Chinese Language Parsing with Maximum-Entropy-Inspired Parser

Heng Lian
Brown University

Abstract

The Chinese language has many special characteristics that make parsing difficult. The performance of state-of-the-art parser is much worse than that for the English language, with an f-score about 10% below that of English. We present the result of a maximum-entropy-inspired parser [3] on Penn Chinese TreeBank 1.0 and 4.0, achieving precision/recall of 78.6/75.6 on CTB1.0 and 79.1/75.0 on CTB 4.0. We also apply the MaxEnt reranker [4] on the 50 best parses and get about 6% error reduction. The parser is also applied directly to unsegmented sentences and also achieves state-of-the-art performance.

1 Introduction

Parsing is an important step in natural language understanding. The output from a parser can be regarded as a low-level preprocessing towards the ultimate goal of letting computers understand human language. While the parsing has been applied successfully on the English language [3, 5, 8], achieving an average precision/recall of nearly 90%, there are few results reported on Chinese. Besides the lack of high-quality treebanks that are required for training the parser, the characteristics of Chinese language itself poses some problems that is not seen in English.

In this work, we apply the maximum-entropy-inspired parser proposed in [3] to the Penn Chinese Treebank [10]. In section 2, we review the maximum-entropy-inspired parser in [3]. In section 3,the MaxEnt reranker [4] is used to improve the performance of the 50-best parser. In section 4, we show how the maximum-entropy-inspired parser can be applied on the unsegmented sentences. Section 5 presents the experimental result. And finally, we conclude in Section 6.

2 Maximum-Entropy-Inspired Parser

The parser used in the experiment is Charniak's maximum-entropy-inpsired parser [3] and the main points are reviewed here.

Like most other successful parsers, we start with a generative model $p(\pi, s)$, where π is the parse tree for a sentence s. The way that a parse tree is generated is as follows. We start from the tree root S (meaning Sentence), and use the context-free grammar for branching. Each expansion is assigned a probability, and the probability of a tree would be the product of the probabilities of all expansions that generate the given sentence. We seek the parse that maximizes the probability $p(\pi, s)$ for the given sentence s. We assign probability to each expansion $L \rightarrow \Delta L_m \ldots L_1 M R_1 \ldots R_n \Delta$, where Δ is the stop symbol and M is the constituent that is the head of this expansion. We assume the Markov model. In the zero order markov model, this is simply

$$p = \prod_i p(L_i|L) \cdot p(M|L) \cdot \prod_i p(R_i|L)$$

And if we want higher order Markov property, we can, for example, additionally condition L_2 on L_1 and M. The 3rd order Markov model is used in the experiment.

The above way of assigning probabilities makes a complete model, but it does not work well in practice, since it does not take into account the history(parent, grandparent) or the lexical information. So you end up assigning to each rule the probability that might look like

$$p(r) = p(t|l, H) \cdot p(h|t, l, H) \cdot p(e|l, t, h, H)$$

, where r is the expansion rule, l is the left hand side of r, h is the head word and t is its tag, e is the right hand side of the expansion rule, and H represents other history information.

The maximum-entropy approach uses carefully designed features to represent each conditional probability. For each conditional probability that appears in the model, the maximum-entropy model specifies that it is of the form

$$p(x|y) = \frac{1}{Z(y)} e^{\lambda_1(x,y)f_1(x,y)+\ldots+\lambda_m(x,y)f_m(x,y)}$$

where f_i is the feature , and λ_i is the weight associated with it, and $Z(y)$ is the so-called partition function that normalizes the probability. The maximum-entropy parser has been developed in [8]. Charniak [3] takes a different approach by noticing that the condition probability specified by the maximum-entropy model is of the product form $p(x|y) = h_0(x, y)h_1(x, y)\ldots h_m(x, y)$. Actually, any conditional probability can be written in product form. As a simple example,

$$p(A|B, C) = p(A)\frac{p(A|B)}{p(A)}\frac{p(A|B, C)}{p(A|B)}$$

The formula as it stands above is just a tautology since the numerator of one factor cancels the denominator of the succeeding factor. But consider the case where one has a factor that is conditioned on a large number of events, say, $\frac{p(A|B,C,D,E,F)}{p(A|B,C,D,E)}$. Remember that these probabilities need to be estimated from the training data, and conditioning on a large number of events will cause the sparse data problem, since it is unreasonable to assume that the joint event $A \cap B \cap C \cap D \cap E \cap F$ will appear a sufficient number of times in the training data to make the estimate accurate. In such cases, you want to condition on less events by keeping only the most relevant ones. So we want to change $\frac{p(A|B,C,D,E,F)}{p(A|B,C,D,E)}$ to, say, $\frac{p(A|B,C,F)}{p(A|B,C)}$, and thus the estimation would be more accurate.

