Korean Treebank Transformation for Parser Training

DongHyun Choi
Dept. of Computer Science
KAIST
Korea
cdh4696@world.kaist.ac.kr

Jungyeul Park
Les Editions
an Amzer Vak
France
park@amzer-vak.fr

Key-Sun Choi
Dept. of Computer Science
KAIST
Korea
kschoi@cs.kaist.ac.kr

Abstract

Korean is a morphologically rich language in which grammatical functions are marked by inflections and affixes, and they can indicate grammatical relations such as subject, object, predicate, etc. A Korean sentence could be thought as a sequence of eojeols. An eojeol is a word or its variant word form agglutinated with grammatical affixes, and eojeols are separated by white space as in English written texts. Korean treebanks (Choi et al., 1994; Han et al., 2002; Korean Language Institute, 2012) use eojeol as their fundamental unit of analysis, thus representing an eojeol as a prepreterminal phrase inside the constituent tree. This eojeol-based annotating schema introduces various complexity to train the parser, for example an entity represented by a sequence of nouns will be annotated as two or more different noun phrases, depending on the number of spaces used. In this paper, we propose methods to transform eojeol-based Korean treebanks into entity-based Korean treebanks. The methods are applied to Sejong treebank, which is the largest constituent treebank in Korean, and the transformed treebank is used to train and test various probabilistic CFG parsers. The experimental result shows that the proposed transformation methods reduce ambiguity in the training corpus, increasing the overall F1 score up to about 9 %.

1 Introduction

The result of syntactic parsing is useful for many NLP applications, such as named entity recognition (Finkel and Manning, 2009), semantic role labeling (Gildea and Jurafsky, 2002), or sentimental analysis (Nasukawa and Yi, 2003). Currently most of the state-of-the-art constituent parsers take statistical parsing approach (Klein and Manning, 2003; Bikel, 2004; Petrov and Klein, 2007), which use manually annotated syntactic trees to train the probabilistic models of each consituents.

Even though there exist manually annotated Korean treebank corpora such as Sejong Treebank (Korean Language Institute, 2012), very few research projects about the Korean parser, especially using phrase structure grammars have been conducted. In this paper, we aim to transform the treebank so that it could be better used as training data for the already-existing English constituent parsers.

Most of Korean treebank corpora use eojeols as their fundamental unit of analysis. An eojeol is a word or its variant word form agglutinated with grammatical affixes, and eojeols are separated by white space as in English written texts (Choi et al., 2011). Figure 1 is one of the example constituent tree from the Sejong Treebank. As can be observed, an eojeol is always determined as a prepreterminal phrase [1]. But this kind of bracketing guideline could cause ambiguities to the existing algorithms for parsing English, because: (1) English does not have the concept of "eojeol", and (2) an eojeol can contain two or more morphemes with different grammatical roles. For example, Korean case par-

[1] A node is a prepreterminal if all the children of this node are preterminals (Part-Of-Speech tags such as `NNP` and `JKG`). Preterminal is defined to be a node with one child which is itself a leaf (Damljanovic et al., 2010).

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 78–88,
Jeju, Republic of Korea, 12 July 2012.

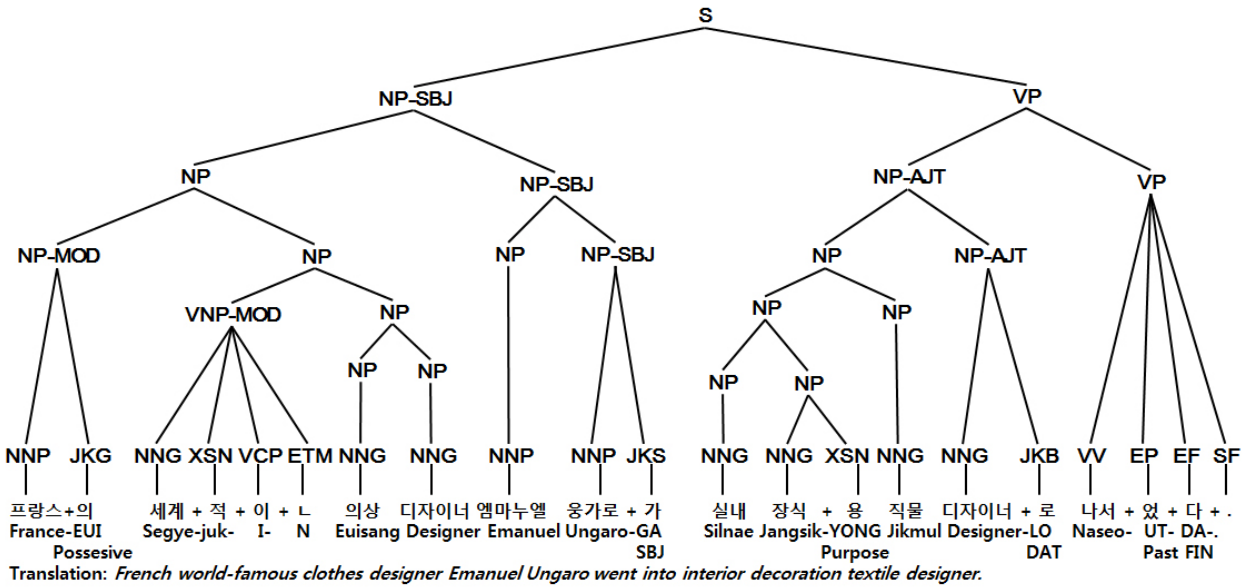

Figure 1: An example constituent tree and morphological analysis result from the Sejong treebank

ticles ('josa') are normally written inside the same eojeol with their argument nouns, but the whole eojeol is always considered as a prepreterminal noun phrase in the Korean treebank, as can be seen in the eojeol *Ungaro-GA*. Considering that the case particles in Korean play important role in determining the syntactic structure of a sentence, this could cause loss of information during the training phase. Moreover, *Emanuel Ungaro* is considered as two different noun phrases, because they simply belong to the two different eojeols (that is, a space exists between eojeols *Emanuel* and *Ungaro-GA*).

