Acquisitions and Applications of Structure Preference Relations in Chinese

Qiang Zhou*, Fuji Ren†, Changning Huang*

*State Key Laboratory of Intelligent Technology and Systems
Dept. of Computer Science, Tsinghua University, Beijing 100084
zhouq@s1000e.cs.tsinghua.edu.cn

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

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: 1) acquired from treebank, 2) acquired from raw texts, are introduced, and some experimental results which prove the availability of the algorithms are also given. At last, we introduce some SPR applications in linguistics and natural language processing, such as preference-based parsing and discovery of representative ambiguous structures, and propose some future research directions.

KEYWORDS: Structure Preference Relation, Ambiguous Structure, Knowledge Acquisition, Preference-Based Parsing, Corpus.

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 absence 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, rather little work has been devoted to describe 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 different 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 these knowledge because of the absence of objective distribution data about them.

In this paper, we propose a new description formalism for ambiguous structures: Structure Preference Relation(SPR) in the paper. 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 sections, 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 SPR from treebank, 2) To discover SPR from raw texts, and give some results of SPR acquisition experiments. Section 4 discussed some applications of SPR, including the preference-based parsing in natural language processing and the discovery of representative ambiguous structures in linguistics research. The final section 5 are conclusions.

2 Basic concepts

Given 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,
- \( S \) is the starting symbol.

We can define 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 Intersecting Constituent(1C) 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 suf-context(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 \( R \): \( P1 \rightarrow AB \) and \( P2 \rightarrow BC \), then we will get a IC B, a PreIC A and a SufIC C.

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

For example, \( IS = \{v^nv n\} \) is a common IS in Chinese.

According to this definition, the intersecting structures are actually 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, 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) can’t 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 the case a) and c) represent compact ISs, and the case b) and d) represent loose ISs.

---

1 We labeled IC in IS with symbol ‘^’ (similarly hereinafter). v—verb, n—noun.
Definition 7. An Structure Preference Relation (SPR) stem is defined as a triple \(<IS, LF, RF>\), where \(IS\) is an intersecting structure, \(LF\) 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) The rule-based structural ambiguity description string, represented by \(IS\).
2) The statistics-based combination instance distribution, represented by \(LF\) and \(RF\).

2.1 Disambiguation characteristics

3 SPR acquisition system

Figure 2 shows the overall structure of our SPR acquisition system. We try to extract useful SPR data from treebank 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 will 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 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 more easier. Because we can focus on developing an efficient tree traveling algorithm to find all possible intersecting structures in the parse tree quickly.

3.1.1 Tree traveling algorithm

A parse tree can be traveled top-down from its root node $R$. A basic traveling 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 subnode $Ni$ of $R$, recurrently call $Tree-Travel(Ni)$;
- Else, return.

3.1.2 Find possible compact ISs

Consider a subtree $PH(RP_1 RP_2 ... RP_n)$ in the parse tree, where $PH$ is the root node of this subtree, $RP_i$, $i \in [1, n]$ are its subnodes. If $RP_i$ is a non-leaf node, and its subtree structure is $RP_i(RP_{i1} RP_{i2} ... RP_{im})$, then the constituent string: $RP_{i1} RP_{i2} ... RP_{im} RP_2 ... RP_n$ can be looked as a possible left combination instance (just like Figure 1a), where $RP_{im}$ is the intersecting constituent. Similarly, if $RP_n$ is a non-leaf node, and its subtree structure is $RP_n(RP_{n1} RP_{n2} ... RP_{nk})$, then the constituent string: $RP_1 RP_2 ... RP_{n-1} RP_{n1} RP_{n2} ... RP_{nk}$ can be looked as a possible right combination instance (just like Figure 1c), where $RP_{n1}$ is the intersecting constituent.

