Automatic Partial Parsing Rule Acquisition Using Decision Tree Induction*

Myung-Seok Choi, Chul Su Lim, and Key-Sun Choi

Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Republic of Korea
{mschoi, cslim}@kaist.ac.kr kschoi@cs.kaist.ac.kr

Abstract. Partial parsing techniques try to recover syntactic information efficiently and reliably by sacrificing completeness and depth of analysis. One of the difficulties of partial parsing is finding a means to extract the grammar involved automatically. In this paper, we present a method for automatically extracting partial parsing rules from a tree-annotated corpus using decision tree induction. We define the partial parsing rules as those that can decide the structure of a substring in an input sentence deterministically. This decision can be considered as a classification; as such, for a substring in an input sentence, a proper structure is chosen among the structures occurred in the corpus. For the classification, we use decision tree induction, and induce partial parsing rules from the decision tree. The acquired grammar is similar to a phrase structure grammar, with contextual and lexical information, but it allows building structures of depth one or more. Our experiments showed that the proposed partial parser using the automatically extracted rules is not only accurate and efficient, but also achieves reasonable coverage for Korean.

1 Introduction

Conventional parsers try to identify syntactic information completely. These parsers encounter difficulties when 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 developed in response to these problems. This technique aims to recover syntactic information efficiently and reliably from unrestricted texts by sacrificing completeness and depth of analysis, and relying on local information to resolve ambiguities [1].

Partial parsing techniques can be roughly classified into two groups. The first group of techniques involves partial parsing via finite state machines [2,3,9,10]. These approaches apply the sequential regular expression recognizer to an input sentence. When multiple rules match an input string at a given position,

* 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.

R. Dale et al. (Eds.): IJCNLP 2005, LNAI 3651, pp. 143–154, 2005.

the longest-matching rule is selected. Therefore, these parsers always produce a single best analysis and operate very fast. In general, these approaches use a hand-written regular grammar. As would be expected, manually writing a grammar is both very time consuming and prone to have inconsistencies.

The other group of partial parsing techniques is text chunking, that is, recognition of non-overlapping and non-recursive cores of major phrases (chunks), by using machine learning techniques [4,7,8,13,15,17]. Since Ramshaw and Marcus [15] first proposed formulating the chunking task as a tagging task, most chunking methods have followed this *word-tagging* approach. In base noun phrase 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[1]:

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

With respect to these approaches, there have been several studies on automatically extracting chunking rules from large-scale corpora using transformation-based learning [15], error-driven pruning [7], the ALLiS top-down inductive system [8]. However, it is not yet clear how these approaches could be extended beyond the chunking task.

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 to extract rules with higher accuracy and broader coverage. We define the partial parsing rules as those that can establish the structure of a substring in an input sentence deterministically. This decision can be considered as a classification; as such, for a substring in an input sentence, a proper structure is chosen among the structures occurred in the corpus, as extended from the *word-tagging* approach of text chunking. For the classification, we use decision tree induction with features of contextual and lexical information. In addition, we use negative evidence, as well as positive evidence, to gain higher accuracy. For general recursive phrases, all possible substrings in a parse tree are taken into account by extracting evidence recursively from a parse tree in a training corpus. We induce partial parsing rules from the decision tree, and, to retain only those rules that are accurate, verify each rule through cross-validation.

In many cases, several different structures are assigned to the same substring in a tree-annotated corpus. Substrings for coordination and compound nouns are typical examples of such ambiguous cases in Korean. These ambiguities can prevent us from extracting partial parsing rules that cover the substrings with more than one substructure and, consequently, can cause the result of partial parsing to be limited to a relatively shallow depth. In this work, we address this problem by merging substructures with ambiguity using an underspecified representation.

[1] This example is excerpted from Tjong Kim Sang [17].

This underspecification leads to broader coverage without deteriorating either the determinism or the precision of partial parsing.

The acquired grammar is similar to a phrase structure grammar, with contextual and lexical information, but it allows building structures of depth one or more. It is easy to understand; it can be easily modified; and it can be selectively added to or deleted from the grammar. Partial parsing with this grammar processes an input sentence deterministically using longest-match heuristics. The acquired rules are then recursively applied to construct higher structures.

