Tree Annotation Tool using Two-phase Parsing to Reduce Manual Effort for Building a Treebank

So-Young Park, Yongjoo Cho, Sunghoon Son, Ui-Sung Song and **Hae-Chang Rim**
College of Computer Software & Media Technology
SangMyung University
7 Hongji-dong, Jongno-gu
SEOUL, KOREA
{ssoya,ycho,shson}@smu.ac.kr

Dept. of CSE
Korea University,
5-ka 1, Anam-dong, Seongbuk-ku
SEOUL, KOREA
{ussong,rim}@nlp.korea.ac.kr

Abstract

In this paper, we propose a tree annotation tool using a parser in order to build a treebank. For the purpose of minimizing manual effort without any modification of the parser, it performs two-phase parsing for the intra-structure of each segment and the inter-structure after segmenting a sentence. Experimental results show that it can reduce manual effort about 24.5% as compared with a tree annotation tool without segmentation because an annotation's intervention related to cancellation and reconstruction remarkably decrease although it requires the annotator to segment some long sentence.

1 Introduction

A treebank is a corpus annotated with syntactic information, and the structural analysis of each sentence is represented as a bracketed tree structure. This kind of corpus has served as an extremely valuable resource for computational linguistics applications such as machine translation and question answering (Lee et al., 1997; Choi, 2001), and has also proved useful in theoretical linguistics research (Marcus et al., 1993).

However, for the purpose of building the treebank, an annotator spends a lot of time and manual effort. Furthermore, it is too difficult to maintain the consistency of the treebank based on only the annotator (Hindle, 1989; Chang et al., 1997). Therefore, we require a tree annotation tool to reduce manual effort by decreasing the frequency of the human annotators' intervention. Moreover, the tool can improve the annotating efficiency, and help maintain the consistency of the treebank (Kwak et al., 2001; Lim et al., 2004).

In this paper, we propose a tree annotation tool using a parser in order to reduce manual effort for building a treebank. Fundamentally, it generates a candidate syntactic structure by utilizing a parser. And then, the annotator cancels the incorrect constituents in the candidate syntactic structure, and reconstructs the correct constituents.

2 Previous Works

Up to data, several approaches have been developed in order to reduce manual effort for building a treebank. They can be classified into the approaches using the heuristics (Hindle, 1989; Chang et al., 1997) and the approaches using the rules extracted from an already built treebank (Kwak et al., 2001; Lim et al., 2004).

The first approaches are used for Penn Treebank (Marcus et al., 1993) and the KAIST language resource (Lee et al., 1997; Choi, 2001). Given a sentence, the approaches try to assign an unambiguous partial syntactic structure to a segment of each sentence based on the heuristics. The heuristics are written by the grammarians so that they are so reliable (Hindle, 1989; Chang et al., 1997). However, it is too difficult to modify the heuristics, and to change the features used for constructing the heuristics (Lim et al., 2004).

The second approaches are used for SEJONG treebank (Kim and Kang, 2002). Like the first

approaches, they also try to attach the partial syntactic structure to each sentence according to the rules. The rules are automatically extracted from an already built treebank. Therefore, the extracted rules can be updated whenever the annotator wants (Kwak et al., 2001; Lim et al., 2004). Nevertheless, they place a limit on the manual effort reduction and the annotating efficiency improvement because the extracted rules are less credible than the heuristics.

In this paper, we propose a tree annotation tool using a parser for the purpose of shifting the responsibility of extracting the reliable syntactic rules to the parser. It is always ready to change the parser into another parser. However, most parsers still tend to show low performance on the long sentences (Li et al., 1990; Doi et al., 1993; Kim et al., 2000). Besides, one of the reasons to decrease the parsing performance is that the initial syntactic errors of a word or a phrase propagates to the whole syntactic structure.

