Natural Language Engineering **6** (2): 163–181. Printed in the United Kingdom

© 2000 Cambridge University Press

Acquisitions and applications of structure preference relations in Chinese

QIANG ZHOU

State Key Laboratory of Intelligent Technology and Systems, Department of Computer Science, Tsinghua University, Beijing 100084
e-mail: zhouq@s1000e.cs.tsinghua.edu.cn

FUJI REN

Faculty of Information Sciences, Hiroshima City University, Japan
e-mail: ren@its.hiroshima-cu.ac.jp

(*Received 10 November 1999; revised 17 March 2000*)

Abstract

In this paper, we propose a new ambiguity representation scheme; Structure Preference Relation (SPR), which consists of useful quantitative distribution information for ambiguous structures. Two automatic acquisition algorithms, the first acquired from a treebank, and the second acquired from raw texts, are introduced, and some experimental results which prove the availability of the algorithms are also given. Finally, we introduce some SPR applications in linguistics and natural language processing, such as preference-based parsing and the discovery of representative ambiguous structures, and propose some future research directions.

1 Introduction

Ambiguity is the common phenomenon in natural language. When we use a Context-Free Grammar (CFG) rule set to parse natural language sentences, we may encounter many ambiguous phenomena due to the limitation of rule descriptions and complexity of real texts. To deal with these problems, many useful methods have been proposed, including:

- To enhance the description capacity of the CFG rules by introducing complex feature formalisms, such as lexical function grammar and general context-free grammar.
- To improve the adaptation of the CFG rules by introducing probability information, such as probabilistic context-free grammar.

But surprisingly, little work has been devoted to describing the ambiguous structures themselves.

Usually, ambiguous structures are found and summarized by linguists. When they analyze the language facts, they may find that some structure instances can have

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

different syntactic explanation in sentences. As the similar instances occur more and more in real texts, the ambiguous structures can be drawn from them accordingly. Although this induction method may be effective, the computational linguists still encounter many difficulties when they try to utilize such knowledge, because of the absence of objective distribution data about it.

In this paper, we propose a new description formalism for ambiguous structures; Structure Preference Relation (SPR). By integrating the occurrence frequencies of different ambiguous combination instances in real texts with the ambiguous structures themselves, it provides useful quantitative distribution information for natural language processing and linguistic research.

In the following, section 2 gives the basic concept of SPR and its distribution characteristics in parse trees. Section 3 introduces two different SPR acquisition algorithms: (1) to find the SPR from a treebank; and (2) to discover SPR from raw texts; and gives some results of SPR acquisition experiments. Section 4 discusses some applications of SPR, including preference-based parsing in natural processing and the discovery of representative ambiguous structures in linguistics research. Section 5 compares our work with the work of others in the area of sentence disambiguation. Section 6 draws conclusions.

2 Basic concepts

Give a context-free grammar $G = (T, N, R, S)$, where:

- T is the set of terminals, i.e. the Part-of-Speech (POS) tagset,
- N is the set of non-terminals, i.e. the syntactic tagset,
- R is the set of CFG rules with the forms $LHP \rightarrow RHP$, where $LHP \in N$, represents the left-hand part of a rule, $RHP \in \{T \cup N\}_*$, represents the right-hand part of the rule, and
- S is the starting symbol.

We can define a Structure Preference Relation (SPR) as follows:

Definition 1
A *First Constituent Set* (*FCS*) comprises all constituents at the first position of *RHP*.

Definition 2
A *Last Constituent Set* (*LCS*) comprises all constituents at the last position of *RHP*.

Definition 3
An *Interesecting Constituent* (*IC*) is an element in the intersection of *FCS* and *LCS*. If it is at the first position of a *RHP*, then the other constituents of the *RHP* are called *IC's sufcontext* (*SufIC*). Otherwise, they are called *IC's pre-context* (*PreIC*). Here, $IC \in \{T \cup N\}$, and $PreIC, SufIC \in \{T \cup N\}_*$.

For example, if we have such two rules in *R*, P1 → AB and P2 → BC, then we will get a *IC* B, a *PreIC* A and a *SufIC* C.

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

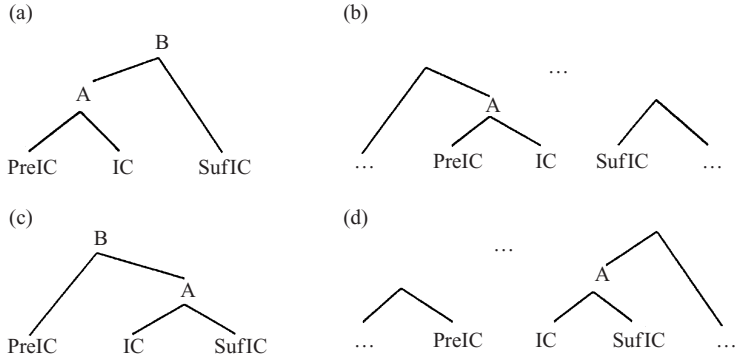


Fig. 1. Distribution of *IS* instances in parse tree.

Definition 4
An *Intersecting Structure* (*IS*) is the constituent string composed by a *PreIC*, an *IC*, and a *SufIC*, i.e. $IS = \{PreIC\ IC\ SufIC\}$.

For example, $IS = \{\text{v }\hat{}\text{v n}^1\}$ is a common *IS* in Chinese.

According to this definition, the intersecting structures are actually the same as the well known ambiguous structures in natural language processing. In the parses of different sentences, they can be instantiated as different combination structures.

Definition 5
The *Left Combination Instances* (*LCIs*) of *IS* are defined as the constituent structures that span *PreIC* and *IC* in *IS*; similarly, the *Right Combination Instances* (*RCIs*) of *IS* are defined as the constituent structures that span *IC* and *SufIC* in *IS*.

If we further analyze the combination structure of the overall *IS* in parse trees, we can find two kinds of different intersecting structures: (1) Compact *IS*, and (2) Loose *IS*.

