Statistical Chinese Parser ICTPROP

Hao Zhang Qun Liu Kevin Zhang Gang Zou Shuo Bai
Institute of Computing Technology
Chinese Academy of Sciences
zhanghao@software.ict.ac.cn

Abstract: Three statistical parsing models, which are incremental augmentations to the conventional PCFG, are presented in this paper. In this sequence of models, wider structural context is taken to condition the derivations. We have applied the models to the task of Chinese parsing. Results on the Penn Chinese Treebank and another treebank of shorter sentences are both reported. The results show that the Labeled Precision and Recall are raised steadily in this approach. For the MT97 treebank, precision/recall are raised from 85.58%/85.66% to 90.72%/90.92%. For the Penn Chinese Treebank, the rise is from 75.71%/70.27% to 77.16%/77.69%. We also present our efficient parsing algorithm that saves over 70% of active edges in comparison with traditional Chart parsers.

1 Introduction

With the emergence of large treebanks, supervised statistical English parsers are achieving promising results. Facing the relatively limited size of Chinese treebanks, we have to start from non-lexicalised parsers. However, the fact that the phrase structures of Chinese is less distinct than English requires us to develop richer parsing models that can take more of the conditions of phrasings into account.

Inspired by our previous work, an improved Chart parsing algorithm "Role Inverse algorithm" (Bai and Zhang 2003) that depicts each symbol position in CFG grammar rules as a unique role, we started to consider that when a category is playing different roles, the different probability distributions of its expanding rules will embody the subtlety of phrasing preferences. This thinking led to the definition of three layers of incremental extensions to PCFG in Section 2.

Corresponding to the extensions at model level, there are also extensions to grammar acquisition (extraction) and parameter estimation. These contents will be discussed in Section 3.

The parsing algorithm is a probabilistic version of the mentioned Role Inverse algorithm. We will compare our optimized algorithm with non-optimized ones in terms of number of active edges produced while parsing to show the efficiency of our parser ICTPROP.

In the last section, detailed report of the performance of our parser will be given and analyzed.

2 Parsing Models

2.1 Current Node Conditioned, Classical PCFG

The symbol system of PCFG G includes:

- A set of terminals, $\{w^k\},\ k = 1, ..., V$
- A set of non-terminals, $\{N^i\}, i = 1, ..., n$

- A start symbol, N^1
- A set of rules, $\{N^i \rightarrow z^j\}$, (z^j is a sequence of terminals and non-terminals)

On the probability aspect, PCFG gives the distribution of all the rules with the same LHS.

$$\forall i \quad \sum_j P(N^i \rightarrow z^j \mid N^i) = 1$$

To compute the probability of a parse tree $P(t)$, we have to make some independence assumptions. However, the assumptions of PCFG are too optimistic. It deems that the expansion of a node in a parse tree is an event independent of any other conditions except the category of the node. So the probability of a parse tree is the multiplication of all the probabilities of individual expansions, regardless of any order. Since all the instances of applied rules form a multiset R, then:

$$P(t) = \prod_{r \in R} P(r \mid LHS(r))$$

2.2 Current Node, Preceding Node Conditioned, P-PCFG

The symbol system is the same as PCFG. We assume that the expansion of a node is dependent on both the category of this node and that of its parent node, or preceding node. The distribution the model gives is:

$$\forall i, k \qquad \sum_j P(N^i \rightarrow z^j \mid < N^i, N^k >) = 1,$$

($< N^i, N^k >$ indicates N^i can be derived directly from N^k)

The computation of $P(t)$ thus becomes:

$$P(t) = \prod_{r \in R} P(r \mid < LHS(r), ParentOf(LHS(r)) >)$$

2.3 Current Node, Preceding Node and Relative Order Conditioned, PORD-PCFG

We further our approach to take the relative order of a node amongst its siblings as another condition.

The distribution in the model is:

$$\forall i, k, ord \quad \sum_j P(N^i \rightarrow z^j \mid < N^i, N^k, ord >) = 1$$

($< N^i, N^k, ord >$ indicates N^i can be derived directly from N^k as its child whose order is ord)

The formula to calculate $P(t)$ is:

$$P(t) = \prod_{r \in R} P(r \mid < LHS(r), ParentOf(LHS(r)), OrderOf(LHS(r)) >)$$

2.4 Preceding Rule Conditioned, PRORD-PCFG

In this model, we condition the expansion of a node on the dominating rule through which this node is derived. PRORD-PCFG conditions expansions on a relatively closed local structural context.