In order to prevent the initial errors from propagating without any modification of the parser, the proposed tool requires the annotator to segment a sentence. And then, it performs two-phase parsing for the intra-structure of each segment and the inter-structure. The parsing methods using clause-based segmentation have been studied to improve the parsing performance and the parsing complexity (Kim et al., 2000; Lyon and Dickerson, 1997; Sang and Dejean, 2001). Nevertheless, the clause-based segmentation can permit a short sentence to be splitted into shorter segments unnecessarily although too short segments increase manual effort to build a treebank.

For the sake of minimizing manual effort, the proposed tree annotation tool induces the annotator to segment a sentence according to few heuristics verified by experimentally analyzing the already built treebank. Therefore, the heuristics can prefer the specific length unit rather than the linguistic units such as phrases and clauses.

3 Tree Annotation Tool

The tree annotation tool is composed of segmentation, tree annotation for intra-structure, and tree annotation for inter-structure as shown in Figure1.

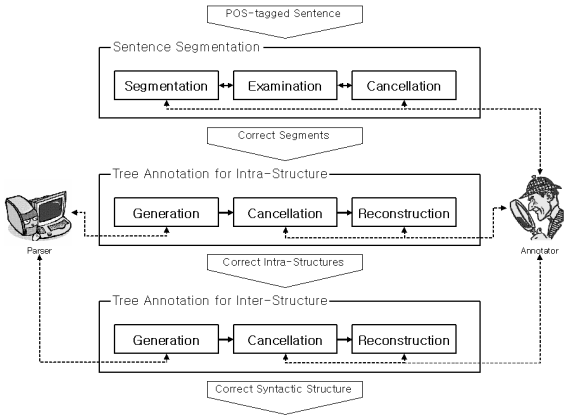


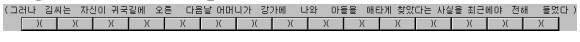
Figure 1: tree annotation tool

3.1 Sentence Segmentation

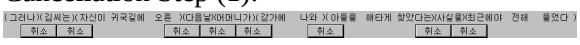
The sentence segmentation consists of three steps: segmentation step, examination step, and cancellation step. In the segmentation step, the annotator segments a long sentence. In the examination step, the tree annotation tool checks the length of each segment. In the cancellation step, it induces the annotator to merge the adjacent short segments by cancelling some brackets. Given a short sentence, the tree annotation tool skips over the sentence segmentation.

As shown in the top of Figure 2, the annotator segments a sentence by clicking the button ")(" between each pair of words. Since the tool regards segments as too short given the segment length 9, it provides the cancel buttons. And then, the annotator merges some segments by clicking the third button, the fifth button, and the sixth button of the middle figure. The bottom figure does not include the fourth button of the middle figure because a new segment will be longer than the segment length 9. When every segment is suitable, the annotator exclusively clicks the confirm button ignoring the cancel buttons.

Segmentation Step:



Cancellation Step (1):



Cancellation Step (2):

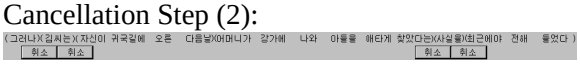


Figure 2: Sentence Segmentation

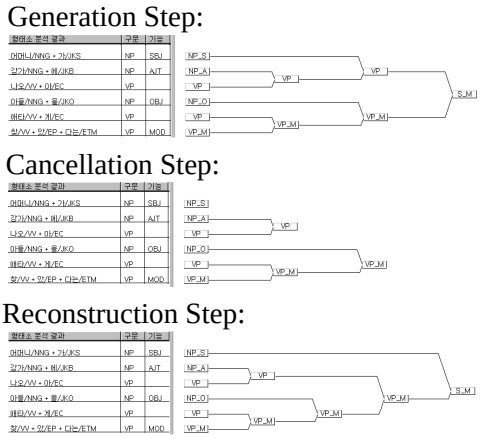


Figure 3: Tree Annotation for Intra-Structure

3.2 Tree Annotation for Intra-Structure

The tree annotation for intra-structure consists of three steps: generation step, cancellation step, and reconstruction step. In the generation step, the parser generates a candidate intra-structure for each segment of a sentence. And then, the tree annotation tool shows the annotator the candidate intra-structure. In the cancellation step, the annotator can cancel some incorrect constituents in the candidate intra-structure. In the reconstruction step, the annotator reconstructs the correct constituents to complete a correct intra-structure.

