Automatic Construction of Korean Verbal Type Hierarchy using Treebank

Sanghoun Song and Jae-Woong Choe

Korea University

Proceedings of the HPSG08 Conference

NICT, Keihanna, Japan

Stefan Müller (Editor)

2008

CSLI Publications

http://csli-publications.stanford.edu/

Abstract

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 graphic word 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.

1 Introduction

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 particular, since the lexical information of verbal lexemes, such as verbs and adjectives, takes an important role in syntactic parsing, argument structures (hereafter ARG-STs) hold the key position in describing a grammar within the HPSG framework, so building up type hierarchies on a large scale should begin with collecting relevant information about ARG-ST.

What we are concerned with in this study is how to build up the verbal type hierarchy in a more efficient way. It is well known that type hierarchy built-up manually usually cost too much time and resources; therefore an alternative method, namely automatic compilation of relevant information is much more preferred.

This study aims to introduce a systematic procedure to collect

† We would like to thank to Jong-Bok Kim, Kiyong Lee, and Jieun Jeon for their help throughout this research. We appreciate the comments on an earlier version of this paper from anonymous readers, and also are thankful for the comments from Hans Uszkoreit, Dan Flickinger, Laurie Poulson, and some other members of the audience during the HPSG conference held at Keihanna, Japan. July 28-30, 2008. Of course, all remaining errors are our own responsibilities.

relevant ARG-STs in Korean verbal system, and to construct the Korean verbal type hierarchy. The procedure will be carried out in a fully automatic way. The data compilation will be based on the results extracted from the *Sejong Korean Treebank*.

This paper is constructed as follows. Chapter 2 provides a brief comparison of ways to extract information of ARG-STs, namely, a traditional manual extraction from dictionaries vs. an automatic extraction from large scale language resources adopted in this study. In chapter 3, we introduce the whole process to get relevant ARG-STs from a Korean Treebank and build up Korean verbal type hierarchy in a systematic and automatic way. Chapter 4 discusses the evaluation of the result of this study with reference to two dictionaries and also with reference to a manually constructed verbal type hierarchy. Chapter 5 is the conclusion of this paper.

2 The Treebank Approach

One way to collect the ARG-ST information of Korean verbal items in a comprehensive way would be to consult the dictionary. For example, the *Yonsei Korean Dictionary*[1] lists the following three types of construction for the adjective *elyep-* 'difficult', a typical 'tough' class verb in Korean.[2]

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

---

[1] The verbal category in this dictionary covers 49,552 entries altogether.

[2] The abbreviations of this paper are as follows.
ACC: Accusative case marker, COMP: Complementizer affix, DAT: Dative case marker, DC: Declarative sentence-type marker, DET: Determiner, DIR: Directive case marker, LOC: Locative case marker, LV: Light verb, NOM: Nominative case marker, NMS: Nominalizer suffix, PAST: Past tense marker

is to make use of some available corpora or Treebanks.[3] 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 she or 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*,[4] lists six different case frames for the same adjective, and in general it is not an easy task to pinpoint the source of the difference.

There are two Korean Treebanks currently available; the *Sejong Korean Treebank* (henceforth SKT) which has been sponsored by the Korean government and the *Penn Korean Treebank* (henceforth PKT) which has been researched at the Univ. of Pennsylvania. The major characteristics of the two are as follows: (i) SKT contains approximately eight hundred thousand graphic words consisting of various genres (e.g. novels, academic articles, etc.), while PKT includes about two hundred thousands of graphic words, which is only composed of military manuals or newspaper articles. (ii) The empty categories are specified in PKT, while there is no empty category in SKT. (iii) Finally, 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. However, since SKT does 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 typical pro-drop style language, allows any element of the sentence be omitted, possibly except for the head. It is one of the most controversial and tough issues in Korean Linguistics to distinguish arguments from adjuncts in Korean as is well documented and discussed in the literature (e.g. Chae 2000).