2 Automatic Rule Acquisition

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

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

This template shows how 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 remainder of an input sentence minus 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. Note that applying a partial parsing rule results in a structure of depth one or more. In other words, the rules extracted by this rule template reduce a substring into a subtree, as opposed to a single non-terminal; hence, the resultant rules can be applied more specifically and strictly.

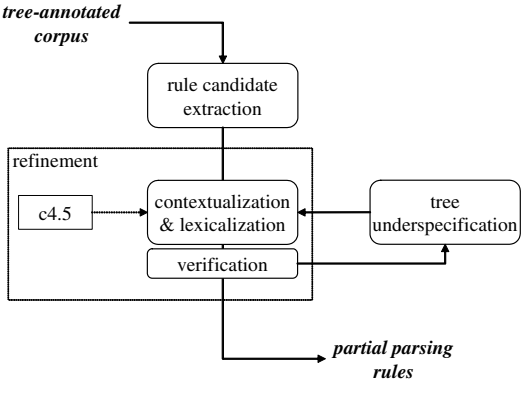

Fig. 1. Procedure for extracting partial parsing rules

Figure 1 illustrates the procedure for the extraction of partial parsing rules. First, we extract all possible rule candidates from a tree-annotated corpus, compliant with the rule template. The extracted candidates are grouped according

to their respective substrings. Next, using the decision tree method, these candidates are enriched with contextual and lexical information. The contextualized and lexicalized rules are verified through cross-validation to retain only those rules that are accurate. The successfully verified accurate rules become the final partial parsing rules. Remaining rules that cannot be verified are forwarded to the tree underspecification step, which merges tree structures with hard ambiguities. As seen in Fig. 1, the underspecified candidates return to the refinement step. The following subsections describe each step in detail.

2.1 Extracting Candidates

From the tree-annotated corpus, we extract all the possible candidates for partial parsing rules in accordance with the rule template. Scanning input sentences annotated with its syntactic structure one by one, we can extract the substructure corresponding to every possible substring at each level of the syntactic structure. We define level 0 as part-of-speech tags in an input sentence, and level n as the nodes whose maximum depth is n. If no structure precisely corresponds to a particular substring, then a null substructure is extracted, which represents negative evidence.

Figure 2 shows an example sentence[2] with its syntactic structure[3] and some of the candidates for the partial parsing rules extracted from the left side of the example. In this figure, the first partial parsing rule candidate shows how the substring 'npp' can be constructed into the substructure 'NP'. S_{null} denotes negative evidence.

The extracted rule candidates are gathered and grouped according to their respective substrings. Figure 3[4] shows the candidate groups. In this figure, G_1 and G_2 are the group names, and the number in the last column refers to the frequency that each candidate occurs in the training corpus. Group G_1 and G_2 have 2 and 3 candidates, respectively. When a particular group has only one candidate, the candidate can always be applied to a corresponding substring

[2] 'NOM' refers to the nominative case and 'ACC' refers to the accusative case. The term 'npp' denotes personal pronoun; 'jxt' denotes topicalized auxiliary particle; 'ncn' denotes non-predicative common noun; 'jco' denotes objective case particle; 'pvg' denotes general verb; 'ef' denotes final ending; and 'sf' denotes full stop symbol. For a detailed description of the KAIST corpus and its tagset, refer to Lee [11]. The symbol '+' is not a part-of-speech, but rather a delimiter between words within a word phrase.

[3] 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—a particle or a verbal-ending—indicates the grammatical role of the word phrase. In the 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 [12]. For example, a word phrase "*na*/npp + *neun*/jxt" is annotated as "(NP *na*/npp) + *neun*/jxt", as in Fig. 2.

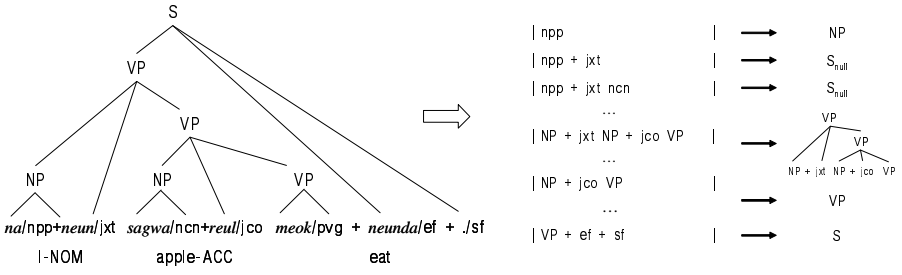

Fig. 2. An example sentence and the extracted candidates for partial parsing rules

