Towards Fully Lexicalized Dependency Parsing for Korean

Jungyeul Park
UMR 6074 IRISA
Université de Rennes 1
Lannion, France
jungyeul.park@univ-rennes1.fr

Daisuke Kawahara
Graduate School of Informatics
Kyoto University
Kyoto, Japan
{dk,kuro}@i.kyoto-u.ac.jp

Sadao Kurohashi
Dept. of Computer Science
KAIST
Daejeon, Korea
kschoi@kaist.edu

Key-Sun Choi

Abstract

We propose a Korean dependency parsing system that can learn the relationships between Korean words from the Treebank corpus and a large raw corpus. We first refine the training dataset to better represent the relationship using a different POS tagging granularity type. We also introduce lexical information and propose an almost fully lexicalized probabilistic model with case frames automatically extracted from a very large raw corpus. We evaluate and compare systems with and without POS granularity refinement and case frames. The proposed lexicalized method outperforms not only the baseline systems but also a state-of-the-art supervised dependency parser.

1 Introduction

Korean dependency parsing has been studied more in comparison with constituent parsing owing to its relatively free word order in Korean (Chung, 2004; Lee and Lee, 2008; Oh and Cha, 2010). A dependency structure is less restricted by the word order because it does not require that one constituent is followed by another. Statistical parsing trained from an annotated dataset has been widespread. However, while there are manually annotated several Korean Treebank corpora such as the Sejong Treebank corpus (SJTree), only a few works on statistical Korean parsing have been conducted. For constituent parsing, (Sarkar and Han, 2002) used a very early version of the Korean Penn Treebank (KTB) to train lexicalized Tree Adjoining Grammars (TAG). (Chung et al., 2010) used context-free grammars and tree-substitution grammars trained on data from the KTB. Most recently, (Choi et al., 2012) proposed a method to transform the word-based SJTree into an entity-based Korean Treebank corpus to improve the parsing accuracy. For dependency parsing, (Chung, 2004) presented a model for dependency parsing using surface contextual information. (Oh and Cha, 2010) developed a parsing model with cascaded chunking by means of conditional random fields learning. (Choi and Palmer, 2011) used the Korean dependency Treebank converted automatically from the SJTree.

In this paper, we start with an unlexicalized Korean dependency parsing system as a baseline system that can learn the relationship between Korean words from the Treebank corpus. Then, we try to improve the parsing accuracy using internal and external resources. For internal resources, we can refine the training dataset for a better representation of the relationship by means of POS tagging granularity. For external resources, we introduce lexical information and propose a lexicalized probabilistic model with case frames. We automatically extract predicate-argument structures from a large raw corpus outside of the training dataset and collect them as case frames to improve parsing performance.

2 Dependency grammars

Converting phrase-structure grammars from the Treebank corpus into dependency grammars is not a trivial task (Wang, 2003; Gelbukh et al., 2005; Candito et al., 2010). We implement a word-to-word conversion algorithm for the Sejong Treebank corpus. Firstly, we assign an anchor for nonterminal nodes using bottom-up breadth-first search. An anchor is the terminal node where each nonterminal node can have as a lexical head node. We use lexical head rules described in (Park, 2006). It assigns only the lexical head for nonterminal nodes at the moment and finds dependencies