Consider the following.

---

[3] For example, Manning (1993) shows a method to acquire subcategorization frames from unlabelled corpora. Sarkar and Zeman (2000) also make use of machine learning techniques for the identification of subcategorization frames, using the Prague dependency Treebank. They use some statistical measures, including *t-score* that we also take advantage of in this study, in their solution to label dependents of a verb as either arguments or adjuncts.

[4] The version used for this study contains 18,618 verbal items.

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

According to the *Yonsei Korean Dictionary*, the ARG-STs of *noh-* 'put' are <NP(*nom*), NP(*acc*), NP(*loc*)> or <NP(*nom*), NP(*acc*)>. Thus, sentence (2a) corresponds to the first ARG-ST that contains a locative case, while sentence (2b) corresponds to the second one without any oblique complements. That is, *chayksang-eyta* 'on the desk' in (2a) is a complement of *noh*, whereas *seylo-lo* 'lengthwise' in (2b) is a mere adjunct according to the standard view. However, both *chayksang-eyta* and *seylo-lo* are tagged as 'NP_AJT' in SKT.

The same problem, though in a lot less degree, crops up in English as is well known. Let us consider 'put' class verbs in (3) taken from Levin (1993:111). According to Levin's explication, sentence (3b) and (3c) sound deviant because the obligatory arguments are omitted. That is, in this example, 'on the desk' functions as a complement.

(3) a. 'John put the book on the desk.'
b. *'John put on the desk.'
c. *'John put the book.'

This kind of linguistic phenomenon has to be taken into consideration in automatic acquisition of the argument structures from corpora. For example, Manning (1993) raises the need for some methodology to verify whether the prepositional phrase 'on the table' in (4) must be an argument of the verb 'put' or not.

(4) 'John put [$_{NP}$ the cactus] [$_{PP}$ on the table].'

In other words, a systematic approach is required to divide dependents of verbs into arguments or adjuncts, even when obtaining argument information automatically.

As a way to cope with this problem of the argument-adjunct distinction, 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 predicate. From the resulting set of candidates (i.e. dependents), we selected only the significant ones as argument structures of the predicate by introducing a statistical method. In a sense we adopted a construction-based method relying on the frequency of the relevant construction. 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.

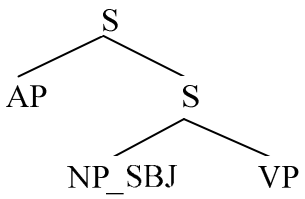
In counting the frequency of ARG-ST, we excluded the verbs or adjectives in the so-called relative clauses in Korean. Relative clauses can raise a troublesome issue in terms of extracting subcategorization frames from corpora, because one of the arguments appears outside of the relative clauses. Unfortunately, there is no way to retrieve its case or functional information with respect to the verbal element in relative clauses. We therefore excluded the verbs or adjectives in relative clauses. Those cases account for approximately 7.5% of all verbal elements in the SKT.

3 Implementation

In this section, we will introduce our basic methodology, step by step, to construct a verbal type hierarchy automatically.

We processed data in Treebanks on the basis of the 'Parse-Tree' algorithm. Data structure of the 'Parse-Tree' algorithm[5] consists of three elements; the mother node (MN), the left daughter node (LDN), and the right daughter node (RDN). Figure (5) represents a typical 'Parse-Tree' structure.

(5)



The first S is the MN of its LDN AP, and its RDN S, while the RDN S, the second S in the tree, is the MN of its LDN NP_SBJ and its RDN VP at the same time. In brief, every node is linked to the head node in a hierarchic binary form.[6]

One of the most prominent distributional characteristics of CFG rules in SKT is that the MN depends upon the RDN almost invariably, which directly reflects the fact that Korean belongs to head-final languages. Therefore, the search paths to extract arguments from a tree structure will be as in the following pictures (6), (9) and (14).