Definition 6
A *Compact Intersecting Structure* (*Compact IS*) can be finally reduced as a constituent (a non-terminal node) in the parse tree; a *Loose Intersecting Structure* (*Loose IS*) cannot be reduced as a constituent (a non-terminal node) in the parse tree.

Figure 1 shows detailed distribution of different *ISs* in parse trees, where cases (a) and (c) represent *compact ISs*, and cases (b) and (d) represent *loose ISs*.

Definition 7
A *Structure Preference Relation* (*SPR*) stem is defined as a triple $\langle IS, LF, RF\rangle$, where *IS* is an intersecting structure, and *LF* and *RF* are the occurrence frequencies of *LCI* and *RCI* in real texts, respectively.

In fact, a SPR stem integrates two kinds of different linguistic knowledge:

[1] We labeled *IC* in *IS* with symbol '^' (similarly hereinafter). v – verb, n – noun.

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

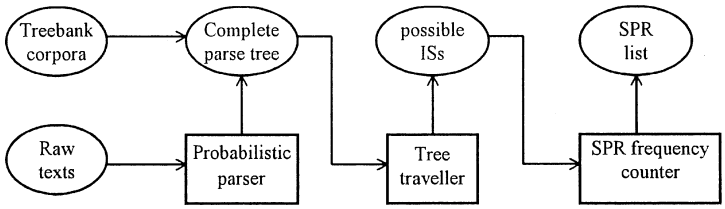


Fig. 2. Overview of SPR acquisition system.

(1) The rule-based structural ambiguity description string, represented as IS.
(2) The statistics-based combination instance distribution, represented as LF and RF.

Its main properties,

- complete description of rule ambiguities: all possible ISs can be enumerated in a certain CFG rule set;
- quantitative distribution of ambiguous phenomena in real texts,

can allow its application in syntactic disambiguation, i.e. the IS information can be used to identify possible ambiguous structures, and the LF and RF information can be used for local-context-based disambiguation.

3 SPR acquisition system

Figure 2 shows the overall structure of our SPR acquisition system. We try to extract useful *SPR* data from treebanks or raw texts by using different acquisition algorithms. Three main processing stages of the algorithms are:

(1) To build a (complete) parse tree.
(2) To travel the parse tree so as to find all possible *ISs*.
(3) To count the SPR preference frequencies for the suitable *ISs*.

In the following sections, we introduce two different *SPR* acquisition algorithms: (1) to search *SPR* data from treebanks; (2) to discover *SPR* data from raw texts, and their experimental results, especially the performance analysis of the *SPR* discovery algorithm. Section 3.1 discusses basic *SPR* searching schemes. Section 3.2 discusses *SPR* discovery strategies. Section 3.3 gives some experimental results.

3.1 SPR search algorithm

If we had treebanks, where every sentence has been annotated with correct syntactic tree, the SPR acquisition task would become much easier. We would then focus on developing an efficient tree travelling algorithm to find all possible intersecting structures in the parse tree quickly.

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

3.1.1 Tree travelling algorithm

A parse tree can be travelled top-down from its root node R. A basic travelling algorithm may be:

Tree-Travel (R)

- If R is a non-leaf node, then
  - Find all possible *compact ISs*;
  - Find all possible *loose ISs*;
  - For every subnodes Ni of R, recurrently call *Tree-Travel* (Ni);
- Else, return.

3.1.2 Find possible compact ISs

Consider a subtree $PH(RP_1\ RP_2 \ldots RP_n)$ in the parse tree, where PH is the root node of this subtree, $RP_i, i \in [1,n]$ are its subnodes. If RP_1 is a non-leaf node, and its subtree structure is $RP_1(RP_{11}\ RP_{12} \ldots RP_{1m})$, then the constituent string $RP_{11}\ RP_{12} \ldots RP_{1m}\ RP_2 \ldots RP_n$ can be looked on as a possible left combination instance (just like figure 1(a)), where RP_{1m} is the intersecting constituent. Similarly, if RP_n is a non-leaf node, and its subtree structure is $RP_n(RP_{n1}\ RP_{n2} \ldots RP_{nk})$, then the constituent string $RP_1\ RP_2 \ldots RP_{n-1}\ RP_{n1}\ RP_{n2} \ldots RP_{nk}$ can be looked on as a possible right combination instance (just like figure 1(c)), where RP_{n1} is the intersecting constituent.

3.1.3 Find possible loose ISs

Consider a subtree $PH(RP_1\ RP_2 \ldots RP_n)$ in the parse tree, where PH is the root node of this subtree, $RP_i, i \in [1,n]$ are its subnodes. Search all adjacent left constituents A_j of PH in the tree. If $(A_j\ PH)$ is not a prefix of a constituent in the tree, i.e. there are no such subtree structures $X(A_j\ PH \ldots)$, then constituent string $A_j\ RP_1\ RP_2 \ldots RP_n$ can be looked on as a possible right combination instance (just like figure 1(d)), where RP_1 is the intersecting constituent. Similarly, search all adjacent right constituents C_j of PH in the tree. If $(PH\ C_j)$ is not a suffix of a constituent in the tree, i.e. there are no such subtree structures $X(\ldots PH\ C_j)$, then constituent string, $RP_1\ RP_2 \ldots RP_n\ C_j$ can be looked on as a possible left combination instance (just like figure 1(b)), where RP_n is the intersecting constituent.

3.2 SPR discovery algorithm

Although a treebank provides convenience for SPR acquisition, it is costly and time-consuming to build a large-scale treebank. Nowadays, some large scale English treebanks, such as the Penn treebank (Marcus et al. 1993), have been built. However, there are still no available large scale Chinese treebanks. So it is very important to explore automatic SPR acquisition techniques from raw texts (annotated with correct tokenization and POS tags).

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

Three key issues in this respect are:

(1) how to get the suitable syntactic trees (or forest) of the raw sentences quickly;
(2) how to efficiently find all possible ISs among different parse trees;
(3) how to accurately compute the expectation frequencies of different ISs in the trees (or forest).

The following sections will give some detailed resolving strategies.

