A Chinese LPCFG Parser with Hybrid Character Information

Wenzhi Xu, Chaobo Sun and Caixia Yuan
School of Computer,
Beijing University of Posts and Telecommunications,
Beijing, 100876 China
{ earl808, sunchaobo}@gmail.com
yuancx@bupt.edu.cn

Abstract

We present a new probabilistic model based on the lexical PCFG model, which can easily utilize the Chinese character information to solve the lexical information sparseness in lexical PCFG model. We discuss in particular some important features that can improve the parsing performance, and describe the strategy of modifying original label structure to reduce the label ambiguities. Final experiment demonstrates that the character information and label modification improve the parsing performance.

1 Introduction

Parsing is an important and fundamental task in natural language processing. The challenge of Chinese parser has been the focus of attention in recent years, and many different kinds of Chinese parsing models are investigated. (Bikel, 2000) adopts Head-Driven model to parse Chinese. (Levy, 2003) analyzes the difficulties of Chinese parsing through comparing the differences between Chinese and English. (Wang, 2006) utilizes shift-reduce approach, dramatically improved the decoding speed of parsing. All these research adopted the same models which are also used in English parser – the models based on the words.

However, there is a big difference between English and Chinese: the expressing unit in English is word, while character is the smallest unit in Chinese. Due to difficulties of word segmentation, especially for different segmenting criteria, many researchers explored parsing Chinese based on characters. The parser of (Luo, 2003) received sentence as input and conducted word segmentation and syntactic parsing at the same time, but they did not utilize the character information in generating subtree; (Zhao, 2009)'s dependency parsing tree totally abandoned the word concept, so the dependency relations are the relations between characters.

We combine both word and character information to gain better performance of parsing. Although the criteria of segmentation are difficult to be unified, different criteria conflict only within the phrases which have little influence on the structure between phrases. So we still use word as our basic unit of parsing. Although word has been proved to be effective in head-driven parser (Collins,1999), the data of word dependence is very sparse. While it is worthy to note that words with similar concept always share the same characters in Chinese. For instance, "科学家(scientist)", "历史学家(historian)", etc., share the same character "家(expert)", since they belong to the same concept "expert in a certain field". So the problem of word sparseness can be solved by combining the character information to some extent.

Throughout this paper, we use TCT Treebank (Zhou, 2004) as experimental data. TCT mainly consists of binary trees, with a few of multi-branch and single-branch trees. Thus, we first transfer all trees to binary trees. Then we use Lexical-PCFG model to exploit the word and character information, and Maximum Entropy Model to calculate the probability of induced trees as (Charinak, 2000). Finally we use CKY-based decoder.

In the following section, we will introduce how to utilize character information in our parsing model and the other features in detail. Section 3 gives experiment results and analysis, which show improvement of our parsing approach. Section 4 presents the conclusion and future work.

2 Lexical PCFG model

2.1 Model Introduction

Starting from the Lexical-PCFG model (Model 2 in Collins, 1999), we propose a new generative process which can conveniently exploit the character information and other features.

Assume P is the label of parent, H is the head child of the rule, and $L1, ..., Ln$ and $R1, ..., Rn$ is the left and right modifiers of H. Then the rule of Lexical-PCFG (LPCFG) can be written as:

$$\begin{aligned} P(hw, ht) \quad & \rightarrow \quad (1) \\ & L_n(lw_n, lt_n)...L_1(lw_1, lt_1)H(hw, ht) \\ & R_1(rw_1, rt_1)...R_n(rw_n, rt_n), \end{aligned}$$

where (hw, ht) represents the head word and head tag of head child, $(lw1, lt1), ..., (lwn, ltn)$ and $(rw1, rt1), ..., (rwn, rtn)$ are the head words and head tags of left and right modifiers, and parent node P's head word and tag are the same as that of H.

