PAPER

Extracting Partial Parsing Rules from Tree-Annotated Corpus : Toward Deterministic Global Parsing

Myung-Seok CHOI[†], ***Nonmember***, **Kong-Joo LEE**[††], ***Member***, **Key-Sun CHOI**[†], ***and*** **Gil Chang KIM**[†], ***Nonmembers***

SUMMARY It is not always possible to find a global parse for an input sentence owing to problems such as errors of a sentence, incompleteness of lexicon and grammar. Partial parsing is an alternative approach to respond to these problems. Partial parsing techniques try to recover syntactic information efficiently and reliably by sacrificing completeness and depth of analysis. One of the difficulties in partial parsing is how the grammar might be automatically extracted. In this paper we present a method of automatically extracting partial parsing rules from a tree-annotated corpus using the decision tree method. Our goal is deterministic global parsing using partial parsing rules, in other words, to extract partial parsing rules with higher accuracy and broader expansion. First, we define a rule template that enables to learn a subtree for a given substring, so that the resultant rules can be more specific and stricter to apply. Second, rule candidates extracted from a training corpus are enriched with contextual and lexical information using the decision tree method and verified through cross-validation. Last, we underspecify non-deterministic rules by merging substructures with ambiguity in those rules. The learned grammar is similar to phrase structure grammar with contextual and lexical information, but allows building structures with more than one depth. Thanks to automatic learning, the partial parsing rules can be consistent and domain-independent. Partial parsing with this grammar processes an input sentence deterministically using longest-match heuristics, and recursively applies rules to an input sentence. The experiments showed that the partial parser using automatically extracted rules is not only accurate and efficient but also achieves reasonable coverage for Korean.

key words: *partial parsing, deterministic parsing, automatic rule extraction, underspecification, decision tree, Korean*

1. Introduction

Conventional parsers, which try to recognize complete syntactic information, face difficulties in processing unrestricted texts, because of ungrammatical sentences, the unavoidable incompleteness of lexicon and grammar, and other reasons like long sentences. Partial parsing is an alternative technique to respond to these problems. It aims to recover syntactic information efficiently and reliably from unrestricted texts by sacrificing completeness and depth of analysis, and relies on local information to resolve ambiguities [1].

Rule-based methods, used in partial parsing techniques via finite state machines [2]–[4], have several advantages over statistical counterparts in that the resulting knowledge is more compact and comprehensible to human developers and results are more easily integrated into existing hand-built symbolic systems [5]. But manually building underlying knowledge is a labor-intensive task and is prone to have inconsistencies.

Recently, there have been several researches on automatically extracting rules from large-scale corpora using several machine learning techniques [6]–[8]. But their work was restricted to the chunking task, especially finding non-recursive and non-overlapping noun phrases.

In this paper, we present a method of automatically extracting partial parsing rules from a tree-annotated corpus using the decision tree method. We define the partial parsing rules as ones that can decide the structure of a substring in an input sentence deterministically. Our goal is toward deterministic global parsing using partial parsing rules. To put it more concretely, we aim to achieve two goals simultaneously in extracting partial parsing rules, though these two goals appeal to be conflicting.

The first goal is to extract partial parsing rules with higher accuracy. We use a rule template that enables to learn not a single non-terminal but a subtree for a given substring. As the rules extracted by this rule template reduce a substring into a subtree, the resultant rules can be more specific and stricter to apply. In addition, we use negative evidence as well as positive evidence in enriching partial parsing rules using the decision model. These approaches can improve the accuracy of the partial parsing rules.

The second goal is to make partial parsing rules cover broader expansion by applying them recursively to a string. In other words, our partial parsing is designed to result in deep depth structures in a similar way as global parsing does. There are many cases that several different structures are assigned to the same substring in a tree-annotated corpus. Substrings for coordination and compound nouns are the typical examples of these ambiguous cases in Korean. These ambiguities can prevent us from extracting partial parsing rules covering the substrings with more than one substructure, and consequently make the result of partial parsing limited to a relatively shallow depth. We address this problem by merging substructures with ambiguity using a underspecified representation in this work.

Manuscript received August 26, 2004.
Manuscript revised December 10, 2004.
[†]Dept. of EECS, KAIST, Korea
[††]School of CIT, KyungIn Women's College, Korea

With this underspecification, we can achieve the second goal without deteriorating the determinism and the precision of partial parsing.

The learned grammar is similar to phrase structure grammar with contextual and lexical information, but allows building structures with more than one depth. It is easy to understand, can be easily modified, and selectively added to or deleted from the grammar. Partial parsing with this grammar processes an input sentence deterministically using longest-match heuristics. The partial parsing rules we acquire are recursively applied to an input sentence, so that a final result of the partial parser looks like that of a global parser.

