Automatic Construction of Korean Verbal Type Hierarchy using Treebank

Sanghoun Song and Jae-Woong Choe
Korea University

The lexical information of verbal lexemes, such as verbs and adjectives, plays an important role in syntactic parsing, because the structure of a sentence mainly hinges on the type of verbal lexemes. The question we address in this research is how to acquire the 'argument structure' (henceforth ARG-ST) of verbal lexemes in Korean. It is well known that manual build-up of type hierarchy usually cost too much time and resources, so an alternative method, namely automatic collection of relevant information is much more preferred. This paper proposes a procedure to automatically collect ARG-ST of Korean verbal lexemes from a Korean Treebank. Specifically, the system we develop in this paper first extracts lexical information of ARG-ST of verbal lexemes from a 0.8 million sized Korean Treebank in an unsupervised way, checks the hierarchical relationship among them, and builds up the type hierarchy automatically. The result is written in an HPSG-style annotation, thus making it possible to readily implement the result in an HPSG-based parser for Korean. Finally, the result is evaluated with reference to two Korean dictionaries and also with respect to a manually constructed type hierarchy in Kim et al. (2006).

1. Problem One of the key issues in writing a comprehensive grammar of a natural language in the HPSG style is how to build up type hierarchies on a large scale. In the case of verbal lexemes in Korean, ARG-ST or case frame plays a significant role in the construction of verbal type hierarchies. One way to collect the ARG-ST information of Korean verbals in a comprehensive way would be to consult the dictionary. For example, the *Yonsei Korean Dictionary* lists the following three types of construction for the adjective *elyep-* 'difficult', a typical 'tough' class verb in Korean.

(1) a. *enehak-i* *elyep-ta.*
linguistics-NM difficult-DC
'Linguistics is difficult.'
b. *nay-ka* *kongpwu-ka* *elyep-ta.*
I-NM study-NM difficult-DC
'It is difficult for me to study.'
c. *enehak-ul* *kongpwu-ha-ki-ka* *elyep-ta.*
linguistics-AC study-LV-NOM-NM difficult-DC
'It is difficult to study linguistics.'

The examples in (1) shows that *elyep-* 'difficult' can be divided into several types according to its ARG-ST; <NP(*nom*)>, <NP(*nom*), NP(*nom*)>, and <S(*nom*)>, which correspond to (1a-c) respectively. An alternative way to collect ARG-ST information on a large scale is to make use of some available Treebanks. Compared to the dictionary based approach, the Treebank approach has at least two obvious advantages. The first is that the Treebank approach would provide the frequency for each ARG-ST as well, which would become crucial for building a stochastic parser. Another advantage of the Treebank approach is that we can minimize the inconsistency or some possible errors in the compilation process of the dictionary. For example, it is up to the judgment of the compiler(s) that s/he selected the three constructions given in (1) for *elyep-*; other compiler(s) could have added another to (1), or even excluded one from (1). In fact, a different dictionary, the *Sejong Electronic Dictionary*, lists six different case frames for the same adjective, and it is almost impossible to pinpoint the source of the difference.

2. Data There are two Korean Treebanks currently available, the *Sejong Korean Treebank* (henceforth SKT) and the *Penn Korean Treebank* (henceforth PKT). The major characteristics of the two can be compared as follows: (i) SKT contains approximately eight hundreds of thousands of words consists of various genres, while PKT includes about two hundreds of thousands of words is composed of only military manuals or newspaper articles. (ii) The empty categories are specified in PKT, while there is no empty category in SKT. (iii) Oblique cases can be tagged as

complements in PKT, whereas in SKT they are excluded from being possible candidates for complements. Between the two, we chose SKT for its size and the balance in its composition. Since SKT do not contain empty categories, it should be noted that the result of this study would likewise be more 'surface-oriented'.

An important problem one faces in dealing with the ARG-ST of the Korean language is the difficulty of differentiating arguments from adjuncts. Korean, a pro-drop style language, allows any element of the sentence be omitted, possibly except for the head. So there is no clear-cut method of distinguishing arguments from adjuncts, as abundantly discussed in the literature (Chae 2000). The same problem crops up in SKT as well. Consider the following.