As mentioned above, our trees are all binary trees. In this case, the LPCFG can be written as:

$$P(hw, ht) \rightarrow H(hw, ht)R(rw, rt), \quad (2)$$

$$P(hw, ht) \rightarrow L(lw, lt)H(hw, ht). \quad (3)$$

Formula 2 and 3 represent that the head child is the left or right child respectively. The probability of the rule is conditioned on the head words and tags of head and its modified children, which is specified as:

$$Pr(P, L, H|hw, ht, lw, lt), \quad (4)$$

and

$$Pr(P, R, H|hw, ht, rw, rt). \quad (5)$$

To calculate these probabilities, we rewrite Equation 4 and 5 by three factors in 6 and 7 using the chain rule.

$$\begin{aligned} & Pr(P, R, H|hw, ht, lw, lt) = \quad (6) \\ & Pr_d(P - DIR|hw, ht, lw, lt) * Pr_h(H|P, hw, ht) \\ & *Pr_m(L - DIR|P, H, hw, ht), \end{aligned}$$

$$\begin{aligned} & Pr(P, R, H|hw, ht, rw, rt) = \quad (7) \\ & Pr_d(P - DIR|hw, ht, rw, rt) * Pr_h(H|P, hw, ht) \\ & *Pr_m(R - DIR|P, H, hw, ht). \end{aligned}$$

in which, DIR=LEFT/RIGHT. DIR in P-DIR is used to discriminate different positions of head child, DIR in L-DIR and R-DIR are used to represent different positions of modifiers.

The calculation processes of Equation 6 and 7 can be interpreted by following generative process. Firstly, the head words and tags of children generate the parent and the head position (the first probability in Equation 6 and 7). We define this probability as the word dependency probability Pr_d: if two words (or characters in words) always appear together in the training data, this probability will be large (< 1); if two words (or characters in words) do not have any dependence in the training data, this probability will be approximately equal to $1/|Y|$, where $|Y|$ is the predicted number of the Pr_d. The second probability generates the head child label (defined as the head child probability Pr_h), we hold that the head word and tag of modifier do not provide information to determine the head child label, so we omit them. The third one produces the modifier label, which is defined as the modifier probability Pr_m, and evaluates the dependency relation between modifier and the head child. We also omit the influence of the head word and tag of modifier.

For example, assume there is a tree as shown in Figure 1. For the rule "vp → v np", head child of parent vp and np are the left child v and right child n respectively, so "组织(organize)" and "专家(expert)" are the head word of vp and np. Thus the LPCFG rule is "vp(组织,v) → v(组织,v)

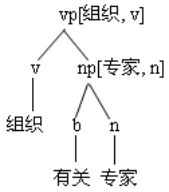

Figure 1: Tree representation of LPCFG rule.

np(专家,n)". The probability can be written as: $Pr_d(vp\text{-LEFT} \mid \text{组织},v,\text{专家},n) * Pr_h(v|vp,\text{组织},v) * Pr_m(np\text{-RIGHT} \mid vp, v, \text{组织},v)$.

2.2 Probability Model and Feature Set

We use Maximum Entropy (ME) Model to compute probabilities of candidate trees. ME model estimate parameters that would maximize the entropy over distributions, meanwhile satisfy certain constraints. These constraints will force the model to reflect characteristic of training data. With the feature function, Maximum Entropy can exploit kinds of features flexibly, some of which are very important to improve the performance of tasks at hand. ME model has been applied successfully in many tasks, such as parser (Charniak, 2000; Luo, 2003), POS tagging (Ratnaparkhi,1996), etc. In our experiment, we use Maxent toolkit developped by Zhang (Zhang, 2004), which uses the LBFGS algorithm for parameter estimation. Details of the model and toolkit can be seen in (Berger, 1996; Zhang, 2004).

Our features consist of four parts: basic features, character features, context features and overlapping features of character and context. Basic features are traditional LPCFG features, including head word, head tag and the label. We extract the first and last characters of a word as the Character features, of course for a single character word the first and last character are the same. Context features are defined as the previous and following POS tags of the current subtree, and these features utilize the information outside of the subtree very well without increasing the complexity of parsing decoder. Overlapping features are the combinations of character features and context features.