The remainder of this paper is organized as follows. In Sect. 2 we briefly review previous work related to this paper. In Sect. 3 we describe our method of automatically extracting partial parsing rules. Section 4 shows experimental results with this method. In Sect. 5 we conclude and propose directions for future research.

2. Related Work

Partial parsing techniques can be roughly classified into two groups depending on their objectives. One is partial parsing via finite state machines [2]–[4], [9]. These approaches apply the sequential regular expression recognizer to an input sentence. Their grammar consists of several levels of the regular grammar. The rules of each level are applied to the output of the previous level. When multiple rules match an input string at a given position, the longest-matching rule is selected. Therefore these parsers always produce a single best analysis and operate very fast. The outputs of these parsers are not necessarily consistent with the constituent structure [1]. Usually, these approaches used a hand-written regular grammar. As easily expected, manually writing a grammar is very time consuming and is prone to have inconsistencies.

The other is text chunking, that is, recognizing non-overlapping and non-recursive cores of major phrases (chunks), using machine learning techniques [6]–[8], [10]–[12]. Since Ramshaw and Marcus [6] proposed formulating the task of chunking as a tagging task, most of the chunking methods have followed this *word-tagging* approach. In base NP chunking, for instance, each word is marked with one of three chunk tags: I (for a word inside an NP), O (for outside of an NP), and B (for between the end of one NP and the start of another) as follows[†]:

(1a) In (early trading) in (Hong Kong) (Monday), (gold) was quoted at ($ 366.50) (an ounce).

(1b) In_O early_I trading_I in_O Hong_I Kong_I Monday_B $,_O$ gold_I was_O quoted_O at_O $\$_I$ 366.50_I an_B ounce_I $._O$

[†]This example is excerpted from Tjong Kim Sang and Veenstra [12].

In addition to this representation (IOB1), tagging schemes such as IOB2, IOE1, IOE2, O+C, IOBES have been proposed [13], [14]. Among these approaches there were several researches on automatically extracting chunking rules from large-scale corpora using several machine learning techniques such as transformation-based learning [6], error-driven pruning [7], ALLiS top-down inductive system [8]. Ramshaw and Marcus [6] and Déjean [8] adopted word-tagging approach, but how a chunk representation is extended to general recursive phrases is not intuitive and the consequent grammar seems to be less readable and less maintainable by human than the finite state grammar. Cardie and Pierce [7] used NP rules that were read from a tree-annotated corpus and pruned based on a benefit score calculated from errors on a corpus. However, extending from base NP chunking, they appear to encounter difficulties when there are two or more different non-terminals for a given substring.

3. Automatic Rule Extraction

To begin with, we define the rule template, the basic format of a partial parsing rule, as follows:

$$left\_context \mid substring \mid right\_context \longrightarrow substructure$$

It means that the *substring* of an input sentence surrounded by the *left_context* and the *right_context* constructs the *substructure*. The *left_context* and the *right_context* are the rest of an input sentence except the *substring*. For automatic learning of the partial parsing rules, the lengths of the *left_context* and the *right_context* are restricted to one respectively. We should note that the result of applying a partial parsing rule is the structure with one or more depths. In other words, the rules extracted by this rule template reduce a substring into not a single non-terminal but a subtree, and hence the resultant rules can be more specific and stricter to apply.

Figure 1 illustrates the procedure for the extraction of partial parsing rules. First, we extract all possible rule candidates according to the rule template from a tree-annotated corpus. Extracted candidates are grouped by their substrings. Next, these candidates are enriched with contextual and lexical information using the decision tree method. The contextualized and lexicalized rules are verified through cross-validation in order to retain only the accurate rules. The successfully verified rules can be the final partial parsing rules. The rest rules which cannot be verified are passed to the step of tree underspecification, which merges tree structures with hard ambiguities. As can be seen in Fig. 1, the underspecified candidates are put back into the refinement step.

The following subsections describe each step in detail.