For example, the tree annotation tool shows the candidate intra-structure of a segment as shown in the top of Figure 3. Assuming that the candidate intra-structure includes two incorrect constituents, the annotator cancels the incorrect constituents after checking the candidate intra-structure as represented in the middle figure. And then, the annotator reconstructs the correct constituent, and the intra-structure is completed as described in the bottom of Figure 3.

3.3 Tree Annotation for Inter-Structure

The tree annotation for inter-structure also consists of three steps: generation step, cancellation step, and reconstruction step. In the generation step, the parser generates a candidate inter-structure based on the given correct intra-structures. And then, the tree annotation tool shows an annotator the candidate syntactic structure which includes both the intra-structures and the inter-structure. In the cancellation step, an annotator can cancel incorrect constituents. In the reconstruction step, the annotator reconstructs correct constituents to complete a correct syntactic structure.

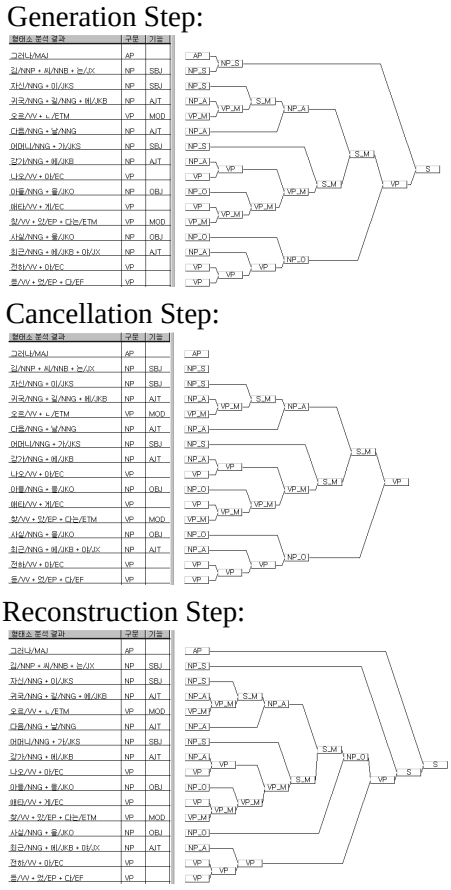


Figure 4: Tree Annotation for Inter-Structure

For example, the tree annotation tool represents the candidate syntactic structure of a sentence as illustrated in the top of Figure 4. Assuming that the candidate syntactic structure includes two incorrect constituents, the annotator cancels the incorrect constituents, and reconstructs the correct constituent. Finally, the intra-structure is completed as described in the bottom of Figure 3.

4 Experiments

In order to examine how much the proposed tree annotation tool reduces manual effort for building a Korean treebank, it is integrated with a parser (Park et al., 2004), and then it is evaluated on the test sentences according to the following criteria. The segment length (*Length*) indicates that a segment of a sentence is splitted when it

is longer than the given segment length. Therefore, the annotator can skip the sentence segmentation when a sentence is shorter than the segment length. The number of segments (*#Segments*) indicates the number of segments splitted by the annotator. The number of cancellations (*#Cancellations*) indicates the number of incorrect constituents cancelled by the annotator where the incorrect constituents are generated by the parser. The number of reconstructions (*#Reconstructions*) indicates the number of constituents reconstructed by the annotator. Assume that the annotators are so skillful that they do not cancel their decision unnecessarily. On the other hand, the test set includes 3,000 Korean sentences which never have been used for training the parser in a Korean treebank, and the sentences are the part of the treebank converted from the KAIST language resource (Choi, 2001). In this section, we analyze the parsing performance and the reduction effect of manual effort according to segment length for the purpose of finding the best segment length to minimize manual effort.