3.2.1 *Preprocess the raw texts*

First, we develop a raw text preprocessor by combining efficient parsing techniques used in the following processing tools:

(1) A statistics-based Chinese parser, which tried to get the best parse tree for every Chinese sentence with correct segmentation and POS tagging information, by using different kinds of statistics extracted from treebank (Zhou 1997b, 1996b).
(2) A Chinese grammar inference tool, which can automatically acquire Chinese Probabilistic Context-Free Grammar (PCFG) rules from raw texts (Zhou and Huang 1998).

Then, we use the preprocessor to parse raw sentences and generate their complete parse trees. Different from the correct parse tree annotated on a treebank sentence, a complete parse tree comprises all possible parse trees for a raw sentence, represented as the Packed Shared Forest (PSF) formation proposed by Tomita (1986). Moreover, every constituent in the PSF is labeled with Inner Probability (IP) and Outer Probability (OP), whose computation makes use of probabilities of Chinese PCFG rules (Zhou and Huang 1998).

3.2.2 *Improved tree travelling algorithm*

The change of the parse formalism from the correct parse tree to PSF brings in a reduction in the searching efficiency of the basic travelling algorithm. Because there are many duplicate constituents among the packed node list, they will cause many redundant searching operations on the same constituent node. To resolve this problem, we add a simple control structure into the basic algorithm: a Searched Node List (SNL). After we finish the searching operation of a constituent node, we put it into a SNL. Only the nodes that are not in the SNL can be searched next. Therefore, an improved travelling algorithm can be built as follows:

Improved-Tree-Travel (*R*)

- If *R* is a non-leaf node and *R* is not in SNL, then
  - Find all possible *compact ISs*;
  - Find all possible *loose ISs*;
  - Add *R* into SNL;
  - For every subnodes *Ni* of *R*, recurrently call *Tree-Travel* (*Ni*);
- Else, return.

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

3.2.3 Compute the expectation frequency of ISs

The distribution frequency of an IS in an annotated correct parse tree is certain, i.e. it has the absolute frequency 1 for every IS instance found. However, the case will be different for PSF, because we cannot know which parse in PSF is the correct one. The only information available under this status is the expectation frequency $E(\mathrm{A} \rightarrow \lambda)$ of a matched constituent $\mathrm{A} \rightarrow \lambda$ in the PSF, which can be computed according to the formula: $E(\mathrm{A} \rightarrow \lambda) = IP(\mathrm{A} \rightarrow \lambda) * OP(\mathrm{A} \rightarrow \lambda)/P(S)$, where $P(S) = \sum_{R \in RS} IP(R \rightarrow \lambda)$, and RS is the set of root nodes in PSF that can generate the complete sentence. Some detailed information about them can be found in Zhou and Huang (1998).

Based on the constituent expectation frequencies, we can compute the distribution frequencies of *IS* as follows:

(1) *For compact ISs*: their distribution frequencies are determined by the expectation frequencies of the major constituent and minor constituent in the intersecting constituent. For example, for a *LCI* of *IS* (figure 1(a)), we have $E(\{\mathrm{PreIC\ IC\ Suf\,IC}\}) = E(\mathrm{A} \rightarrow \mathrm{PreIC\ IC}) \cdot E(\mathrm{B} \rightarrow \mathrm{A\ Suf\,IC})$.

(2) *For loose ISs*: their distribution frequencies are determined by the expectation frequencies of the minor constituent and its left (or right) adjacent constituent. For example, for a *LCI* of *IS* (figure 1(b)), we have $E(\{\mathrm{PreIC\ IC\ Suf\,IC}\}) = E(\mathrm{A} \rightarrow \mathrm{PreIC\ IC}) \cdot E(\mathrm{Suf\,IC})$, where $E(\mathrm{Suf\,IC}) = \sum E(\mathrm{Suf\,IC} \rightarrow \lambda)$.

3.3 Experimental evaluation

The experimental corpora were extracted from a small Chinese treebank developed at Peking University (Zhou 1996b). It comprises 5573 Chinese sentences, 64,430 Chinese words and 89,492 Chinese characters. All sentences in the corpus were annotated with correct syntactic trees, part-of-speech tags and tokenization information. Therefore, the following different kinds of SPR acquisition experiments can be easily carried out on them:

- Experiment 1: to search SPR stems from treebank sentences.
- Experiment 2: to discover SPR stems from raw sentences.

In experiment 1, we obtained an SPR list with 4021 different stems (SPR list A). In experiment 2, we obtained an SPR list with 6091 different stems (SPR list B1). By setting a frequency threshold $\mu = 0.25$ and excluding all SPR stems whose total frequency are less than μ, we obtained an SPR list with 3832 different stems (SPR list B2).

To test the performance of the SPR discovery algorithm, we divide SPR list A into three subsets, according to the Total Frequency (TF)[2] of the SPR stem:

(1) High frequency set: $\mathrm{TF} > 10$
(2) Middle frequency set: $1 < \mathrm{TF} \leq 10$
(3) Low frequency set: $\mathrm{TF} \leq 1$

[2] The Total Frequency (TF) of a SPR stem is defined as the sum of its Left combination Frequency (LF) and Right combination Frequency (RF), i.e. $\mathrm{TF} = \mathrm{LF} + \mathrm{RF}$.

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

Table 1. *SPR recalls of the automatic discovery algorithm*

	TF > 10	1 < TF ≤ 10	TF ≤ 1	Total
Number of stems in SPR list A	785	1852	1384	4021
Number of stems occurred in SPR list B2	773	1591	840	3204
SPR recall ($\mu = 0.25$)	0.9847	0.8591	0.6069	0.8361
Number of stems occurred in SPR list B1	783	1789	1199	3771
SPR recall ($\mu = 0$)	0.9975	0.9660	0.8663	0.9378

Then, the SPR recall of the automatic discovery algorithm can be computed on these three subsets. To sum up, we can determine the reliability and availability of the terms of the SPR discovery algorithm, based on the comparison of SPR data automatically discovered with the SPR data extracted from treebanks in recall and distribution of common ISs.