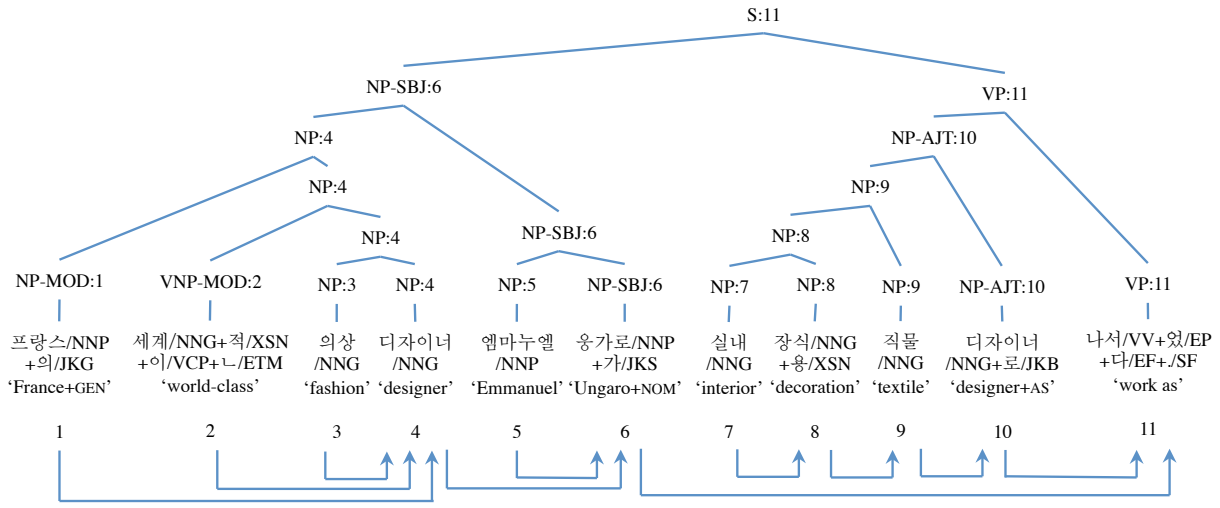

Figure 1: Example of the original SJTree (above) and its dependency representation (below) for the example sentence 'The world-class French fashion designer Emanuel Ungaro worked as an interior textile designer.': The address of terminal nodes (underneath) and the anchor of nonterminal node (on its right) are assigned using lexical head rules. The head of the terminal node 1 is the node 4, which is the anchor of the parent of the parent node (NP:4). The head of the terminal node 4 is the node 6 where the anchor of its ancestor node is changed from itself (NP-SBJ:6). The head of the terminal node 11 is itself where the anchor of the root node and itself are same (S:11).

would be in the next step. Lexical head rules give priorities to the rightmost child node, which inherits in general the same phrase tag. On the other hand, in the case of VP VP for the construction of the main predicate and the auxiliary verb, the leftmost child node is exceptionally assigned as an anchor.

Then, we can find dependency relations between terminal nodes using the anchor information. The head is the anchor of the parent of the parent node of the current node (For example, terminal nodes 1, 2, 3, 5 and 7 in Figure 1). If the anchor of the parent of the parent node is the current node and if the parent of the parent node does not have the right sibling, the head is itself (the anchor of the root node and itself are same) (Terminal node 11), or the head is the anchor of its ancestor node where the anchor is changed from itself to other node (Terminal nodes 4, 6, 8 and 10). If the anchor of the parent of the parent node is the current node and if the parent of the parent node has another right sibling, the head is the anchor of the right sibling. The last condition is for the case of an auxiliary verb construction where the leftmost child node is assigned as an anchor. Assigning the lexical anchor and finding dependencies at the separated step enables arguments for the verb to be correctly dependent on the main verb and the main verb to be dependent on the auxiliary verb in the ambiguous annotation scheme in the SJTree.[1] Figure 1 shows the original SJTree phrase structure and its corresponding converted representation in dependency grammars.

3 Parsing Model

Our parsing model gives a probability to each possible dependency tree T for a sentence $S = e_1, e_2, ..., e_n$, where e_i is a Korean word. The model finally selects the dependency tree T^* that maximizes $P(T|S)$ as follows:

$$T^* = \arg\max_T P(T|S). \quad (1)$$

[1] (Oh and Cha, 2010; Choi and Palmer, 2011) also introduced an conversion algorithm of dependency grammars for the SJTree. (Choi and Palmer, 2011) proposed head percolation rules for the SJTREE. However, we found some errors such as S related rules, where it gives lower priority to S than VP. It would fail to assign a head node correctly for S → VP S. Moreover, they did not consider auxiliary verb constructions annotated as VP in the SJTREE. According to their head rules, arguments for the main verb are dependent on the auxiliary verb instead of the main verb because of the annotation of the corpus (in general, VP → VP VP where the former VP in RHS is for the main verb and the latter VP is for the auxiliary verb). (Oh and Cha, 2010) corrected such ambiguities as a post-processing step (personal communication, August 2012).

We use the CKY algorithm to decode the dependency trees by employing bottom-up parsing and dynamic programming. $P(T|S)$ is defined as the product of probabilities as follows:

$$P(T|S) = \prod_{E_{pa} \in T} P(E_{pa}, dist|e_h), \quad (2)$$

where E_{pa} represents a clause dominated by a predicate or a genitive nominal phrase, e_h represents the head Korean word of E_{pa}, and $dist$ is the distance between E_{pa} and e_h. Instead of specifying the actual distance, it is classified into three bins: 1, 2 – 5, and 6 –. If the dependent Korean word appears right next to the head, the distance is 1. If it appears between 2 and 5, the distance is 2. If it appears past 6, the distance is 6. $P(T|S)$ is calculated in the similar way as (Kawahara and Kurohashi, 2006a). We describe the outline of this model below. Each probability in equation (2) is decomposed into two ways according to the type of E_{pa}. If E_{pa} is a clause dominated by a predicate, it is decomposed into a predicate-argument structure (content part) PA_m and a function part f_m. e_h is also decomposed into a content part c_h and a function part f_h.

