AUTOMATIC GRAMMAR PARTITIONING FOR SYNTACTIC PARSING[1]

Po Chui Luk*, Fuliang Weng, and Helen Meng***

*****Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong, Hong Kong, China
**Robert Bosch Corporation
Research and Technology Center, Palo Alto, CA94304
{pcluk, hmmeng}@se.cuhk.edu.hk, fuliang.weng@rtc.bosch.com

Abstract

This paper begins by reviewing an effort in grammar decomposition (or grammar partitioning) for natural language to achieve modular parsing. We then propose a novel automatic approach for grammar partitioning, instead of the manual partitioning used in systems. Experiments are conducted with GLR parsing for the Wall Street Journal (WSJ) corpus in the Penn Treebank where single grammar GLR parsing is impractical. Our results show that the automatic grammar partitioning method based on the mutual information criterion fares better than a random partitioning method, and exhibits efficiency in parsing and high parse coverage.

1 Introduction

Due to the complexity of natural language itself, the grammar that describes the language can be very complex. The traditional way of using a single parser and single grammar leads to inefficiency and complication in natural language system. Using the co-occurrence feature in the natural language, grammar rules can be automatically clustered into sub-grammars to achieve modular parsing.

Studies are related to grammar decomposition for modular parsing have been reported recently. Amtrup [1] introduced an approach that distributes grammar rule applications in multiple processors within a chart parser framework. Zajac et al [9] presented a modular unification-based parsing, a monolithic grammar is decomposed into sub-grammars by constituent classification, and sub-grammars are combined to build a parsing application. Basili et al [2] proposed a modular parsing framework through the idea of grammar stratification (verb subcategorization). However, their grammar decomposition is conducted manually.

Our previous work showed that grammar partitioning helps reduce the overall parsing table size when compared to using a single grammar in Air Travel Information Systems (ATIS) domain [4]. In this paper, we propose an *automatic grammar partitioning* method for syntactic parsing based on the WSJ sentences in the Penn Treebank. We use the sub-parsers with partitioned grammars to parse the test data in the WSJ.

2 Grammar Partitioning

2.1 Definitions and Concepts

We first review some definitions and concepts used in the paper for easier readability, focusing on context-free grammars (CFG)[7]. The entire grammar is partitioned into sub-grammars based on its production rules. The interaction among different sub-grammars is through non-terminal sets, i.e. INPUT and OUTPUT, and a *virtual terminal* technique. The virtual terminal *vt* is essentially a non-terminal, but acts as if it were a terminal. The INPUT of a sub-grammar is a set of non-terminals that were previously parsed by other sub-grammars. The OUTPUT of a sub-grammar are those non-terminals that were parsed based on this sub-grammar and used by other sub-grammars as their INPUT symbols.

2.2 Automatic Grammar Partitioning Method

Based on the discussion of [8], grammar partitioning can be seen as the reverse process of grammar clustering. Our partitioning procedure begins with the set of the finest grammar partition, i.e., each partition contains exactly one grammar rule. We form a partition for every grammar rule. For instance, if a grammar partition only contains Rule 3, the INPUT set of Rule 3 is the non-terminal NP and the OUTPUT set is

[1] This work is supported by Intel Corp., and it was carried out while the second author was with Intel China Research Center.

PP-DIR. For training on the parsed sentence of Penn Treebank in Figure 1, we use a calling matrix in Figure 2 to record the calling frequencies between sub-grammars. The OUTPUT and INPUT of the sub-grammars are in the columns and rows of the matrix respectively. For the calling matrix, the entry at row *i* and column *j* is the frequency of sub-grammar *i* calling sub-grammar *j*. We start with sub-grammars having an empty INPUT set (zero row in the calling matrix), duplicate and merge them for all of its caller sub-grammars.

2.3 Calculating Mutual Information

We intend the partitioned grammars to have close interactions within each sub-grammar, but infrequent interactions between different sub-grammars. In this paper, we use the Mutual Information of caller/callee co-occurrences between two sub-grammars to measure the closeness of the interactions [3].