Table 1 shows the results. As can be seen, almost all high frequency and about 85% middle-frequency SPR stems can be discovered automatically. For low-frequency SPR stems, the recall is about 61%, but it is also in conformity with the linguistic facts. As most of these SPR stems are special structures in a treebank, they should have lower expectation frequencies in PSF. Therefore, most of them can be excluded by setting a threshold $\mu = 0.25$. Even under this threshold condition, the overall recall of the automatic discovery algorithm reaches 84%. It indicates that SPR data acquired automatically are reliable.

To analyze the distribution difference of different SPR stems in lists A and B2, we respectively selected 10 SPR stems with the highest total frequencies in each list, and computed their Left combination Probability (LP) and Right combination Probability (RP).[3] Table 2 shows the results. Due to the diversity of the frequency computation methods of two SPR acquisition algorithms, there are some differences in the LF and RF values. However, the same ISs co-occurring in these two sets have almost the same LP and RP values, except IS 'v ^v n'. It indicates that SPR data acquired automatically reflect the virtual linguistic facts better.

4 Some applications of SPR

4.1 Preference-based parsing

In PCFGs, probabilities are associated with the grammar rules. This introduces new opportunities for disambiguation. For instance, among the different parsing results of an ambiguous structure, if one has a higher probability and another has a much lower probability, then we can remove the less likely constituent safely, because it will probably not be part of the correct parse. This technique is called *preference-based parsing* (Church and Mercer 1993).

[3] LP = LF/TF, RP = RF/TF.

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

Table 2. *Top 10 stems with the highest total frequencies in two SPR lists*

SPR list A					SPR list B2				
IS[a]	LF	RF	LP	RP	IS	LF	RF	LP	RP
d ^vp wD vp	487.00	3.00	0.99	0.01	m ^q n	469.07	0.00	1.00	0.00
m ^q n	476.00	0.00	1.00	0.00	d ^v np	0.08	365.96	0.00	1.00
r ^vp wD vp	441.00	3.00	0.99	0.01	d ^vp wD vp	361.61	4.00	0.99	0.01
v ^v n	140.00	292.00	0.32	0.68	r ^vp wD vp	337.54	5.94	0.98	0.02
d ^v vp	24.00	377.00	0.06	0.94	v ^v n	147.66	134.24	0.52	0.48
v ^v np	203.00	182.00	0.53	0.47	v ^v np	151.87	128.95	0.54	0.46
d ^v np	3.00	343.00	0.01	0.99	d ^v n	0.00	272.72	0.00	1.00
pp ^v v	0.00	326.00	0.00	1.00	r ^q n	272.29	0.00	1.00	0.00
d ^v n	2.00	302.00	0.01	0.99	d ^v vp	17.03	246.64	0.06	0.94
d ^v v	3.00	300.00	0.01	0.99	v ^np wD vp	243.19	0.03	1.00	0.00

[a] The POS and syntactic tags used in this table are briefly described as follows. Some detailed information about our POS and syntactic tagsets can be found in Zhou (1996b).

[POS tags]: d – adverb, wD – punctuation(comma), m – numeral, q – classifier, n – noun, v – verb, r – pronoun.

[Syn tags]: vp – verb phrase, np – noun phrase, pp – preposition phrase, dj – simple sentence.

In fact, the left and right combination probabilities in SPR stems can be looked upon as a kind of good local preference information. In the following sections, we introduce their applications in optimizing our current Chinese probabilistic parser. Section 4.1.1 introduces the basic matching algorithm in the parser. Section 4.1.2 analyzes its problems, and proposes some solving strategies. Section 4.1.3 describes the detailed optimization algorithm. Section 4.1.4 gives some experimental results.

4.1.1 Basic matching algorithm

The Chinese probabilistic parser proposed by Zhou (1997b) tries to parse Chinese sentences through the following three stages: (1) constituent boundary prediction (Zhou 1996a); (2) bracket matching; (3) statistical disambiguation. Among these processes, the bracket matching algorithm forms the connection bridge between the preceding and following stages. Its basic operation is to match left-hand bracket with suitable right-hand brackets. The algorithm scans the input sentences from left to right, and generates the complete parse trees (or forest) of them bottom-up, under the restriction of constituent boundary tags and matching restriction regions discovered in the preceding stage.

Three basic control structures of the algorithm are as follows:

(1) Bracket Matching Stack (BMS), which saves all information on the constituent boundary tags and matching restriction regions of the input sentence, with a function just like the graph-structure stack in Tomita's algorithm (Tomita 1986).

(2) Packed Shared Forest (PSF), which saves all of the constituents generated by bracket matching operations.

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

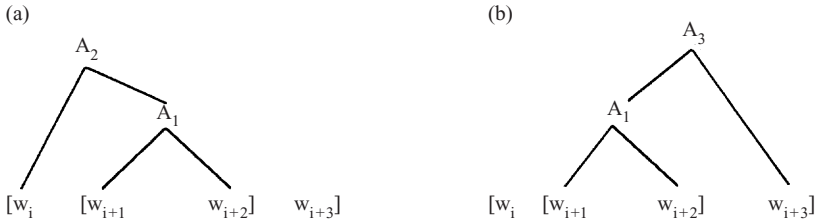


Fig. 3. The matching results of a segment.

(3) Pending Constituent List (PCL), which saves all of the pending constituents for any further matching process, just like the agenda in the chart parsing algorithm (Winograd 1983).

Some detailed descriptions of the algorithm can be found in Zhou (1997a).

4.1.2 Problems and resolving strategies

The problem of the current matching algorithm is its lower efficiency due to many redundant constituents generated during the bracket matching process.

Consider such a segment in the input sentence after constituent boundary prediction: $[w_i\ [w_{i+1}\ w_{i+2}]\ w_{i+3}]$. After we get the matched constituent A_1, two possible matched constituents, A_2 and A_3, can be generated through bracket matching operations (figure 3).[4] However, only one of them will be the correct parse, the other may be redundant. In an input sentence with many words, a redundant matched constituent may cause many unnecessary constituents to be generated later, and increase parsing perplexity. So an efficient approach is to remove them as early as possible.

According to the idea of preference-based parsing, if we can determine the preference combination relations of constituent A_1 from among local context w_i and w_{i+3}, we will easily optimize the current matching algorithm by safely pruning the less likely matching operations in the local context.

