Unsupervised Chinese Phrase Parsing Based on Tree Pattern Mining

Xiaotian Zhang
Shanghai Jiao Tong University
xtian.zh@gmail.com

Hai Zhao
Shanghai Jiao Tong University
zhaohai@cs.sjtu.edu.cn

Abstract

This paper investigates unsupervised phrase parsing for Chinese. It is the first time according to our best knowledge that unsupervised data oriented (U-DOP) model described in (Bod06) is fully and exactly re-implemented. Our U-DOP implementation achieves the similar results on CTB as reported in (Bod06). Moreover, using the evaluation measure in (Bod07), our system achieves the highest F1 score among all the existing POS-sequence based unsupervised phrase parsing models. We also give a detailed comparison between the performance of our U-DOP system and the unsupervised CCL parser (Seg07) in terms of prediction accuracy on different kinds of phrases.

1 Introduction

Unsupervised syntactic parsing can be thought as finding patterns in the training data and learn a probabilistic model of the syntactic structure. There exists mainly four unsupervised parsing models CCM (KM02), DMV (KM04), UDOP (Bod06), CCL (Seg07). One major difference lies in the different definition of component when assigning probability to a syntactic structure. In the case of CCM (KM02), probability is assigned by the constituent and non-constituent sequences of parts-of-speech in the structure as well as the contexts of these sequences. In the case of U-DOP (Bod06), the probability of a syntactic structure is the product of the probabilities of the subtrees. In the case of DMV, the probability is assigned to the two parts-of-speech being joined by a link. As for CCL, a new link based representation of syntactic structure is developed and probability of the new links is calculated. Among these unsupervised parsing models, UDOP is an all subtree approach which achieves higher F score than CCM, and DMV, but not comparable with CCL because CCL parses from plain text while the others parse from pos sequences.

A major part of the U-DOP is the n-best CYK parsing. According to (HC05), they developed three algorithm for n-best parsing and the third one is a generation of the algorithm developed by (JM00). Based on their comparison, the third algorithm beat the other two in average parsing speed and only has an advantage over Jimenez's algorithm in heap size when n is less than 64 in their experiments. Because we want to calculate the top 100 best trees when parsing, we choose to implement Jimenez's algorithm which is easier to implement. As described in (JM00). The main idea of this algorithm is that firstly the best parse tree is computed using the CYK algorithm, and then a large number of alternative parse trees in order by weight (or probability) can be obtained and this takes only a small fraction of the time required by the first step.

In this paper, we implement U-DOP as described in (Bod06). It is the first time according to our best knowledge that U-DOP is fully re-implemented.

2 Implementation of U-DOP [1]

There are three steps in U-DOP model.

[1]The implementation could be downloaded from http://sourceforge.net/projects/udop/

The first step of U-DOP is to enumerate all possible binary trees for training sentences and generate PCFG rules. For example, there will be five binary trees in all for the pos tag sequence "NNS VBD JJ NNS". For all the training sentences, calculate all the possible binary trees and sample from these generated trees randomly because the whole possible trees will be too numerous to handle with. As in (Bod06), for sentences of 7 words we randomly sample 60% of the trees, and for sentences of 8, 9 and 10 words we sample respectively 30%, 15% and 7.5% of the trees. Then PCFG rules are generated from the sampled trees as (Goo96). For each nonterminal node A_j with children B_k and C_l in the tree, eight PCFG rules will be generated as follows:

$$\begin{array}{ll} A_j \rightarrow BC(1/a_j) & A \rightarrow BC(1/a) \\ A_j \rightarrow B_kC(b_k/a_j) & A \rightarrow B_kC(b_k/a) \\ A_j \rightarrow BC_l(c_l/a_j) & A \rightarrow BC_l(c_l/a) \\ A_j \rightarrow B_kC_l(b_kc_l/a_j) & A \rightarrow B_kC_l(b_kc_l/a) \end{array}$$

where b_k and c_l are the number of non-trivial subtrees of B_k and C_l respectively and $a_j = (b_k + 1)(c_l + 1)$. According to the properties of the PCFG rules, each derivation of the subtrees will have a isomorphic PCFG derivation with equal probability, so a n-best CYK parsing algorithm could be used to find the n-best parses based on the PCFG rules.