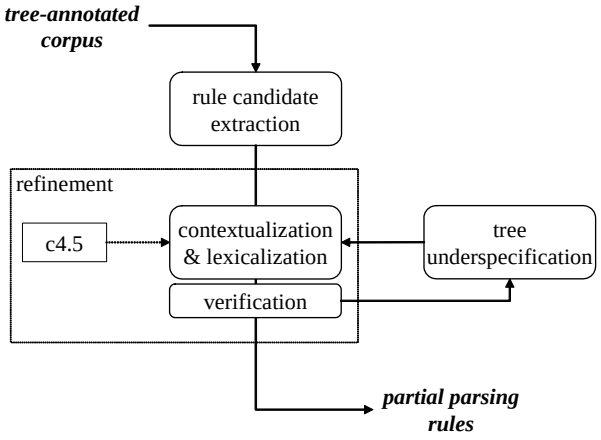

Fig. 1 Procedure for extracting partial parsing rule

3.1 Extracting candidates

From the tree-annotated corpus we extract all possible candidates of partial parsing rules according to the rule template. In this step only part-of-speech information is considered, and in later steps additional contextual and lexical information will be used to enrich the candidates, which is a general-to-specific approach. A syntactic structure can be regarded as a sequence of levels from part-of-speech tags (level 0) in a bottom-up manner. We define a level as all nodes with the same depth in a syntactic structure. Scanning an input sentence annotated with its syntactic structure one by one, we can extract the substructure responding to every possible substring, in each level of the syntactic structure. If there is no structure that responds to a substring exactly, then a null substructure is extracted, which represents negative evidence.

Figure 2 shows an example sentence[†]with its syntactic structure[††]and some of the candidates of the partial parsing rules extracted from this example. In this figure, the first candidate of the partial parsing rules means that the substring 'npp' surrounded by 'bos' and '+ jxt' can be constructed into the substructure 'NP'. S_{null} denotes negative evidence.

[†]'NOM' stands for nominative case and 'ACC' accusative case. 'npp' denotes personal pronoun, 'jxt' topicalized auxiliary particle, 'ncn' non-predicative common noun, 'jco' objective case particle, 'pvg' general verb, 'ef' final ending, and 'sf' full stop symbol. For a detailed description of KAIST corpus and its tagset, see Lee [15]. 'bos' stands for the begin of sentence, 'eos' the end of sentence, and '+' is not a part-of-speech but a delimiter between words within a word-phrase.

[††]In Korean, a word-phrase, similar to bunsetsu in Japanese, is defined as a spacing unit with one or more content words followed by zero or more functional words. A content word indicates the meaning of the word-phrase in a sentence, while a functional word—particle or verbal-ending—indicates the grammatical role of the word-phrase. In KAIST corpus used in this paper, a functional word is not included in the non-terminal that the preceding content word belongs to, following the restricted representation of phrase structure grammar for Korean [16]. For example, a word-phrase "나/npp + 는/jxt" is annotated as "(NP 나/npp) + 는/jxt" in Fig. 2.

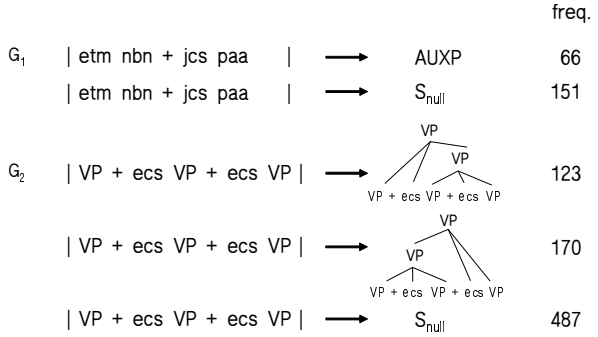

Fig. 3 Groups of the candidates of partial parsing rules

Extracted rule candidates are gathered and grouped by their substrings. The contextual information in the extracted candidates is temporarily hidden in the representation of the rule groups. Figure 3[†††]shows the groups of the candidates. In this figure, G_1 and G_2 are the names of the groups and the number in the last column is the frequency of each candidate occurring in the training corpus. The groups G_1 and G_2 have 2 and 3 candidates, respectively. When the number of candidates in a group is only one, the candidate can be always applied to the corresponding substring deterministically. On the other side, in the case that there is more than one candidate in a group, we should enrich those candidates with contextual and lexical information in order to make them distinctive from each other in applying to a substring.

3.2 Refining candidates

This step refines ambiguous candidates with contextual and lexical information so as to make them unambiguous.

First of all, they need to be annotated with contextual and lexical information occurring in the training corpus as shown in Fig. 4. In this figure, we realize that the substring with lexical information such as '수(way)/nbn' unambiguously constitutes the substructure 'AUXP'. We use the decision tree method, C4.5 [17], to select the important contextual and lexical information that can contribute effectually toward making the partial parsing rules unambiguous. The features used in the decision tree method are lexical information of each terminal or non-terminal for the *substring*, and parts-of-speech and lexical information for

[†††]'etm' denotes adnominalizing ending, 'nbn' non-unit bound noun, 'jcs' subjective case particle, 'paa' attributive adjective, 'ecs' subordinate conjunctive ending, and 'AUXP' auxiliary phrase.

