GRAMMAR PARTITIONING AND PARSER COMPOSITION FOR NATURAL LANGUAGE UNDERSTANDING

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

*Human-Computer Communications Laboratory
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
**Intel China Research Center
{pcluk, hmmeng}@se.cuhk.edu.hk, fuliang.weng@intel.com

ABSTRACT

This paper presents an approach for natural language understanding, which integrates *multiple* sub-grammars and sub-parsers; in contrast with the traditional *single* grammar and parser approach. The use of GLR(k) parsers for natural language understanding is hampered by the problem of exponential growth of the parsing table size as the size of grammar rules increases. Hence, we propose to *partition* a grammar into multiple sub-grammars. For each sub-grammar we generate its own parsing table together with its specialized GLR sub-parser. The total size of the sub-grammars' parsing tables is much smaller than the size of the parsing table of the unpartitioned grammar. A *parser composition algorithm* then combines the sub-parsers' outputs to produce an overall parse that is identical to that produced by a single parser. Results based on natural language queries in the Air Travel Information System (ATIS) domain shows that this is a viable and efficient approach applicable to *both* English and Chinese.

1. INTRODUCTION

Modular parsing architectures are receiving an increasing amount of attention. Among others, Abney [1] proposed a two-level chunking parser, which converts an input sentence into the chunks and then uses an attacher to convert these chunks into a parse tree. Amtrup [2] introduced an approach that distributes grammar rule applications in multiple processors within a chart parser framework. Weng and Stolcke [11] presented a general schema for partitioning a grammar into sub-grammars and the combination of parsers for sub-grammar to achieve modular parsing. Ruland et al [7] developed a multi-parser multi-strategy architecture for noisy spoken languages.

The Generalized LR (GLR) parser is an efficient parser that can be used to handle context-free grammars (CFGs), a backbone for many natural language processing systems. However, Earley [3] has proved that the number of states of the LR parsers could grow exponentially with the size of a CFG[1]. This hampers the use of the GLR parser for large-scale natural language systems. To tackle this problem, we propose to adopt the *grammar partitioning* approach, where a large grammar is partitioned into multiple sub-grammars. This reduces the total number of states of the parser, as well as the computational time for generating the parsing table. This parsing architecture is modular, since all the parsing tables of the sub-parsers are constructed separately. Hence if any sub-grammar needs to be modified, we only need to regenerate its corresponding parsing table, instead of the parsing table for the whole grammar. This eases the process of grammar development, promotes (sub-) grammar reuse, and consequently enhances the scalability of natural language understanding systems to more complex domains, and the portability across application domains. In this work we use GLR parsers as our basic parsing mechanism. Each sub-grammar has its own corresponding sub-parser, and we use a *parser composition* approach [10] to combine the sub-parser outputs to produce an overall parser for the input sentences. This paper explores the advantages of our grammar partitioning and parser composition framework, in comparison with the conventional un-partitioned grammar and single parser approach.

2. GRAMMAR PARTITIONING

We use the definitions in Weng and Stolcke [11] for grammar partitioning, a generalization of Korenjak's [5]. For simplicity, we partition the grammar based on the non-terminals of a CFG. The concept of *virtual terminal* is introduced, and denoted with prefix *vt*. The virtual terminal is essentially a non-terminal, but acts as if it were a terminal. Suppose we need to partition grammar G in Figure 1. We select non-terminal *NP* for partitioning. The production rules with *NP* in their left-hand side (LHS) are distinguished from the original entire grammar. Then all *NP* in the right-hand side (RHS) of the original grammar rules are replaced by the virtual terminal symbol *vtNP*. These two sets of grammar rules are converted into the reduced form.[2] As a result, the grammar is partitioned into two sub-grammars as shown in Figure 1. The interaction among different sub-grammars is through non-terminal sets – INPUT and OUTPUT. For a sub-grammar, its INPUT is a set of virtual terminals that were previously parsed by other sub-grammars. The OUTPUT of a sub-grammar is a (set of) non-terminal(s) that were parsed based on this sub-grammar and used by other sub-grammars as their INPUT symbols. In other words, we may view a partitioned subset of production rules of a grammar as a function, it takes virtual terminals in INPUT as its input, and returns a (set of) non-terminal(s) in OUTPUT as its output. A directed *calling graph* for the sub-grammar is then defined as (V, E). V is a sub-grammar set that contains all partitioned sub-grammars, and E={(A, B)}, where A and B are the sub-grammars in V, with the overlap of the OUTPUT of B and the INPUT of A being nonempty. The calling graph of sub-grammar G_S and G_{NP} is shown in Figure 2. The INPUT of G_S is *vtNP* and OUTPUT of G_{NP} is *NP*. Therefore, the directed edge from G_S to G_{NP} is added to the calling graph.