(6) illustrates the main process to acquire arguments with grammatical cases, such as nominatives or accusatives; if a node includes a

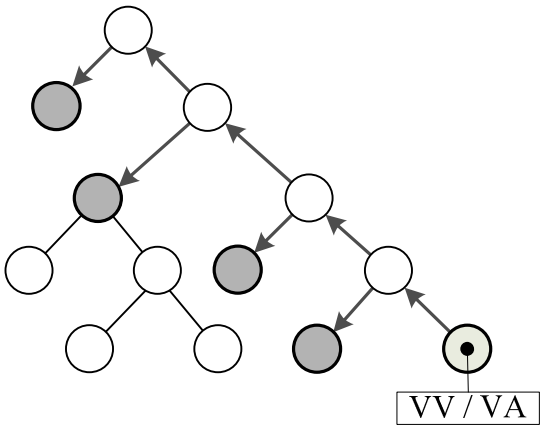
---

[5] Technically speaking, the 'Parse-Tree' algorithm is grounded upon a stack on the principle of 'Last In First Out' (LIFO). The stack has two basic operations; 'push' and 'pop'. The former adds a new node to the top of the stack, and the latter removes and returns the top node on the stack.

[6] SKT adopted a strict binary format for its hierarchical analyses.

verb ‘VV’ or an adjective ‘VA’, the node is the starting position for a search.

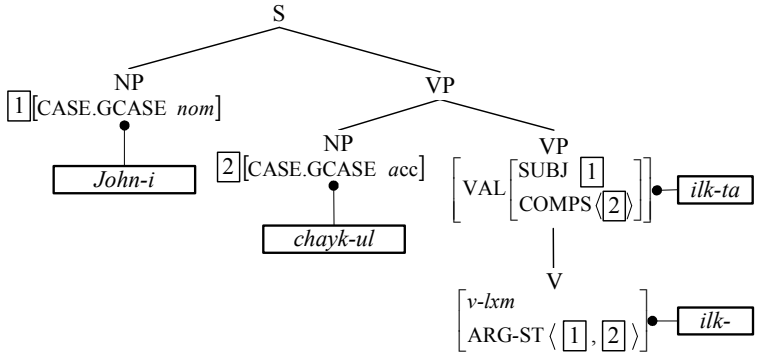
(6)



The algorithm traverses up the tree, checking the left node of its ancestor nodes repeatedly, and collecting relevant cases: if the left node can be a 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’, is annotated on each node in SKT in most cases, this process can be carried out with consistency. For instance, let us take a look at sentence (7) in which a typical transitive verb is used. The corresponding tree derivation will be as (8).

(7) *John-i* *chayk-ul* *ilk-ta*.
John-NOM book-ACC read-DC
‘John reads a book.’

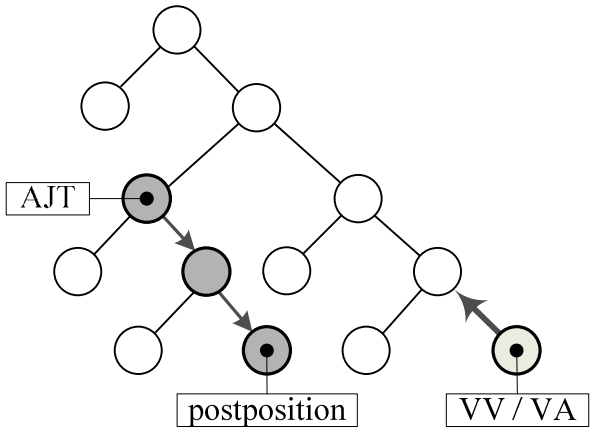
(8)



In (8), VP that contains the main verb *ilk*- ‘read’ will be the starting point. First, *chayk* ‘book’ with an accusative case is taken as a relevant dependent of *ilk*-, and next, ‘John’ in the subject position is also taken. After going through further procedure, <NP(*nom*), NP(*acc*)> is added as an ARG-ST of the verb *ilk*-.

