment error. Starred example is correct, and alternative is parse error

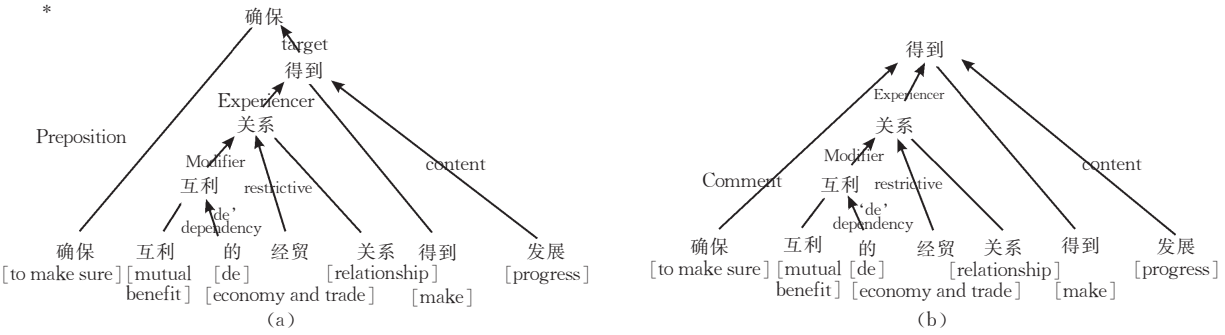


Figure 7 Relative clause error. Starred example is correct, and alternative is parse error

From the examples, we can see that the parser neglects some structure constrains, because we assume that the merging operations are dependent only on the two merged nodes. Further improvements are needed to solve this kind of error. More structure features should be added to model in future parsers, for example, valence of verbs, the surrounding words of the two merged node, sub nodes of the merged node, and so on.

7 Conclusion

We have presented a semantic parser, which could discover the semantic dependency relations in a sentence. The parser is evaluated on a large corpus annotated with semantic dependency, and shows good performance and robustness in parsing unrestricted text. In addition, the parser is easy to build, because the parameter of the parser can be automatically learned from semantic dependency annotated corpus, which is much easier to build than the corpus annotated with phrase structure. This method is also applicable to parse semantic relations for other languages.

Much further work will be needed. Many more features of sentences could be integrated in the statistical parsing model, which we have discussed in section 6. Then semantic class auto-tagging should be integrated in the parser.

References

1 Marcus M., Santorini B. *et al*.. Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics, 1993, 19(2): 313～330

2 Collins M.. Three generative, lexicalised models for statistical parsing. In: Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics, Madrid, 1997, 16～23

3 Ratnaparkhi A.. A linear observed time statistical parser based on maximum entropy models. In: Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing, Brown University, Providence, 1997, 1～10

4 Charniak, Eugene. A maximum-entropy-inspired parser. In: Proceedings of the North American Chapter of the Association for Computational Linguistics, 2000, 132～139

5 Roark B.. Probabilistic top-down parsing and language modeling. Computational Linguistics, 2001, 27(2): 249～276

6 Zhou Qiang. A statistics-based Chinese parser. In: Proceedings of the 5th Workshop on Very Large Corpora, Beijing, 1997, 4～15

7 Zhou Ming. A block-based dependency parser for unrestricted Chinese text. In: Proceedings of the 2nd Chinese Language Processing Workshop attached to ACL2000, Hong Kong, 2000, 78～84

8 Baker, Collin F., Fillmore, Charles J., Lowe, John B.. The Berkeley FrameNet project. In: Proceedings of the COLING-ACL, Montreal, 1998, 86～90

9 Li Ming-Qin, Li Juan-Zi, Dong Zhen-Dong, Wang Zuo-Ying, Lu Da-Jin. Building a large Chinese corpus annotated with semantic dependency. In: Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing, Sapporo, 2003, 84～91

10 Mei Jia-Ju, Zhu Yi-Ming Gao Yun-Qi, Yin Hong-Xiang ed.. Tong Yi Ci Ci Lin. Shanghai: Shanghai Cishu Publisher, 1983 (in Chinese)
(梅家驹, 竺一鸣, 高蕴琦, 殷鸿翔编. 同义词词林. 上海: 上海辞书出版社, 1983)

11 Zhang Jian-Ping. A study of language model and understanding algorithm for large vocabulary spontaneous peech recognition [Ph. D. dissertation]. Department of Electronic Engineering, Tsinghua University, Beijing, 1999

12 Li Juan-Zi. Chinese statistcal parser based on semantic dependencies. Tsinghua Science and Technology, 2002, 7(6): 591～595

13 Dong Zhen-Dong, Dong Qiang. Construction of a knowledge system and its impact on Chinese research. Contemporary Linguistics, 2001, (3): 33～44

14 Chen S., Goodman J.. An empirical study of smoothing techniques for language modeling. In: Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, Santa Cruz, California, 1996, 310～318

LI Ming-Qin, born in 1977, Ph. D. candidate. Her research interests include speech recognition, natural language processing, and semantic parsing.

LI Juan-Zi, born in 1964, Ph. D., associate professor. Her main research interests include natural language understanding, Chinese information processing, knowledge discovery and management on the Web.

WANG Zuo-Ying, born in 1935, professor and Ph. D. supervisor. His research interests focus on signal processing and pattern recognition.

LU Da-Jin, born in 1928, professor, Ph. D. supervisor. His current research interests include radar signal processing and speech signal processing.