Suppose G_i and G_j are two sub-grammars in a grammar partition with n sub-grammars, the mutual information of these two sub-grammars is defined as:

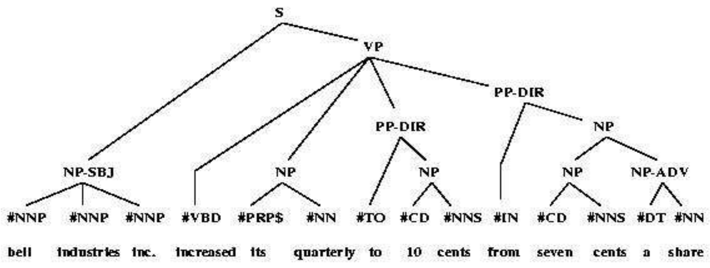
$$MI(G_i,G_j) = \log\frac{P(G_i,G_j)}{P(G_i,*)P(*,G_j)} = \log\frac{Freq(G_i,G_j)}{\sum_{j=1}^{n} Freq(G_i,G_j)\sum_{i=1}^{n} Freq(G_i,G_j)} + \log\sum_{i,j=1}^{n} Freq(G_i,G_j) \quad (1)$$

Where, *Freq*(G_i, G_j) and $P(G_i,G_j)$ are the frequency and probability of G_i calling G_j, $P(G_{i,*})$ the probability of G_i being a caller, and $P(*,G_j)$ the probability of G_j being a callee, as shown in following equations.

$$P(G_i,G_j) = \frac{Freq(G_i,G_j)}{\sum_{i,j=1}^{n} Freq(G_i,G_j)} \qquad P(G_i,*) = \frac{\sum_{j=1}^{n} Freq(G_i,G_j)}{\sum_{i,j=1}^{n} Freq(G_i,G_j)} \qquad P(*,G_j) = \frac{\sum_{i=1}^{n} Freq(G_i,G_j)}{\sum_{i,j=1}^{n} Freq(G_i,G_j)}$$

For each merge iteration, we find the maximum value of $MI(G_i, G_j)$ in the calling matrix, and merge the corresponding sub-grammars pair together to form a larger sub-grammar. For example, in Figure 1, partition 3 (Rule 3: PP-DIR → #IN NP) is merged with partition 4 (Rule 4: NP → NP NP-ADV), because they have the highest Mutual Information, $MI(G_3, G_4)$, in the matrix of Figure 2. The merge operation is iterated until the process reaches the maximum number of 2,000 iterations. To prevent partitions being too large, a size up-bound is used for stopping merges during the iterations. To prevent partitions being too small, in post-processing stage, we further merge together the very small ones that share the same OUTPUT set. The grammar size is defined as follows with *length(x)* is the length of string *x*:

$$|G| = \sum_{(A\to\alpha)\in P} length(A\alpha) \quad (2)$$


S
VP
PP-DIR
PP-DIR
NP
NP-SBJ
NP
NP
NP
NP-ADV
#NNP #NNP #NNP #VBD #PRP$ #NN #TO #CD #NNS #IN #CD #NNS #DT #NN
bell industries inc. increased its quarterly to 10 cents from seven cents a share

1: S → NP-SBJ VP
2: VP → #VBD NP PP-DIR PP-DIR
3: PP-DIR → #IN NP
4: NP → NP NP-ADV
5: NP-ADV → #DT #NN
6: PP-DIR → #TO NP
7: NP → #CD #NNS
8: NP → #PRP$ #NN
9: NP-SBJ → #NNP #NNP #NNP

Figure 1: An example parse tree drawn from the WSJ sentences and its corresponding grammar rules.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	Σ
1	0	1	0	0	0	0	0	0	1	*2*
2	0	0	1	0	0	1	0	1	0	*3*
3	0	0	0	1	0	0	0	0	0	*1*
4	0	0	0	0	1	0	1	0	0	*2*
5	0	0	0	0	0	0	0	0	0	*0*
6	0	0	0	0	0	0	1	0	0	*1*
7	0	0	0	0	0	0	0	0	0	*0*
8	0	0	0	0	0	0	0	0	0	*0*
9	0	0	0	0	0	0	0	0	0	*0*
Σ	*0*	*1*	*1*	*1*	*1*	*1*	*2*	*1*	*1*	*9*