The distribution in the model is:

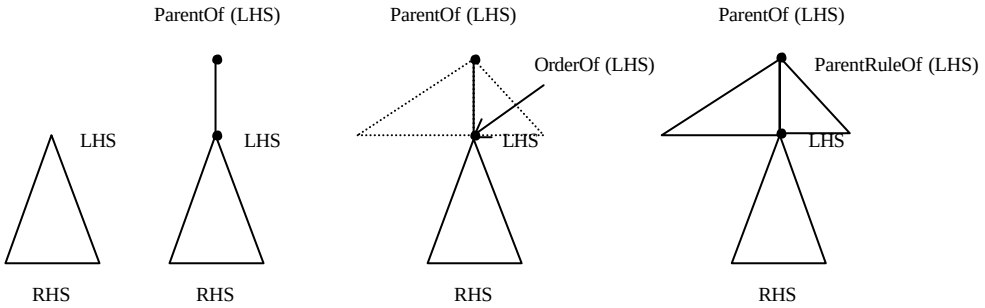
$$\forall i, r, ord \quad \sum_j P(N^i \to \mathbf{z}^j \mid < N^i, r, ord >) = 1,$$

($< N^i, r, ord >$ indicates N^i is at the *ord*-th position of RHS of rule r)

We compute $P(t)$ as:

$$P(t) = \prod_{r \in R} P(r \mid < LHS(r), ParentRuleOf(LHS(r)), OrderOf(LHS(r)) >)$$

In figure 1, the expansion of conditioned structural context is demonstrated.



(1) $P(r \mid LHS(r))$

(2) $P(r \mid < LHS(r), ParentOf(LHS(r)) >)$

(3) $P(r \mid < LHS(r), ParentOf(LHS(r)), OrderOf(LHS(r)) >)$

(4) $P(r \mid < LHS(r), ParentRuleOf(LHS(r)), OrderOf(LHS(r)) >)$

Figure 1: Incremental Extensions to PCFG

3 Grammar Extraction & Parameter Estimation

3.1 Treebank Grammar and Its Extensions

Extraction of PCFG from a treebank is straightforward. We count the occurrences of rules in the bracketed corpus at first. Then we use maximum likelihood estimation to estimate the probabilities, as in (Charniak 96)：

$$\hat{P}(N^i \to \mathbf{z}^j) = \frac{C(N^i \to \mathbf{z}^j)}{\sum_k C(N^i \to \mathbf{z}^k)}$$

Because structural context is introduced in a uniform way, a mapping technique may be applied to reduce the conditional events (derivations) to independent ones. The additional preparing work is to map all the labels of intermediate nodes in the treebank to "conditioned labels" first. In essence, this is actually to systematically split one category into multiple subcategories to fit different types of structural context.

The mapping is only necessary for non-terminal and non-root nodes in the trees. Since in our models, the root derives without constraints of any other nodes and the terminals are not capable of derivation.

(1) P-PCFG mapping:

Append the category label of its parent node to its own as the postfix.

(2) PORD-PCFG mapping:

After P-PCFG mapping, append the relative order number as the postfix.

(3) PRORD-PCFG mapping:

After PORD-PCFG mapping, append the string of labels of siblings as the postfix.

Our mapping method (1) is the same as (Johnson 98). But we generalized it to a greater extent in the consequent mappings.

The consequence of mappings is demonstrated in figure 2.

In the mapped treebanks, the previously conditional derivations are but independent derivations of more specific categories. The MLE can still be applied in the same manner as before. However, we have to deal with sparse data problem in models 2 to 4.