Many researchers have used word association techniques, such as the mutual information measure, in preference-based parsing. For example, to disambiguate prepositional phrase attachments in English, Hindle and Rooth (1993) used the relative strength of association of the preposition, with verbal and nominal heads estimated on the basis of distribution in an automatically parsed corpus. To deal with the problem of data sparseness, Basili, Pazienza and Velardi (1993) used semantic tags to cluster similar words.

We try to use some simple combination frequencies, such as the LF and RF data in SPR stems, for local-context-based disambiguation. It has the following advantages:

(1) High coverage: all possible local ambiguous structures in the sentences can be found by matching with the IS strings in the SPR list.

[4] If we assumed that (t_1A_1) and (A_1t_{i+3}) are the correct RHP structures in PCFG rules.

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

(2) Quick adaptation: all the ambiguous distribution data can be automatically acquired from different kinds of raw texts by using the SPR discovery algorithm.

The next section will introduce its implementation in our optimized matching algorithm.

4.1.3 Optimized matching algorithm

The implementation of the optimized matching algorithm should deal with the following questions:

(1) How to determine the local context for a pending constituent?
(2) How to guarantee the reliability of the applied SPR data?
(3) How to integrate different optimization strategies into former matching algorithm efficiently?

The following sections give detailed discussions on the above points.

Determine the local context. The local context of a pending constituent A is made up by its adjacent left and right matching stems. Due to the left-to-right parsing scheme of the current matching algorithm, we can easily get the left matching stems by searching the PSF and BMS, because all of them have been generated before we get the constituent A. The difficulty is how to get the right matching stems, because under the current parsing state, the matching operation in the right region of A has not been processed.

Consider a sentence segment $[w_i\ [_A\ w_{i+1}\ w_{i+2}]\ [w_{i+3}\ w_{i+4}]$. For the pending constituent A, its real right matching stems may be the constituent B generated by the matching operation upon word w_{i+3} and w_{i+4}. However, this constituent cannot be obtained at present, so we cannot carry out local optimization upon constituent A. To resolve this problem, we propose a delaying selection scheme, whose basic idea is as follows.

First, we set a Delayed Constituent List (DCL). If we encounter a pending constituent whose right matching stems can't be determined under the current parsing state, we move it from PCL to DCL. Then we continue to parse the sentence from left-to-right. Whenever a new matched constituent is generated, we must check whether there are suitable right matching stems for every delayed constituent in DCL. If there are, then we can get it from DCL and insert it into PCL again. Thus, through the coordination of PCL and DCL, we can guarantee that every pending constituent can acquire a suitable local context for optimization.

Select suitable SPR threshold. The SPR threshold (β) is set to restrict the application scope of SPR list, i.e. only the SPR stems in the list that satisfy the formulas $|LP - RP| > \beta$ can be used for local optimization. Under such a restriction, we can implement optimization strategies as follows: if $LP > RP$, then select a left combination structure in priority, and discard the right combination structure; otherwise,

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

1. Get a pending constituent A from PCL;
2. Search its right matching stems (RMSs) in BMS and PSF;
3. If cannot find, put A into DCL and goto 8;
4. Get one of its left matching stems (LMSs) in BMS and PSF;
5. Check the local preference of A under context {LMS A RMS}, controlled by SPR threshold β;
6. If left preference, then carry out matching operation: [LMS A]; else goto 7;
7. If there are any other left matching stems, then goto 4; else got 8;
8. If PCL is not empty, then goto 1, else return;

Fig. 4. Basic procedure of the optimized matching algorithm.

select a right combination structure in priority, and discard the left combination structure.

In general, if we-use a tight (large) threshold, and remove only these constituents that are much less probable under local contexts, our parser will run only slightly faster than those without optimization (i.e. $\beta = 1$), while performance measure such as precision and recall will remain virtually unchanged. On the other hand, if we use a loose (small) threshold, and remove constituents that are almost as probable as the prior constituents under local contexts, then we can get a considerable speedup, but at the cost of a reduction in accuracy. So the key issue is to find the trade-off between accuracy and time. In section 4.1.4, we introduce an experiment to select suitable SPR thresholds.

Basic optimization procedure. By integrating the above optimization strategies into our former matching algorithm (Zhou 1997a), we develop an optimized matching algorithm. Figure 4 shows its basic control procedure, where steps 2 and 3 realize the delaying selection scheme, and steps 5 and 6 implement local-preference-based optimization by adding up priority judgements through SPR data under certain thresholds.

4.1.4 Experimental evaluation

The experiment is designed to test the application effect of SPR information to trade-off between accuracy and time in our Chinese probabilistic parser. Its training and test sets are extracted from the same treebank used for SPR acquisition (section 3.1). They comprise 5071 and 506 Chinese sentences, respectively.

For measuring parse accuracy, we use the following metrics defined by Zhou (1996b, 1997b):

(1) Matched Recall (MR).
(2) Matched Precision (MP).
(3) Crossing Brackets (CBs).
(4) Labeled Precision (LP).

For measuring parse time, we use the following metrics:

(1) Total of matched constituents (MCTotal): all the matched constituents generated through matching operations.

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

Table 3. *Experimental results of the optimized parser with different SPR thresholds*

β	MR (%)	MP (%)	CBs	LP (%)	MCSum	PTSum	CPU time (Sec.)
1	89.93	90.01	0.87	95.17	1.19×10^4	6.53×10^6	228
0.9	89.87	90.00	0.87	95.21	8.55×10^3	1.91×10^5	184
0.8	90.10	90.24	0.84	95.22	8.39×10^3	9.44×10^4	166
0.7	90.08	90.21	0.85	95.27	8.25×10^3	6.92×10^4	179
0.6	90.31	90.44	0.82	95.31	8.13×10^3	4.46×10^4	173
0.5	90.40	90.53	0.82	95.31	8.08×10^3	4.02×10^4	170
0.4	90.57	90.70	0.80	95.30	8.03×10^3	3.57×10^4	158
0.3	90.50	90.64	0.81	95.32	7.94×10^3	3.11×10^4	166
0.2	90.46	90.58	0.81	95.29	7.93×10^3	3.10×10^4	176
0.1	90.48	90.60	0.81	95.24	7.82×10^3	2.03×10^4	171

(2) Total of parse trees (PTTotal): all the complete parse trees[5] in PSF, whose computation formula can be found in Zhou (1996b).
(3) CPU time: the elapsed CPU time to parse the input sentences (three processing stages).