$MI(G_1, G_2)$	$=\log[1/(2\times1)]+\log(9)$
$MI(G_1, G_9)$	$=\log[1/(2\times1)]+\log(9)$
$MI(G_2, G_3)$	$=\log[1/(3\times1)]+\log(9)$
$MI(G_2, G_6)$	$=\log[1/(3\times1)]+\log(9)$
$MI(G_2, G_8)$	$=\log[1/(3\times1)]+\log(9)$
$\mathbf{MI(G_3, G_4)}$	$\mathbf{=\log[1/(1\times1)]+\log(9)}$
$MI(G_4, G_5)$	$=\log[1/(2\times1)]+\log(9)$
$MI(G_4, G_7)$	$=\log[1/(2\times2)]+\log(9)$
$MI(G_6, G_7)$	$=\log[1/(1\times2)]+\log(9)$

Figure 2: The calling matrix of the grammar and its computed mutual information.

3 Composing Sub-parsers for Multiple Grammars

In our system, each sub-grammar has its own LR(1) parsing table and GLR parser, and all the sub-parsers are composed to obtain an overall parse of the input sentences through a lattice with multiple granularities (LMG), an interface to record the INPUTs and OUTPUTs of all sub-parsers [8]. The particular parser composition method used is the bottom-up cascading algorithm.

To improve the efficiency, the left-corner virtual terminals frequencies are collected from the training data for each sub-grammar. Given an incoming input edge on an LMG, this frequency information is used to rank the priority of invocation of sub-parsers in an N-best list. The sub-parsers that are in the N-best list are invoked, and all the rest sub-parsers are discarded. However, each sub-parser could end at any edge in the LMG. To avoid creating too many new edges on the LMG for each parser invocation, we only select one resultant parse tree as an output parse if the parser returns successfully.

4 Experiments

We only report the comparison between GLR parser sizes of automatic partitioned grammars and the sizes of a randomly partitioned grammar partition. This is similar to the LALR(1) case in [5], where excessive long time is required to compute GLR parsing tables for the monolithic Penn Treebank grammar, and therefore the single GLR parser is not included in the comparison.

4.1 Automatic Grammar Partitioning by the Mutual Information Criterion (MI)

Automatic grammar clustering was trained on sections 02 to 21 of the Wall Street Journal of Penn Treebank, which contains 41,603 sentences. There are totally 20,696 rules extracted from the training sentences. The maximum sub-grammar size is set to 1,000. There are two parameters for the iterative merge operation: the minimum calling frequency of two sub-grammars to be merged and the maximum number of iterations. The former is set to 4 in order to avoid sparse data, and the latter to 2,000.[2] This way, we obtained 357 sub-grammars.

4.2 Random Grammar Partitioning without any Heuristics (Random)

As a controlled experiment, we randomly cluster the extracted grammar rules into clusters. To obtain a comparable number of clusters as the one using the mutual information criterion, we partition the 20,696 rules into 351 sub-grammars with 59 rules per cluster. No rule duplicate in any clusters.

4.3 Parser Sizes

We used the same LR(1) parsing table generator to construct LR(1) tables for these two partitions generated using the above two methods. The size of the parsers for a grammar partition is the sum of the number of states in all sub-parsers. The sizes for the automatic partitioning and the random partitioning are 231,907 and 469,484 respectively. So, the former is only less than half of the latter. Figure 3 show the histograms of the number of states and the number of states/rule in the two grammar partitions. In both cases, the grammar partitioned with MI is much smaller in both measures (number of states, and states/rule). This clearly indicates that using MI for grammar partitioning, the overall parsing table sizes can be much saved. Notice that the smaller numbers of states and the ratios imply good determinism of the sub-parsers.

4.4 Parsing Results

We use our multi-parser framework to parse the WSJ sentences in Penn Treebank's section 23. 2,403 sentences with length no more than 50 words are selected from section 23. The average input sentence length is 20.4 POS tags. For the N-best list described in section 3, N is set to 3 so that only the top 3 ranked sub-parsers are used for invocation in both partitions. The parsing output of each input sentence is an LMG. Since there may be multiple paths in the LMG, a Viterbi algorithm is used to find the shortest path in the LMG to represent the sentence. Table 1 shows the parsing coverage and speed on the test set. Full parse means there is a parse tree covering the whole input query. The experiment was performed on Intel Pentium III 1GHz CPU, 512 Mbytes memory. There is tradeoff between the parse coverage and the parsing speed.