In this paper, we propose methods to refine the Sejong treebank which is currently the largest Korean treebank corpus. The methods are aimed at decreasing the ambiguities during the training phase of parsers, by separating phrases which are integrated into the same prepreterminal phrase due to the reason that they happen to be in the same eojeol, and integrating phrases into the same prepreterminal phrase which are separated because they happen to be in different eojeols. The refined datasets are trained and tested against three state-of-the-art parsers, and the evaluation results for each dataset are reported.

In section 2, the work about Korean parsers are briefly introduced. Sejong treebank is described with more detailed explanation in section 3, while the methods to transform the treebank are introduced in section 4. In section 5 the evaluation results of the transformed treebank using the three existing state-of-the-art parsers are introduced with an error report, and we discuss conclusions in section 6.

2 Related Work

There were some trials to build Korean constituent parsers, but due to the lack of appropriate corpus those trials were not able to acheive a good result. (Smith and Smith, 2004) tried to build a Korean parser by bilingual approach with English, and achieved labeled precision/recall around 40 % for Korean. More recently, (Park, 2006) tried to extract tree adjoining grammars from the Sejong treebank, and (Oh et al., 2011) build a system to predict a phrase tag for each eojeol.

Due to the partial free word order and case paricles which can decide the grammatical roles of noun phrases, there exist some works to build statistical dependency parsers for Korean. (Chung, 2004) presented a dependency parsing model using surface contextual model. (Choi and Palmer, 2011) converted the Sejong treebank into the dependency treebank, and applied the SVM algorithm to learn the dependency model.

NNG	General noun	IC	Interjection	JKQ	Quotational CP	XSV	Verb DS
NNP	Proper noun	MM	Adnoun	JX	Auxiliary PR	XSA	Adjective DS
NNB	Bound noun	MAG	General adverb	JC	Conjunctive PR	XR	Base morpheme
NP	Pronoun	MAJ	Conjunctive adverb	EP	Prefinal EM	SN	Number
NR	Numeral	JKS	Subjective CP	EF	Final EM	SL	Foreign word
VV	Verb	JKC	Complemental CP	EC	Conjunctive EM	SH	Chinese word
VA	Adjective	JKG	Adnomial CP	ETN	Nominalizing EM	NF	Noun-like word
VX	Auxiliary predicate	JKO	Objective CP	ETM	Adnominalizing EM	NV	Verb-like word
VCP	Copula	JKB	Adverbial CP	XPN	Noun prefix	NA	Unknown word
VCN	Negation adjective	JKV	Vocative CP	XSN	Noun DS	SF,SP,SS,SE,SO,SW	

Table 1: POS tags used in Sejong treebank (CP: case particle, EM: ending marker, DS: derivational suffix, PR: particle, SF SP SS SE SO: different types of punctuations, SW: currency symbols and mathematical symbols. Table borrowed from (Choi and Palmer, 2011))

Apart from the Sejong Treebank, there are few other Korean treebanks available. The KAIST treebank (Choi et al., 1994) contains constituent trees about approximately 30K sentences from newspapers, novels and textbooks. Also, the Penn Korean Treebank (Han et al., 2002) contains 15K constituent trees constructed from the sentences of newswire and military domains. The proposed methods are evaluated using the Sejong treebank because it is the most recent and the largest Korean treebank among those which is currently available.

3 Sejong Treebank

The Sejong treebank is the largest constituent treebank in Korean. It contains approximately 45K manually-annotated constituent trees, and their sources cover various domains including newspapers, novels and cartoon texts. Figure 1 shows an example of the Sejong constituent tree.

The tree consists of phrasal nodes and their functional tags as described in table 2. Each eojeol could contain one or more morphemes with different POS tags (Table 1 shows the POS tagset). In most cases, eojeols are determined by white spaces. As stated in its bracketing guidelines, the Sejong treebank uses eojeols as its fundamental unit of analysis [2]. This means that an eojeol is always treated as one prepreterminal phrase. This could cause confusions to the training system, because an eojeol could contain many morphemes which have very different grammatical roles, as can be seen in the example of *Ungaro-GA* - word *Ungaro* is a noun, where the nominative case particle *GA* suggests that this eojeol is used as a subject.

Table 2 shows phrase tags and functional tags used to construct the Sejong treebank. Some phrases are annotated with functional tags to clarify their grammatical role inside the sentence. There are three special phrase tags beside those in table 2: X indicates phrases containing only case particles or ending markers, L and R indicate left and right parenthesis.