Take Chinese sentence "委员会/n 由/p 农业部/n 组织/v 有关/b 专家/n 组建/v 完成/v (Committee is composed of experts organized by the Ministry of Agriculture)" for example, the corresponding rule is "vp(组织,v) → v(组织,v) np(专家,n)", the feature template of the example sentence is shown in Table 1.

When applying the character information, it is worthwhile to note that character is always combined with the POS tag of the word since the sense of single character varies as word's POS tag changes. For example, the sense of "爱(love)" in verb "爱护(care and protect)" and noun "爱情(love)" is different. Of course the sense discussed here is reflected in the dependency of words: "爱护(care and protect)" can be followed by some nouns which are objects, while "爱情(love)" can not.

For the multi-branch tree, (Collins,1999) calculates the probability of the left or right modifier with a feature which represent whether there are modifiers between current modifier and the head child (distance feature). But in the situation of binary tree, it is obvious that current modifier is unlikely to follow other modifiers. Since the representation of binary tree conforms to the X-bar theory of Chomsky, we can modify the head child label to get this non-local information in binary tree. For instance, a multi-branche tree rule "vp1 → pp d vp2" corresponding to these two binary tree rules: "vp3 → pp vp4" and "vp4 → d vp5" (the index numbers of the vb here stand for different vp). So when we calculate the probability of pp with the multi-branch situations, d lies between pp and vp2. While in binary tree situation, we cannot catch this information between pp and vp4. However, we can modify vp4 to vp-LEFT, which means there is a modifier at the left child of vp4, then we get the similar effect in (Collins, 1999). We call this as the head position labeling.

2.3 Label splitting and Head Position Modifying

(Klein, 2003) improves the performance of parser via splitting the POS tag in corpus. We split the non-terminal label using the same approaches (assuming the POS tag is terminal label). The need of label splitting is that the corpus does not sufficiently consider different situations and treat them

Table 1: Feature templates and symbol explanation.

	Pr_d (vp-LEFT)		Pr_h (v)		Pr_m (np-RIGHT)	
basic	lw rw	组织专家	p	vp	p h	vp v
feature	lw lt rw rt	组织v 专家n	p hw	vp 组织	p h hw	vp v组织
			p hw ht	vp 组织v	p h hw ht	vp v 组织v
			p ht	vp v	p h ht	vp v v
char.	lw frc rt	组织专n	p fhc ht	vp 组v	p h fhc ht	vp v 组v
feature	other combinations	···	p lhc ht	vp 织v	p h lhc ht	vp v 织v
	flc lt frc rt	组v 专n				
	other combinations	···				
context	lw rw pt1 at1	组织专家n v	p pt1	vp n	p h pt1	vp v n
feature	lw lt rw rt pt1 at1	组织v 专家n n v	p pt1 pt2	vp n p	p h pt1 pt2	vp v n p
			p at1	vp v	p h at1	vp v v
			p at1 at2	vp v v	p h at1 at2	vp v v v
			p pt1 at1	vp n v	p h pt1 at1	vp v n v
			p hw pt1 at1	vp 组织v n	p h hw pt1 at1	vp v组织v n
overlap	lw frc rt pt1 at1	组织专n n v	p fhc ht pt1 at1	vp 组v n v	p h fhc ht pt1 at1	vp v 组v n v
feature	other combinations	...	p lhc ht pt1 at1	vp 织v n v	p h lhc ht pt1 at1	vp v 织v n v
	flc lt frc rt pt1 at1	组v 专n n v				
	other combinations	...				

Symbol Explanation

frc lrc	the first and last characters of the head word of the right child
pt at	previous and following POS tag of current subtree, number indicates the position
flc llc	the first and last characters of the head word of the left child
fhc lhc	the first and last characters of the head word

as the same label which results in ambiguity. Furthermore, in our experiment, the corpus that we adopt is binary tree. Though the rule set in binary tree is closed, it brings stronger independent assumption (Jonson,1998). Thus splitting the label can make the node label represent more information from descendants. Just like the intuition of head position labeling, this is also one method to utilize the non-local information. We mainly consider these modifying as follows.