[1] $S \rightarrow A_i$ ($1 \le i \le n$), $A_i \rightarrow a_j A_i$ ($1 \le i \ne j \le$n), $A_i \rightarrow a_i B_i | b_i$ ($1 \le i \le n$) and $B_i \rightarrow a_j B_i | b_i$ ($1 \le i, j \le n$)

[2] A reduced grammar with production rules has no unused terminals or non-terminals.

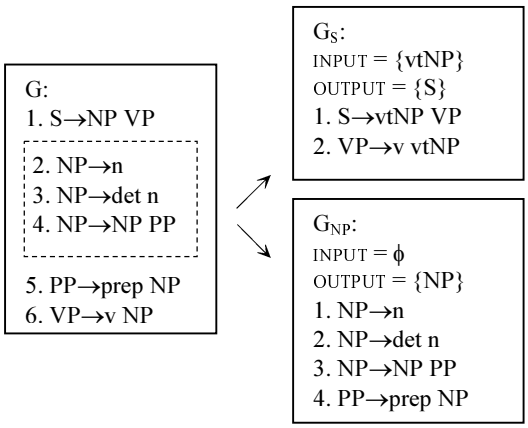

Figure 1: The entire grammar G is partitioned into sub-grammars G_{NP} and G_S.

In this particular investigation, partitioned sub-grammar is assigned with a *level index (ID)*. A sub-grammar is assigned to level i if its INPUT are virtual terminals at the level $< i$, where $i \geq 0$. The master sub-parser is assigned with the highest level index.

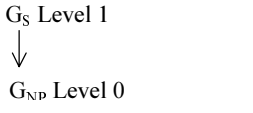

Figure 2: The calling of sub-grammars G_S and G_{NP}.

3. PARSER COMPOSITION

In this paper, two parser composition algorithms – *parser composition by cascading* and *parser composition with predictive pruning* are investigated empirically, with GLR parsers as their sub-parsers.

A *lattice with multiple granularities (LMG)*, a directed acyclic graph (DAG), is introduced to serve as a common interface among sub-parsers. For an LMG, its nodes are either virtual terminals or terminals and are linked by its transitions. Figure 3 shows an example. If a sub-parser takes an LMG as its input and parses it successfully, it creates a corresponding virtual terminal, and places it on the LMG. For example, *vtS* together with a parse tree is shown in Figure 3. Hence the LMG provides a good book-keeping for parser composition.

Since sub-parser needs to handle an LMG as input, we modify our GLR parsing algorithm in a way similar to that proposed by Tomita [9], such that the parser can handle an input lattice (in addition to an input string). The lattice is parsed on topological order and may end within a sentence.

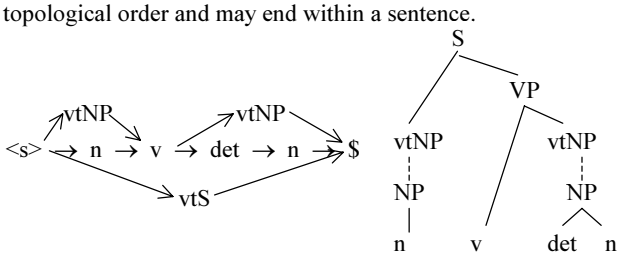

Figure 3: Example of an LMG and its parse tree with *vtS* as the output virtual terminal.

3.1. Parser Composition by Cascading

Parser composition by cascading is a bottom-up parsing algorithm. The mechanism takes an input sentence and converts it into an LMG. The parsers at the lowest level are activated to parse the LMG, and leave their corresponding virtual terminals on the LMG if they parse it successfully. If no sub-parsers of a given level can produce any new virtual terminals at any position in the LMG, the cascading process moves to the next level. This parsing process continues until the highest level sub-parsers get activated. Since the process starts from the lowest level to the highest level, level by level, we call it *cascading*.

3.2. Parser Composition with Predictive Pruning

In the cascading composition, sub-parsers get invoked at every position in the input LMG. Some of the created virtual terminals do not contribute to the final parse forest. To avoid excessive invocation, an alternative composition algorithm is implemented – *parser composition with predictive pruning*.