Phrase-level tags		Functional tags	
S	Sentence	SBJ	Subject
Q	Quotative clause	OBJ	Object
NP	Noun phrase	CMP	Complement
VP	Verb phrase	MOD	Modifier
VNP	Copula phrase	AJT	Adjunct
AP	Adverb phrase	CNJ	Conjunctive
DP	Adnoun phrase	INT	Vocative
IP	Interjection phrase	PRN	parenthetical

Table 2: Phrase tags used in Sejong treebank.

4 Transforming Methods: from Eojeol-based to Entity-based

In this section, we describe the methods to transform the annotation schema of the Korean treebank from eojeol-based to entity-based using the examples of the Sejong treebank.

4.1 Method 1: POS Level Preprocessing

Before starting the actual transforming process, the system first detects emails, phone numbers and dates

[2]The bracketing guidelines could be requested from the Sejong project, but available only in Korean

based on their unique POS patterns. If the system detects a sequence of morphemes matching with one of predefined POS patterns inside an eojeol, then it groups those morphemes into one entity and tags it as a noun. This procedure aims to reduce the ambiguity of the corpus by reducing many miscellaneous mrophemes which in fact forms one phone number, email address or date information into one entity. Figure 2 shows an example of an eojeol whose five morphemes toghether represent one date, and its transformation result.

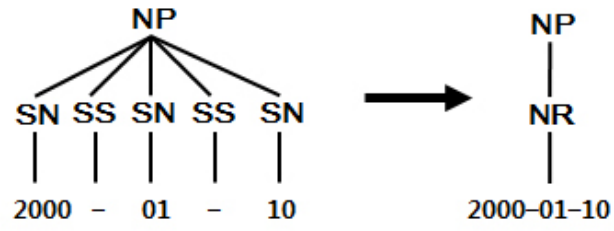

Figure 2: Example of an eojeol containing date: five morphemes are merged into one morpheme representing date.

Also, the morphemes representing chinese characters (POS: `SH`) and other foreign characters (POS: `SL`) are considered as nouns, since they are normally used to rewrite Korean nouns that have their foreign origin such as Sino-Korean nouns.

4.2 Method 2: Detecting NPs inside an Eojeol

Although an eojeol is considered to be one prepreterminal phrase as a whole, many eojeols contain separated noun components inside them. For example, a noun phrase *Ungaro-GA* in Figure 3 consists of a separated noun component *Ungaro* in it, plus josa *GA*. The system separates noun components from other endings and case particles, creates a new phrase containing those words and tags it as an NP. By doing so, the boundaries of the NP are more clarified - before transforming prepreterminal NPs could contain case particles and endings, but after the transformation it is not possible. Also the internal syntactic structures of phrases are revealed, providing more information to the parser.

4.3 Method 3: Finding Arguments of Josa

In this step, the system tries to find out the actual argument of each josa. For example, in figure 4 the

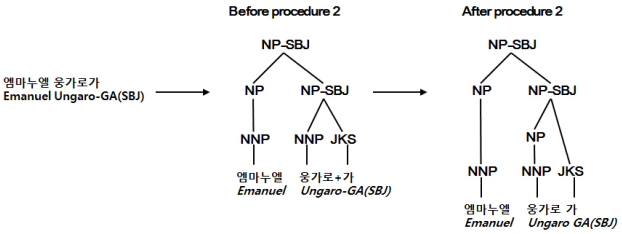

Figure 3: Detecting NP inside an eojeol: Case of a verb phrase

actual argument of the nominative josa *GA* is the whole person name *Emanuel Ungaro*, not only *Ungaro*. The system tries to find out the actual argument of each josa by using a rather simple heuristic:

1. Traverse the constituent parse tree in bottom-up, right-to-left manner.
2. If a phrase node is NP, its parent is also NP, and it directly dominates josa(s), then:
   (a) Create a new NP.
   (b) Attach the node to that NP, except the josa(s).
   (c) Attach all the other children of the parent node to the newly-created NP.
   (d) Remove all the children of the parent, and attach the new NP and remaining josa part to the parent node.
3. After the procedure ends, find and remove redundant NPs, if exist.

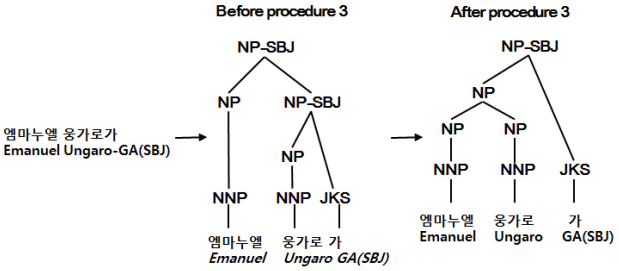

Figure 4: Example of applying the transformation heuristic

Method 3 is dependent on method 2, since method 2 first determines boundary of NPs which do not include any case particles.

4.4 Method 4: Integrating a Sequence of Nouns into One NP

Some of entities represented as sequences of nouns are considered as two or more separated noun

phrases since their components belong to the different eojeols. This could be problematic because an entity could sometimes be written without any whitespace between its component nouns. Figure 5 shows one of the case: person name *Emanuel Ungaro* is considered as two separated NPs since there exists a whitespace between a noun *Emanual* and a noun *Ungaro*. In this step, we aim to solve this problem.