3.1.3 Find possible loose ISs

Consider a subtree $PH(RP_1 RP_2 ... RP_n)$ 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 not such subtree structures $X(A_j PH ...)$, then constituent string: $A_j RP_1 RP_2 ... RP_n$ can be looked as a possible right combination instance (just like Figure 1d), 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 not such subtree structure $X(... PH C_j)$, then constituent string: $RP_1 RP_2 ... RP_n C_j$ can be looked as a possible left combination instance (just like Figure 1b), where $RP_n$ is the intersecting constituent.
3.2 SPR discovery algorithm

Although 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 Penn treebank[MSM93], have been built. But there are still not any 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).

Two 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

Firstly, 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 sentences with correct segmentation and POS tagging information, by using different kinds of statistics extracted from treebank([ZQ97b], [ZQd96]).
2) A Chinese grammar inference tool, which can automatically acquired Chinese probabilistic context-free grammar(PCFG) rules from raw texts[ZH97].

Then, we use the preprocessor to parse raw sentences and generate their complete parse trees. Different from the correct parse tree annotated on 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 PSF are labeled with inner probability(IP) and outer probability(OP), whose computation makes use of probabilities of Chinese PCFG rules[ZH97].

3.2.2 Improved tree traveling algorithm

The change of the parse formalism from the correct parse tree to PSF brings into the reduction of searching efficiency of the basic traveling 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 in basic algorithm: a Searched Node List(SNL). After we finish the searching operation of a constituent node, we put it into SNL. Only the nodes that are not in SNL can be searched next. Therefore, an improved traveling 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.
### 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. But the case will be different for PSF, because we can’t know which parse in PSF is the correct one. The only information available under this status is the expectation frequency $E(A \rightarrow \lambda)$ of a matched constituent $A \rightarrow \lambda$ in PSF, which can be computed according to the formula: $E(A \rightarrow \lambda) = IP(A \rightarrow \lambda) \cdot OP(A \rightarrow \lambda) / P(S)$, where $P(S) = \sum_{R \in RS} IP(R \rightarrow \lambda)$, $RS$ is the set of root nodes in PSF that can generate the complete sentence. Some detailed information about them can be found in [ZH97].

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 1a), we have: $E(\{PreIC IC SufIC\}) = E(A \rightarrow PreIC IC) \cdot E(B \rightarrow A SufIC)$

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 1b), we have: $E(\{PreIC IC SufIC\}) = E(A \rightarrow PreIC IC) \cdot E(SufIC)$, where $E(SufIC) = \sum E(SufIC \rightarrow \lambda)$

### 3.3 Experimental evaluation

The experimental corpus were extracted from a small Chinese treebank developed in Peking University[ZQd96]. It comprise 5573 Chinese sentences, 64430 Chinese words and 89492 Chinese characters. All sentences in the corpus were annotated with correct syntactic trees, part-of-speech tags and tokenization information. Therefore, the following two 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 the experiment 1, we got a SPR list with 4021 different stems(SPR list A). In the experiment 2, we got a 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 $\mu$, we got a SPR list with 3832 different stems(SPR list B2).

In order 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 SPR stem:

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

Then, the SPR recall of automatic discovery algorithm can be computed on these three subsets. Table 1 shows the results. As can be seen, almost all high-frequency and about 85% middle-frequency SPR stems can be automatically discovered. For low-frequency SPR stems, the recall is about 61%. But it is also in conformity with the linguistic facts. Because most of these SPR stems are special structures in treebank, they should have lower expectation frequencies in PSF. Therefore most of them can be excluded by setting threshold $\mu=0.25$. Even if under this threshold

---

$^2$ 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. $TF = LF + RF$. 

---

6
condition, the overall recall of the automatic discovery algorithm also reaches 84%. It indicates that SPR data automatically acquired are reliable.