[2] If we set the maximum sub-grammar size be 500 and minimum calling frequency be 4, the iterative merge operation will end around 2000 iterations (no more two sub-grammars can be merged). To control the processing time for merging, the maximum number of iterations allowed is capped at 2000 and the maximum grammar size is capped at 1000.

The low parse coverage of random partitioned grammars suggests using leftmost frequencies is not enough to give a good prediction. Our further experiment will study the changes in parsing results if increasing N.

	MI	**Random**
Full parse	92.1%	45.3%
Parsing speed	4.19seconds/sentence	0.68 seconds/sentence

Table 1: The parsing results of the WSJ test data (section 23) in the Penn Treebank.

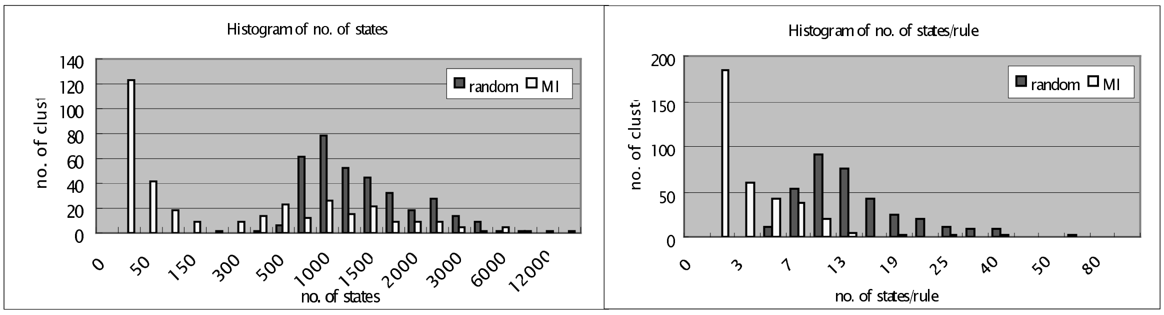


Figure 3: The histograms of number of states and states per rule in the partitioned grammar set.

5 Conclusions

In this paper, we have presented our automatic grammar partitioning algorithm and experimented with the WSJ sentences of the Penn Treebank. Grammar rules are derived from the parse trees in the training data. In our novel automatic grammar partitioning algorithm, Mutual Information criterion is used to cluster rules to form sub-grammars. Experimental results show that this automatic grammar partitioning approach can reduce the parsing table size by more than 50%, compared with a random decomposition method. When the partitioned grammars are used for GLR parsers to parse the testing data, the results are encouraging on two counts: (1) the GLR parser was impractical using a monolithic grammar for Penn Treebank task, while it is quite feasible using our automatic partitioned approach; and (2) we achieved a parse coverage of 92% for the test set with the automatic partitioning method, which suggests that our parsing framework is generalizable to unseen test data.

References

[1] Amtrup, J., "Parallel Parsing: Different Distribution Schemata for Charts", In Proceedings of the 4th IWPT, Prague, Sep 1995.

[2] Basili, R., Pazienza, M.T. and Zanzotto, F.M., "Customizable Modular Lexicalized Parsing", In Proceedings of 6th IWPT, February, 2000

[3] Church, K. and Hanks, P., "Word Association Norms, Mutual Information, and Lexicography". In Computational Linguistics, Vol. 16, No. 1., pages 22-29, 1990.

[4] Luk, P.C., Meng, H. and Weng, F., "Grammar Partitioning and Parser Composition for Natural Language Understanding", In Proceedings of the ICSLP, October, 2000.

[5] Moore, R.C., "Improved Left-Corner Chart Parsing for Large Context-Free Grammars", In Proceedings of 6th IWPT, February 2000.

[6] Tomita, M., Efficient Parsing for Natural Language, Kluwer Academic Publishers, MA, 1985.

[7] Weng, F.L. and Stolcke, A. "Partitioning Grammars and Composing Parsers", In Proceedings of 4th IWPT, 1995. For the full paper, see web page http://www.speech.sri.com/people/fuliang/publications.html.

[8] Weng, F.L., Meng, H. and Luk, P.C., "Parsing a Lattice with Multiple Grammars", In Proceedings of 6th IWPT, February 2000.

[9] Zajac, R. and Amtrup, J.W., "Modular Unification-Based Parsers", In Proceedings of 6th IWPT, February, 2000.