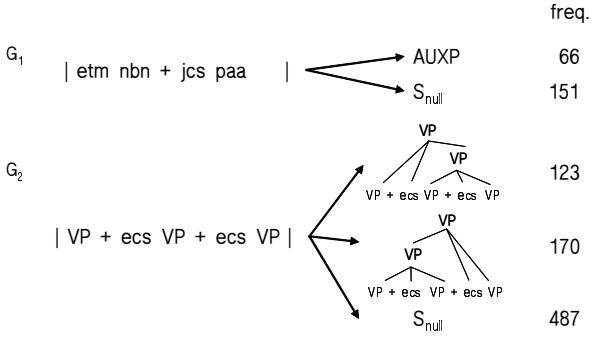

Fig. 3. Groups of partial parsing rules candidates

deterministically. In contrast, if there is more than one candidate in a particular group, those candidates should be enriched with contextual and lexical information to make each candidate distinct for proper application to a corresponding substring.

2.2 Refining Candidates

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

First, each candidate needs to be annotated with contextual and lexical information occurring in the training corpus, as shown in Fig. 4. In this figure, we can see that a substring with lexical information such as '*su*/nbn' unambiguously constitutes the substructure 'AUXP'. We use the decision tree method, C4.5 [14], to select the important contextual and lexical information that can facilitate the establishment of unambiguous partial parsing rules. The features used in the decision tree method are the lexical information of each terminal or

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

sal/pvg +	| *r*/etm ***su***/nbn + *ga*/jcs ***iss***/paa	| + *da*/ef → AUXP
i/jp +	| *r*/etm ***su***/nbn + *ga*/jcs ***eop***/paa	| + *da*/ef → AUXP
nolla/pvg +	| *n*/etm *jeok*/nbn + *i*/jcs *iss*/paa	| + *da*/ef → S_{null}
wanjeonha/paa +	| *n*/etm *geot*/nbn + *i*/jcs *eop*/paa	| + *go*/ecc → S_{null}
kkeutna/pvg +	| *n*/etm *geut*/nbn + *i*/jcs *ani*/paa	| + *ra*/ecs → S_{null}
ik/pvg +	| *neun*/etm *geut*/nbn + *i*/jcs *jot*/paa	| + *da*/ef → S_{null}
ha/xsv +	| *r*/etm *nawi*/nbn + *ga*/jcs *eop*/paa	| + *da*/ef → S_{null}

Fig. 4. Annotated candidates for the G_1 group rules

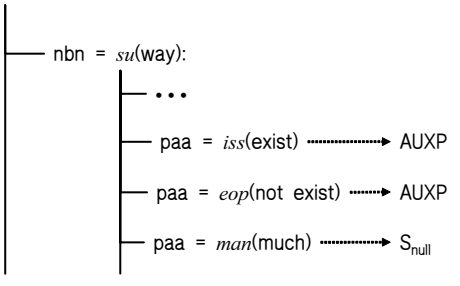

Fig. 5. A section of the decision tree

non-terminal for the *substring*, and the parts-of-speech and lexical information for 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 a section of the decision tree learned from our example substring. The deterministic partial parsing rules in Fig. 6 are extracted from the decision tree. As shown in Fig. 6, only the lexical entries for the second and the fourth morphemes in the substring are selected as additional lexical information, and none of the contexts is selected in this case. We should note that the rules induced from the decision tree are ordered. Since these ordered rules do not interfere with those from other groups, they can be modified without much difficulty.

|	etm ***su***/nbn + jcs ***iss***/paa	|	⟶ AUXP
|	etm ***su***/nbn + jcs ***eop***/paa	|	⟶ AUXP
|	etm ***su***/nbn + jcs ***man***/paa	|	⟶ S_{null}

Fig. 6. Partial parsing rules extracted from a section of the decision tree in Fig. 5

After we enrich the partial parsing rules using the decision tree method, we verify them by estimating the accuracy of each rule to filter out less deterministic rules. We estimate the error rates (%) of the rule candidates via a 10-fold cross validation on the training corpus. The rule candidates of the group with an error rate that is less than the 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 as the final partial parsing rules, because the estimated error rate of the group was higher than the threshold. The candidates in G_2 are set aside for tree underspecification processing. Using the threshold θ, we can control the number of the final partial parsing rules and the ratio of the precision/recall trade-off for the parser that adopts the extracted partial parsing rules.