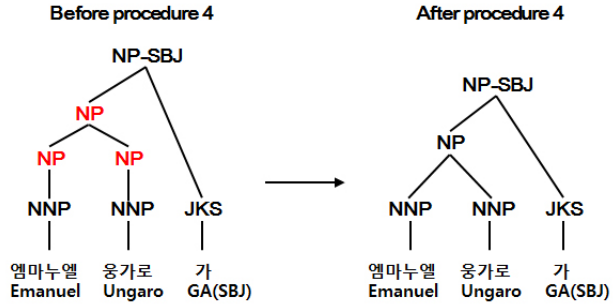

Figure 5: Integrating sequence of nouns representing one entity into one prepreterminal noun phrase

The system finds out an NP which has two NP children which dominates only the noun preterminal children. If the system finds such an NP, then it removes NP children and attaches their children directly to the found NP. Figure 5 shows an application example of the method.

This method is dependent on method 3, since this method assumes that an NP with its parent also NP does not have any case particles - which cannot be hold if method 3 is not applied.

4.5 Method 5: Dealing with Noun Conjunctions

The system tries to enumerate the noun conjunctions, rather than expressing those conjunctions in binary format. Current Sejong treebank expresses noun conjunctions in binary format - that is, to express the constituent tree for noun conjunctions, the nonterminal node has one NP child on its left which contains information about the first item of the conjunction, and the rest of conjunctions are expressed on the right child. Figure 6[3] shows an example of the Sejong constituent tree expressing the noun conjunctions, and its transformed version.

[3] *Mike-WA_(CNJ) Speaker-GA_(NOM) Jangchak-DOI-UH IT-DA.* ('Microphone and speaker are installed.')

Korean Sentence	마이크와 스피커가 장착되어 있다.
Pronounciation	*Mike-WA(CNJ) Speaker-GA(SBJ) JangChak-DOI-UH IT-DA.*
English translation	Microphone and speaker are installed.

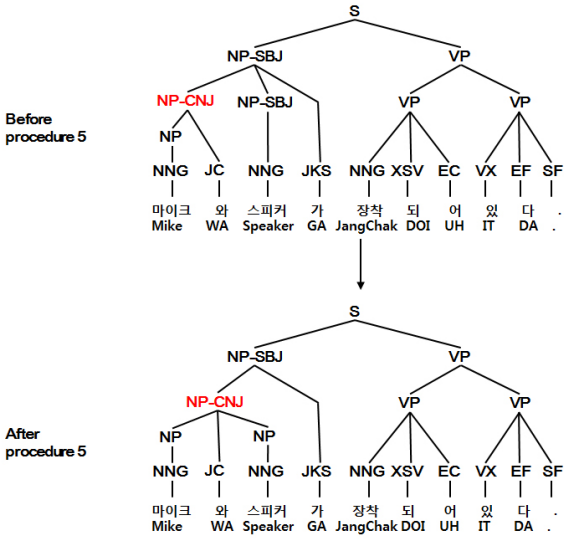

Figure 6: Enumerating Noun Conjunctions

By converting noun conjunctions into rather the 'enumerated' forms, two benefits could be gained: first, the resultant constituent tree will resemble more to the Penn-treebank constituent trees. Since most of the existing English parsers are trained on the Penn Treebank, we can expect that the enumerated form of conjunctions will more 'fit' to those parsers. Second, the conjunctions are expressed in much more explicit format, so the human users can more easily understand the conjunctive structures inside the constituent trees.

4.6 Method 6: Re-tagging Phrase Tags

In this step, the system re-tags some of phrase tags to clarify their types and to decrease training ambiguities. For example, a noun phrase with and without case particles should be distinguished. The system re-tags those noun phrases with case particles to JSP [4] to distinguish them from the pure noun phrases which consist of only nouns. Also, VP-MOD and VNP-MOD are re-tagged to DP, since they have very similar lexical formats with existing DPs. NP-MOD is converted into JSP-MOD - most of them consist of a NP with josa JKG, forming possesive cases. S-MOD remains as S-MOD if its head is JSP-MOD:

[4] It stands for a 'Josa Phrase'.

otherwise, it is also re-tagged to a DP. Figure 7[5] shows a re-tagging example.

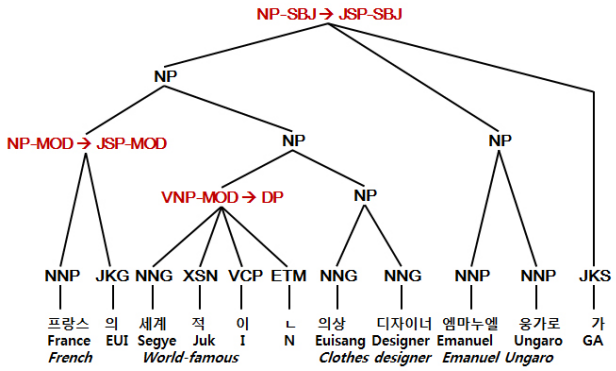

Figure 7: Example of retagging phrase tags: VP-MOD to DP, NP-MOD to JSP-MOD, and NP-SBJ to JSP-SBJ.

5 Evaluations

In this section, several experiment results using the standard F1 metric ($2PR/(P+R)$) are introduced to show the effect of each transforming method, and the most frequently shown error cases are explained.