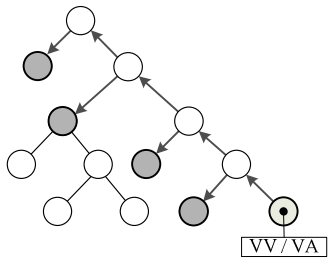
(2) a. *Mia-ka* *yenphil-ul* *chayksang-eyta* *noh-ass-ta.*
Mia-NM pencil-AC desk-LOC put-PAST-DC
'Mia put a pencil on the desk.'
b. *Mia-ka* *yenphil-ul* *seylo-lo* *noh-ass-ta.*
Mia-NM pencil-AC length-DIR put-PAST-DC
'Mia put a pencil lengthwise.'

According to the *Yonsei Korean Dictionary*, the ARG-ST of *noh-* 'put' is <NP(*nom*), NP(*acc*), NP(*loc*)> or <NP(*nom*), NP(*acc*)>. Thus, note that *chayksang-eyta* 'on the desk' in (2a) is a complement of *noh*, whereas *seylo-lo* 'lengthwise' is a mere adjunct according to the standard view. However, both *chayksang-eyta* and *seylo-lo* are tagged as 'NP_AJT' in SKT. As a way to cope with this problem, we took a practical, construction based approach in this study. We first took the ARG-ST in its broadest sense, thus including every possible NPs, VPs, and Ss that are dependent on a verbal. From the resulting set of 'argument structure' candidates, we selected only the significant ones as ARG-ST of the verbal by introducing a simple statistical method. In a sense we adopted a construction based method relying on the frequency of the relevant construction, namely, the 'argument structure'. Note that we do not distinguish arguments from adjuncts in its original sense, nor we distinguish between oblique cases from grammatical cases. This again reflects our surface-oriented and frequency-based approach.

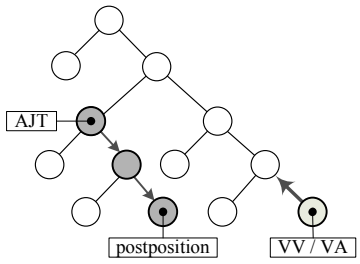
In counting the frequency of ARG-ST, we excluded the verbs or adjectives in the so-called relative clauses. In a relative clause, one of the arguments appears on the surface as the head noun, but there is no way to retrieve its case or function information with respect to the verbal element in the relative clause. Those cases comprise 7.5% of all verbal elements in SKT.

3. Implementation One of the most prominent distributional characteristics of CFG rules in SKT is that the mother node depends upon the right daughter node almost invariably, which directly reflects the fact that Korean belongs to head-final languages. Therefore, the search path to extract arguments from a tree structure will be as in the following pictures.

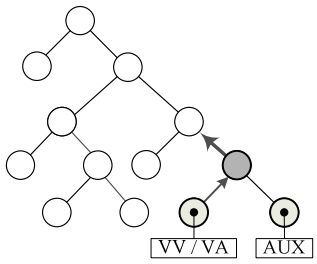
(3)



(4)



(5)



(3) illustrates the main process to acquire arguments with grammatical cases, such as nominatives or accusatives; if a node includes a verb 'VV' or an adjective 'VA', the node is the starting position of searching. By traversing the left node of its ancestor nodes reflexively, relevant cases are collected: if the left node can be the member of ARG-ST of the verbal lexeme, the node becomes an element of candidate set of ARG-ST. Since information about the function, such as 'SBJ' or 'OBJ', are annotated on each node in SKT in most cases, this process can be carried out with consistency.

(4) indicates how the candidate set of ARG-ST takes NPs with oblique cases as its element. If a left node of an ancestor node of verbal lexeme is tagged as 'AJT', the node is the starting point.

Since oblique cases in Korean largely hinge on postpositions attached to NP just as oblique cases in English hinge on prepositions, if the final right daughter node contains a postposition, the final node also becomes an element of candidate set. Oblique cases in Korean are shown in (6), adapted from Sohn (1999:213). On the basis of (6), some heuristic assumptions which substitute a postposition with its representative form are applied as a way to deduce representative types of oblique cases

(6) dat 'to' (*ey*, *eykey*, *hanthay*…), loc 'on, at, in' (*eyta*, *eytaka*…), src 'on, at, in' (*eyse*, *eykeyse*…), abl 'from' (*pwuthe*…), dir 'towards' (*lo*, *ulo*…), inst 'with' (*lo*, *ulo*…), cmt 'with' (*wa, lang*…), con 'and, or' (*mye*, *na*…), comp 'than' (*pota*), eqt 'as, like' (*chelem*…)

(5) is for troublesome cases which are due to peculiarities of Korean verbal system. In complex predicates in Korean (e.g. 'verb + auxiliary'), the ARG-ST of the sentence is determined by the main verb (Sells 1998, Kim and Yang 2007, etc.). For example, in (7) where *mek-* 'eat' combines with *siph-* 'would like to', both *Mia* and *ppang* 'bread' are analyzed as arguments of *mek-*.