To select suitable SPR threshold, we first parse the sentences in the test set by setting 10 different thresholds from 0.1 to 1. Table 3 shows the experimental results, where the first row ($\beta = 1$) is actually the performance data of the former parser without local optimization. It can be used as a baseline for analysis.

Notice that as the SPR threshold decreases (from 0.9 to 0.1), the parsing speed increases slightly, shown by the decrease in MC Total, PT Total and CPU time metrics. This tendency is consistent with our expectation. As the restriction upon SPR stems is loosened, more and more SPR stems can be used for local optimization, so that less likely matching operations can be removed.

However, for parsing accuracy, we can see an interesting tendency in Table 3. As the SPR threshold (β) decreases, it increases slowly, until it reaches a summit (when $\beta = 0.4$), then decreases gradually. The reason can be analyzed as follows.

When β is larger, the applied SPR stems show a stronger local preference. They can be used to safely prune the ambiguous structures with less likely probabilities under local contexts as early as possible. Thus, the following statistics-based disambiguation algorithm can efficiently select a best parse tree from the remaining constituents with more likely probabilities under local contexts; but when β decreases beyond a certain limit, many correct constituents will be pruned due to the application of some less reliable SPR stems under local contexts. Therefore, the parsing accuracy will be reduced.

Therefore, it will be suitable to select $\beta = 0.4$ as the SPR threshold in our optimized matching algorithm. Under this condition, we get such encouraging optimization results: the CPU time, total of matched constituents and total of parse trees are

[5] A complete tree covers all words of an input sentence.

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

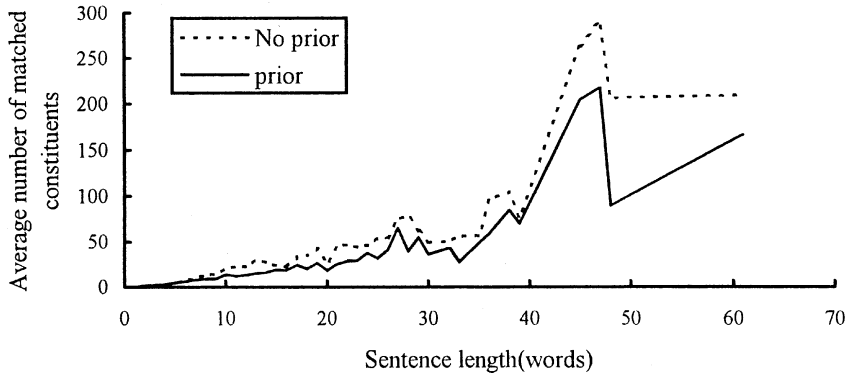


Fig. 5. Average number of matched constituents vs. sentence length.

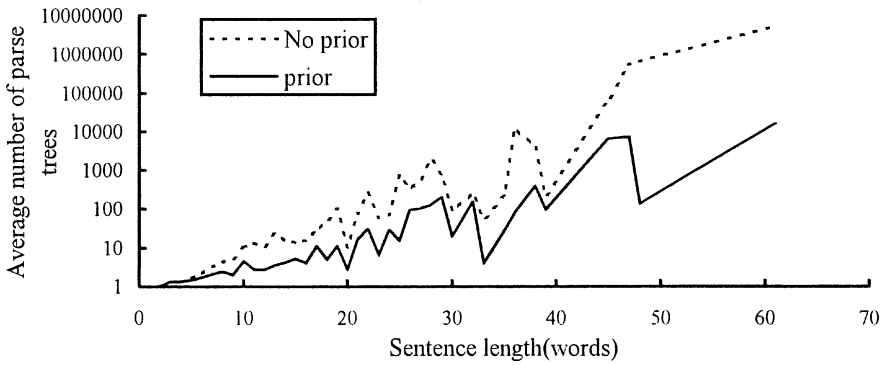


Fig. 6. Average number of parse trees vs. sentence length.

reduced 30.7%, 32.5% and 99.5%, respectively, and the parsing accuracy also increases slightly.

To compare the optimization effect of sentences with different lengths, we compute the average number of MCTotal and PTTotal in test sets. Figures 5 and 6 show the results, where $\beta = 0.4$. As can be seen, for simple sentences (with no more than 20 words), the optimization algorithm does not show a distinct improvement in parse efficiency, but for complex sentences (with more than 20 words), the improvement in parse efficiency is obvious, because in complex sentences, more local contexts can be optimized by SPR information. The removal of one constituent under local context in the early stages of tree generation will cause the removal of thousands of subtrees comprising the constituent in the final PSF of the complex sentence. Table 4 indicates the obvious optimization effects of the longest sentence (with 61 words) in the test set.

4.2 Discover representative ambiguous structures

Intuitively, the representative ambiguous structures in natural language may have the following characteristics:

(1) They are the common structures in real texts.

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

Table 4. *The different parsing results of a complex sentence* (*with 61 words*) *before and after optimization*

β	MR (%)	MP (%)	CBs	LP (%)	MCSum	PTSum	CPU time (Sec.)
No prior	80.00	76.60	9	94.44	2.09×10^2	5.22×10^6	4
Prior	82.22	78.72	8	94.59	1.66×10^2	1.61×10^4	4

(2) Every ambiguous instance of them will occur with almost the same frequency.
(3) It will be very difficult to disambiguate them correctly, if we only use some simple linguistic knowledge, such as part-of-speech information under local context.

For example, the prepositional phrase attachment problem describes a representative ambiguous structure 'V NP PP' in English.

Now, by using SPR data, we can give a quantitative definition for representative ambiguous structure in natural language processing.