2.3 Dealing with Hard Ambiguities: The Underspecified Representation

The group G_2 in Fig. 3 has one of the attachment ambiguities, namely, consecutive subordinate clauses. Figure 7 shows sections of two different trees extracted from a tree-annotated corpus. The two trees have identical *substrings*, but are analyzed differently. This figure exemplifies how an ambiguity relates to the 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, that are difficult to resolve with local information. The resolution usually requires lexical co-occurrence, global context, or semantics. Such ambiguities can deteriorate the precision of partial parsing or limit the result of partial parsing 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. In this paper, we refer to them as *internal syntactic ambiguities*. We manually examined the patterns of the internal syntactic ambiguities, which were found in the KAIST corpus as they could not be refined automatically due to low estimated accuracies. During the process, we observed that few internal syntactic ambiguities could be resolved with local information.

In this paper, we handle internal syntactic ambiguities by merging the candidates using tree intersection and making them underspecified. This underspecified representation enables an analysis with broader coverage, without deterio-

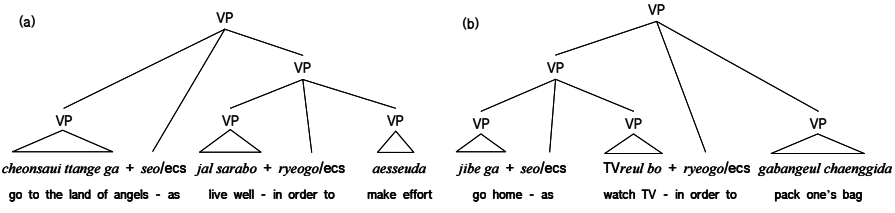

Fig. 7. Examples of internal syntactic ambiguities

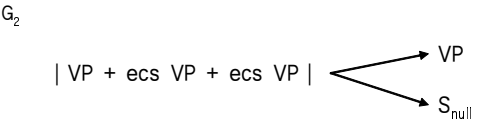

Fig. 8. Underspecified candidates

rating the determinism or 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 group G_2. In this figure, the first two rules in G_2 are reduced to the merged 'VP'. Underspecified candidates are also enriched with contextual and lexical information using the decision tree method, and they are verified through cross-validation, as described in Sect. 2.2. The resolution of internal syntactic ambiguities is forwarded to a module beyond the partial parser. If necessary, by giving all possible structures of underspecified parts, we can prevent a later processing from re-analyzing the parts. Any remaining candidates that are not selected as the partial parsing rules after all three steps are discarded.

3 Experimental Results

We have performed experiments to show the usefulness of automatically extracted partial parsing rules. For our evaluations, 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. Then, if any matches are found, the parser applies the first-matching rule in the group to the corresponding substring, because the rules induced from the decision tree are ordered.

In our experiments, we used the KAIST tree-annotated corpus [11]. The training corpus contains 10,869 sentences (289,362 morphemes), with an average length of 26.6 morphemes. The test corpus contains 1,208 sentences, with an average length of 26.0 morphemes. The validation corpus, used for choosing the threshold, θ, contains 1,112 sentences, with an average length of 20.1 morphemes, and is distinct from both the training corpus and the test corpus.

The performance of the partial parser was evaluated using PARSEVAL measures [5]. The F measure, a complement of the E measure [16], is used to combine precision and recall into a single measure of overall performance, and is defined as follows:

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

In the above equation, β is a factor that determines the weighting of precision and recall. Thus, $\beta < 1$ is used to weight precision heavier than recall, $\beta > 1$ is used to weight recall heavier than precision, and $\beta = 1$ is used to weight precision and recall equally.