3.2 **Witten-Bell Backoff Smoothing.**

Mapping only answers the question of how to represent the appearances of conditional rules. For the unseen ones, we have to apply some kind of smoothing technique. We choose Witten-Bell backoff as the smoothing method, because we want to put more weight on the rules that are in terms of relative frequency closely correlated with more detailed history. The backoff is carried out in the reverse order of model extension. Details are in (Chen and Goodman 98)

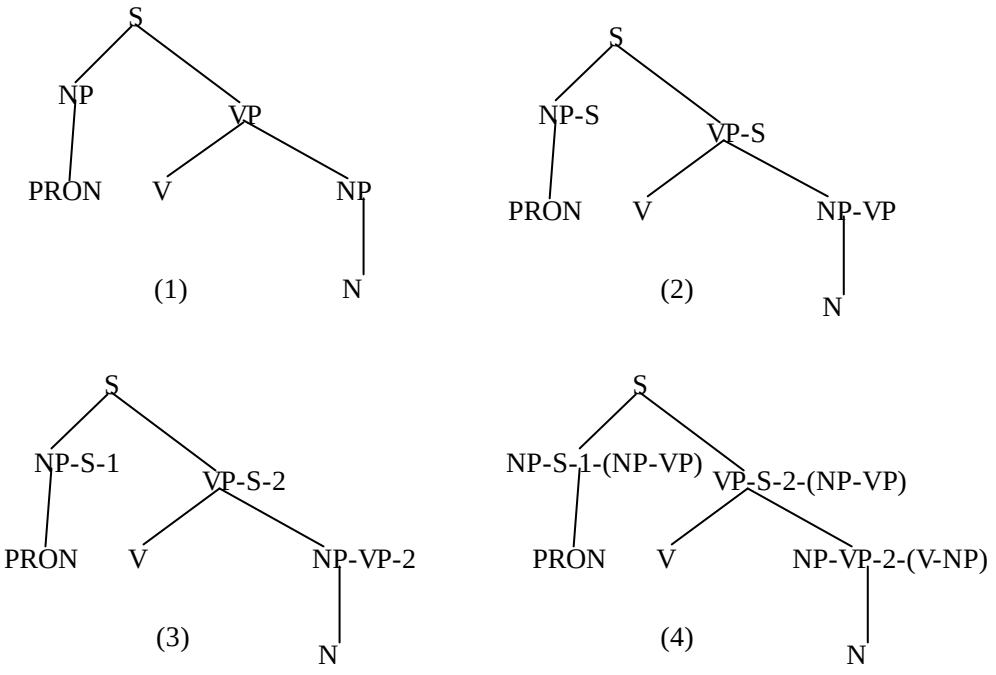


Figure 2 Example of label mapping

With mapping and smoothing, we can generate different levels of probabilistic grammars, but all in the same representation of classical PCFG. The parser may just be a PCFG one. Another additional work to do after parsing is to strip off all the postfixes of labels in the parse trees.

4 **Parsing Algorithm**

We adopted Chart parsing algorithm as our framework of parsing algorithm. To effectively

deal with the large treebank grammars, two optimizations are applied. One is a categorical edge-pruning technique, while the other prunes edges based on prefix probability.

4.1 Role Inverse Algorithm: an Improved Chart Parsing Algorithm

Let G be a CFG grammar without null-productions, and number all the rules from 1 to the total number, number all the positions in a rule from left to right as the offset to the leftmost position. We use $x.y$ to represent such a role that is the y-th constituent of the x-th rule. The functions below will be used frequently:

$Length(x)$: RHS length of rule x ,

$Left(x)$: LHS category of rule x ,

$Cat(x.y)$: The category that plays role $x.y$ (a terminal or non-terminal),

Obviously, categories and roles have a one-to-many mapping.

Definition 1 Role Inverse function $I(C,t)$ produces the set of possible roles a proposed category C can play when it meets terminal t to its right as the look-ahead symbol:

$$I(C,t)=\{x.y \mid Cat(x.y)=C) \wedge (y<Length(x) \rightarrow t \in FIRST(Cat(x.y+1))) \wedge (y=Length(x) \rightarrow t \in FOLLOW(Left(x)))\} \quad (1)$$

Definition 2 Rule Starting function $S(C,t)$ produces the set of possible rules a predicted category C can expand when a terminal t is the look-ahead:

$$S(C,t)=\{x \mid Left(x)=C \wedge t \in FIRST(Cat(x.1)\} \quad (2)$$

Actually, $S(C,t)$ is a special case of $I(C,t)$, when y in (1) is always 0.