Definition 8
Given a frequency threshold α and a relative probability threshold β, we can call such an intersecting structure in a SPR stem a *Representative Ambiguous Structure* (*RAS*), if it satisfies the following conditions: (1) $\mathrm{LF} + \mathrm{RF} > \alpha$; (2) $|\mathrm{LP} - \mathrm{RP}| \leq \beta$.

The frequency threshold can be empirically set according to the size of training corpora, but the relative probability threshold is especially set so that it can be the same as the SPR threshold defined in section 4.1. In fact, the SPR threshold can be looked on as a measure of disambiguation difficulty under local context. Only those ambiguous structures that can be easily disambiguated according to its local POS and syntactic tags information due to their high priority combination can be used in the optimized procedure of preference-based parsing. Thus, all the excluded ambiguous structures will just comprise the set of RAS. In future, if some detailed disambiguation knowledge for RAS, such as lexical-based or semantics-based knowledge, can be automatically learned and applied in the current preference-based parser, we can expect that the parsing performance will improve further.

By setting $\alpha = 10$ and $\beta = 0.4$, we get 56 RASs from SPR list A (in section 3.3), which accounts for about 1.39% of different SPR stems in it. Compared with some lists of Chinese syntactically ambiguous structures summarized by linguists (Zhu 1980; Huang 1985), we find that they cover most common ambiguous structures in contemporary Chinese texts.

Table 5 lists the top 10 representative ambiguous structures among them with the highest frequencies. They indicate some disambiguation difficulties in Chinese syntactic parser:

(1) *Predicate-complement structure*
Due to the complex syntactic functions of Chinese verbs, there are two possible parsing results for the IS 'v1 ^v2 n'. One makes 'v2' the complement of 'v1',

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

Table 5. *Top 10 RASs found in the Chinese treebank with 5573 sentences*

ISs	LF	RF	LP	RP
v ^v n	140.00	292.00	0.32	0.68
v ^v np	203.00	182.00	0.53	0.47
dj wD ^vp wD vp	69.00	121.00	0.36	0.64
v ^np vp	111.00	53.00	0.68	0.32
dj wD ^dj wD dj	44.00	85.00	0.34	0.66
v ^n uJDE n	50.00	51.00	0.50	0.50
dj wD ^vp wD dj	49.00	36.00	0.58	0.42
v ^v vp	33.00	34.00	0.49	0.51
v ^v r	19.00	41.00	0.32	0.68
d ^a uJDI	17.00	29.00	0.37	0.63

and the 'n' the object of the predicate-complement structure (or verb phrase) 'v1 v2'. The other makes 'n' the object of 'v2', and the verb phrase 'v2 n' the verbal object of 'v1'. To disambiguate, it may need some lexical and collocation information on Chinese verbs. The similar ISs in Table 5 are 'v ^v np', 'v ^np vp', 'v ^v vp' and 'v ^v r'.

(2) *Serial sentences*
A characteristic of the Chinese language is that several simple sentences (or verb phrases) can be combined as a complex sentence without any conjunctions. Therefore, some semantic information may be needed to determine the correct relations among these simple sentences; for example, to disambiguate the ambiguous structures 'dj wD ^vp wD vp', 'dj wD ^dj wD dj', and 'dj wD ^vp wD dj' in Table 5.

(3) *Prepositional attribute clause*
The attribute in a Chinese sentence is prepositional. We can easily find its right boundary by identifying the auxiliary word '*De*' in most cases. However, it is very difficult to find its correct left boundary due to there not being any obvious starting words, like wh-nouns or wh-adverbs in English. This characteristic brings in difficulties of disambiguating the IS 'v ^n uJDE n'.

Though developing a searching tool, we can automatically extract all RAS instances from raw texts. Based on these, many detailed disambiguation rules for RAS, especially lexical-based and semantics-based information, can be efficiently learned manually and automatically. Figure 7 shows its overall structure.

5 Discussion

In statistical parsing, probability plays an important role for disambiguation. A good probabilistic language model must deal with the following questions: (1) How to define the meaning of probability; (2) How to acquire the probabilistic distributions; (3) How to apply them for disambiguation. The different answers to these questions may result in different disambiguation efforts.

The Probabilistic Context-Free Grammar (PCFG) is a commonly used statistical

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

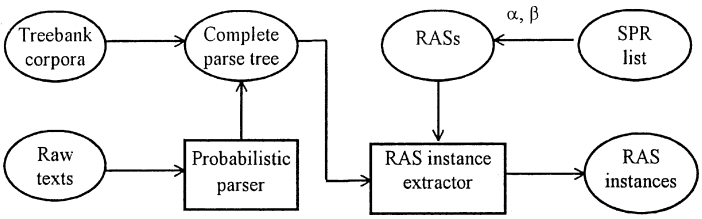


Fig. 7. Overview of a RAS instance extractor.

language model. It extends the standard Context Free Grammar (CFG) formalism by assigning a probability to each production. The rule probabilities can be acquired through two different kinds of inference strategies: (1) to directly obtain useful syntactic statistics from treebanks (Charniak 1996); (2) to automatically acquire syntactic knowledge from raw texts by an iterative algorithm, such as the commonly used EM (Expectation Maximization) algorithm (Lari and Young 1990; Carroll and Charniak 1992; Pereira and Schabes 1992; Shih et al. 1993). When we use the PCFG rules to parse sentences, every parse tree can be assigned a probabilistic score through the production of the scores of all its subtrees with the probability of the PCFG rule to generate the tree. Thus, the disambiguation can be achieved by selecting the parse tree with the maximum score. This is an overall disambiguation process. Some similar disambiguation models include statistical decision trees (Magerman 1995), Bigram dependence model (Collins 1996), and the maximum entropy model (Ratnaparkhi 1997).

Different to PCFG rules, the structure preference relations try to describe the preference combination phenomena of different ambiguous structures in local context. Their special formalisms (i.e. the ambiguous structures integrated with the occurrence frequencies of different combination instances in real texts) determine that they have a different disambiguation capability with PCFG rules. The tentative experiments (in section 4.1.4) show they can be used to prune away lots of unnecessary constituents in local context, and the optimization of parsing efficiency can be obtained through the SPR-based local-context disambiguation and the PCFG-based overall disambiguation.