$$\begin{aligned} P(E_{pa}, dist|e_h) &= P(PA_m, f_m, dist|c_h, f_h) \\ &= P(PA_m|f_m, dist, c_h, f_h) \times P(f_m, dist|c_h, f_h) \\ &\approx P(PA_m|f_m, c_h) \times P(f_m, dist|f_h) \end{aligned} \quad (3)$$

The first term in the last equation represents a fully-lexicalized generative probability of the predicate-argument structure. This probability is calculated based on automatically compiled case frames in the same way as (Kawahara and Kurohashi, 2006a). The second term of the last equation is a generative probability of function morphemes, in which f_m and f_h are defined as the POS patterns of morphemes for the predicate of E_{pa} and the head Korean word e_h, respectively. This probability is estimated from training data using maximum likelihood estimation. If E_{pa} is a genitive nominal phrase, it consists of a Korean word that is decomposed into c_m and f_m. Its probability is defined similarly as follows:

$$\begin{aligned} P(E_{pa}, dist|e_h) &= P(c_m, f_m, dist|c_h, f_h) \\ &= P(c_m|f_m, dist, c_h, f_h) \times P(f_m, dist|c_h, f_h) \\ &\approx P(c_m|c_h) \times P(f_m, dist|f_h). \end{aligned} \quad (4)$$

The first term in the last equation represents a fully-lexicalized generative probability of the genitive nominal phrase. This probability is calculated from the constructed database of N_1 *ui* N_2 (N_2 *of* N_1) structures. The second term is the same as the second term in equation (3). In our experiments, we use an unlexicalized parsing model as a baseline. This unlexicalized model regards the above lexicalized probabilities as uniform and actually calculates the product of generative probabilities of function morphemes, $P(f_m, dist|f_h)$.

4 POS Sequence Granularity

Given that Korean is an agglutinative language, a combination of Korean words is very productive and exponential. Actually, a larger dataset would not alleviate this issue. The number of POS patterns would not converge even with a corpus of 10 million words in the Sejong morphologically analyzed corpus. The wide range of POS patterns in words is mainly due to the fine-grained morphological analysis results, where they show all possible segmentations divided into lexical and functional morphemes. For example, most Korean language resources to represent Korean morphological analyses including the SJTree would analyze the word *kimkyosunim* ('Professor Kim+HON') as *kim*/NNP + *kyosu*/NNG + *nim*/XSN ('Kim + professor + HON'). Instead of keeping the fine-grained morphological analysis results, we simplify POS sequences as much as possible using the linguistically motivated method. It would be helpful if we can refine the dataset for a better representation of the relationship. We introduce four level POS granularity: PUNC, MERG, CONT and FUNC.

PUNC: Punctuation marks (denoted as SF for periods, question marks and exclamation points, SP for commas and SE for ellipsis) and non-punctuation marks (for example, a period in the number is equally denoted as SF such as 3/SN + ./SF + 14/SN) are distinguished. Recurrent punctuation marks such as .../SE + .../SE in the word are also merged into a single symbol.

MERG: Special characters such as mathematical characters denoted as SW are merged into an entity with adjacent morphemes. Other non-Korean characters such as SL (Roman letters), SH (Chinese char-

acters) and SN (cardinal numbers) are either merged into an entity with adjacent morphemes or are considered as nouns when they appear alone. Secondly, all suffixes are merged with adjacent morphemes. Functional morpheme-related refining rules are described as follows with the number of occurrences in the SJTree.

The nominal prefix (XPN) and suffix (XSN) with adjacent morphemes are merged into the POS of the corresponding morphemes (17,955 cases). The noun derivational suffix (ETN) with precedent morphemes is merged into the noun (5,186 cases). The non-autonomous lexical root (XR) is merged into the following POS (5,322 cases). The verb and adjective derivational suffix (XSV and XSA) with precedent morphemes are merged into the verb and adjective (20,178 and 9,096 cases, respectively). The adjective and the adverbial derivation *gey*/EC are merged into the adverb (2,643 cases). Refinement rules are applied recursively to all POS tags until there are no rules to apply. For example, *soljk*/XR + *ha*/XSA + *gey*/EC is applied both according to the XR rule and the adverbial derivation rule to become *soljikhagey*/MAG ('frankly').