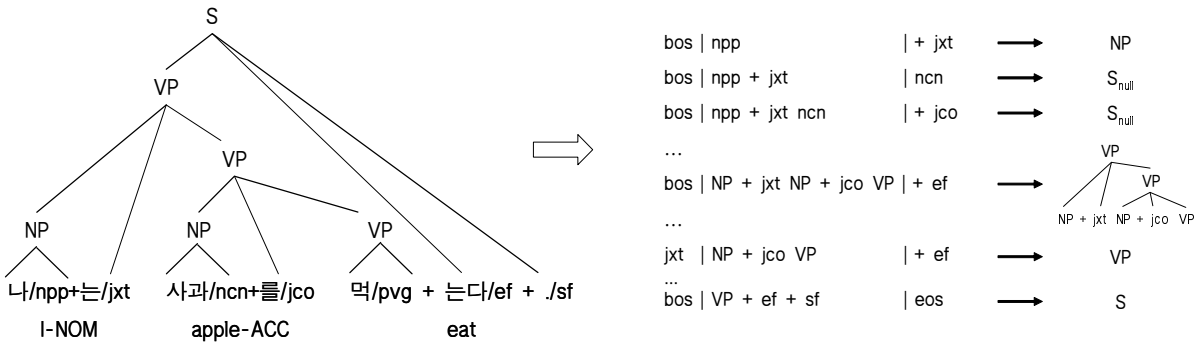

Fig. 2 Example sentence and the extracted candidates of partial parsing rules

	etm nbn + jcs paa	
살/pvg +	ㄹ 수 + 가 있	+ 다/ef → AUXP
이/jp +	ㄹ 수 + 가 없	+ 다/ef → AUXP
놀라/pvg +	ㄴ 적 + 이 있	+ 다/ef → S_{null}
완전하/paa +	ㄴ 것 + 이 없	+ 고/ecc → S_{null}
끝나/pvg +	ㄴ 것 + 이 아니	+ 라/ecs → S_{null}
읽/pvg +	는 것 + 이 좋	+ 다/ef → S_{null}
하/xsv +	ㄹ 나위 + 가 없	+ 다/ef → S_{null}

Fig. 4 Annotated candidates for the rules in the group G_1

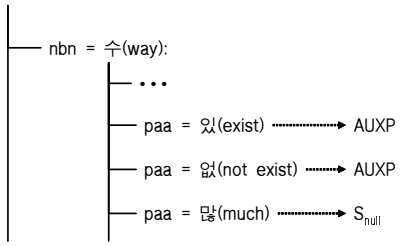

Fig. 5 The part of the decision tree

etm 수/nbn + jcs 있/paa	⟶	AUXP
etm 수/nbn + jcs 없/paa	⟶	AUXP

Fig. 6 Partial parsing rules extracted from Fig. 5

the *left_context* and the *right_context*. Lexical information of a non-terminal is defined as the part-of-speech and lexical information of its headword.

Figure 5 shows the part of the decision tree learned from our example substring. From the above decision tree, the deterministic partial parsing rules in Fig. 6 are extracted. As shown in Fig. 6, only the lexical entries for the second and the fourth morphemes in the substring are selected as the additional lexical information, and none of the contexts are selected in this case.

After we enrich the partial parsing rules using the decision tree method, we verify them by estimating their accuracies in order to filter out less deterministic rules. We estimate the error rates (%) of the rule candidates by 10-fold cross validation on the training corpus. The rule candidates in the only group with the error rate less than a predefined threshold θ can be extracted to the final partial parsing rules. The candidates in the group G_2 in Fig. 3 could not be extracted to the final partial parsing rules, because its estimated error rate is higher than the threshold. The candidates in G_2 are set aside for the processing of tree underspecification. We can control the number of the final partial parsing rules and the level of precision/recall trade-off of the parser that adopts the extracted partial parsing rules through the threshold θ.

We should note that the rules inducted from the decision tree are ordered. Because these ordered rules do not interfere with those from other groups, they can be modified without much difficulty.

3.3 Dealing with hard ambiguities: the underspecified representation

The group G_2 in Fig. 3 is one of attachment ambiguities, consecutive subordinate clauses. Figure 7 shows the part of two different trees extracted from a tree-annotated corpus. Two trees have the identical *substrings*, but are analyzed differently. This figure exemplifies that the ambiguity relates to lexical association between verb phrases, which is difficult to annotate in rules. There are many other syntactic ambiguities such as coordination and noun phrase bracketing difficult to resolve with local information. The resolution usually requires lexical co-occurrence, global context, semantics, and so on. These ambiguities can deteriorate the precision of partial parsing, or make the result of partial parsing limited to a relatively shallow depth.