When a caller sub-parser reads a node (virtual terminal or terminal) in the LMG, the sub-parser that gets activated must have the node as its left corner and must be able to return a virtual terminal *predicted* by the caller sub-parser. Notice that the left corner can be either a terminal or a virtual terminal. For our implementation, each virtual terminal's left corner is pre-computed before parsing. Any sub-parsers in the caller's INPUT set that do not satisfy these predictive conditions are *pruned*. The parsing starts with the master GLR sub-parser at the leftmost lattice node, and ends when it reaches the final node of the LMG. Since the activated sub-parsers must satisfy the caller sub-parser's predictive constraint and the ones that do not satisfy the constraint are pruned, it is called *predictive pruning*.

4. EXPERIMENTS

4.1. Experimental Corpus

For our experiments, we used the ATIS-3 Class A queries, a subset of the ATIS corpus (Air Travel Information System) [6]. In addition, we translated the English ATIS queries into Cantonese Chinese, as a parallel corpus to verify our results. Examples are shown in Table 1.

The training set has 1564 queries; the 1993 test set has 448 queries; and the 1994 test set has 444 queries. Each query has a corresponding SQL query for database access.

English:	*Show me the most expensive one way flight from detroit to westchester county*
Chinese:	話俾我知由底特律飛去西赤斯特城最貴既單程航機
English:	*Show me the flights arriving on baltimore on june fourteenth*
Chinese:	話俾我知係六月十四號飛到巴的摩爾既航機

Table 1: Examples of English queries in the ATIS3 corpus. We translated these into Cantonese Chinese to form a parallel corpus as illustrated.

4.2. ATIS Grammar Development

Our ATIS grammar is a set of context-free rules. The grammar contains both semantic and syntactic structures. The low level grammar rules are mainly semantic concepts typical of ATIS corpus, such as CITY-NAME, CLASS-TYPE, MONTH-NUMBER, etc. They are obtained by a semi-automatic grammar induction algorithm [8]. These higher level grammar rules describe phrases, such as a time phrase, flight preposition phrase, etc. The SENTENCE-level grammar rules, however, are generated using a data-driven approach described below, because the semi-automatic algorithm does not produce reliable candidates due to sparse training data.

Parser composition by cascading produces a lattice for every training sentence. We derive top-level (SENTENCE-level) grammar rules by picking up the "best" path through the lattice using the shortest-path algorithm [4]. Such rules maximize the coverage of our training set. Thus we formed grammars for the English and Chinese corpora, with un-partitioned and partitioned grammar sizes listed in Table 2. Examples of English ATIS-3 rules include:

S → ASK FLIGHT_NP| … (SENTENCE-level rules)
ASK → show me|list|tell me|give me|…
FLIGHT_NP → FLIGHT FLIGHT_PP
FLIGHT → flight|flights|flight number|…
FLIGHT_PP → DEPARTURE|ARRIVAL|…
DEPARTURE → leaving CITY_NAME|…
CITY_NAME → phoneix|new york|seattle|…

4.3. Partitioning the ATIS Grammar

Following our grammar partitioning scheme described in the previous section, we manually partitioned our rules sets into sub-grammars by creating virtual terminals such as vtSTATE-NAME, vtCITY-NAME, vtAIRPORT-NAME, etc. Most of these virtual terminals are based on semantic concepts, so that they can be easily grouped together.

Grammar Statistics	**No Partitioning**	
	English	**Chinese**
# rules	1650	1538
# terminals	602	515
# non-terminals	97	85
# virtual terminals	N/A	N/A
# parsing table states	72,869	29,734
Grammar Statistics	**Partitioned Grammar**	
	English	**Chinese**
# rules	1818	1637
# terminals	602	515
# non-terminals	97	85
# virtual terminals	65	63
# parsing table states	3,350	3,894

Table 2: Grammar statistics based on the original un-partitioned grammar and a partitioned grammar.

Table 2 provides a comparison between the statistics of the unpartitioned and partitioned grammars. We observe that the partitioned grammar has a larger number of rules. This is because some rules have been duplicated across multiple sub-grammars. The duplicated non-terminals do not constitute the output of any sub-grammars. For example, production rule DIGIT → *one* may appear in sub-grammar of COST, TIME, etc.

The LR(1) parsing table generator is used to construct the LR(1) tables from all grammars, partitioned or un-partitioned. The total number of states (rows) in the parsing table for partitioned grammars is the sum of the number of states in all sub-parsers. From Table 2, the unpartitioned English grammar has 72,869 states and the unpartitioned Chinese grammar has 29,734 states in their parsing tables. In comparison, the partitioned English grammar has 3,350 states and the partitioned Chinese grammar has 3,894 states in total. Grammar partitioning greatly reduces parsing table sizes. This decreases the computation required to generate the parsing tables, space and memory to store the tables, and time to access the tables during parsing. This result shows that grammar partitioning is more desirable for large scale natural language processing.