The idea of Role Inverse algorithm is to construct a set of $I(C,t)$ table and $S(C,t)$ table to filter the roles a category may take. The reduction of proposed roles will be reflected in a Chart as reduced edges. An efficient twins-graph algorithm that makes the construction of parsing table cost-effective is explained in detail in (Bai and Zhang 2003).

In Chart paring, an edge is represented as $[i, j, A \rightarrow a \bullet Bb]$. If $A \rightarrow aBb$ is numbered as x, and $|a|=y$, we can encode it as $[i, j, x.y]$, which is called role code. ($[i, i, x.0]$ encodes the starting rules)

Role inverse parsing algorithm, in comparison with classical Chart algorithm (Russell and Norvig 1995):

- init
  add edge $[0,0, S' \rightarrow \bullet S]$ ($[0,0,0.0]$);
- repeatedly add edge, until no edge can be added with the following actions
  for edge $[i, j, A \rightarrow a \bullet Bb]$ ($[i, j, x.y]$),
  - pre action:
    if z: $B \rightarrow g$ satisfies $z \in S(B, string[j+1])$, then add edge $[j, j, B \rightarrow \bullet g]$ ($[j, j, z.0]$);
  - scan action:
    if $string[j+1]$ belongs to category B, and $x.y+1 \in I(B, string[j+2])$, then add edge $[i, j+1, A \rightarrow aB \bullet b]$ ($[i, j+1, x.y+1]$);

- comp action：
  if there is an edge $[j,k,B \rightarrow F \bullet]$ ， and $x.y+1 \in I(B,string[k+1])$, then add edge $[i,k,A \rightarrow aB \bullet b]$ ($[i,k,x.y+1]$);

- if $[0,n,S' \rightarrow S \bullet]$ ($[0,n,0.1]$) appears then a successful parse is produced

4.2 Prefix Probability Maximization

The probabilistic version of Role Inverse Algorithm is still a strict left-to-right parsing algorithm. Before a new terminal is scanned, all the partial Viterbi parses left to the scan point is selected, and the maximum probability of every active edge is also available. Because what we want is the global Viterbi parse, any active edge that is impossible to lead to this goal can be pruned. The pruning criteria are:

For all active edges that

(1) have the same starting point and end at the scan point,

(2) have the same postfix to be expected,

(3) have the same category to be reduced to,

select the one with the maximum probability and prune all the others.

The reason is these edges have the same behavior in the following process and the one whose probability is currently the highest among them will always be the highest in future development. So we select it and defeat others as soon as possible.

4.3 Result with two edge-pruning techniques applied

In the following chart, the pruning effect of role filtering and prefix filtering is demonstrated. The x-axis indicates the length of sentences to be parsed. The y-axis indicates the percent of active edges relative to that of a Chart algorithm without the two optimizations. The chart tells us the longer the sentence to be parsed, the eminent the pruning effect. On average, our algorithm saves over 70% of active edges, which compose the majority of edges.