CONT: All content morphemes in the word are merged together. For example, the sequence of the different type of nouns in a word is merged as a single noun with the priority of proper noun (NNP) > common noun (NNG) > dependent noun (NNB). For example, the sequence of NNP and NNG such as *masan*/NNP + *yek*/NNG ('Masan station'), is merged into an NNP. The sequence of the different type of verbs in a word is also merged as a single verb. The difference between MERG and CONT is the nature of merged morphemes. MERG concerns about merging functional morphemes and CONT about merging lexical morphemes.

FUNC: All functional morphemes are merged together. For example, *eoss*/EP (PAST) + *da*/EF ('DECL') is merged into a single verbal ending *eossda*/EF.

5 Exploiting Lexical Information

This section aims at exploiting lexical information and proposes a lexicalized probabilistic model with case frames aggregated from predicate-argument structures and the database of *N* of *N* structures to improve the parsing system.

5.1 Constructing case frames

It is difficult to make wide-coverage predicate-argument structures manually. Therefore, it is necessary to compile them automatically from a large corpus for our purpose. We introduce two methods using POS patterns and parsed corpora to extract case frames automatically from a raw corpus. We then apply clustering to the extracted predicate-argument structures to produce case frames.

Firstly, we use POS patterns to select predicate-argument structures after automatically assigning POS tags to a raw corpus. The key criteria for determining the predicate-argument structures are the appearance of the final or conjunctive verbal endings (denoted as EC and EF, respectively). Using functional morphemes, we are able to detect the end of predicate-argument structures in the sentence. In Figure 1, we can find two case markers agglutinated to NPs for the predicate *naseo+ss+da*: *-ga* and *-ro* for nominative and adverbial case markers (JKS and JKB). Therefore, we can select the predicate-argument structure composed of *ungaro+ga* ('Ungaro+NOM') and *designer+ro* ('designer+AS') as arguments for the verb *naseo* ('work as'). Our algorithm for selecting predicate-argument structures using POS patterns is described below. All arguments with case markers except JKG (genitive) and JC (connective postpositions) are extracted as a predicate-argument structure. JX (auxiliary postpositions) are not extracted because they can be interpreted either nominative or accusative and it becomes ambiguous.

```
var pa
while w_i in the sentence do
    if w_i ends with case markers then
        pa += w_i;
    else if w_i contains final or conjunctive verbal
    endings && pa is not NULL then
        print and initialize pa;
    else if w_i contains other verbal endings then
        initialize pa;
    else
        do nothing;
    end
end
```

Secondly, to use parsed corpora, we employ the method proposed in (Kawahara and Kurohashi,

2006b) and re-implement it to extract case frames. A large corpus is automatically parsed and case frames are constructed from modifier-head examples in the resulting parsed corpus. Then we extract dependency lists depending on their head as follows. Dependency lists consist of *modifier*$_1$ ... *modifier*$_n$ *head* where $n >= 1$. Then, we select dependency lists only if the head is a predicate such as *unggaroga_6 dijaineoro_10 naseoeossda_11*.

Thereafter, predicate-argument structures are clustered to merge similar ones, as described in (Kawahara, 2005). We distinguish predicate-argument structures by the predicate and its closest case instance to the predicate as described in Figure 2[2]: In order to merge predicate-argument structures we introduce similarities between two structures calculated by the production of the similarities between arguments and the ratio of common instances.

$$Similarty_{case\_frames} = sim_{cf} \cdot ratio_{ci} \quad (5)$$

We use the semantic hierarchy of nouns from Korean CoreNet (Choi, 2003) for the similarities between two instances of arguments. CoreNet is composed of 2,937 concepts represented by *kortermnum* (k_{num}). A cipher of k_{num} tells a hierarchy depth. For instance, COUNTRY (k_{num}: 11125) has ORGANIZATION (1112) as a parent concept (hypernym). *hanguk* (11125, 'Korea) and *namhan* (11125, 'South Korea) share COUNTRY (11125) as a concept. Therefore, similarity between two instances is obtained as follows. $common$ is the shared length of k_{num} for i_1 and i_2[3]:

$$sim_{inst} = \frac{len_{knum}(common * 2)}{len_{knum}(i_1) + len_{knum}(i_2)} \quad (6)$$