5.1 Experiments using the Sejong Treebank

The proposed transformation methods are applied to the Sejong treebank, and the converted treebanks are used to train and test three different well-known statistical parsers, namely Stanford parser (Klein and Manning, 2003), Bikel-Collins parser (Bikel, 2012) and Berkeley parser (Petrov et al., 2006). To figure out the effect of each method, all six methods are sequentially applied one by one, and each version of the treebank is used to train and test each parser. The baseline treebank is the original Sejong treebank without any transformations. For the Korean head word extraction which will be used during parsing, the head percolation rule of (Choi and Palmer, 2011) is adapted. According to that paper, particles and endings were the most useful morphemes to determine dependencies between eojeols. Based on the observation, their rules are changed so that they give the best priorities on those morphemes. We use the preprocessing method described in (Park, 2006) for training trees. It replaces symboles with Penn-Treebank-like tags and corrects wrong morpheme boundary marks within the eojeol. Methods are applied cumulatively; for example, symbol 'M 1-6' means the version of a treebank to which method 1, 2, 3, 4, 5 and 6 are applied cumulatively.[6]

System	Corpus	P	R	F1
Stan.	**Baseline**	67.88%	61.77%	64.69%
	M 1	68.34%	61.93%	64.98%
	M 1-2	71.78%	67.50%	69.58%
	M 1-3	71.28%	67.91%	69.56%
	M 1-4	71.06%	67.08%	69.01%
	M 1-5	71.35%	67.27%	69.26%
	M 1-6	75.85%	72.07%	73.92%
Bikel.	**Baseline**	74.81%	70.39%	72.53%
	M 1	74.87%	70.45%	72.59%
	M 1-2	77.05%	73.84%	75.41%
	M 1-3	75.87%	72.88%	74.34%
	M 1-4	75.33%	72.10%	73.68%
	M 1-5	75.29%	72.18%	73.70%
	M 1-6	73.70%	71.05%	72.35%
Berk.	**Baseline**	75.25%	72.72%	73.96%
	M 1	74.54%	71.97%	73.23%
	M 1-2	77.27%	75.05%	76.14%
	M 1-3	75.60%	73.19%	74.38%
	M 1-4	75.69%	73.32%	74.49%
	M 1-5	76.53%	74.30%	75.40%
	M 1-6	78.60%	76.03%	77.29%

Table 3: Evaluation results of parsers, with various transformed versions of the Sejong treebank.

Table 3 shows the experimental results on each version of the treebanks using each parser. Since the corpus covers various domains (i.e. the style of sentences is not homogeneous.), we perform 10-fold cross-validation for our experiments. **Stan.** represents Stanford parser, **Bikel.** represents Bikel-Collins parser, and **Berk.** means Berkeley parser. For the Berkeley parser, we set the number of iteration as two for latent annotations. In this set of experiments, only phrase tags are the target of training and testing, not including functional tags.

As can be observed from the evaluation result, the performance is improved due to methods 2 and 6 are quite big compared to the effect of other four

[5] See Figure 1 for its transcription and translation.

[6] As pointed out by reviewers, we are planning the reversibility of transformations to be evaluated on the same trees for meaning comparison.

System	Corpus	P	R	F1
Stan.	**Baseline**	71.48%	69.40%	70.43%
	M 1	71.89%	69.75%	70.81%
	M 1-2	75.90%	73.44%	74.65%
	M 1-3	72.32%	69.76%	71.02%
	M 1-4	72.37%	69.97%	71.16%
	M 1-5	72.80%	70.28%	71.52%
	M 1-6	72.32%	69.81%	71.05%
Bikel.	**Baseline**	69.65%	66.80%	68.19%
	M 1	69.73%	66.97%	68.32%
	M 1-2	74.33%	71.90%	73.09%
	M 1-3	63.94%	64.57%	64.25%
	M 1-4	63.95%	65.04%	64.49%
	M 1-5	64.09%	65.05%	64.57%
	M 1-6	62.94%	64.16%	63.54%
Berk.	**Baseline**	76.82%	75.28%	76.04%
	M 1	76.73%	75.06%	75.89%
	M 1-2	79.59%	77.91%	78.74%
	M 1-3	75.24%	72.16%	73.67%
	M 1-4	75.02%	73.01%	74.00%
	M 1-5	75.58%	73.61%	74.58%
	M 1-6	74.37%	71.93%	73.13%

Table 4: Evaluation results of parsers, with phrase tags and functional tags together as learning target.

methods. Especially, the performance increase due to the method 6 strongly suggests that Sejong phrase tagsets are not enough to distinguish the types of phrases effectively. Except those two methods, only the method 5 increases the overall performance slightly, and methods 1, 3 and 4 do not have any significant effect on the performance or even sometimes decrease the overall performance.

Although the usage of functional tags is different from that of phrase tags, the Sejong treebank has a very rich functional tag set. Considering the results of the previous experiments, it is highly likely that some of phrasal information is encoded into the functional tags. To prove that, another set of experiments is carried out. In this time, parsers are trained not only on phrase tags but also on functional tags. Table 4 shows the evaluation results.

As can be observed, by keeping functional tags to train and test parsers, the baseline performance increases 3 to 6 % for the Stanford and Berkeley parsers. Only the performance of the Bikel parser is decreased - it is highly possible that the parser fails to find out the appropriate head word for each possible tag, because the number of possible tags is increased greatly by using the functional tags along with the phrase tags.

In both set of experiments, the method 3 decreases the overall performance. This strongly suggests that finding the actual argument of josa directly is quite a challenging work. The performance drop is considered mainly because the branching problem at the higher level of the constituent tree is counted twice due to the josa.