Rule candidates with these ambiguities mostly have several different structures assigned to the same substrings under the same non-terminals, which we will call them *internal syntactic ambiguities* in this paper. Table 1 shows the patterns of the internal syntactic ambiguities that were found in KAIST corpus but unable to be refined because of low estimated accuracy. We observe by examining the patterns manually that

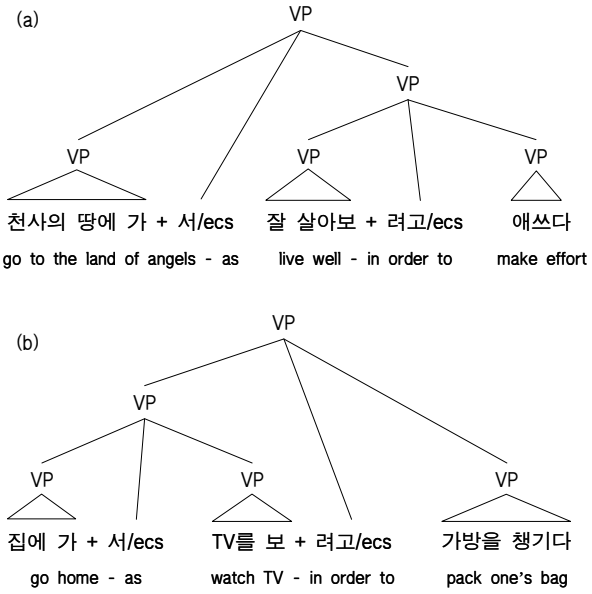

Fig. 7 Examples of internal syntactic ambiguities

Table 1 Patterns of internal syntactic ambiguities

Type	Freq.	Examples
conjunction	856	ADVP VP + ecc VP + ecc VP
attachment	830	VP + etm NP + jcm NP
NP bracketing	26	ncn ncn ncn
tagging error	11	

G_2'	| VP + ecs VP + ecs VP | $\longrightarrow$	VP
	| VP + ecs VP + ecs VP | $\longrightarrow$	S_{null}

Fig. 8 Underspecified candidates

few internal syntactic ambiguities can be resolved with local information.

In this paper, we handle internal syntactic ambiguities by merging the candidates by tree intersection and making them underspecified. This underspecified representation enables the analysis with broader coverage without deteriorating the determinism and the precision of partial parsing. Since only different structures under the same non-terminal are merged, the underspecification does not harm the structure of higher nodes. Figure 8 shows the underspecified candidates of G_2. In this figure, first two rules in G_2 are reduced into the merged 'VP'. Underspecified candidates are also enriched with contextual and lexical information by the decision tree method and verified through cross-validation as described in Sect. 3.2. The resolution of internal syntactic ambiguities is handed over to a module beyond partial parsing. If necessary, by giving all possible structures of underspecified parts we can prevent a later processing from re-analyzing the parts. The rest candidates that cannot be picked up as the partial parsing rules by all the three steps are thrown away.

4. Experiments

To show usefulness of the partial parsing rules automatically extracted in this paper, we made some experiments. For evaluation, we implemented a naive partial parser using TRIE indexing to search the partial parsing rules. The input of the partial parser is a part-of-speech tagged sentence and the result is usually the sequence of subtrees. At each position in an input sentence, the parser tries to choose a rule group using longest-match heuristics, and then, if any, applies the first-matching rule in the group, because the rules inducted from the decision tree are ordered.

For our experiments, we used KAIST tree-annotated corpus [15]. The training corpus contains 10,869 sentences (289,362 morphemes) with the average length of 26.6 morphemes, while the test corpus contains held-out 1,208 sentences whose average length is 26.0 morphemes. The validation corpus, used for choosing the threshold θ, contains 1,112 sentences with the average length of 20.1 morphemes, and is mutually exclusive with both the training corpus and the test corpus.

The performance of the partial parser is evaluated under PARSEVAL measures [18]:

$$labeled\ precision(LP) = \frac{number\ of\ correct\ constituents\ in\ partial\ parse}{number\ of\ constituents\ in\ partial\ parse}$$

$$labeled\ recall(LR) = \frac{number\ of\ correct\ constituents\ in\ partial\ parse}{number\ of\ constituents\ in\ correct\ parse}$$

where the *correct constituent* has to span the same sequence of words and has the same label as a constituent in the correct parse. F measure, a complement of the E measure [19], has been used to combine both into a single measure of overall performance, and is defined as follows:

$$F_\beta = \frac{(\beta^2+1) * LP * LR}{\beta^2 * LP + LR}$$

where β is a factor that determines the weighting of precision and recall. $\beta < 1$ is used for giving precision a larger weight than recall, $\beta > 1$ for giving recall a larger weight than precision, and $\beta = 1$ for equal weighting of precision and recall. We chose a value of $\beta = 1$ for comparison.