In fact, according to the different frequency distribution, the ambiguous structures described in SPR lists can be divided into two sets, i.e. those which can (set 1) or cannot (set 2) be reliably disambiguated on the basis of local part-of-speech and syntactic tags. The idea here is that more detailed knowledge of the syntactic or lexical features will enable the model to better disambiguate those special ambiguous structures in set 2 in the future. A representative example is the prepositional phrase attachment problem in English. If PP is always attached to NP in the local context (V NP PP), the accuracy is 59% (Collins and Brooks 1995). When some detailed lexical or semantic knowledge, such as the word associations (Hindle and Rooth, 1993) or semantic tags (Basili et al. 1993), and a new statistical model, such as the back-off model (Collins and Brooks 1995), are used, the accuracy can be improved to about 80%.

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

6 Conclusions

In this paper, we introduce the basic concepts of a quantitative description formalism for ambiguous structures, Structure Preference Relation (SPR), propose two different SPR acquisition algorithms, and discuss some SPR applications in preference-based parsing and linguistic research. Some new techniques employed in the paper are summarized as follows:

(1) The method of linking SPR instances with different subtree structures in parses ensures that all possible intersecting structures can be found through a recurrent tree travelling algorithm.
(2) The sentence preprocessor based on statistics-based parsing and grammar inference techniques provides an efficient method to generate the complete parse trees (or forest) upon raw sentences with correct segmentation and POS tag information. The complete parse trees not only provide essential information for the improved tree travelling algorithm to discover possible ISs, but also supply the expectation frequency computation schemes with useful constituent inner and outer probabilities.
(3) The setting of SPR relative probability threshold establishes a quantitative measure of disambiguation difficulty for different ambiguous structures. Accordingly most ambiguous structures can be safely disambiguated through the preference-based parsing technique. Therefore, we can focus on exploring disambiguation knowledge about only the representative ambiguous structures.

Although the SPR acquisition and application experiments showed some encouraging results, the experimental corpora was still too small. In future, we want to use the SPR discovery algorithm in more Chinese raw texts, and generate a new SPR list. Most Chinese representative ambiguous structures and their combination instances can be expected to be automatically acquired from them. Thus, we can further explore the disambiguation rules of RASs by using machine learning and corpus-based techniques. We hope this work can give great impetus to research into Chinese linguistics and natural language processing.

Acknowledgements

The work was supported by the Chinese national natural science foundation (Grant No. 69705005) and Chinese postdoctoral research foundation. The work has also been partly supported by a grant for international Scientific Research (09044179) from the Education Ministry of Japan. Thanks to our anonymous reviewers for their comments.

References

Basili, R., Pazienza, M. T. and Velardi, P. (1993) Semi-automatic extraction of linguistic information for syntactic disambiguation. *Applied Artificial Intelligence* **7**: 339–364.

Carroll, G. and Charniak, E. (1992) Learning probabilistic dependency grammars from labelled text. *AAAI Fall Symposium Series: Probabilistic Approaches to Natural Language* 25–32.

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press

Collins, M. J. (1996) A new statistical parser based on bigram lexical dependencies. *Proc. of ACL-34*, 184–191.

Collins, M. and Brooks, J. (1995) Prepositional phrase attachment through a backed-off model. *Proc. and Third Workshop on Very Large Corpora*, Cambridge, MA, 27–38.

Church, K. W. and Mercer, R. L. (1993) Introduction to the Special Issues on Computational Linguistics Using Large Corpora. *Computational Linguistics* **19**(1): 1–24.

Huang, G. (1985) Ambiguous phrases in contemporary Chinese. *Language Research* **1**: 69–89.

Hindle, D. and Rooth, M. (1993) Structural ambiguity and lexical relations. *Computational Linguistics* **19**(1): 103–120.

Lari, K. and Young, S. J. (1990) The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Compute Speech and Language* **4**(1): 35–56.

Magerman, D. M. (1995) Statistical decision-tree models for parsing. *Proc. ACL-33*, 276–303.

Marcus, M. P., Marcinkiewicz, M. A. and Santorini, B. (1993) Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* **19**(2): 313–330.

Pereira, F. and Schabes, Y. (1992) Inside-Outside reestimation from partially bracketed corpora. *Proc. of ACL-30*, 128–135.

Ratnaparkhi, A. (1997) A linear observed time statistical parser based on maximum entropy models. In: C. Cardie and R. Weischede (editors), *Second Conference on Empirical Methods in Natural Language processing* (EMNLP-2), Somerset, New Jersey.

Shih, H. H., Young, S. J. and Waegner, N. P. (1993) An inference approach to grammar construction. *Computer Speech and Language* **9**(3): 235–256.

Tomita, M. (1986) *Efficient Parsing for Natural Language – A Fast Algorithm for Practical System*. Boston: Kluwer Academic.

Winograd, T. (1983) *Language as a cognitive process. Vol 1, Syntax*. Reading, MA: Addison-Wesley 116–129.

Zhou, Q. (1996a) A model for automatic prediction of Chinese phrase boundary location. *Journal of Software* **7** (Supplement): 315–322.

Zhou, Q. (1996b) *Phrase Bracketing and Annotating on Chinese Language Corpus. PhD. dissertation*, Peking University.

Zhou, Q. (1997a) Implementation of Chinese parsing algorithm based on bracket matching principle. In: L. Chen and Q. Yuan (editors), *Language Engineer*, Beijing: Isinghua University Press, 194–200.

Zhou, Q. (1997b) A statistics-based Chinese parser. *Proc. Fifth Workshop on Very Large Corpora*, Beijing, China, August 18, 4–15.

Zhou, Q. and Huang, C. (1998) An inference approach for Chinese probabilistic context-free grammar. *Chinese Journal of Computers* **21**(5): 385–392.

Zhu, D. (1980) Ambiguous phenomena of Chinese syntax. *Chinese Language* **2**: 81–90.

https://doi.org/10.1017/S1351324900002448 Published online by Cambridge University Press