Of course, strictly speaking, now we don't have exact equality in the above display. But arguably, one can assume it is not far from equality. Throwing out normalizing constant also sometimes appears in a slightly different framework in computer vision literature to reduce computational burden [9]. And by conditioning on fewer events, we can hope to alleviate the problem of sparse data.

3 MaxEnt Reranker

Machine learning technique is recently used to improve the performance of a parser [6]. We use the reranker in [4] which seems to give better results. In order to use the reranker, the modified version of the maximum-entropy-inspired parser must be used which produces 50 parses for each sentence with their respect probability. The reranker tries to assign a new probability to each one of these 50 parses. Additional features are used for this task and the probability is defined through

$$\log \frac{p(\pi_1)}{p(\pi_2)} = \frac{exp\{\theta \cdot f(\pi_1)\}}{exp\{\theta \cdot f(\pi_2)\}}$$

where f is the vector of features that are used in the reranker and θ is the vector of weights that need to be fitted, π_1 and π_2 are just 2 parses among the 50 best produced by the parser. For the training data, 10-fold cross-validation is used to compute the 50-best parses for each sentence s in the training set. And we train the reranker to select the best parse according to the f-score of the 50-best parses. (The best parse selected by the reranker need not be the correct parse, because the 50-best parses may not include the correct parse.) After fitting the parameter θ, the reranker is applied on the 50 best parses for the test sentences, and select the one parse with highest probability. This is the same as selecting the parse with highest $\theta \cdot f$.

In practice, a penalty term $J(\theta) = c||\theta||_2$ must be used to prevent overfitting. So the final objective function that need to be minimized during training is

$$-\sum_i \log p_\theta(\pi_i^b) + J(\theta)$$

, where π_i^b is the best parse among the 50 best according to the f-score. There are a large number of features selected during training and that really slows down the reranker. Automatic feature selection can be achieved by using in the penalty term L_1 norm of θ instead of the L_2 norm $J(\theta) = c||\theta||_1$, but this possibility is not explored in this experiment.

4 Character-based Parsing

One major difference between Chinese and most western languages is that the words in Chinese is not delimited by white-spaces. There has been significant research on Chinese word segmentation. In this work, we directly apply the maximum-entropy-inspired parser on the treebank by first transforming the treebank as stated in the following.

We convert the original parse tree into a tree in which the terminals consist of a single character instead of words. For any tag X in the original Treebank, we add 4 additional tags: Xf, Xl, Xm, and Xs. Xf is the tag for the first character of a multi-character word, Xl is the tag for the last character of a multi-character word, Xm is the tag for the characters in between. Finally, we use Xs as the tag for a single-character word. Now the original tags becomes non-terminals in the new tree.

After transforming the training parse trees in this way, we can then directly apply the parser on the transformed treebank and everything goes through as before.

5 Experimental Result

We use both CTB1.0 (3485 sentences in total) and CTB4.0(12334 sentences in total) in the experiment. The treebank is divided into training set, test set, and development test set with the same splitting as in [1]. The development set is used in the EM algorithm to compute the weights for the expected-frequency interpolation weights of conditional probabilities [2]. Final results are summarized in table 1. For comparison, the results in [1] are reproduced in table 2. We can see that the our parser performs marginally better in all cases. The reranker further increases the f-score by about 1.4%(table 3). For the character-based experiment, the result is compared with [7], which is the only other character-based parser we found.