The parsing result can be affected by the predefined threshold θ (described in Sect. 3.2), which can control the accuracy of the partial parser as well as the number of the extracted rules. Table 2 shows the number of the extracted rules and how precision and recall trade off for the validation corpus as the threshold θ is varied. As can be seen, the smaller the threshold θ, the

Table 2 Precision/Recall with respect to the threshold θ for the validation corpus

θ	# of rules	precision	recall	$f_{\beta=1}$
6	18,638	95.5%	72.9%	82.7%
11	20,395	95.1%	75.1%	83.9%
16	22,650	94.2%	78.0%	85.3%
21	25,640	92.6%	83.3%	87.7%
26	28,180	92.0%	84.7%	88.2%

higher the precision and the lower the recall. As the threshold θ became larger, they went in reverse. From a partial parser's point of view, the precision is generally favored over the recall. In this paper, we choose θ as 11 whose precision was over 95% although $f_{\beta=1}$ was the highest when θ was 26.

Table 3 presents the precision and the recall of the partial parser for the test corpus when the threshold θ was 11. In baseline grammar we selected the most probable structure for a given substring in each group of candidates, and the grammar 'depth 1 rule only' is the set of the rules extracted along with the restriction that only the single depth substructure is permitted in the rule template. The grammar 'underspecified' is the final version of our partial parsing rules, and 'not underspecified' is the extracted rules without the underspecification processing. Both the grammar PCFG and Lee [15] are statistical full parsers of Korean, and Lee enriched the grammar using contextual and lexical information in order to improve the accuracy of a parser. Both of them were trained and tested on the same corpus as ours for comparison. The performance of not underspecified and the underspecified grammars was greatly improved compared to the baseline and PCFG that do not adopt contextual and lexical information in their rules. Not underspecified rules performed better than depth 1 rules. This indicates that increasing the depth of a rule is helpful in partial parsing, as in the case of a statistical full parsing, Data-Oriented Parsing [20]. Comparing the underspecified rules with not underspecified ones, we can see that the underspecification leads to broader coverage, that is, higher recall. The precision of the underspecified rules was above 95%, which may be interpreted as 19 out of 20 structures are correct when a parser generates 20 phrases. However, its recall went down far beyond that of the statistical full parser [15]. When we set θ as 26, the underspecified grammar slightly outperformed that of the full parser in terms of $f_{\beta=1}$, although our partial parser does not always produce one complete parse tree[†]. It follows from what has been said thus far that our parser can be a partial parser with the high precision and can come close to a full parser with the same performance as that of a statistical full parser, depending on the threshold θ.

[†]In the test corpus, the percentage that our partial parser (θ=26) produced one complete parse tree was 70.9%. When θ=11, the percentage was 35.9%.

Table 3 Experimental result of the partial parser for Korean

Grammar	LP	LR	$f_{\beta=1}$
baseline	73.0%	72.0%	72.5%
depth 1 rule only	95.2%	68.3%	79.6%
not underspecified	95.7%	71.6%	81.9%
underspecified	95.7%	73.6%	83.2%
underspecified (in case θ=26)	92.2%	83.5%	87.6%
PCFG	80.0%	81.5%	80.7%
Lee [15]	87.5%	87.5%	87.5%

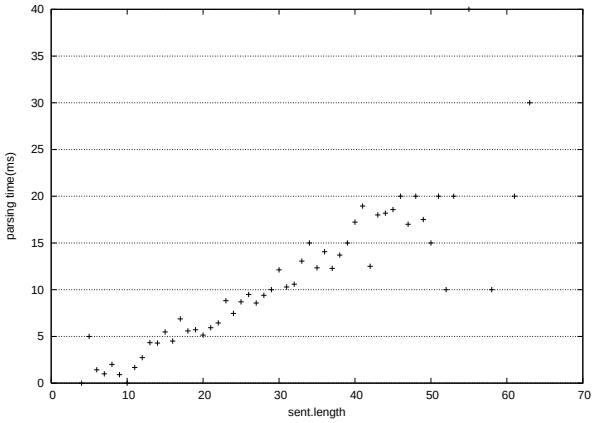

Fig. 9 Time spent in parsing

The current implementation of our parser has an $O(n^2 m_r)$ worst case time complexity in the case of a skewed binary tree, where n is the length of input sentence and m_r is the number of rules. Because m_r is the constant, much more than two elements are reduced to subtrees with one or more depths in each level of parsing, and, different from full parsing, the number of recursion in partial parsing seems to be limited, we can parse in near-linear time. In our parser, the maximum number of recursion was 10 and the average number of recursion was 4.47. Figure 9 shows the time spent in parsing as a function of the sentence length[††].