Then, we calculate similarities between arguments of the same case marker in two predicate-argument structure as follows:

$$sim_{arg} = \frac{\sum_{x=1}\sum_{y=1} sim_{inst} \cdot \sqrt{|e_x||e_y|}}{\sum_{x=1}\sum_{y=1} \sqrt{|e_x||e_y|}} \quad (7)$$

where e_x and e_y are the number of the occurrences of the instance example e of the same case maker. The ratio of common instances is calculated as follows:

$$ratio_{ci} = \frac{\sum_{i=1} \sqrt{|e_x|}}{\sum_{j=1} \sqrt{|e_y|}} \quad (8)$$

where i is the number of the occurrences of the instance examples of the same case marker and j is the number of the occurrences of the instance examples of the all case marker.

5.2 Constructing the database of N_1 *ui* N_2 structures

We also integrate lexical information on Korean noun phrases of the form N_1 *ui* N_2, which roughly corresponds to N_2 *of* N_1 in English. Even though Korean genitive marker *ui* does not have a broad usage as much as *no* in Japanese as described in (Kurohashi and Sakai, 1999), it sometime does not modify the immediate constituent such as *Kyungjiui meylonhyang binwuleul* ('Melon-flavored soap of Kyungji') where *Kyungjiui* modifies *binwuleul* instead of *meylonhyang*. The N_1 *ui* N_2 structure is very useful to recognize the meaning of natural language text can improve head-modifier relationships between genitive nouns.

6 Experiment and Results

6.1 Parsing results

We use the Sejong Treebank corpus (SJTree) in our experiment.[4] We use standard dataset split for training, development and testing. We report here final evaluation results on the baseline unlexicalized parsing and different POS granularities. We crawl news articles published in 2007 from the website of *Chosun Ilbo*[5] (literally, 'Korea Daily News'), which is one of the major newspapers in Korea to integrate lexical information. We collect 212,401 pages and extract Korean sentences. We acquire a raw corpus with over three million sentences. Then, we use the Espresso POS Tagger and Dependency Parser for Korean to assign POS and parse sentences to extract POS patterned and parsed case frames.[6] We extract the database of

[2] {*inbu*$_3$} ('worker') means that the instance *inbu* has 3 occurrences.

[3] $common = 0$ if either i_1 or i_2 is not included in CoreNet.

[4] Differently from other Korean Treebank corpora, the SJTree contains non-sentences such as noun phrases. We select only complete sentences. We also remove erroneous sentences in the SJTree using heuristics such as non-defined POS tags and not-well-formed morpheme and POS tag pairs.

[5] http://www.chosun.com

[6] http://air.changwon.ac.kr/research/software

CF$_1$:	{*inbu*$_3$}:*i*	{*cha*$_2$,*teuleok*$_1$}:*ey*	{*gabang*$_5$}:*eul*	*silneunda.*
	{worker}:NOM	{car,truck}:LOC	{bag}:ACC	load
CF$_2$:		{*teuleok*$_2$}:*ey*	{*jim*$_3$}:*eul*	*silneunda.*
		{truck}:LOC	{baggage}:ACC	load

Figure 2: Predicate-argument structures distinguished by the predicate and its closest case instance

		UAS
Baseline system		71.735%
POS granularity	PUNC	73.714%
	MERG	76.993%
	CONT	81.515%
	FUNC	82.702%

Table 1: Evaluation results on unlexicalized parsing

		UAS
Lexical information	CF-parsed + NoN	86.037%
	CF-pos + NoN	86.433%

Table 2: Evaluation results on lexicalized parsing with FUNC

N_1 *ui* N_2 structures from the same corpus. During building case frame structures, we ignore JX postpositions (mostly for topic markers) which can be interpreted as either NOM or ACC. Instead, we explicitly specify this ambiguity in the input to let the parser consider both cases to select the correct one. For case frame structures extracted without the subject, we intentionally insert the dummy subject to represent the complete construction of the sentence without any missing constituents.

6.2 Discussion

The basic parsing model is directly based on the POS patterns of words. If some sentences have POS patterns that are not seen in the training dataset, our baseline system cannot handle them. By introducing POS sequence granularity we can increase recall and eventually it makes the dataset more parsable with less untrained POS sequences. Integrating lexical information is prominent. We can increase precision and it can fix many predicate-argument dependency errors in unlexicalized parsing. Results with case frames extracted from the automatically parsed corpus are slightly lower than results with POS patterned case frames because the nature of the corpus. The automatically parsed corpus contains inevitably much more errors than the POS tagged corpus. Moreover, the simpler method using POS patterns can guarantee less errors contained case frames. Filtering out erroneously parsed sentences and building case frame structures only using reliable sentences would yield better results.