Table 1  SPR recalls of the automatic discovery algorithm

<table>
<thead>
<tr>
<th>TF&gt;10</th>
<th>1&lt;TF&lt;=10</th>
<th>TF&lt;=1</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of stems in SPR list A</td>
<td>785</td>
<td>1852</td>
<td>1384</td>
</tr>
<tr>
<td>Number of stems occurred in SPR list B2</td>
<td>773</td>
<td>1591</td>
<td>840</td>
</tr>
<tr>
<td>SPR recall(μ=0.25)</td>
<td>0.9847</td>
<td>0.8591</td>
<td>0.6069</td>
</tr>
<tr>
<td>Number of stems occurred in SPR list B1</td>
<td>783</td>
<td>1789</td>
<td>1199</td>
</tr>
<tr>
<td>SPR recall (μ=0)</td>
<td>0.9975</td>
<td>0.9660</td>
<td>0.8663</td>
</tr>
</tbody>
</table>

To analyze the distribution difference of different SPR stems in list 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). Table 2 shows the results. Due to the diversity of the frequency computation methods of two SPR acquisition algorithms, there are some difference in LF and RF values. But the same ISs co-occurring in these two sets have almost same LP and RP values, except IS “v ^v n”. It indicates that SPR data automatically acquired reflect the virtual linguistic facts better.

To sum up, from the comparison of SPR data automatically discovered with the SPR data found from treebanks in recalls and distribution of common ISs, we can conclude the reliability and availability of the SPR discovery algorithm.

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

<table>
<thead>
<tr>
<th>IS4</th>
<th>LF</th>
<th>RF</th>
<th>LP</th>
<th>RP</th>
<th>IS</th>
<th>LF</th>
<th>RF</th>
<th>LP</th>
<th>RP</th>
</tr>
</thead>
<tbody>
<tr>
<td>d ^vp wD vp</td>
<td>487.00</td>
<td>3.00</td>
<td>0.99</td>
<td>0.01</td>
<td>m ^q n</td>
<td>469.07</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>m ^q n</td>
<td>476.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>d ^v np</td>
<td>0.08</td>
<td>365.96</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>r ^vp wD vp</td>
<td>441.00</td>
<td>3.00</td>
<td>0.99</td>
<td>0.01</td>
<td>d ^v p wD vp</td>
<td>361.61</td>
<td>4.00</td>
<td>0.99</td>
<td>0.01</td>
</tr>
<tr>
<td>v ^v n</td>
<td>140.00</td>
<td>292.00</td>
<td>0.32</td>
<td>0.68</td>
<td>r ^vp wD vp</td>
<td>337.54</td>
<td>5.94</td>
<td>0.98</td>
<td>0.02</td>
</tr>
<tr>
<td>d ^v vp</td>
<td>24.00</td>
<td>377.00</td>
<td>0.06</td>
<td>0.94</td>
<td>v ^v n</td>
<td>147.66</td>
<td>134.24</td>
<td>0.52</td>
<td>0.48</td>
</tr>
<tr>
<td>v ^v np</td>
<td>203.00</td>
<td>182.00</td>
<td>0.53</td>
<td>0.47</td>
<td>v ^v np</td>
<td>151.87</td>
<td>128.95</td>
<td>0.54</td>
<td>0.46</td>
</tr>
<tr>
<td>d ^v np</td>
<td>3.00</td>
<td>343.00</td>
<td>0.01</td>
<td>0.99</td>
<td>d ^v n</td>
<td>0.00</td>
<td>272.72</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>pp ^v v</td>
<td>0.00</td>
<td>326.00</td>
<td>0.00</td>
<td>1.00</td>
<td>r ^q n</td>
<td>272.29</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>d ^v n</td>
<td>2.00</td>
<td>302.00</td>
<td>0.01</td>
<td>0.99</td>
<td>d ^v vp</td>
<td>17.03</td>
<td>246.64</td>
<td>0.06</td>
<td>0.94</td>
</tr>
<tr>
<td>d ^v v</td>
<td>3.00</td>
<td>300.00</td>
<td>0.01</td>
<td>0.99</td>
<td>v ^vp wD vp</td>
<td>243.19</td>
<td>0.03</td>
<td>1.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

3 LP = LF / TF,  RP = RF / TF.
4 The POS and syntactic tags used in this table are briefly describes as follows. Some detailed information about our POS and syntactic tagsets can be found in [ZQd96].

[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.
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 higher probability and another has much more 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[CM93].