Since the estimator of DOP is biased and inconsistent as pointed out in (Joh02), we rectified it as described in (Bod03) by adding the correction factor α which is the number of times nonterminals of type A occur in the training data, and experiment shows this correction factor increases the F score by 13% on CTB.

$$\begin{array}{ll} A_j \rightarrow BC(1/a_j) & A \rightarrow BC(1/a\alpha) \\ A_j \rightarrow B_kC(b_k/a_j) & A \rightarrow B_kC(b_k/a\alpha) \\ A_j \rightarrow BC_l(c_l/a_j) & A \rightarrow BC_l(c_l/a\alpha) \\ A_j \rightarrow B_kC_l(b_kc_l/a_j) & A \rightarrow B_kC_l(b_kc_l/a\alpha) \end{array}$$

Since Bod didn't clarify that whether the rules are generated from pos tag sequences with or without punctuation, we tried to generate $PCFG$ rules from WSJ10 both with and without punctuation respectively and find experiment using punctuation could reproduce $14.8 * 10^6$ distinct PCFG rules mentioned in (Bod06). And moreover, experiment shows the performance of using PCFG rules including punctuation marks is better than that of using rules generated from pos tag sequences excluding punctuation marks, which indicates the use of punctuation in predicting syntactic structures. We extract $6.2 * 10^6$ rules from CTB10 v3.0 in all.

When tested on the test sentence, U-DOP will find the n best most probable parses for the test pos tag sequences by the CYK n-best parsing algorithm using the PCFG rules generated. Here we adapt the CYK n best algorithm described in (JM00). The main idea of this algorithm is that after the best parse tree has been computed using the CYK algorithm, a large number of alternative parse trees in order by weight (or probability) can be obtained in a small fraction of the time required by the CYK algorithm to find the best parse tree. Firstly, we have to take the negative log of the probabilities of the rules and thus convert the problem to finding the trees with n smallest weights.

We use $A_{i:k}$ to denote the nonterminal node in the tree which dominates from the ith pos tag to the kth pos tag. $\Upsilon^n(A_{i:k})$ denotes the candidate set of nonterminal node $A_{i:k}$ and $T^n(A_{i:k})$ denotes the nth best tree which is the smallest one among $\Upsilon^n(A_{i:k})$. Use CYK parsing algorithm to get the the most probable tree $T^1(A_{i:k})$ and then use the following recursive equations to calculate the $T^n(A_{i:k})$ from $T^{n-1}(A_{i:k})$.

Let

$$\begin{aligned} \Upsilon^1(A_{i:k}) = \{ < A_{i:k}, T^1(B_{i:j}), T^1(C_{j+1:k}) > \\ : \exists A \rightarrow BC \, \forall \, j \, s.t \, i \leq j < k \} \end{aligned} \tag{1}$$

For $n > 1$, let us assume that

$$T^{n-1}(A_{i:k}) =< A_{i:k}, T^p(B_{i:j}), T^q(C_{j+1:k}) > \tag{2}$$

Then if $q = 1$

$$\begin{aligned} \Upsilon^n(A_{i:k}) = (\Upsilon^{n-1}(A_{i:k}) - T^{n-1}(A_{i:k})) \\ \cup \{ < A_{i:k}, T^p(B_{i:j}), T^{q+1}(C_{j+1:k}) > \} \\ \cup \{ < A_{i:k}, T^{p+1}(B_{i:j}), T^q(C_{j+1:k}) > \} \end{aligned} \tag{3}$$

otherwise

$$\begin{aligned} \Upsilon^n(A_{i:k}) = (\Upsilon^{n-1}(A_{i:k}) - T^{n-1}(A_{i:k})) \\ \cup \{ < A_{i:k}, T^p(B_{i:j}), T^{q+1}(C_{j+1:k}) > \} \end{aligned} \tag{4}$$

Then we have

$$T^n(A_{i:k}) = argmin_{T \in \Upsilon^n(A_{i:k})} W(T)$$

After calculating the top 100 best trees, the last step of U-DOP is to sum the probabilities of the trees of the same structure, because different deviation of subtrees may lead to parses of the same structure. Then the tree structure with the largest probability is the predicted phrase structure of the test sentence.

3 Experiments

We evaluate the performance of U-DOP as described in (Bod06; Kle05). The definitions of unlabeled precision (UP) and recall (UR) of a proposed corpus $P = [P_i]$ against a gold corpus $G = [Gi]$ are:

$$UP(P,G) \equiv \frac{\sum_i |brackets(P_i) \cap brackets(G_i)|}{\sum_i |brackets(P_i)|}$$

$$UR(P,G) \equiv \frac{\sum_i |brackets(P_i) \cap brackets(G_i)|}{\sum_i |brackets(G_i)|} \quad (5)$$

$$UF_1(P,G) = \frac{2}{UP(P,G)^{-1} + UR(P,G)^{-1}}$$

Moreover, Bod pointed out in MLCS07, London that " In evaluating U-DOP, we had to binarize test-set trees in Penn Treebank otherwise the f-scores become meaningless (Kle05; KM02; KM04)" . Because the test-set trees are more flat than the binary predicted trees by U-DOP, it tends to have lower precision and higher recall if evaluated as above without binarizing the test set. And testing on binarized test-set will certainly increase the number of right brackets predicted. However, on one hand, the results of the CCM and DMV model are reproduced without binarizing the test set by a current available open source implementation[2] and the Figure 6 in (KM05) also implies that CCM is not evaluated on binarized test set. On the other hand, all binary branching bracketings of a sentence have the same number of brackets, and thus evaluation on binarized test-set trees will lead to equal UP and UR if UR is equal to the number of brackets shared by the predict tree and binarized test tree divided by the total binarized gold brackets, which is contradictory to the results reported in (Bod06; Kle05). Otherwise, if UR is calculated via dividing the number of shared brackets by the number of test-set brackets before binarizing, the recall score is also meaningless since binarizing generates new brackets that don't exist in the gold brackets.