4.1 Parsing Performance According to Segment Length

For the purpose of evaluating the parsing performance given the correct segments, we classify the constituents in the syntactic structures into the constituents in the intra-structures of segments and the constituents in the inter-structures. Besides, we evaluate each classified constituents based on labeled precision, labeled recall, and distribution ratio. The labeled precision (*LP*) indicates the ratio of correct candidate constituents from candidate constituents generated by the parser, and the labeled recall (*LR*) indicates the ratio of correct candidate constituents from constituents in the treebank (Goodman, 1996). Also, the distribution ratio (*Ratio*) indicates the distribution ratio of constituents in the intra-structures from all of constituents in the original structure.

Table 1 shows that the distribution ratio of the constituents in the intra-structures increases according to the longer segment length while the distribution ratio of the constituents in the inter-structures decreases. Given the segment length 1, the constituents in the inter-structures of a sentence are the same as the constituents of the sentence because there is no evaluated constituent on intra-structure of one word. In the same way, the constituents in the intra-structures of a sentence are the same as the constituents of the sentence given the segment length 28 because all test sentences are shorter than 28 words.

Table 1: Parsing Performance

	Intra-Structure			Inter-Structure		
Length	LP	LR	Ratio	LP	LR	Ratio
1	0.00	0.00	0.00	87.62	86.06	100.00
2	100.00	93.42	52.25	74.45	74.08	47.75
3	100.00	97.27	66.61	60.63	58.55	33.39
4	98.93	96.47	74.27	62.11	59.71	25.73
5	97.68	96.05	79.47	65.30	63.65	20.53
6	96.50	95.47	83.24	67.88	66.68	16.76
7	95.45	94.45	86.12	70.84	69.81	13.88
8	94.34	93.10	88.47	74.24	73.23	11.53
9	93.41	92.36	90.40	76.85	76.25	9.60
10	92.65	91.47	92.01	78.99	78.31	7.99
11	91.91	90.65	93.43	81.53	80.72	6.57
12	91.19	89.86	94.59	84.39	83.60	5.41
13	90.54	89.23	95.62	86.92	85.97	4.38
14	89.87	88.61	96.54	88.82	88.19	3.46
15	89.34	87.99	97.30	90.39	89.41	2.70
16	88.98	87.65	97.97	90.50	89.86	2.03
17	88.64	87.27	98.51	91.64	89.61	1.49
18	88.37	86.99	98.98	92.92	90.84	1.02
19	88.15	86.76	99.34	92.97	91.30	0.66
20	87.98	86.57	99.58	92.00	91.62	0.42
21	87.83	86.42	99.76	92.73	92.36	0.24
22	87.76	86.36	99.83	93.60	93.20	0.17
23	87.74	86.36	99.89	94.46	93.81	0.11
24	87.69	86.27	99.94	97.67	94.74	0.06
25	87.65	86.26	99.97	100.00	95.45	0.03
26	87.63	86.24	99.99	100.00	100.00	0.01
27	87.63	86.16	99.99	100.00	100.00	0.01
28	87.62	86.06	100.00	0.00	0.00	0.00

As described in Table 1, the labeled precision and the labeled recall decrease on the intra-structures according to the longer segment length because both the parsing complexity and the parsing ambiguity increase more and more. On the other hand, the labeled precision and the labeled recall tend to increase on the inter-structures since the number of constituents in the inter-structures decrease, and it makes the parsing problem of the inter-structure easy. It is remarkable that the labeled precision and the labeled recall get relatively high performance on the inter-structures given the segment length less than 2 because it is so easy that the parser generates the syntactic structure for 3 or 4 words.

4.2 Reduction Effect of Manual Effort

In order to examine the reduction effect of manual effort according to segment length, we measure the frequency of the annotator's intervention on the proposed tool, and also classify the frequency into the frequency related to the intra-structure and the frequency related to the inter-structure.