Lastly, we manually examined the first 100 or so errors occurring in the test corpus. Table 4 shows the error classes. In spite of underspecification, the errors related to conjunctions and attachments were most frequent. The errors of conjunctions were mostly caused by *substrings* not occurring in the training corpus, while the cases of attachments lacked contextual or lexical information for a given *substring*. These errors can be partly resolved by increasing the size of the corpus, but it seems not that they can be resolved completely in partial parsing. The errors of noun phrase bracketing were related to two or more consecutive nouns and could be classified into two categories. One was split-

[††]This result was obtained using a Linux machine with Pentium III 700MHz processor.

Table 4 Error classes

Class	Freq.
conjunction	24
attachment	20
noun phrase bracketing	9
date, time, unit	8
caused by errors of the subtree	22
incorrectly tagged or inherently ambiguous	17
incorrect sentence	1

ting one NP erroneously because, in Korean, various modifiers including clauses can often pre-modify a noun that is a modifier of a head noun. The other was combining two separate NPs. The latter seems to be solved by learning, from other lexical resources, negative constraints about nouns that cannot occur in the first, intermediate, or last position of NP. Varying forms of date, time and unit expression are difficult to all occur in the corpus, but they appear to be used in a regular way. For those expressions, manually encoded rules may be effective in partial parsing. Besides, there were many errors caused by incorrectly tagged sentences or inherently ambiguous sentences. We should note that many unrecognized phrases included the expression not occurring in the training corpus. This is obviously because our grammar cannot handle not occurring substrings; hence alleviating the sparseness in *sequences* is the subject for a future study.

5. Conclusion

In this paper we proposed a method of extracting the partial parsing rules automatically from a tree-annotated corpus using the decision tree method. By allowing rules to construct a subtree with more than one depth, enriching rule candidates with contextual and lexical information using the decision tree method, and verifying them through cross-validation, highly accurate partial parsing rules can be extracted. By merging substructures with ambiguity in non-deterministic rules using an underspecified representation, we can handle syntactic ambiguities difficult to resolve with local information, such as coordination, noun phrase bracketing ambiguities. The learned grammar is similar to phrase structure grammar with contextual and lexical information, and is easy to understand and modify. Partial parsing with this grammar processes an input sentence deterministically using longest-match heuristics, and recursively applies rules to an input sentence. The experiments showed that the partial parser using automatically extracted rules is not only accurate and efficient but also achieves reasonable coverage.

A partial parser based on the extracted grammar in this paper does not resolve hard ambiguities such as coordination, attachment ambiguities at all, but tries to resolve the ambiguities considered as accurate enough, which is controlled by the threshold θ. It can recognize more complex or recursive phrases than a chunker and analyze their internal structure with retaining accuracy. Therefore our partial parser can be a more practical solution for collocation extraction (especially with syntactic relations), corpus annotation, terminology extraction, etc. than a full parser or a chunker: full parsers try to recognize complete syntactic information even when only inaccurate rules can be applied, which correct syntactic relations cannot be distinguished from incorrect ones caused by low-precision rules; a common chunker gives only small pieces of information because of its limited coverage, even though the result of the chunker has higher precision.

Our method suffers more from data sparseness than *word-tagging* approaches in text chunking because we deal with *sequences* in rules. In Korean, an agglutinative language, a word-phrase (alternatively called as a word-unit or an "eojeol") is a composition of one or more content words and zero or more functional words. In most cases content words in a word-phrase can be normalized into one content word, and the case of functional words seems analogous. Using this property and other methods such as part-of-speech clustering we are planning to alleviate sparseness in sequences. Lexical association is helpful in analyzing the underspecified parts of a partial parse. We are working on the integration of lexical association into the partial parser.

Acknowledgments

This research was supported in part by the Ministry of Science and Technology, the Ministry Of Culture and Tourism, and the Korea Science and Engineering Foundation in Korea.

References

[1] S.P. Abney, "Part-of-speech tagging and partial parsing," in Corpus-Based Methods in Language and Speech, ed. S. Young and G. Bloothooft, Kluwer Academic Publishers, 1996.
[2] S.P. Abney, "Partial parsing vis finite-state cascades," Proceedings of the ESSLLI '96 Robust Parsing Workshop, 1996.
[3] J.R. Hobbs, D. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, and M. Tyson, "Fastus: A cascaded finite-state transducer for extracting information from natural-language text," in Finite-State Language Processing, ed. E. Roche and Y. Schabes, pp.383–406, The MIT Press, 1997.
[4] S. Aït-Mokhtar and J.P. Chanod, "Incremental finite-state parsing," Proceedings of Applied Natural Language Processing, 1997.
[5] R. Mooney and C. Cardie, "Symbolic machine learning for natural language processing," ACL '99 Tutorial, 1999.
[6] L.A. Ramshaw and M.P. Marcus, "Text chunking using transformation-based learning," Proceedings of Third Wordkshop on Very Large Corpora, pp.82–94, 1995.
[7] C. Cardie and D. Pierce, "Error-driven pruning of treebank grammars for base noun phrase identification," Proceedings of 36th Annual Meeting of the Association for Compu-