4.4 Composing ATIS Sub-parsers

After grammar partitioning, we need to compose all the sub-parsers in order to obtain an overall parse for the input. We use the two parser composition algorithms to combine sub-parsers – cascading composition and predictive composition.

Our parsing framework with multiple sub-grammars is compared with a single GLR parser with single grammar. Results on grammar coverage and natural language understanding are shown in Tables 3 and 4:

- **Grammar coverage**:

Statistics for English and Chinese ATIS are shown in Table 3. *Full parse* means there is a parse tree covering the whole input query. *Partial parse* means there is at least one parse chunk covering part of query. *No parse* means there is no parse at all for the query.

The first row of Table 3 shows that the full parse coverage are the same for the different parsing strategies in English and Chinese queries. This reflects the consistency of our parsing framework. Since our SENTENCE-level grammar rules are derived from the shortest paths in the training lattices, they can reach high full parse coverage in the training sets. However, these rules tend to be rather specific in structure, and may contribute to the relatively low full parse coverage in the test sets.

The single GLR parser approach cannot create any partial parse. However, composed sub-parsers place virtual terminals on the LMG if they successfully generate sub-parses. Hence partial parses can be obtained from parser composition. When parser composition by cascading is used, a partial parse can end at any position of the input query. When the predictive composition is used, sub-parsers are activated only if they satisfy the predictive constraints. Therefore, cascading obtains the highest percentage of partially parsed queries and the single GLR obtains no partially parsed queries.

- **Natural Language Understanding Performance**

For evaluating the accuracy performance, the output parses are converted to semantic frames. This is straightforward for the single GLR approach, since it produces a single parse forest/tree. With grammar partitioning, the output of the parsers is an LMG. There are multiple paths from the sentence START to sentence END. We apply the shortest-path algorithm in [4] to find the "best path" in the LMG for the input sentence. Our semantic interpreter walks through the parse trees that attached to the virtual terminals in the best path, and extracts semantic information to form a semantic frame. The contents of the frame are compared against with "reference" semantic frame, derived from the list of attributes in the corresponding SQL query from the ATIS corpus.

Results for natural language understanding are shown in Table 4. *Full match* refers to queries with exact matches between the generated semantic frame and the reference semantic frame. *Partial match* refers to cases where the fraction of matching concepts in the semantic frame is between zero and one, where insertion, deletion and substitution errors are all penalized. *No match* refers to the situation when the concept error rate equals or exceeds 100% for the sentence.[3]

From Table 4, we found that the single GLR approach has the highest concept error rate, since it produces no partial parses, when the input query is ungrammatical. Conversely, both composition algorithms for partitioned grammars produce partial parses on LMG and attain a higher natural language

[3] Insertion errors may cause the concept error rate to exceed 100%.

understanding performance. In addition, cascading performs better than predictive, because cascading attempts to parse chunks of the input at all lattice nodes, while predictive pruning invokes virtual terminals only if they abide to the left corner predictive constraints.

5. CONCLUSIONS

In this paper, we have presented a modular parsing framework – *grammar partitioning and parser composition,* based on the ATIS corpora, which can be applied to both English and Chinese. We presented experimental results that show that grammar partitioning can drastically reduce parsing table size by an order of magnitude, when compared with a single CFG derived from the ATIS training sets. Both the single GLR parser and the parser composition approaches gave the same full parse coverage on the test sets, but parser composition produces more partial parses than single GLR parser. Therefore, parser composition can obtain a higher understanding accuracy. We will be implementing probabilistic parsing to rank multiple parse outputs as our next step.