Table 1. Precision/Recall with respect to the threshold, θ, for the validation corpus

θ	# of rules	precision	recall	$F_{\beta=0.4}$
6	18,638	95.5	72.9	91.6
11	**20,395**	**95.1**	**75.1**	**91.7**
16	22,650	94.2	78.0	91.6
21	25,640	92.6	83.3	91.2
26	28,180	92.0	84.7	90.9

Table 2. Experimental results of the partial parser for Korean

Grammar	precision	recall	$F_{\beta=0.4}$	$F_{\beta=1}$
baseline	73.0	72.0	72.9	72.5
depth 1 rule only	95.2	68.3	90.3	79.6
not underspecified	95.7	71.6	91.4	81.9
underspecified	**95.7**	**73.6**	**91.9**	**83.2**
underspecified (in case θ=26)	92.2	83.5	90.9	87.6
PCFG	80.0	81.5	80.2	80.7
Lee [11]	87.5	87.5	87.5	87.5

The parsing result can be affected by the predefined threshold, θ (described in Sect. 2.2), which can control both the accuracy of the partial parser and the number of the extracted rules. Table 1 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, a lower threshold, θ, corresponds to a higher precision and a lower recall. A higher threshold corresponds to a lower precision and a higher recall. For a partial parser, the precision is generally favored over the recall. In this paper, we used a value of 11 for θ, where the precision was over 95% and $f_{\beta=0.4}$ was the highest. The value of this threshold should be set according to the requirements of the relevant application.

Table 2 presents the precision and the recall of the partial parser for the test corpus when the threshold, θ, was given a value of 11. In the baseline grammar, we selected the most probable structure for a given substring from each group of candidates. The "depth 1 rule only" grammar is the set of the rules extracted along with the restriction, stating that only a substructure of depth one is permitted in the rule template. The "underspecified" grammar is the final version of our partial parsing rules, and the "not underspecified" grammar is the set of the rules extracted without the underspecification processing. Both PCFG and Lee [11] are statistical full parsers of Korean, and Lee enriched the grammar using contextual and lexical information to improve the accuracy of a parser. Both of them were trained and tested on the same corpus as ours was for comparison. The performance of both the "not underspecified" grammar and the "underspecified" grammar was greatly improved compared to the baseline grammar and PCFG, neither of which adopts contextual and lexical information in their rules. The "not underspecified" grammar performed better than

the "depth 1 rule only" grammar. 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 [6]. Comparing the "underspecified" grammar with the "not underspecified" grammar, we can see that underspecification leads to broader coverage, that is, higher recall. The precision of the "underspecified" grammar was above 95%. In other words, when a parser generates 20 structures, 19 out of 20 structures are correct. However, its recall dropped far beyond that of the statistical full parser [11]. When we set θ to a value of 26, the underspecified grammar slightly outperformed that of the full parser in terms of $f_{\beta=1}$, although the proposed partial parser does not always produce one complete parse tree[5]. It follows from what has been said thus far that the proposed parser has the potential to be a high-precision partial parser and approach the performance level of a statistical full parser, depending on the threshold θ.

The current implementation of our parser has a $O(n^2 m_r)$ worst case time complexity for a case involving a skewed binary tree, where n is the length of the 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 of depth one or more in each level of parsing, and, differing from full parsing, the number of recursions in the partial parsing seems to be limited[6], we can parse in near-linear time. Figure 9 shows the time spent in parsing as a function of the sentence length[7].

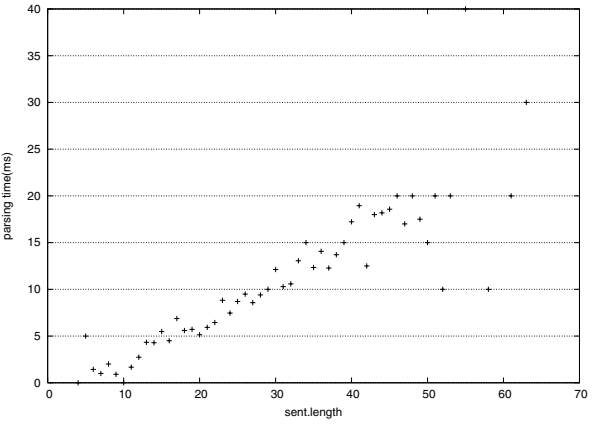

Fig. 9. Time spent in parsing

Lastly, we manually examined the first 100 or so errors occurring in the test corpus. In spite of underspecification, the errors related to conjunctions and

[5] 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%.