First of all, we split the label vp. There are three kinds of verb phrases: the first one is that there is modifier ahead (such as advp); the second phrase consists of an object; while the third one has the form of two verbs or verb plus an auxiliary word. The formal two situations can not follow any object any more (some double-object verb phrase may be continued to contain object, but their POS label is different with common verb), the vp in last situation can be followed by object (there maybe actually no object). If we do not discriminate these situations, it will be easy to result in dividing the object into two objects during parsing test, just as shown in Figure 2. However, if we modify vp in the third situation into vb, then this difference can be discriminated well. We take a simple statistics as an example to illustrate the sense. Assume our object is np, rule "vp → vp np" appears for 5,284 times in corpus before modifying, while it present only 166 times after modifying.

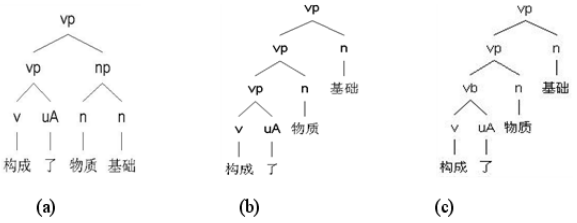

Figure 2: Parsing Result Example: (a) is a correct tree, (b) is a wrong one, while the probability may be not small enough, (c) is also wrong, but the probability is very small due to the symbol vb.

Secondly, we also split the np tag. We notice that a noun phrase, which consists of non-noun (phrase) modifier (such as ADJP, PP) and a noun (phrase), is always the final noun phrase but rarely part of another noun phrase. So we transform the np, which has the non-noun (phrase) modifier, to

nm. From the statistics of corpus, we find rule "np → np n" occurs for 4,502 times, while "np → nm n" only appears 826 times.

Finally, we change the head position of preposition phrase. The head position of preposition phrases in corpus mostly is the phrase behind the preposition, but we found the grammar of preposition phrase is much related to the preposition. Take the preposition "以(by)" and "对(to)" as example, these two prepositions occur for 755 and 1,300 times respectively. In our corpus, 98.7% of preposition phrases with 以(by)" are the modifiers of verb phrases, while only 57.2% of phrases with "对(to)" appear as the modifiers of verb phrases, and the remaining 42.8% are the modifiers of noun phrases.

3 Experiment Result and Analysis

Our experiments are conducted on the TCT corpus, which is used as the standard data of the CIPS-SIGHAN Parser 2010 bakeoff. We omit the sentences with length 1 during training and testing. Performance on the test corpus is evaluated with the standard measures from (SIGHAN REPORT, 2010).

We submit two results for the parsing bakeoff: one is single model we described in Section 2, another is reranking model, which is an attempt to apply a perceptron algorithm to rerank the 50-best result produced by the ME model.[1] Table 2 shows the result of our parser compared with the top one in this bakeoff. Since the parser we built is strictly dependent on the POS tags, the precision of POS tagging has a harsh effect on the overall parsing performance.

The performance of the rerank model is lightly lower than that of the single model. The most likely reason is that the features we count on are far from enough, and the informative features proved to be useful in (Charniak and Johnson, 2003) are not yet included in our discriminative ranker. Besides, the rank model we used is a simple perceptron learner, more delicated model, such as ME model used in (Charniak and Johnson, 2003), might improve the result.

[1]More details can be found in (Charniak and Johnson, 2003; Huang, 2008). The features we used include ParentRule, RightBranch, Rule, Heads, WProj described in (Charniak and Johnson, 2003).

Table 3: Results of different features with no limit sentence length.

feature set	LR	LP	F	CB	0CB	2CB
basic	80.19	79.61	79.90	1.20	56.10	83.49
+ch	81.91	81.38.	81.65	1.10	58.34	84.95
+cont	85.53	85.34	85.44	0.83	65.62	88.86
+ch + cont	86.17	85.94	86.06	0.80	66.61	89.62
+ch + cont + ol	86.34	86.13	86.24	0.79	66.65	89.81
+ch + cont + ol + cwd	86.47	86.26	86.37	0.78	66.73	89.87
+ch + cont + ol + cwd + cm	87.03	86.77	86.90	0.75	67.06	90.36
+ch + cont + ol + cwd + cm + hpl	87.20	86.94	87.07	0.74	67.43	90.40

ch=character feature, cont=context feature
ol=overlap feature, cwd=coordinate word dependence
cm=corpus modifying, hpl= head position label