(7) *Mia-ka* *ppang-ul* [*mek-ko* *siph-ta*].
Mia-NM bread-AC eat-COMP would like to-DC
'Mia would like to have bread.'

In this case, the starting point of the search path is the parent node of the verbal lexeme, which is marked as a dark circle in (5).

3.1 Algorithms In order to handle the cases presented so far, we have implemented a computer program module, coded in the ANSI C++ programming language. There are two major algorithms to extract the candidate set of ARG-ST from SKT; one is the 'Parse Tree' algorithm pseudo-coded in (8), the other is the 'Traverse' algorithm to treat (3), (4), and (5).

(8)
```
1: parse_tree(n):
2:   n→left = n→right = n→parent = NIL:
3:   if n is not a terminal node:
4:      n→right = pop()
5:      n→left = pop()
6:      if n→left is NIL:
7:         n→left = n→right
8:         n→right = NIL
9:   n→left→parent = n→right→parent = n
10:  push(n)
```

(9)
```
1: traverse(n):
2:   if n is not NIL:
3:      get_argst(n→parent)
4:      traverse(n→left)
5:      traverse(n→right)

1: get_argst(n):
2:   if next(n) is AUX:            …(5)
3:      n = n→parent
4:   while n is not NIL:
5:      get_arg(n→left)            …(3)
6:      get_postposition(n→right)  …(4)
7:       n = n→parent
```

3.2 ARG-ST Sets of ARG-ST of verbal lexemes extracted so far need further process for two reasons. One is that SKT does not discern between oblique NPs as arguments and those as adjuncts. Hence, it is necessary to decide whether an oblique case is qualified for the element of ARG-ST or not. The other is because there is no empty category in SKT; therefore, it is not clear whether a surface ARG-ST is saturated with underlying arguments or not. The previous studies that seek to acquire subcategorization frames from corpora have proposed various solutions to this kind of puzzles. Among them, Sarkar and Zeman (2000), who concentrate on the filtering of adjuncts from observed data, employ some stochastic techniques as a device to distinguish valid ARG-STs from invalid ones. In line with their proposal, in order to obtain ARG-STs on the basis of a single criterion, we also use a statistical device, in particular, *t-score* considering it is quite simple to apply and suffices to our purpose. If a frequency of each ARG-ST of an entry is given, *t-score* will be calculated as (10).

(10) $$t = \frac{(m-x)\sqrt{N}}{s}$$ m : the mean of frequencies, x : each frequency,
N : the number of ARG-STs, s : the standard deviation of frequencies

Then each *t-score* is compared with the cut-off value presented at 25% significance level in the *t-distribution* table. If *t-score* is smaller than the cut-off, that means the ATG-ST is not meaningless; therefore, it is regarded as one of the valid ARG-STs. We tested a couple of cut-off values and

settled with the given one for now as the most appropriate one based on our intuition. It could be an arbitrary decision and obviously needs further research, but the way the cut-off value applies to each verbal lexeme is fixed and consistent.

As an example of the selection process, let us take *elyep-* 'difficult'. It had originally 28 ARG-STs, as given in (11). On the base of *t-score*, however, only four ARG-STs are considered as candidates for building up the type hierarchy.

(11) *elyep*/VA

<NP(*nom*), VP(*nom*)>	86
<NP(*nom*)>	51
<S(*nom*)>	11
<NP(*nom*), VP(*nom*), NP(*dat*)>	10
<NP(*nom*), NP(*dir*)>	7
<NP(*nom*), NP(*dat*)>	5
<NP(*nom*), VP(*nom*), NP(*src*)>	4
<NP(*nom*), NP(*comp*)>	3
...	