[6] In our parser, the maximum number of recursion was 10 and the average number of recursion was 4.47.

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

attachments were the 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 that they cannot be resolved completely with partial parsing. In addition, there were errors related to noun phrase bracketing, date/time/unit expression, and either incorrectly tagged sentences or inherently ambiguous sentences. For date, time, and unit expressions, manually encoded rules may be effective with partial parsing, since they appear to be used in a regular way. We should note that many unrecognized phrases included expressions not occurring in the training corpus. This is obviously because our grammar cannot handle unseen substrings; hence, alleviating the sparseness in the *sequences* will be the goal of our future research.

4 Conclusion

In this paper, we have proposed a method of automatically extracting the partial parsing rules from a tree-annotated corpus using a decision tree method. We consider partial parsing as a classification; as such, for a substring in an input sentence, a proper structure is chosen among the structures occurred in the corpus. Highly accurate partial parsing rules can be extracted by (1) allowing rules to construct a subtree of depth one or more; (2) using decision tree induction, with features of contextual and lexical information for the classification; and (3) verifying induced rules through cross-validation. By merging substructures with ambiguity in non-deterministic rules using an underspecified representation, we can handle syntactic ambiguities that are difficult to resolve with local information, such as coordination and noun phrase bracketing ambiguities. Using a threshold, θ, we can control the number of the partial parsing rules and the ratio of the precision/recall trade-off of the partial parser. The value of this threshold should be set according to the requirements of the relevant application. Our experiments showed that the proposed partial parser using the automatically extracted rules is not only accurate and efficient, but also achieves reasonable coverage for Korean.

References

1. Abney, S.P.: Part-of-speech tagging and partial parsing. Corpus-Based Methods in Language and Speech. Kluwer Academic Publishers (1996)
2. Abney, S.P.: Partial parsing via finite-state cascades. Proceedings of the ESSLLI '96 Robust Parsing Workshop (1996) 8–15
3. Aït-Mokhtar, S., Chanod, J.P.: Incremental finite-state parsing. Proceedings of Applied Natural Language Processing (1997) 72–79
4. Argamon-Engelson, S., Dagan, I., Krymolowski, Y.: A memory-based approach to learning shallow natural language patterns. Journal of Experimental and Theoretical AI **11**(3) (1999) 369–390

5. Black, E., Abney, S., Flickenger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., Strzalkowski, T.: A procedure for quantitatively comparing the syntactic coverage of English grammars. Proceedings of the DARPA Speech and Natural Language Workshop (1991) 306–311
6. Bod, R.: Enriching Linguistics with Statistics: Performance Models of Natural Language. Ph.D Thesis. University of Amsterdam (1995)
7. Cardie, C., Pierce, D.: Error-driven pruning of treebank grammars for base noun phrase identification. Proceedings of 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (1998) 218–224
8. Déjean, H.: Learning rules and their exceptions. Journal of Machine Learning Research **2** (2002) 669–693
9. Hindle, D.: A parser for text corpora. Computational Approaches to the Lexicon. Oxford University (1995) 103–151
10. Hobbs, J.R., Appelt, D., Bear, J., Israel, D., Kameyama, M., Stickel, M., Tyson, M.: Fastus: A cascaded finite-state transducer for extracting information from natural-language text. Finite-State Language Processing. The MIT Press (1997) 383–406
11. Lee, K.J.: Probabilistic Parsing of Korean based on Language-Specific Properties. Ph.D. Thesis. KAIST, Korea (1998)
12. Lee, K.J., Kim, G.C., Kim, J.H., Han, Y.S.: Restricted representation of phrase structure grammar for building a tree annotated corpus of Korean. Natural Language Engineering **3**(2) (1997) 215–230
13. Muñoz, M., Punyakanok, V., Roth, D., Zimak, D.: A learning approach to shallow parsing. Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Copora (1999) 168–178
14. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers (1993)
15. Ramshaw, L.A., Marcus, M.P.: Text chunking using transformation-based learning. Proceedings of Third Wordkshop on Very Large Corpora (1995) 82–94
16. van Rijsbergen, C.: Information Retrieval. Buttersworth (1975)
17. Tjong Kim Sang, E.F.: Memory-based shallow parsing. Journal of Machine Learning Research **2** (2002) 559–594