5.2 Experiments using the Penn Korean Treebank

To show the effect of the transformation methods more clearly, the Penn Korean Treebank (Han et al., 2002) is used as another treebank for experimentation: (Chung et al., 2010) describes about major difficulties of parsing Penn Korean Treebank. The same three parsers are trained and tested using the treebank. Due to the different annotation guidelines and different tagsets, transformation methods 1, 5 and 6 cannot be applied on the treebank. Thus, only method 2, 3 and 4 are used to transform the treebank. Table 5 shows the evaluation results.

System	Corpus	P	R	F1
Stan.	**Baseline**	82.84%	80.28%	81.54%
	M 2	85.29%	83.25%	84.26%
	M 2-3	84.52%	82.71%	83.61%
	M 2-4	84.52%	82.92%	83.72%
Bikel.	**Baseline**	81.49%	78.20%	79.81%
	M 2	75.82%	74.47%	75.13%
	M 2-3	73.50%	69.66%	71.53%
	M 2-4	73.45%	69.66%	71.51%
Berk.	**Baseline**	85.11%	81.90%	83.47%
	M 2	83.40%	81.04%	82.20%
	M 2-3	82.36%	80.52%	81.43%
	M 2-4	82.97%	81.28%	82.12%

Table 5: Evaluation on Penn Korean Treebank.

The overall performance of training the Penn Korean treebank is higher than that of the Sejong treebank. There could be two possible explanations. First one is, since the Penn Korean treebank tries to follow English Penn treebank guidelines as much

as possible, thus annotation guidelines of the Korean Penn treebank could be much "familiar" to the parsers than that of the Sejong treebank. The second explanation is, since the domain of the Penn Korean treebank is much more restricted than that of the Sejong treebank, the system could be trained for the specific domain. The best performance was gained with the Stanford parser, with the treebank transformed by method 2. Actually, (Chung et al., 2010) also investigated parsing accuracy on the Penn Korean treebank; the direct comparison could be very difficult because parsing criteria is different.

5.3 Error Analysis

In this section, some of the parsing error cases are reported. Berkeley parser trained with the Sejong treebank is used for error analysis. Both phrase tags and functional tags are used to train and test the system.

5.3.1 Locating Approximate Positions of Errors

As the first step to analyze the errors, we tried to figure out at which points of the constituent tree errors frequently occur – do the errors mainly occur at the bottom of the trees? Or at the top of the trees? If we can figure out approximate locations of errors, then the types of errors could be predicted.

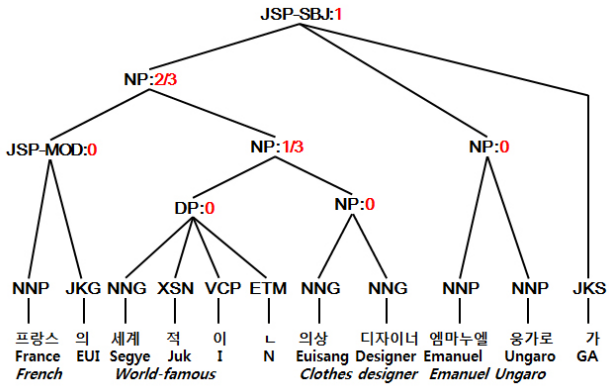

Figure 8: Example of assigning levels to each phrasal node.

To define the level of each nonterminal node of the constituent tree, the following rules are used:

- The level of prepreterminal node is 0.
- The levels of other phrasal nodes are defined as: the maximal level of their children + 1.
- Once the levels of all the phrasal nodes are calculated, normalize the levels so that they have the values between 0 and 1.

Figure 8 shows an example of constituent tree with levels assigned to its phrasal nodes. All the prepreterminal nodes have level value 0, and the topmost node has level 1.

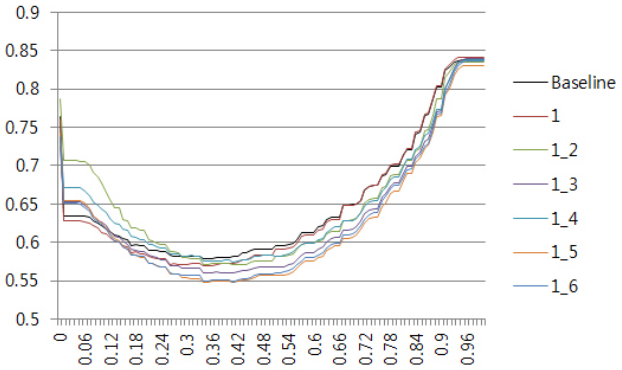

Figure 9: Performance of the system on each level of the parse tree

Once the levels are assigned to each constituent tree, only those constituents with levels larger than or equal to the predefined threshold μ are used to evaluate the system. μ are increased from 0 to 1 with value 0.01. Higher μ value means that the system is evaluated only for those constituents positioned at the top level of the constituent tree.

Figure 9 shows the evaluation results. X-axis represents the value of μ, and Y-axis represents the F1-score. As can be observed, most of the errors occur at the mid-level of the constituent trees. Also, the effects of some methods are explicitly shown on the graph. For example, method 2 greatly increases the performance at low level of the constituent tree, suggesting improved consistency in detemining prepreterminal NP nodes. Also, it is shown that the proposed methods does not affect the performance of mid-level and top-level constituent decisions - this suggests that the future works should be more focused on providing more information about those mid-level decision to the treebank annotation.