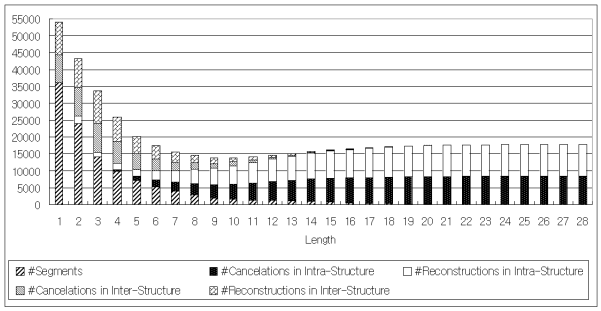


Figure 5: Reduction of Manual Effort

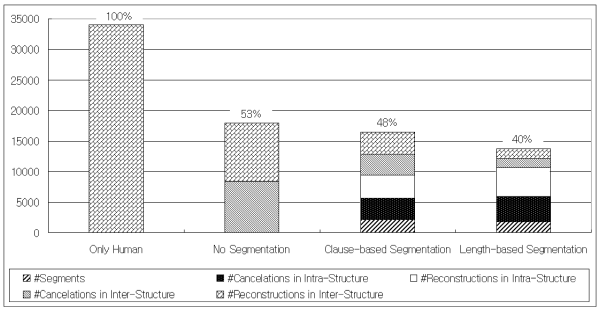


Figure 6: Comparison with Other Models

As shown in Figure 5, the total manual effort includes the number of segments splitted by the annotator, the number of constituents cancelled by the annotator, and the number of constituents reconstructed by the annotator.

Figure 5 shows that too short segments can aggravate the total manual effort since the short segments require too excessive segmentation work. According to longer length, the manual effort on the intra-structures increase because the number of the constituents increase and the labeled precision and the labeled recall of the parser decrease. On the other hand, the manual effort on the inter-structures decreases according to longer length on account of the opposite reasons. As presented in Figure 5, the total manual effort is reduced best at the segment length 9. It describes that the annotation's intervention related to cancellation and reconstruction remarkably decrease although it requires the annotator to segment some sentences. Also, Figure 5 describes that we hardly expect the effect of two-phase parsing based on the long segments while the short segments require too excessive segmentation work.

4.3 Comparison with Other Methods

In this experiment, we compare the manual effort of the four methods: the manual tree annotation tool (*only human*), the tree annotation tool using the parser (*no segmentation*), the tree annotation tool using the parser with the clause-based sentence segmentation (*clause-based segmentation*), and the tree annotation tool using the parser with the length-based sentence segmentation (*length-based segmentation*) where the segment length is 9 words.

As shown in Figure 6, the first method (*only human*) does not need any manual effort related to segmentation and cancellation but it requires too expensive reconstruction cost. The second method (*no segmentation*) requires manual effort related to cancellation and reconstruction without segmentation. As compared with the first method, the rest three methods using the parser can reduce manual effort by roughly 50% although the parser generates some incorrect constituents. Furthermore, the experimental results of the third and fourth methods shows that the two-phase parsing methods with sentence segmentation can reduce manual effort more about 9.4% and 24.5% each as compared with the second method.

Now, we compare the third method (*clause-based segmentation*) and the fourth method (*length-based segmentation*) in more detail. As represented in Figure 6, the third method is somewhat better than the fourth method on manual effort related to intra-structure. It show that the parser generates more correct constituents given the clause-based segments because the intra-structure of the clause-based segments is more formalized than the intra-structure of the length-based segments. However, the third method is remarkably worse than the fourth method on manual effort related to inter-structure. It describes that the third method can split a short sentence into shorter segments unnecessarily since the third method allows the segment length covers a wide range. As already described in Figure 5, too short segments can aggravate the manual effort. Finally, the experimental results shows that the length-based segments help the tree annotation tool to reduce manual effort rather than the clause-based segments.