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

4.1.1 Basic matching algorithm

The Chinese probabilistic parser proposed in [ZQ97b] tries to parse the Chinese sentences through the following three stages: 1) Constituent boundary prediction[ZQ96], 2) Bracket matching, 3) Statistical disambiguation. Among them, bracket matching algorithm forms the connection bridge between the preceding and following stages. Its basic operation is to matching left hand bracket with suitable right hand bracket. The algorithm scans the input sentences 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 of constituent boundary tags and matching restriction regions of the input sentence, with the function just like graph-structure stack in Tomita algorithm[MT86].
2) Packed Shared Forest(PSF), which saves all constituents generated by bracket matching operations.
3) Pending Constituent List(PCL), which saves all pending constituents await for further coming matching process, just like the agenda in chart parsing algorithm[TW83].

Some detailed descriptions of the algorithm can be found in [ZQ97a].

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 bracket matching process.

Consider such a segment in the input sentence after constituent boundary prediction: \([w_1 \ [w_{i+1} \ w_{i+2}] \ w_{i+3}]\). After we get the matched constituent \(A_1\), two possible matched constituent: \(A_2\) and \(A_3\) can be generated through bracket matching operations(Figure 3)\(^5\). But only one of them will be the correct parse, the other may be redundant. In an input sentence with many words, a redundant

\(^5\) If we assumed that \((t_i \ A_1)\) and \((A_1 \ t_{i+3})\) are the correct RHP structures in PCFG rules.
matched constituent may cause many unnecessary constituents to be generated later and increase parsing perplexity. So an efficient approach is to remove them as earlier as possible.

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

Many researchers used word association techniques, such as 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, R. Basili, M. T. Pazienza and P. Velardi(1993) used semantic tags to cluster some similar words.

In our opinions, some simple combination frequencies, such as the LF and RF data in SPR stems, can also provide useful preference information for local-context-based disambiguation. 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 determinate 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 will give detailed discussions about them.

4.1.3.1 Determine the local context

The local context of a pending constituent $A$ is made up by its adjacent left matching stems and right matching stems. Due to the left-to-right parsing scheme of current matching algorithm, we can easily get the left matching stems by searching 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 operations in the right region of $A$ has not be processed.

Consider such a sentence segment: $[w_i \ [A \ w_{i+1} \ w_{i+2}] \ w_{i+3}]$. 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 $w_{i+4}$. But this constituent can’t get now. So we can’t carry out local
optimization upon constituent A. To resolve this problem, we proposed a delaying selection scheme, whose basic idea is described 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 current parsing state, we move it from PCL to DCL. Then we continue to parse the sentence left-to-right. Whenever a new matched constituent is generated, we must check whether there are suitable right matching stem for every delayed constituent in DCL or not. If there are, then we can get it from DCL and insert into PCL again. Thus, through the coordination of PCL and DCL, we can guarantee that every pending constituent can acquire suitable local context for optimization.

4.1.3.2 Select suitable SPR threshold

The SPR threshold ($\beta$) is set to restrict the application scope of SPR list, i.e. only the SPR stems in the list that satisfies the formula: $|\text{LP-RP}| > \beta$, can be used for local optimization. Under such restriction, we can implement optimization strategies as follows: If LP>RP, then select left combination structure in priority and discard the right combination structure; Otherwise, select 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 reduction in accuracy. So the key issue is to find the tradeoff between accuracy and time. In section 4.1.4, we will introduce an experiment to select suitable SPR thresholds.

4.1.3.3 Basic optimization procedure

By integrating the above all optimization strategies into our former matching algorithm [ZQ97a], we develop an optimized matching algorithm. Figure 4 shows its basic control procedure, where step 2 and 3 realize the delaying selection scheme, and step 5 and 6 implement local-preference-based optimization by adding up priority judgment through SPR data under certain threshold.

1. Get a pending constituent A from PCL;
2. Search its right matching stems (RMSs) in BMS and PSF;
3. If can’t 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 $\beta$;
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 goto 8;
8. If PCL is not empty, then goto 1; else return.