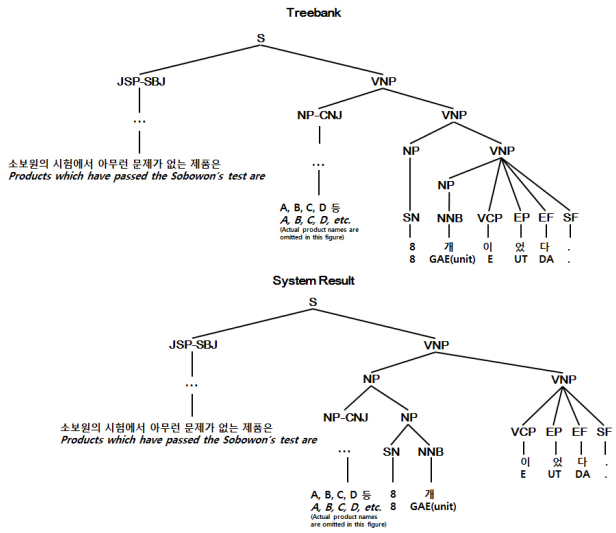

Figure 10: Example of NP boundary detection error. Part of parse tree as well as name of the enumerated products are omitted to more clearly show the example itself.

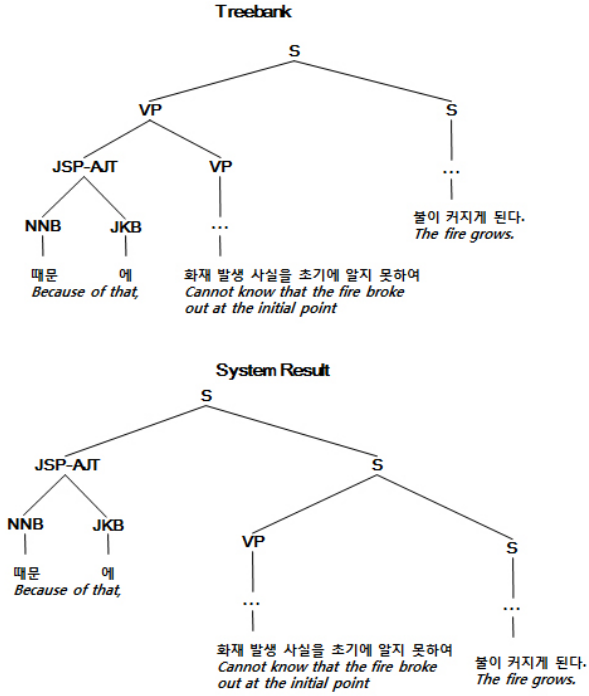

Figure 11: Example of a phrase (**JSP-AJT**) which failed to find its right modifee.

5.3.2 Frequent Error Cases

In this section, four major parsing error cases are described.

Detecting Boundaries of NP. Although the method 4 tries to find and gather the sequence of nouns which actually belong to one NP, it misses some of the cases. Figure 10 shows such example. Some parts of the tree are omitted using the notation '...' to show the example more simply. Although it is counted as the parser error, the result of the parser is more likely to be an answer - the number of those products is 8, not their action. The Sejong treebank tree is annotated in that way because the number '8' and bound noun *Gae* ('unit'), representing as units, are separated by a space. To detect such kind of separated NPs and transform them into one NP will be our next task.

Finding an Appropriate Modifee. Some phrases modifying other phrases were failed to find their appropriate modifees. Figure 11 shows an example of such kind of error case.

Detecting an Appropriate Subject of the Sentence. This case frequently occurs when a sentence is quotated inside the other sentence. In this case, the subject of quotated sentence is often considered as the subject of the whole sentence, because the quotated sentences in Korean are usually first stated and then the subject of the whole sentence shows up. Figure 12 shows an example of the erroneously detected subject.

The Wrongly-tagged Topmost Node. Some of Sejong treebank trees have phrases which are not tagged as **S** as their topmost nodes. This could cause confusion during the training. Figure 13 shows such example.

6 Conclusion and Future Work

Although there exist some manually-annotated large-enough constituent treebanks such as Sejong treebank, it was hard to apply the algorithms for English parsers to Korean treebanks, because they were annotated in eojeol-based scheme, which concept does not exist in English. In this paper, we showed the possibility of acquiring good training and testing results with the existing parsers trained using the existing Korean treebanks, if it undergoes some simple transforming procedures. The error analysis result shows that, indeed the proposed method improves the performance of parser at the lower level of constituent tree.