In order to make clear how different features effect the parser performance, we conducted experiments on the TCT data provided by CIPS-ParEval-2009 for Chinese parser bakeoff [2], since the sentences in CIPSParEval-2009 are given with head words and gold-standard POS tags. The results of our parser are given in Table 3. From Table 3 we can see that character features bring the improvement of F score for 1.75 compared with the basic features (line 2 vs line 3), and for 0.8 after adding the context features (line 4 vs line 6). These results show that character features can improve the model with basic features very well. After applying the context features, character features can still bring improvement, which states that character features can solve the ambiguities that can not be solved by the context features.

One likely reason why character information is helpful is that the character can partly represent the meaning of word and can partly resolve the sparseness problem of word dependence as been observed in the work of (Kang, 2005). Kang calculated the statistics for 50,000 double characters words and divided the methods of constructing word into 8 types according to the relations of meaning between word and characters:

(1) A+B=A=B (2) A+B=A
(3) A+B=B (4) A+B=C
(5) A+B=A+B (6) A+B=A+B+D
(7) A+B=A+D (8) A+B=D+B

A and B stand for the meaning of the two characters which are used to construct the word. C is a totally new meaning and D represents an ad-

[2]http://www.ncmmsc.org/CIPS-ParsEval-2009/index.asp, the first workshop on Chinese Syntactic Parsing Evaluation, November, 2009.

Table 2: Results of different features with no limit sentence length.

	"B+C"-P	"B+C"-R	"B+C"-F1	"B+C+H"-P	"B+C+H"-R	"B+C+H"-F1	POS-P
Top one	85.42	85.35	85.39	83.69	83.63	83.66	93.96
Single	74.86	76.05	75.45	71.06	72.20	71.63	87.00
Rerank	74.48	75.64	75.05	70.72	71.81	71.26	87.00

Table 4: The relation between the meaning of words and characters.

type	1	2	3	4	5	6	7	8
word number	4035	1031	297	4201	14455	23562	2780	1886
rate(%)	7.71	1.97	0.57	8.02	27.60	44.99	5.31	3.60

ditional meaning. The expression after the first "=" is the meaning of the word, and the symbol "+" indicates the melding of meaning. For example, A+B=A+D indicates that the word retains the meaning of character A, and adds new meaning D. The distribution of each type in the dataset is shown in Table 4. From Table 4 we can see that type 4, i.e., there are no relation between characters and word, occupies only 8.02%. This data proves that the word inherits the meaning from the characters which are used to construct the word. However, the relations are really complicated. For example, some words only inherit the meaning of formal characters and others of the last characters. This might be the reason why character information does not have very obvious effect as expected.

In our parsing model, context features are really helpful to the parsing accuracy. Different with the decision method in (Rantnaparkhi, 1999) and (Wang, 2006), and reranking in (Collins, 2000) which all can utilize the context of current subtree very well (not only the POS tag), the CYK decoding algorithm restricts our context features. However, we can conveniently exploit the POS tags around the current subtree without increasing the complexity of decoding and thus improve the performance.

Commonly, each subtree has only one head word. However, we notice that the two head words of two coordinate children are equivalent, as illustrated in Figure 3. We assume that the parent node of these two children is A and the two head word are all the head words of A. When A is the child of the parent node B, all the head words in A can be dependent with the other head

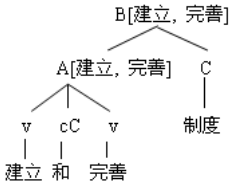

Figure 3: Example of dependence between coordinate words.

words of another child C. When A is still the head child of B, the head words of B are also the same as A. Then we can extract more word dependence data. For example, A have two head words "建立(construct)" and "完善(complete)", and "制度(rule)" is the head word of C, then we consider that "建立(construct)" and "完善(complete)" are all dependent with "制度(rule)". Meanwhile, A is also the head child of B, and the head words of B are also "建立(construct)" and "完善(complete)". During the decoding, we choose the most probable dependence as the dependence probability of B. From the result, we can see that this strategy yields 0.17 improvements in the F score.