Next, (9) 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 becomes the starting point.

(9)



Since oblique cases in Korean largely hinge on postpositions attached to NP just as oblique cases in English hinge on prepositions, if the final RDN contains a postposition, the final node also becomes an element of candidate set. Oblique cases in Korean determined by postpositions are given in the table below, which is adapted from Sohn (1999:213).

case	**postposition**	**meaning**
dative	*ey, eykey, hanthey, tele…*	'to'
locative	*ey, eykey, hanthey, eyta…*	'on,at,in'
source	*eyse, eykeyse…*	'on,at,in'
ablative	*pwuthe, lopwuthe , sepwuthe…*	'from'
directive	*lo, ulo…*	'towards'
instrumental	*lo, ulo, ulosse…*	'with'
comitive	*wa, kwa, hako, lang…*	'with'
connective	*mye, imye, wa, na…*	'in addtion to, and ,or'
comparative	*pota*	'than'
equative	*chelem, kathi, mankhum…*	'as, like, as much as'

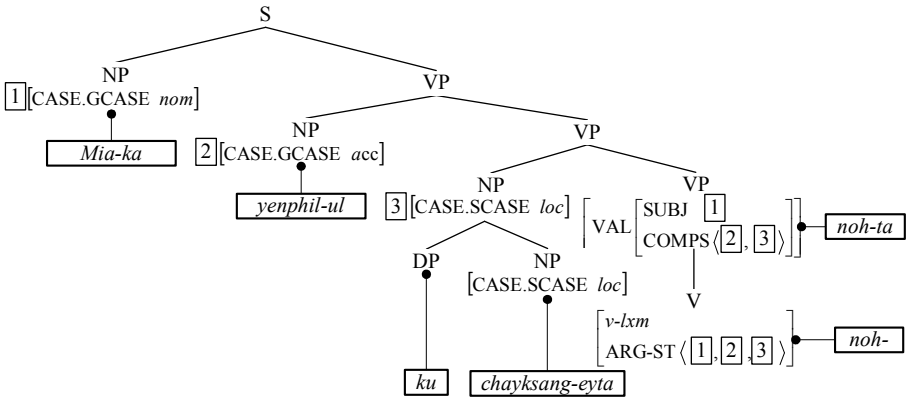
Table 1 : Postpositions in Korean

On the basis of the above, some heuristic assumptions to substitute a postposition with its representative form are taken as a way to deduce representative types of oblique cases. Let us take an example that includes an oblique noun phrase. In (10), *chayksang-eyta* 'on the desk' is coded with a locative case.

(10) *Mia-ka yenphil-ul ku chayksang-**eyta** noh-ta*.
Mia-NOM pencil-ACC DET desk-**LOC** put-DC
'Mia puts a pencil on the desk.'

In this case, it would be more plausible to regard this NP as a complement of the main verb, as was discussed in Section 2, though it is annotated as an adjunct in SKT. (11) stands for the derivation of (10).

(11)

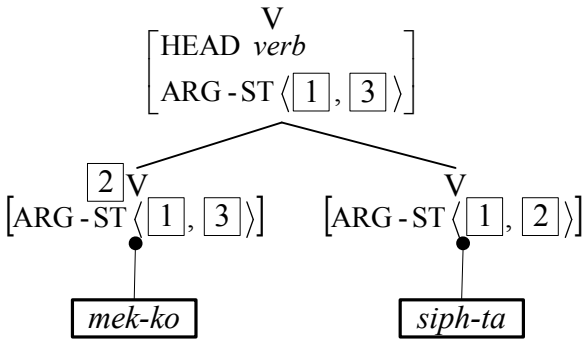


Based on our search path to collect dependents, in the above structure, the NP 3 will be the starting point. And then, the search path goes through its right daughter, finding a postposition such as a locative case marker *-eyta*. As a result of the previous and this procedures, <NP(*nom*), NP(*acc*), NP(*loc*)> will be added as an ARG-ST of the verb *noh-* 'put'. Essentially, the compilation of oblique dependents, in our system, largely depends on the appearance of postposition.