6 Conclusion

We have reported the results we get by applying the maximum-entropy-inspired parser on the Penn CTB. The performance we observe is better than previously obtained results. The MaxEnt reranker on the 50-best parser gives slightly better performance but requires much more additional computation time. Also, the treebank is converted so that the parser can be applied in a character-based approach, so the word segmentation task is subsumed under the framework of parsing. Character-based parsing is an important problem that current algorithms cannot produce satisfactory result on, and deserves more research effort. The overall result is significantly worse than that on the English treebank. Hopefully the availability of a higher quality tagging and bracketing treebank would lead to more encouraging results.

Appendix A

Here we list a few changes that need to be done in order for the maximum-entropy-inspired parser to work on the Chinese Treebank.
1. There are sentences in the Chinese Treebank that consist of 2 sub-sentences , so the bracketing looks like ((IP ...) (IP ...)). The code needs to be changed in order to read in this kind of trees.
2. The end character of a Chinese word is used to guess the POS if the word is not seen in the training set. Since the treebank files are GB coded, we should use a **string** to store the Chinese character instead of **char**.
3. Chinese has a different punctuation system and that needs to be changed whenever punctuation is used in the program. This include ccInd.C, tree_noopenQl/r in treeHistSf.C/edgeSubFns.C/fhSubFns.C, scorePunctuation() in InputTree.C, finalPunc() and effEnd() in ChartBase.C, ccInd() in Edge.C

Treebank	≤ 40 words		
	LR	LP	F
1.0	79.9	81.9	80.9
4.0	77.5	81.4	79.4
	all sentences		
	LR	LP	F
1.0	75.6	78.6	77.1
4.0	75.0	79.1	77.0

Table 1: Parsing results of the maximun-entropy-inspired parser.

Treebank	≤ 40 words		
	LR	LP	F
1.0	78.0	81.2	79.6
4.0	76.9	81.1	78.9
	all sentences		
	LR	LP	F
1.0	74.4	78.5	76.4
4.0	74.7	79.0	76.8

Table 2: Parsing result from Dan Bikel's parser.

Treebank	Parser Result	Reranker Result
1.0	77.1	78.4
4.0	77.0	78.4

Table 3: Reranking results on Charniak 50-best parses. Only the f-score is reported here.

Treebank	LR	LP	F
1.0	68.8	76.2	70.7
4.0	67.6	71.7	69.6

Table 4: Parsing results of the maximum-entropy-inspired parser on unsegmented sentences

Treebank	Parser	LR	LP	F
1.0	This Report	77.8	79.7	78.8
1.0	Fung04	76.1	74.4	75.2

Table 5: Parsing result of the maximum-entropy-inspired parser on unsegmented sentences, compared to Fung's result. The numbers reported consider POS-tagged words to be constituents.

4. The code in trainRs.C assumes that there are at least 500 sentences for the EM algorithm, which is not always available for small treebank.
5. I also implemented a different headFinder.C, using the head finding rule in [1].
6. In CTB, the punctuation is tagged with **PU**, it would be better to use the actual punctuation as the tag, as in the English treebank.

References

[1] Bikel, D. 2004. On the Parameter Space of Lexicalized Statistical Parsing Models. *Ph.D Thesis, University of Pennsylvania*

[2] Charniak, E. 1996. Expected-frequency interpolation. Department of Computer Science, Brown Univerisity, Technical Report CS96-37, 1996

[3] Charniak, E. 2000. A maximum-entropy-inspired parser. In *The Proceedings of the North American Chapter of the Association for Computational Linguistics,* 132-139

[4] Charniak, E. and Johson, M. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking.

[5] Collins, M. 1997. Three generative lexicalized models for statistcal parsing. In *Proceedings of the 35th Annual Meeting of the ACL.* 16-23.

[6] Collins, M. 2000. Discriminative reranking for natural language parsing. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML 2000),* 175-182

[7] Fung, P. and Ngai, G. et al. 2004. A maximum entropy Chinese parser augmented with transformation-based learning. In *ACM Transactions on Asian Language Processing,* 3(2), 159-168, 2004

[8] Ratnaparkhi, A. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning 34(1999), 151-175*

[9] Tappen, M. and Freeman, W et al. 2002. Recovering intrinsic images from a single image. MIT AI Lab Technical Report 2002-015, 2002.

[10] Xia, F. and Palmer, M. et al. 2000. Developing guidelines and ensuring consistency for Chinese text annotation. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation* Athens, 2000