So a meaningful evaluation measure using binarized test-set is that UR is still calculated as above while UP is calculated as follows,

$$UP(P,G) \equiv \frac{\sum_i |brackets(P_i) \cap brackets(B_i)|}{\sum_i |brackets(P_i)|} \quad (6)$$

where $B = [B_i]$ is the binarized version of the gold corpus [3]. We just use right branching of the test-set trees.

model	UP	UR	F1
CCM	34.6	64.3	45.0
DMV	35.9	66.7	46.7
DMV+CCM	33.3	62.0	43.3
UDOP(Bod)	36.3	64.9	46.6
UDOP*			42.8
$UDOP_{non-binarized}(ours)$	34.4	53.2	41.8
$UDOP_{binarized}(ours)$	42.3	53.2	47.1
CCL	50.1	51.1	50.6

Table 1: Test Results on CTBv3.0

model	UP	UR	F1
$UDOP_{non-binarized}(ours)$	40.0	55.7	46.5

Table 2: Test Results on CTBv5.0

We use the version 3.0 of CTB to test our implementation as in (KM02; KM04) and compare the F1 score of different models in Table 1.[4]

The experiment on CTB10 v3.0 (about 2137 sentences) takes nearly 52 hours on a computer with Intel core X5560@2.80GHz. We also test our system on version 5.0 of CTB and achieve F score of 46.5. The UP is calculated by equation 5 in $UDOP_{non-binarized}$ and by equation 6 in $UDOP_{binarized}$.

[2]It is available from http://www.cs.famaf.unc.edu.ar/ francolq/en/proyectos/dmvccm

[3]The details on how to binarize the test-set trees are not given in (Bod06)

[4]The version of CTB used in (Bod06) is not identified.

phrase	percent	U-DOP	CCL	shared
CP	4.0%	0.146	0.099	0.012
DNP	2.7%	0.246	0.234	0
NP	13.2%	0.396	0.209	0.125
NP-OBJ	9.2%	0.319	0.271	0.114
NP-PN	2.5%	0.198	0.154	0.037
NP-SBJ	10.9%	0.372	0.321	0.156
PP	1.4%	0.244	0.100	0.022
QP	2.9%	0.750	0.190	0.168
VP	37.2%	0.228	0.316	0.113

Table 3: The Prediction Accuracy of Major Kinds of Phrases by U-DOP and CCL On CTB10 v3.0

Here we also record the accuracy of each kind of phrases predicted by U-DOP (my implementation) and CCL(Seg07) when testing on CTB10v3.0. The fifth column of Table 3 is the percentage of brackets shared by U-DOP and CCL in predicting each kind of phrases. From Table 3, we could see that U-DOP performs much worse in predicting VP than CCL when tested on CTB. Since VP takes up the highest proportion among all kinds of phases, U-DOP is inferior to CCL in average even if it performs better in most of other aspects when tested on CTB.

4 Conclusion

In all, we report our implementation of U-DOP in detail especially in terms of dealing punctuation, CYK n-best parsing algorithm, and evaluation measures. Experiment results show our implementation achieves the higher f-score that others if evaluated acording to (Bod07). Moreover, the comparison between U-DOP and CCL shows the defect of U-DOP in predicting VP.

References

Rens Bod. An efficient implementation of a new dop model. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, pages 19–26, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

Rens Bod. Unsupervised parsing with u-dop. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 85–92, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Rens Bod. Linguistic relevance of unsupervised data-oriented parsing. In *Machine Learning and Cognitive Science of Language Acquisition*, June 2007.

Joshua Goodman. Efficient algorithms for parsing the dop model, 1996.

Liang Huang and David Chiang. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Parsing '05, pages 53–64, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

Víctor M. Jiménez and Andrés Marzal. Computation of the n best parse trees for weighted and stochastic context-free grammars. In *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*, pages 183–192, London, UK, 2000. Springer-Verlag.

Mark Johnson. Squibs and discussions: the dop estimation method is biased and inconsistent. *Comput. Linguist.*, 28:71–76, March 2002.

Dan Klein. *The unsupervised learning of natural language structure*. PhD thesis, Stanford, CA, USA, 2005. AAI3162386.

Dan Klein and Christopher D. Manning. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 128–135, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. Corpus-based induction of syntactic structure: models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. Natural language grammar induction with a generative constituent-context model. *Pattern Recogn.*, 38:1407–1419, September 2005.

Yoav Seginer. Fast unsupervised incremental parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 384–391, 2007.