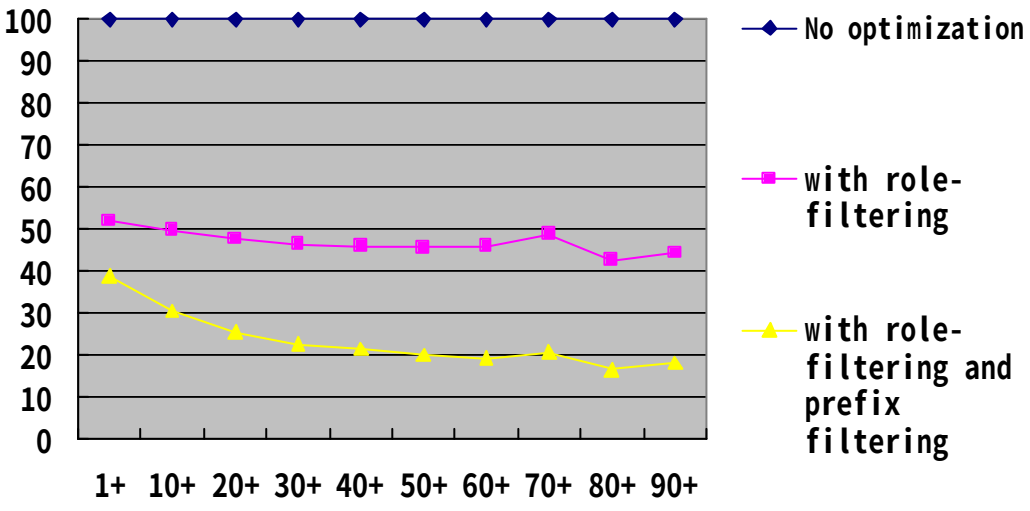


Figure 3 Effect of Edge-Pruning

5 Results and Conclusions

We experimented on two Chinese treebanks to test the performance of our extended PCFG parser ICTPROP. The first treebank MT97 (Liu 98) is a treebank of short sentences with average length of 8 words. We got very satisfying results on this small data set. (See table 2-1) The second treebank Penn Chinese Treebank (Xia et al. 2000) has an average sentence length of 30 words and presents a much greater challenge to the parser. Unfortunately, the last model PRORD-PCFG is not feasible without pruning of rules. However, in the experiments, we didn't try to prune any rules blindly. We replaced PRORD-PCFG with a less rich model that only attaches the labels of adjacent brother nodes as postfix during mapping (3) to approximate the ideal model. The result on Penn Treebank is not so dramatic as expected. We analyzed the reason and concluded that:

(1) The sparse data problem is serious without cutting of extremely infrequent rules when extending PCFG. We need to treat the rules differently while extending them to conditional ones.

(2) Without lexicalisation, there will be an upper bound of precision/recall. We plan to use some kind of words clustering algorithm to group the words automatically first and then build a richer model based on the word groups. If successful, we may approximate lexicalisation to some extent.

The following table shows the results in terms of LP (Labelled Precision), LR (Labelled Recall), CB (Crossing Brackets), 0CB (percentage of 0CB parses), 1CB (percentage of 1 or 0 CB parses), as described in (Manning and Schütze 99)

Table 2-1 MT97 Treebank Result（2400 training, 671 testing）

	PCFG	P-PCFG	PORD-PCFG	PRORD-PCFG
LP	85.58	88.88	88.91	90.72
LR	85.66	89	91.13	90.92
CBs	0.86	0.53	0.43	0.40
0CB	58.57	72.43	76.9	78.99
1CB	76.30	85.4	88.08	88.97

Table 2-2 Penn Chinese Treebank Result (2200 training, 662 testing)

	PCFG	P-PCFG	PORD-PCFG	PRORD-PCFG*
LP	75.71	77.76	77.73	77.16
LR	70.27	73.02	75.3	77.69
CBs	3.42	2.90	2.79	2.79
0CB	24.81	31.16	31.56	32.70
1CB	40.24	49.47	48.1	48.73

*Approximation

References

[1] Charniak, E. 1996. Treebank grammars. Technical Report CS-96-02, Department of Computer Science, Brown University.

[2] Charniak, E. 1997. Statistical parsing with a context-free grammar and word statistics. In Proceedings of NCAI-1997, pp 598--603.
[3] Fei Xia, Martha Palmer, Nianwen Xue, Mary Ellen Okurowski, John Kovarik, Fu-Dong Chiou, Shizhe Huang, Tony Kroch, and Mitch Marcus. Developing guidelines and ensuring consistency for Chinese text annotation. In Proceedings of the Second International Conference on Language Resources and Evaluation (LREC-2000), Athens,Greece, 2000.
[4] Johnson, M. 1998. The effect of alternative tree representations on tree bank grammars. In Proceedings of the Joint Conference on New methods in Language Processing and Computational Natural Language Learning (NeMLaP3/CoNLL'98), pp 39--48.
[5] Manning, C., Schütze, H. 1999. Foundations of Statistical Natural Language Processing. MIT Press.
[6] Russell, S.J., Norvig, P. Artificial Intelligence: a modern approach. N.J.: Prentice-Hall, 1995. 696~703
[7] Shuo Bai, Hao Zhang. Role Inverse Algorithm: an Improved Chart Parsing Algorithm. To appear in Journal of Software, 2003(3)
[8] Stanley F. Chen, Joshua Goodman. An Emprirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, 1998.
[9] Ying Liu. A Hybrid Approach to Chinese-English Machine Translation. Doctoral Dissertation. Institute of Computing Technology, Chinese Academy of Sciences.1998