The third search path is for rather troublesome cases, such as complex predicates which consist of a verb plus an auxiliary. In that case, the ARG-ST of the sentence is determined by the main verb. Kim (2004) provides an analysis of Korean auxiliary constructions within the HPSG framework. According to his analysis, since what is responsible for the argument structure in Korean complex predicates is not an auxiliary but the main verb, the mother-category inherits the ARG-ST from the main verb directly. For example, in (12), taken from Kim (2004), where *mek-* 'eat' combines with *siph-* 'would like to', both *John* and *ppang* 'bread' are analyzed as arguments of *mek-*, not the auxiliary *siph-*, as presented in (13).

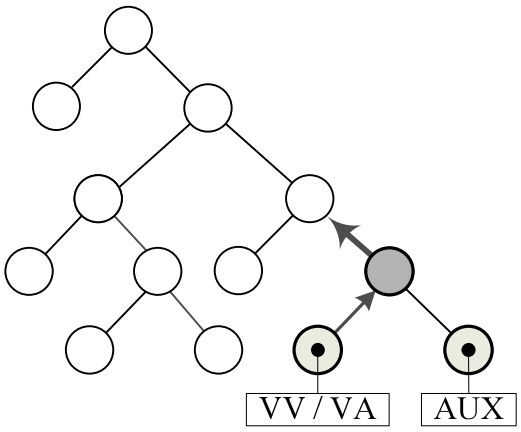
(12) *John-i ppang-ul* [$_{V}$[$_{V}$***mek**-ko*] [$_{V}$***siph**-ta*]].
John-NOM bread-ACC eat-COMP would like to-DC
'John would like to eat bread.'

(13)

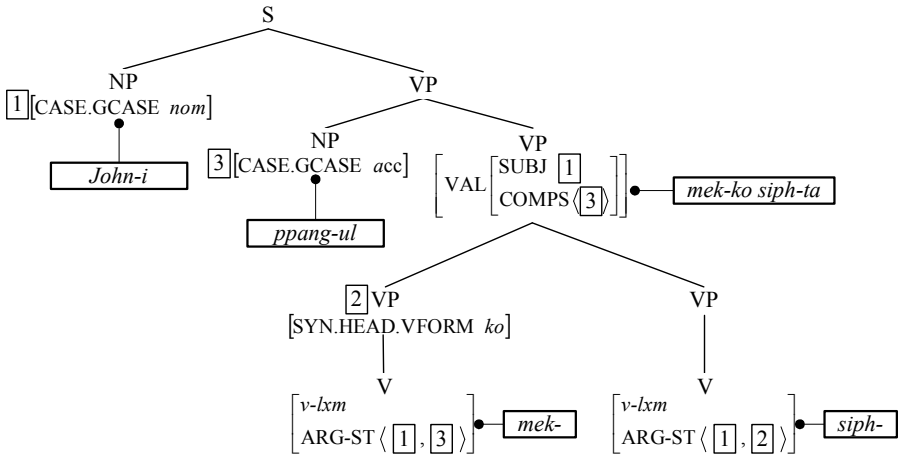


The starting point to collect dependents in complex predicates, therefore, is different from the previous cases. In this case, the starting point of the search path is the parent node of the verbal lexeme, which is marked as a dark circle in (14).

(14)



(15)



In the above diagram, which shows a kind of complex predicate, the starting point turns into the upper node of both the main verb *mek-* 'eat' and the corresponding auxiliary *siph-* 'would like to'. Then, the same procedure as in (6) will be applied so that we can get the pertinent ARG-ST of *mek-* as <NP(*nom*), NP(*acc*)>, which are represented by 1 and 3, respectively, in the above diagram.