tational Linguistics and 17th International Conference on Computational Linguistics, pp.218–224, 1998.
[8] H. Déjean, "Learning rules and their exceptions," Journal of Machine Learning Research, vol.2, pp.669–693, 2002.
[9] D. Hindle, "A parser for text corpora," in Computational Approaches to the Lexicon, ed. B.T.S. Atkins and A. Zampolli, pp.103–151, Oxford University Press, 1995.
[10] S. Argamon-Engelson, I. Dagan, and Y. Krymolowski, "A memory-based approach to learning shallow natural language patterns," Journal of Experimental and Theoretical AI, vol.11, no.3, pp.369–390, 1999.
[11] M. Muñoz, V. Punyakanok, D. Roth, and D. Zimak, "A learning approach to shallow parsing," Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Copora, pp.168–178, 1999.
[12] E.F. Tjong Kim Sang, "Memory-based shallow parsing," Journal of Machine Learning Research, vol.2, pp.559–594, 2002.
[13] E.F. Tjong Kim Sang and J. Veenstra, "Representing text chunks," Proceesing of 9th Conference of the European Chapter of the Association for Computational Linguistics, Bergen, Norway, pp.173–179, 1999.
[14] K. Uchimoto, Q. Ma, M. Murata, H. Ozaku, and H. Isahara, "Named entity extraction based on a maximum entropy model and transformation rules," Proceedings of 38th Annual Meeting of the Association for Computational Linguistics, Hong Kong, pp.326–335, 2000.
[15] K.J. Lee, Probabilistic Parsing of Korean based on Language-Specific Properties, Ph.D. thesis, KAIST, Dept. of CS, 1998.
[16] K.J. Lee, G.C. Kim, J.H. Kim, and Y.S. Han, "Restricted representation of phrase structure grammar for building a tree annotated corpus of korean," Natural Language Engineering, vol.3, no.2, pp.215–230, 1997.
[17] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1993.
[18] E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J.Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski, "A procedure for quantitatively comparing the syntactic coverage of english grammars," Proceedings of the DARPA Speech and Natural Language Workshop, Pacific Grove, CA, pp.306–311, 1991.
[19] C. van Rijsbergen, Information Retrieval, Buttersworth, 1975.
[20] R. Bod, Enriching Linguistics with Statistics: Performance Models of Natural Language, University of Amsterdam: Institute for logic, language and computation, Amsterdam, 1995.

Myung-Seok Choi received the B.S. degree in 1996 and the M.S. degree in 1998 in Computer Science from Korea Advanced Institute of Science and Technology (KAIST), Korea. He is currently working toward the Ph.D. degree at the Division of Computer Science, Department of Electrical Engineering and Computer Science, KAIST. His research interests include natural language parsing, information retrieval, and machine learning.

Kong-Joo Lee received the B.S. degree in Computer Science from the Sogang University, Korea, in 1992, the M.S. degree in 1994 and the Ph.D. degree in 1998 in Computer Science from KAIST. During 1998 through 2002 she was working for Microsoft, Korea. In 2003, she was a faculty member of the Department of Computer Science in Ewah Womans University. Since 2004, she has been a faculty member of the School of Computer Information Technology in KyungIn Women's College, Korea. Her research interests include information retrieval, natural language parsing, and machine translation.

Key-Sun Choi received the B.S. degree in Mathematics from Seoul National University, Korea, in 1978, the M.S. degree in 1980 and the Ph.D. degree in 1986 in Computer Science from KAIST. From 1985 to 1986, he was a professor in Hangook University for Foreign Studies. In 1987 and 1988, he was an invited researcher in NEC C&C, Japan. Since 1988, he has been a faculty member of the Division of Computer Science, Department of Electrical Engineering and Computer Science and also a director of KORTERM, KAIST. His research interests include natural language processing, machine translation, information retrieval, and terminology.

Gil Chang Kim received the B.S. degree in Electrical Engineering from University of Michigan, U.S., in 1963, the M.S. degree in 1966 and the Ph.D. degree in 1969 in Mathematics from University of Texas Austin, U.S. From 1969 to 1970, he was a researcher in NASA, U.S. In 1970 and 1971, he was a researcher in USAF Systems Command, U.S. During 1972 through 1973, he was an invited researcher in IBM T.J. Watson research center, U.S. In 1986 and 1987, he was an invited researcher in NEC C&C, Japan. Since 1971, he has been a faculty member of the Division of Computer Science, Department of Electrical Engineering and Computer Science, KAIST. His research interests include natural language processing and machine translation.