REFERENCES

1. Abney, S., "Parsing by Chunks", In Principle-Based Parsing: Computation and Psycholinguistics, R. C. Berwick et al. (eds), Kluwer Academic Publishers, 1991.
2. Amtrup, J., "Parallel Parsing: Different Distribution Schemata for Charts", In Proceedings of the 4th International Workshop on Parsing Technologies, Prague, pages 12-13, Sep 1995.
3. Earley, J., An Efficient Context-free Parsing Algorithm, PhD thesis, Carnegie Mellon University, 1968.
4. Hillier, F. S. and Lieberman, G. J, Introduction to Operations Research, McGraw-Hill, Inc., 6 edition, 1995.
5. Korenjak, A., "A Practical Method for Constructing LR(k)", CACM 12, 11, 1969.
6. Price, P., "Evaluation of Spoken Language Systems: The ATIS Domain", The 3rd European Conference on Speech Communication and Technology, 1993.
7. Ruland, T., Rupp, C., Spilker, J., Weber, H., and Worm, K., "Making the Most of Multiplicity: A Multi-Parser Multi-Strategy Architecture for the Robust Processing of Spoken languages", In Proceedings of ICSLP, 1998.
8. Siu, K. C. and Meng, H., "Semi-Automatic Acquisition of Domain-Specific Semantic Structures", In Proceedings of Eurospeech, 1999.
9. Tomita, M., "An Efficient Word Lattice Parsing Algorithm for Continuous Speech Recognition", In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1569-1572, Apr 1986.
10. Weng, F., Meng, H. and Luk, P.C., "Parsing a Lattice with Multiple Grammars", In Proceedings of the 6th International Workshop on Parsing Technologies, 2000.
11. Weng, F. and Stolcke, A., "Partitioning grammar and Composing Parsers", In Proceedings of the 4th International Workshop on Parsing Technologies, Sep 1995.

Based on the English ATIS-3 Corpus	**Training set**			**Test set 93**			**Test set 94**		
	Unpart.	**Partitioned**		**Unpart.**	**Partitioned**		**Unpart.**	**Partitioned**	
	GLR	**CAS**	**PP**	**GLR**	**CAS**	**PP**	**GLR**	**CAS**	**PP**
Full parse (%)	99.4	99.4	99.4	60.9	60.9	60.9	62.4	62.4	62.4
Partial parse (%)	0	0.6	0.5	0	39.1	34.2	0	37.6	35.1
No parse (%)	0.6	0	0.1	39.1	0	4.9	37.6	0	2.5
Based on the Chinese ATIS-3 Corpus	**Training set**			**Test set 93**			**Test set 94**		
	Unpart.	**Partitioned**		**Unpart.**	**Partitioned**		**Unpart.**	**Partitioned**	
	GLR	**CAS**	**PP**	**GLR**	**CAS**	**PP**	**GLR**	**CAS**	**PP**
Full parse (%)	98.7	98.7	98.7	44.6	44.6	44.6	57.4	57.4	57.4
Partial parse (%)	0	1.3	1.0	0	55.4	51.3	0	42.6	37.4
No parse (%)	1.3	0	0.3	55.4	0	4.0	42.6	0	5.2

Table 3: Grammar coverage for the English and Chinese (shaded) ATIS-3 corpora. CAS abbreviates parser composition by cascading; PP is parser composition with predictive pruning, and GLR is the use of a single GLR parser (no partitioning).

Based on the English ATIS-3 Corpus	**Training set**			**Test set 93**			**Test set 94**		
	Unpart.	**Partitioned**		**Unpart.**	**Partitioned**		**Unpart.**	**Partitioned**	
	GLR	**CAS**	**PP**	**GLR**	**CAS**	**PP**	**GLR**	**CAS**	**PP**
Error rate in semantic concepts (%)	8.0	7.5	7.8	41.2	9.9	33.8	40.4	11.7	29.1
Full match (% of sentence)	82.4	82.7	82.4	56.3	87.7	60.5	54.5	77.0	59.2
Partial match (% of sentence)	15.0	15.3	15.2	4.0	8.0	9.8	7.7	18.9	20.3
No match (% if sentence)	2.6	2.0	2.4	39.7	4.2	29.7	37.8	4.1	20.5
Based on the Chinese ATIS-3 Corpus	**Training set**			**Test set 93**			**Test set 94**		
	Unpart.	**Partitioned**		**Unpart.**	**Partitioned**		**Unpart.**	**Partitioned**	
	GLR	**CAS**	**PP**	**GLR**	**CAS**	**PP**	**GLR**	**CAS**	**PP**
Error rate in semantic concepts (%)	10.3	7.9	8.6	58.6	11.0	25.6	44.9	12.7	28.4
Full match (% of sentence)	77.6	80.8	80.2	37.5	78.6	61.6	49.8	72.7	58.6
Partial match (% of sentence)	18.5	16.9	17.1	6.0	16.7	23.2	7.4	22.3	24.1
No match (% if sentence)	3.9	2.2	2.7	56.5	4.7	15.2	42.8	5.0	17.3

Table 4: Performance in language understanding of English and Chinese (shaded) ATIS-3 corpora. CAS abbreviates parser composition by cascading; PP is parser composition with predictive pruning, and GLR is the use of a single GLR parser (no partitioning).