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 given in (16), the other is the 'Traverse' algorithm to treat (6), (9), and (14). Let us look into the algorithm of building up the 'Parse-Tree' structure.

(16)
```
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)
```

If there is a new node which is not yet processed (line 1), the left of the node, the right of the node, and the parent of the node are assigned a NULL value (line 2). If the node is not a terminal node (i.e. a non lexical entry) (line 3), the left and right of the node are assigned a value popped from the stack (line 4-5). Since there can be a node without its right, in that case (line 6), this algorithm swaps left with right and assigns a NULL value to the right (line 7-8). The current node naturally becomes the parent node of both its LDN and its RDN (line 9). Finally, this algorithm pushes the node processed so far into the stack in order to link with other nodes (line 10).

(17) and (18) are our 'Traverse' algorithms to collect relevant elements of verbs or adjectives recursively. In (18), line 2 is for the third search path represented in (14), line 5 is for the first search path in (6), and line 6 is for the second search path for oblique cases, shown in (9).

(17)
```
1:  traverse(n):
2:     if n is not NIL:
3:       get_argst(n→parent)
4:       traverse(n→left)
5:       traverse(n→right)
```

(18)
```
1:  get_argst(n):
2:     if next(n) is AUX:              …(14)
3:        n = n→parent
4:     while n is not NIL:
5:        get_arg(n→left)              …(6)
6:        get_postposition(n→right)    …(9)
7:        n = n→parent
```

Based upon these algorithms, we could extract dependents of verbal lexemes from treebanks in an unsupervised way.

3.2 ARG-ST

Sets of ARG-ST of verbal lexemes extracted so far need further process for two reasons. One is that SKT, as stated before, does not discern between oblique NPs as arguments and those as adjuncts. Hence, it is necessary to decide whether an oblique case is qualified to be an element of the ARG-ST or not. The other is that 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 filtering of adjuncts from observed data, employ some stochastic techniques as a way 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* since it is quite simple to apply and suffices to our purpose. If the elements and their frequency value of each ARG-ST of a verbal entry is given, *t-score* will be calculated on the basis of the formula (19), where *m* is short for 'the mean of frequencies,' *x* means 'each frequency,' *N* stands for 'the number of ARG-STs,' and *s* is for 'the standard deviation of frequencies.'

(19)
$$t = \frac{(m - x)\sqrt{N}}{s}$$

Then each *t-score* is compared with the cut-off value presented at 25% significance level in the *t-distribution* table.[7] If *t-score* is smaller than the cut-off point, that means the ARG-ST is not meaningless; therefore, it is regarded as one of the valid ARG-STs.

As an example of the selection process, let us take *elyep-* 'difficult'.

---

[7] 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.

It had originally 28 ARG-STs, as given in (19)[8], before applying *t-score*.

(20) *elyep*/VA

<VP(nom)>	85
<NP(nom)>	49
<S(*nom*)>	11
<VP(*nom*), NP(*dat*)>	10
<NP(*nom*), NP(*dir*)>	6
<NP(*nom*), NP(*dat*)>	5
<NP(*nom*), VP(*nom*), NP(*src*)>	4
...	

After applying *t-score*, however, only four ARG-STs are considered as candidates for building up the type hierarchy, as shown below.

(21) *elyep*/VA

<VP(*nom*)>	85
<NP(*nom*)>	49
<S(*nom*)>	11
<VP(*nom*), NP(*dat*)>	10

Let us compare our result with the description of the same adjective in the *Yonsei Korean Dictionary,* which was previously shown in (1). In (22), we added ARG-ST information to each example in (1) for the purpose of comparison with (21).