Only small numbers of research projects about statistical parsing have been conducted using the same Treebank corpus. (Oh and Cha, 2010; Choi and Palmer, 2011) used the early version of the Sejong Treebank and obtained up to 86.01% F_1 score and 85.47% UAS, respectively. (Choi et al., 2012) obtained 78.74% F_1 score for phrase structure parsing. Our current results outperform previous work. We also test MaltParser[7] on the same dataset and we obtain 85.41% for UAS. It still shows the better performance of our proposed method. The advantage of our proposed system is the capability of adding lexicalized information from external corpora.

7 Conclusion

In this paper, we improved Korean dependency parsing accuracy using various factors, including POS granularity changes and lexical information. We refined the training dataset for a better representation of the relationship between words. We also introduced the use of lexical information. The accuracy was improved and it shows promising factors. The lexical knowledge extracted from a much bigger corpus would be interesting to pursue when seeking further improvement opportunities pertaining to the deep processing of Korean sentences.

Acknowledgments

This work was supported by the Industrial Technology International Cooperation Program (FT-1102, Creating Knowledge out of Interlinked Data) of MKE/KIAT, and the IT R&D program of MSIP/KEIT (10044494, WiseKB: Big data based self-evolving knowledge base and reasoning platform).

[7] http://www.maltparser.org

References

Marie Candito, Benoît Crabbé, and Pascal Denis. 2010. Statistical French Dependency Parsing: Treebank Conversion and First Results. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Language Resources Association (ELRA).

Jinho D. Choi and Martha Palmer. 2011. Statistical Dependency Parsing in Korean: From Corpus Generation To Automatic Parsing. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages*, pages 1–11, Dublin, Ireland, October. Association for Computational Linguistics.

DongHyun Choi, Jungyeul Park, and Key-Sun Choi. 2012. Korean Treebank Transformation for Parser Training. In *Proceedings of the ACL 2012 Joint Workshop on Statistical Parsing and Semantic Processing of Morphologically Rich Languages*, pages 78–88, Jeju, Republic of Korea, July 12. Association for Computational Linguistics.

Key-Sun Choi. 2003. CoreNet: Chinese-Japanese-Korean WordNet with Shared Semantic Hierarchy. In *Proceedings of Natural Language Processing and Knowledge Engineering*, pages 767–770, 26-29 October, 2003.

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.

Alexander Gelbukh, Sulema Torres, and Hiram Calvo. 2005. Transforming a Constituency Treebank into a Dependency Treebank. *Procesamiento del Lenguaje Natural*, 35:145–152.

Daisuke Kawahara and Sadao Kurohashi. 2006a. A Fully-Lexicalized Probabilistic Model for Japanese Syntactic and Case Structure Analysis. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 176–183, New York City, USA, June. Association for Computational Linguistics.

Daisuke Kawahara and Sadao Kurohashi. 2006b. Case Frame Compilation from the Web using High-Performance Computing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC2006)*, pages 1344–1347.

Daisuke Kawahara. 2005. *Automatic Construction of Japanese Case Frames for Natural Langage Understanding*. Ph.D. thesis, Kyoto University, July.

Sadao Kurohashi and Yasuyuki Sakai. 1999. Semantic Analysis of Japanese Noun Phrases - A New Approach to Dictionary-Based Understanding. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 481–488, College Park, Maryland, USA, June. Association for Computational Linguistics.

Yong-Hun Lee and Jong-Hyeok Lee. 2008. Korean Parsing using Machine Learning Techniques. In *Proceedings of the Korea Computer Congress (KCC) 2008*, pages 285–288, Jeju, Korea.

Jin-Young Oh and Jeong-Won Cha. 2010. High Speed Korean Dependency Analysis Using Cascaded Chunking. *Simulation Journal*, 19(1):103–111.

Jungyeul Park. 2006. *Extraction automatique d'une grammaire d'arbres adjoints à partir d'un corpus arboré pour le coréen*. Ph.D. thesis, Université Paris 7 - Denis Diderot, mai.

Anoop Sarkar and Chung-Hye Han. 2002. Statistical Morphological Tagging and Parsing of Korean with an LTAG Grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ 6)*, pages 48–56, Venice, Italy.

Wen Wang. 2003. *Statistical Parsing and Language Modeling based on Constraint Dependency Grammar*. Ph.D. thesis, Purdue University, December.