Label splitting can also improve the performance. However, modifying the labels need much linguistic knowledge and manual work. (Petrov, 2006) proposed an automated splitting and merging method. As an attempt, we tested the effectiveness of it in our parser empirically. When tested on the TCT data provided by CIPS-ParsEval-2009 for Chinese parser, bakeoff the label spitting improve the F1 measure from 0.864 to 0.869.

4 Conclusion and Future Work

This paper presents a new lexical PCFG model, which can synthetically exploit the word and character information. The results of experiment prove the effectiveness of character information.

Also our model can utilize the context features and some non-local features which can dramatically improve the performance.

In future work, we need improve the decoding algorithm to exploit more complex features. As the parser we build is greatly dependent on the preprocessing result of word segmentation, POS tagging and head labeling, a critical direction of future work is to do word-segmentation, POS tagging, head detection and parsing in a unified framework. Besides, as for the K-best reranking, we should take into account more informative features and more powerful reranking model.

Acknowledgment

This research has been partially supported by the National Science Foundation of China (NO. NSFC90920006). We also thank Xiaojie Wang, Huixing Jiang, Jiashen Sun and Bichuan Zhang for useful discussion of this work.

References

A.L. Beger, S. A. D Pietra, and V.J.D Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 39–71.

D.M. Bikel and D. Chiang. 2000. Two statistical parsing models applied to the Chinese Treebank. *In Proceedings of the Second Chinese Language Processing Workshop*, 1–6.

E. Charniak. 2000. A maximum-entropy-inspired parser. *In Proceedings of the 1st NAACL*, Seattle, WA, 132–139.

E. Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. *In Proceddings of the Fourteenth National Conference on Artificial Intelligence*, Menlo Park, CA. 598–603.

X. Chen, C.N. Huang, M. Li, and C.Y. Kit. 2009. Better Parser Combination. *In CIPS ParsEval*, Beijing, China. 81–90.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing.* Ph.D. Dissertation, University of Pennsylvania.

M. Collins. 2000. Discriminative reranking for natural language parsing. *In Proceedings of ICML 2000*, 175–182.

S.Y. Kang, X.X. Xu, and M.S. Sun. 2005. The Research on the Modern Chinese Semantic Word-Formation. *Journal of Chinese Language and Computing*, 103–112.

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. *In Proceedings of ACL 2003*, 423–430.

L. Huang. 2008. Forest Reranking: Discriminative Parsing with Non-Local Features. *In Proceedings of ACL 2008*, 586–594.

D. Klein and C.D. Manning. 2003. Accurate unlexicalized parsing. *In Proceedings of ACL 2003*, 423–430.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. *In Proceedings of ACL 2005*, 173–180.

R. Levy and C.D. Manning. 2003. Is it harder to parse Chinese, or the Chinese Treebank? *In Proceedings of ACL 2003*, 439–446.

X.Q. Luo. 2003. A maximum entropy Chinese character-based parser. *In Proceedings of EMNLP 2003*, 192–199.

M. Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 613–632.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. *In Proceedings of ACL 2006*, 433–440.

A. Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. *In Proceedings of EMNLP 1996*, 133–142.

A. Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 503–512.

M.W. Wang, K. Sagae, and T. Mitamura. 2006. A Fast, Accurate Deterministic Parser for Chinese. *In Proceedings of ACL 2006*, 425–432.

L. Zhang. 2004. *Reference Manual*. Maximum Entropy Modeling Toolkit for Python and C++.

H. Zhao. 2009. Character-Level Dependencies in Chinese: Usefulness and Learning. *In Proceedings of 12th ECACL 2009*, 879–887.

SIGHAN REPORT. 2010. SIGHAN REPORT ON TASK2. *In Proceedings of CIPS-SIGHAN 2010*, Beijing, China.

Q. Zhou. 2004. Annotation Scheme for Chinese Treebank. *Journal of Chinese Information Processing*, 1–8.