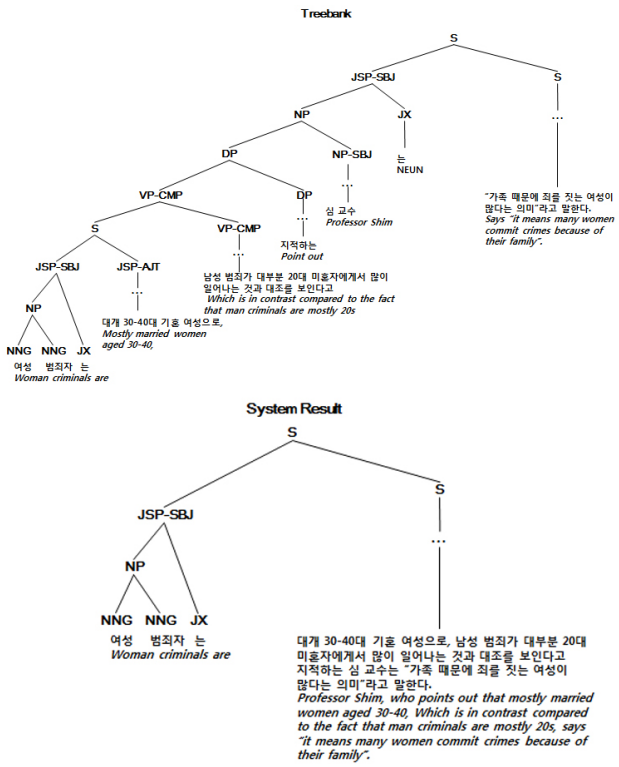

Figure 12: Example of a wrongly-detected subject.

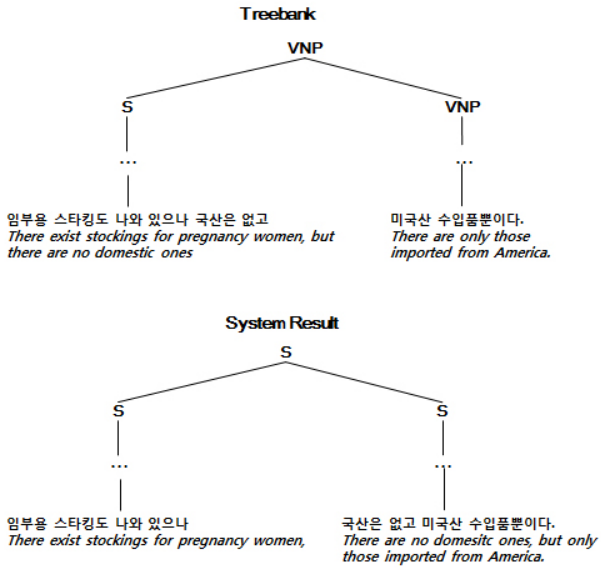

Figure 13: Example of the wrongly-tagged topmost node. Some trees in the treebank have Non-**S** topmost phrase nodes.

Although there exists a performance gain due to the transforming methods, there are still many gaps for improvement. The evaluation results and error analysis results suggests the need to define the phrase tagset of Sejong treebank in more detail. Also, the transforming methods themselves are not perfect yet - we believe still they could be improved more to increase consistency of the resultant treebanks.

We will continuously develop our transforming methods to improve the parsing result. Furthermore, we are planning to investigate methods to determine the appropriate "detailedness" of phrase tag set, so that there are no missing information due to too small number of tags as well as no confusion due to too many tags.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2011-0026718)

References

Dan Bikel. 2004. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models*. Ph.D. thesis, University of Pennsylvania.

Dan Bikel. 2012. Bikel parser. `http://www.cis.upenn.edu/~dbikel/software.html`.

Jinho D. Choi and Martha Palmer. 2011. Statistical dependency parsing in Korean: From corpus generation to automatic parsing. In *The Second Workshop on Statistical Parsing of Morphologically Rich Languages*, pages 1–11.

Key-Sun Choi, Young S. Han, Young G. Han, and Oh W. Kwon. 1994. KAIST tree bank project for Korean: Present and future development. In *Proceedings of the International Workshop on Sharable Natural Language Resources*, pages 7–14.

Key-Sun Choi, Isahara Hitoshi, and Maosong Sun. 2011. Language resource management – word segmentation of written texts – part 2: Word segmentation for Chinese, Japanese and Korean. In *ISO 24614-2*. ISO.

Tagyoung Chung, Matt Post, and Daniel Gildea. 2010. Factors affecting the accuracy of korean parsing. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 49–57, Los Angeles, CA, USA, June. Association for Computational Linguistics.

Hoojung Chung. 2004. *Statistical Korean Dependency Parsing Model based on the Surface Contextual Information*. Ph.D. thesis, Korea University.

Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. 2010. Identification of the question focus: Combining syntactic analysis and ontology-based lookup through the user interaction. In *Proceedings of 7th Language Resources and Evaluation Conference (LREC)*, pages 361–368.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *NAACL '09 Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Chung-Hye Han, Na-Rae Han, Eon-Suk Ko, Heejong Yi, and Martha Palmer. 2002. Penn Korean treebank: Development and evaluation. In *Proceedings of the 16th Pacific Asia Conference on Language, Information and Computation*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 423–430.

Korean Language Institute. 2012. Sejong treebank. `http://www.sejong.or.kr`.

Tetsuya Nasukawa and Jeonghee Yi. 2003. Sentiment analysis: capturing favorability using natural language processing. In *Proceedings of the 2nd international conference on Knowledge capture*, pages 70–77.

Jin Young Oh, Yo-Sub Han, Jungyeul Park, and Jeong-Won Cha. 2011. Predicting phrase-level tags using entropy inspired discriminative models. In *2011 International Conference on Information Science and Applications (ICISA)*, pages 1–5.

Jungyeul Park. 2006. Extraction of tree adjoining grammars from a treebank for Korean. In *Proceedings of the 21st International Conference on computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 73–78.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the COLING-ACL 2006*, pages 433–440.

David A. Smith and Noah A. Smith. 2004. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proceedings of the EMNLP*, pages 49–56.