(22) a. **<NP(*nom*)>**

enehak-i *elyep-ta.*
linguistics-NOM difficult-DC
'Linguistics is difficult.'

b. **<NP(*nom*), NP(*nom*)>**

nay-ka *kongpwu-ka* *elyep-ta.*
I-NOM study-NOM difficult-DC
'It is difficult for me to study.'

c. **<S(*nom*)>**

enehak-ul *kongpwu-ha-ki-ka* *elyep-ta.*
linguistics-ACC study-LV-NMS-NOM difficult-DC
'It is difficult to study linguistics.'

It turns out that while (22a) and (22c) are included in our result, (22b), <NP(*nom*), NP(*nom*)>, is not. The most frequent type in (20), <VP(*nom*)>,

[8] The numbers on the right side are the frequency value for each item in SKT.

is not given in (22), but perhaps it can be considered as a case of (22c),[9] though the distribution of <S(*nom*)> and <VP(*nom*)> in SKT should not be ignored. <VP(*nom*), NP(*dat*)>, whose frequency value is 10, is not reflected in (22). Perhaps it has something to do with the the difference on the status of 'NP(*dat*)', that is, whether it should be treated as a valid argument or not.

Then the main and clear difference between (21) and (22) would be (22b), which does not appear in (21). In fact, it appeared only once in SKT. It is very interesting to note that the construction given in (22b) is the so-called multiple nominative construction, one of the most hotly and widely debated topics in Korean linguistics, as it is claimed to show one of the major characteristics of the Korean language. Therefore, the significance and implication of the difference regarding (22b) would need further investigation, which we leave for future research.

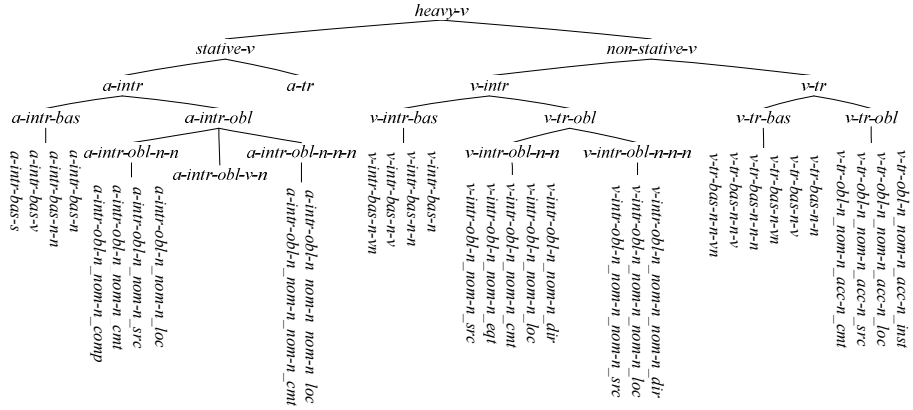
3.3 The Type Hierarchy

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

---

[9] In SKT, the difference between an S and a VP is the presence or absence of the nominative marked NP on the surface. So, the example in (22c), which would treated as a case of <S(*nom*)> in the *Yonsei Korean Dictionary*, is to be considered as <VP(*nom*)> in (21) as the nominalized clause *enehak-ul kongpwu-ha-ki-ka* 'to study linguistics' lacks its internal subject on the surface.

(23)



To begin with, our system generate only three types; *regular-v*, *stative-v*, and *non-stative-v*. By 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*)>, presented in (4a), 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. We also designed our system to be a stringent or shallow one, minimizing unnecessary branches in the hierarchy. For example, the v-*tr-obl-n-n-n* type is deleted after the whole type hierarchy is built up, because the type has no subtypes. That is, after a type hierarchy has been built up once, our system gets rid of types without subtypes from the tentative hierarchy, and minimizes the depth of hierarchy.