(12) *elyep*/VA

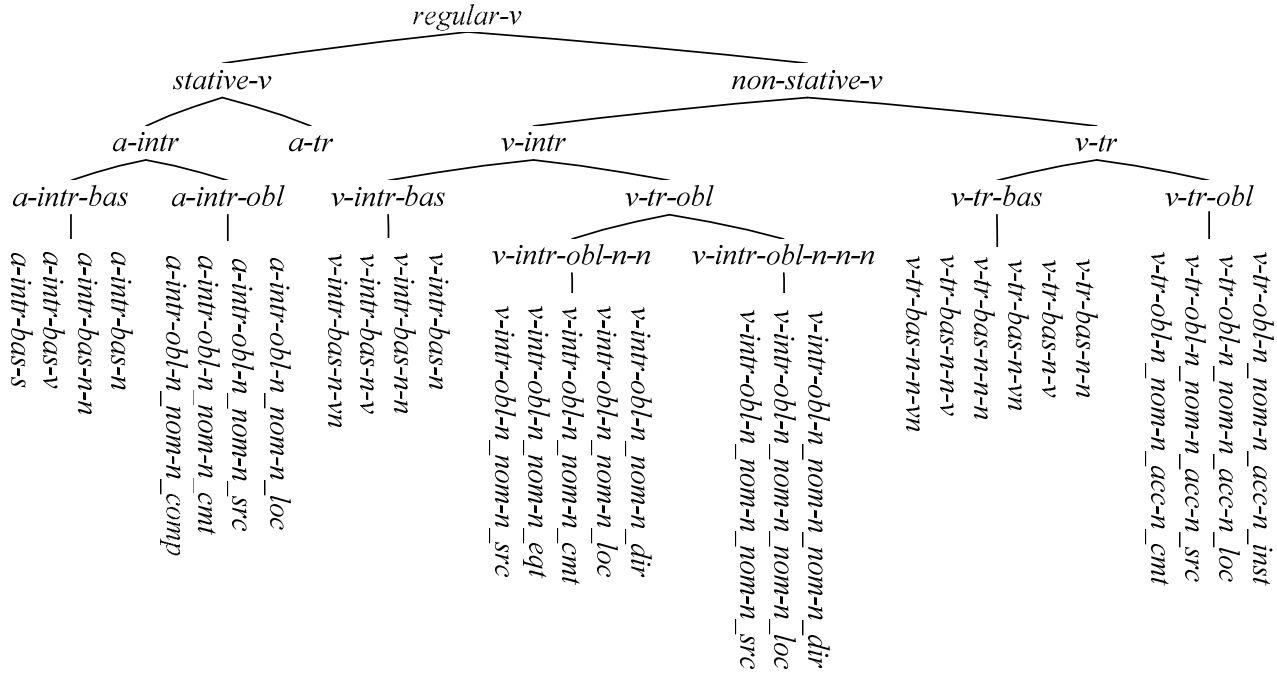
<NP(*nom*), VP(*nom*)>	86
<NP(*nom*)>	51
<S(*nom*)>	11
<NP(*nom*), VP(*nom*), NP(*dat*)>	10

Let us compare (12) with (1). While (1a) and (1c) are included in (12), (1b), namely, <NP(*nom*), NP(*nom*)>, is not. Importantly, the dictionary, though compiled on the basis of a large scale corpus data, does not reflect the most frequent type <NP(*nom*), VP(*nom*)>.

3.3 The Type Hierarchy After the valid set of ARG-STs is acquired, our system draws the type hierarchy of verbal lexemes automatically. There are six depths in our automatic-drawn type hierarchy. The top node of the hierarchy is *regular-v*, which is divided into two subtypes at the second depth; *stative-v* for adjectives and *non-stative-v* for verbs. Types in the third depth are divided according to transitivity, and types in the fourth depth are divided according to whether the ARG-ST of the lexeme can contain oblique cases. If an oblique case can appear in the ARG-ST, *-obl-* is attached to the type name; otherwise, *-bas-* is attached. The fifth depth classifies types into subtypes in conformity with the category of arguments; such as NP, VP, or S. Finally, the last depth is related to the case of arguments, such as *nom*, *acc*, or *dat*.