5 Conclusion

In this paper, we propose the tree annotation tool which performs two-phase parsing for the intra-structure of each segment and the inter-structure after segmenting a sentence. The proposed tree annotation tool has the following characteristics. First, it can reduce manual effort to build a treebank. Experimental results show that it can improve approximately 60.0% and 24.5% as compared with the manual tree annotation tool and the tree annotation tool using one phase parsing. Second, it can prevent the initial syntactic errors of a word or a phrase from propagating to the whole syntactic structure without any modification of the parser because it takes sentence segmentation. Third, it can shift the responsibility of extracting the reliable syntactic rules to the parser. For future works, we will try to develop an automatic segmentation method to minimize manual effort.

Acknowledgements

This work was supported by Ministry of Education and Human Resources Development through Embedded Software Open Education Resource Center(ESC) at Sangmyung University.

References

Byung-Gyu Chang, Kong Joo Lee, Gil Chang Kim. 1997. Design and Implementation of Tree Tagging Workbench to Build a Large Tree Tagged Corpus of Korean. *In Proceedings of Hangul and Korean Information Processing Conference*, 421-429.

Ki-Sun Choi. 2001. "KAIST Language Resources ver. 2001." *The Result of Core Software Project from Ministry of Science and Technology*, http://kibs.kaist.ac.kr. (written in Korean)

Shinchi Doi, Kazunori Muraki, Shinichiro Kamei and Kiyoshi Yamabana. 1993. Long sentence analysis by domain-specific pattern grammar. *In Proceedings of the 6th Conference on the European Chapter of the Association of Computational Linguistics*, 466.

Joshua Goodman. 1996. Parsing Algorithms and Metrics. *In Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp.177–183.

Donald Hindle. 1989. Acquiring disambiguation rules from text. *In Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 118-125.

Sungdong Kim, Byungtak Zhang and Yungtaek Kim. 2000. Reducing Parsing Complexity by Intra-Sentence Segmentation based on Maximum Entropy Model. *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 164-171.

Ui-Su Kim, and Beom-Mo Kang. 2002. Principles, Methods and Some Problems in Compiling a Korean Treebank. *In Proceedings of Hangul and Korean Information Processing Conference 1997*. 155-162.

Yong-Jae Kwak, Young-Sook Hwang, Hoo-Jung Chung, So-Young Park, Hae-Chang Rim. 2001. FIDELITY: A Framework for Context-Sensitive Grammar Development. *In Proceedings of International Conference on Computer Processing of Oriental Languages*, 305-308.

Kong Joo Lee, Byung-Gyu Chang, Gil Chang Kim. 1997. Bracketing Guidlines for Korean Syntactic Tree Tagged Corpus Version 1. *Technical Report CS/TR-976-112, KAIST, Dept. of Computer Science*.

Wei-Chuan Li, Tzusheng Pei, Bing-Huang Lee and Chuei-Feng Chiou. 1990. Parsing long English sentences with pattern rules. *In Proceedings of the 13th International Conference on Computational Linguistics*, 410-412.

Joon-Ho Lim, So-Young Park, Yong-Jae Kwak, and Hae-Chang Rim. 2004. A Semi-automatic Tree Annotating Workbench for Building a Korean Treebank. *Lecture Note in Computer Science*, 2945:253-257.

Caroline Lyon and Bob Dickerson. 1997. Reducing the Complexity of Parsing by a Method of Decomposition. *In International Workshop on Parsing Technology*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313-330.

So-Young Park, Yong-Jae Kwak, Joon-Ho Lim, Hae-Chang Rim, and Soo-Hong Kim. 2004. Partially Lexicalized Parsing Model Utilizing Rich Features. *In Proceedings of the 8th International Conference on Spoken Language Processing*, 3:2201-2204.

Erik F. Tjong Kim Sang and Herve Dejean. 2001. Introduction to the CoNLL-2001 Shared Task: Clause Identi-fication. *In Proceedings of the Conference on Natural Language Learning*, 53-57.