Let us now consider *elyep*- 'difficult' mentioned above. As shown before, there are four ARG-STs which fall under *elyep*-; <VP(*nom*)>, <NP(*nom*)>, <S(*nom*)>, and <VP(*nom*), NP(*dat*)>. Since *elyep*- is an adjective, all four belong to *a-intr* type in the above hierarchy (23). Among them, since the last one, <VP(*nom*), NP(*dat*)>, takes an oblique case (i.e. datives) as its argument, it belongs to the *a-intr-obl* type. The others that do not take any kind of oblique cases as their argument come under the *a-intr-bas* type. Table in the below shows the matching between them. Note that if there are no subtypes under a node, the node will be discarded in order to make the hierarchy as shallow as possible. For example, although an ARG-ST <VP(*nom*), NP(*dat*)> seems to belong to the *a-intr-obl-v_nom-n_dat* type, its type is specified as *a-intr-obl-v-n*, because there are no sister type that shares its parent type.

ARG-ST	type	frequency	proportion
<VP(nom)>	*a-intr-bas-v*	85	42.3%
<NP(nom)>,	*a-intr-bas-n*	49	24.4%
<S(nom)>,	*a-intr-bas-s*	11	5.5%
<VP(*nom*), NP(*dat*)>.	*a-intr- obl-v-n*	10	5.0%

Table 2 : Types of elyep- 'difficult'

All in all, the result of this study consists of two parts. One is the whole type hierarchy of verbal lexemes in Korean. There are 50 types in the resulting type hierarchy. The other is the set of lexical information of verbal lexemes, which includes information about frequency. The result of our analysis includes 915 verbal entries (91 adjectives and 824 verbs). Since an adjective or a verb can belong to two or more types, the total number of lexicons is 1,572. Each ARG-ST has its own frequency value. Since the results of our study are written in a type definition language, it would be possible to implement the result in an HPSG-based parser, such as the LKB system.

4 Evaluation

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 type hierarchy, built up manually, proposed in Kim et al. (2006) (*eval3*). In order to evaluate the results of our analysis, we make use of *precision*, *recall*, and *F-measure* (Manning and Schütze 1999:268) as given below.[10]

(24) $$precision = \frac{tp}{tp + fp}$$

(25) $$recall = \frac{tp}{tp + fn}$$

(26) $$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}}$$

[10] According to Manning and Schütze (1999), *precision* is defined as 'a measure of the proportion of selected items that the system got right', *recall* is defined as 'the proportion of the target items that the system selected', and *F-measure* is one of 'the combined measures of *precision* and *recall*'. In the formula (26), *P* is short for *precision, R* means *recall.* And as for α, 'α = 0.5' is normally selected.

The comparison was done as follows; 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 does not appear in 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%

Table 3 : Evaluations

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, according to our random sample comparison, appears to match reasonably well with the information provided in two of the available resources, though a more thorough and in-depth comparison would be necessary.

We have taken a very practical and surface-oriented approach in selecting ARG-STs that form the basis of the type hierarchy, thus obviating 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.

We believe the analysis given in this study brings up some specific and interesting questions and issues for more theoretically oriented linguistics as well as for computational linguistics. Discussion of these and related issues, and their implications, would certainly need further investigation.

References

Chae, Hee-Rahk. 2000. Complements vs. Adjuncts (in Korean). *Studies in Modern Grammar* 19:69-85.

Kim, Jong-Bok and Jaehyung Yang. 2004. A constraint-based approach to Korean auxiliary constructions and its computational implementation. *Language Research* 40(1): 195-226.

Kim, Jong-Bok et al. 2006. Building up Korean Verbal Hierarchy. Paper presented at *Conference of Korean Lexicology*, Seoul.

Levin, Bath. 1993. *English Verb Classes and Alternations: a Preliminary Investigation.* Chicago: University Of Chicago Press.

Manning, Christopher D.. 1993. Automatic Acquisition Of A Large Subcategorization Dictionary From Corpora. Paper presented at *The 31st Annual Meeting of the Association for Computational Linguistics*, Columbus, Ohio.

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*.

Sohn, Ho-Min. 1999. *The Korean Language*. Cambridge: Cambridge University Press.