To begin with, our system generate only three types; *regular-v*, *stative-v*, and *non-stative-v*. With checking all verbal lexemes which appear ten or more times in SKT, the type hierarchy automatically branches out whenever a new type comes out. For example, *noh* 'put' <NP(*nom*), NP(*acc*), NP(*loc*)> which belongs to *v-tr-obl-n_nom-n_acc-n_loc* generates four types hierarchically, if there has not been corresponding types yet; *v-tr*, *v-tr-obl*, *v-tr-obl-n-n-n*, and itself.

The whole type hierarchy that our system built up is sketched out below.



Therefore, the result of this study consists of two parts. One is the whole type hierarchy of verbal lexemes in Korean. The other is the set of lexical information of verbal lexemes, which includes information about frequency. Both of them are written in a type definition language, as stated before.

4. Evaluation The result includes 914 verbal entries (90 adjectives and 824 verbs). Since an adjective or a verb can belong to two or more types, the total number of lexicons is 1,578. Each ARG-ST has its own frequency, as well. On the other hand, there are 45 types in the resulting type hierarchy. In order to evaluate these results, we make use of *precision*, *recall*, and *F-measure* (Manning and Schütze 1999:268).

(13) $$precision = \frac{tp}{tp + fp}$$ (14) $$recall = \frac{tp}{tp + fn}$$ (15) $$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}}$$

As a way to check how well our result fits with other known language resources, we compared our ARG-STs with three available resources separately, the *Yonsei Korean Dictionary (eval1)*, the *Sejong Korean Electronic Dictionary (eval2)*, and also a manually built type hierarchy proposed in Kim et al. (2006) (*eval3*). After selecting at random one hundred entries from our list, we observed the differences. If an ARG-ST of our results is compatible with that of the *Yonsei Korean Dictionary* or the *Sejong Korean Electronic Dictionary*, *tp* (true positives) will increase. If an ARG-ST of our results is missing that of the dictionary, *fn* (false negatives) will increase. In the reversed cases, *fp* (false positives) will increase. Let us call this evaluation process *eval1* and *eval2*, respectively. The following table shows the comparison.

	eval1	*eval2*	*eval3*
precision	80.66%	79.01%	55.56%
recall	79.35%	71.50%	62.50%
$F_{\alpha=0.5}$	80.00%	75.07%	58.82%

The values of *eval1* and *eval2* are fairly high, which are at the similar level reported in Sarkar and Zeman (2000). On the other hand, the values of *eval3* are relatively low. We have yet to sort out where the major source of the difference lies.

5. Conclusion In this paper we have proposed a method of automatically building up a type hierarchy for verbal lexemes based on parsed corpora. We introduced algorithms to collect all the possible ARG-ST and its frequency for a given verbal lexeme, to select appropriate ARG-STs from the candidate set, and finally to build a comprehensive type hierarchy for Korean verbal lexemes. The type hierarchy we have reached in this study matches reasonably well with the information provided in two of the available resources.

We have taken a very practical and surface-oriented approach in selecting ARG-STs that form the basis of the type hierarchy, thus avoiding the difficult task of resolving the argument-adjunct distinction problem in Korean. There is also certain flexibility in the selection process: for example, the significance level we chose was at 25%, a very loose one, but if we choose the significance level at a stricter level, say, 10%, or 5%, the result would be a much more simple type hierarchy. On the other hand, if we choose a yet looser one, the resulting type hierarchy would be a much more fine-grained and complex one.

References

Chae, Hee-Rahk. 2000. Complements vs. Adjuncts (in Korean). *Studies in Modern Grammar* 19:69-85. − Kim, Jong-Bok and Jaehyung Yang. 2007. On the Syntax and Semantics of the Bound Noun Constructions: with a Computational Implementation. Paper presented at *PACLIC 21*, Seoul. − Kim, Jong-Bok et al. 2006. Building up Korean Verbal Hierarchy. Paper presented at *Conference of Korean Lexicology*, Seoul. − Manning, Christopher D. and Hinrich Schütze (1999) *Foundations of Statistical Natural Language Processing*. Cambridge: The MIT Press. − Sarkar, Anoop and Daniel Zeman. 2000. Automatic Extraction of Subcategorization Frames for Czech. Paper presented at *COLING-2000*. − Sells, Peter. 1998. Structural Relationships within Complex Predicates. Paper presented at *The 11th meeting of the International Circle of Korean Linguists*, Hawaii. − Sohn, Ho-Min. 1999. *The Korean Language*. Cambridge: Cambridge University Press.