Figure 4  Basic procedure of the optimized matching algorithm
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 set and test set 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 in ([ZQd96], [ZQ97b]):
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.
2) Total of parse trees (PTTotal): all the complete parse trees \(^6\) in PSF, whose computation formula can be found in [ZQd96].
3) CPU time: the elapsed CPU time to parse the input sentence (three processing stages).

In order to select suitable SPR threshold, we firstly parse the sentences in 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 the 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 MCTotal, PTTotal and CPU time metrics. This tendency is consistent with our expectation. As the restriction upon SPR stems is loosed, more and more SPR stems can be used for local optimization, so that more and more less likely matching operations can be removed.

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

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

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

When \(\beta\) is larger, the applied SPR stems show stronger local preference. They can be used to safely prune the ambiguous structures with less likely probabilities under local contexts as early as

\(^6\) A complete tree covers all words of an input sentence.
possible. Thus, the following statistics-based disambiguation algorithm can efficiently select a best parse tree from the remained constituents with more likely probabilities under local contexts. But when $\beta$ 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 reduced 30.7%, 32.5% and 99.5% respectively, and the parsing accuracy also increases slightly.

![Figure 5](image1.png)  
**Figure 5**  Average number of matched constituents vs. sentence length

![Figure 6](image2.png)  
**Figure 6**  Average number of parse trees vs. sentence length

In order to compare the optimization effect of sentences with different length, we compute their average number of MCTotal and PTTotal in test set. Figure 5 and Figure 6 show the results, where $\beta=0.4$. As can be seen, for simple sentences (with no more than 20 words), the optimization algorithm doesn't show very 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. And the remove of one constituent under local context in the early stage of tree generation will cause to the remove 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 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;
2) Every ambiguous instances of them will occur with almost same frequencies;
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 $\alpha$ and a relative probability threshold $\beta$, we can call such a intersecting structure in a SPR stem as a Representative Ambiguous Structure (RAS), if it satisfies the following conditions: $1) \text{LF+RF} > \alpha,$ $2) \| \text{LP} - \text{RP} \| \leq \beta$.

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 as same as the SPR threshold defined in section 4.1. In fact, the SPR threshold can be looked 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 comprise as the set of RAS just enough. In the future, if some detailed disambiguation knowledge for RAS, such as lexical-based or semantics-based knowledge, can be automatically learned and applied in current preference-based parser, we can expect that the parsing performance will improve furthermore.

By setting $\alpha=10$ and $\beta=0.4$, we get 56 RASs from the SPR list A (in section 3.3), account for about 1.39% of different SPR stems in it. Comparing with some lists of Chinese syntactic ambiguous structures summarized by linguists ([Zhu80], [HGY85]), we find they covers 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’ as the complement of ‘v1’ and the ‘n’ as the object of the predicate-complement structure (or verb phrase) “v1 v2”. The other makes ‘n’ as the object of ‘v2’ and the verb phrase “v2 n” as the verbal object of ‘v1’. To disambiguate it may need some lexical and collocation information of Chinese verbs. The similar ISs in Table 5 are: “v ^v np”, “v ^np vp”, “v ^v vp”, and “v ^v v”.

2) Serial sentences:

A characteristics of the Chinese language is that several simple sentences (or verb phrase) 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. But it is very difficult to find its correct left
boundary due to there are not any obvious starting words like wh-nouns or wh-adverbs in English.
This characteristics brings into difficulties to disambiguate the IS “v ^n uJDE n”.

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

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

Though developing a searching tool, we can automatically extract all RAS instances from raw
texts. Based on them, 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 Conclusions

In this paper, we introduce the basic concepts of a quantitative description formalist 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 all
possible intersecting structures can be found through a recurrent tree traveling algorithm.
2) The sentence preprocessor based on statistics-based parsing and grammar inference techniques provides 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 traveling 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 established a quantitative measure of disambiguation difficulty for different ambiguous structures. According to it, 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 the future, we want to carry out SPR discovery algorithm in much more Chinese raw texts and generate 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 these work can give great impetus to the research of Chinese linguistics and natural language processing.

Acknowledgments

The work was supported by Chinese national natural science foundation 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.

References


