Statistical Dependency Parsing in Korean: From Corpus Generation To Automatic Parsing

Jinho D. Choi
Department of Computer Science
University of Colorado at Boulder
choijd@colorado.edu

Martha Palmer
Department of Linguistics
University of Colorado at Boulder
mpalmer@colorado.edu

Abstract

This paper gives two contributions to dependency parsing in Korean. First, we build a Korean dependency Treebank from an existing constituent Treebank. For a morphologically rich language like Korean, dependency parsing shows some advantages over constituent parsing. Since there is not much training data available, we automatically generate dependency trees by applying head-percolation rules and heuristics to the constituent trees. Second, we show how to extract useful features for dependency parsing from rich morphology in Korean. Once we build the dependency Treebank, any statistical parsing approach can be applied. The challenging part is how to extract features from tokens consisting of multiple morphemes. We suggest a way of selecting important morphemes and use only these as features to avoid sparsity. Our parsing approach is evaluated on three different genres using both gold-standard and automatic morphological analysis. We also test the impact of fine vs. coarse-grained morphologies on dependency parsing. With automatic morphological analysis, we achieve labeled attachment scores of 80%$^+$. To the best of our knowledge, this is the first time that Korean dependency parsing has been evaluated on labeled edges with such a large variety of data.

1 Introduction

Statistical parsing has recently been popular in the NLP community; most state-of-the-art parsers take various statistical approaches to achieve their performance (Johnson and Ural, 2010; Nivre and McDonald, 2008). The biggest advantage of statistical parsing over rule-based parsing is that it can automatically adapt to new domains, genres, or languages as long as it is provided with enough and proper training data. On the other hand, this can also be the biggest drawback for statistical parsing because annotating such training data is manually intensive work that may be costly and time consuming.

For a morphologically rich language like Korean, dependency parsing shows some advantages over constituent parsing. Unlike phrase structure that is somewhat restricted by word-order (e.g., an object needs to be followed by a verb), dependency structure does not enforce such restrictions (e.g., an object is a dependent of a verb in any position), which makes it more suitable for representing flexible word-order languages. Korean is known to be a flexible word-order language in that although it generally follows the SOV (subject-object-verb) construction, it still accepts sentences following different orders. This is because subjects and objects are usually attached to case particles, so locating them in different positions does not create too much ambiguity. Furthermore, these morphemes (case particles along with many others) often give important clues about dependency relations to their heads, which become very helpful for dependency parsing.

To perform statistical dependency parsing, we need sufficiently large training data. There is not much training data available for dependency structure in Korean. However, there is a Treebank, called the Sejong Treebank[1], containing a large number of constituent trees in Korean (about 60K sentences),

[1] http://www.sejong.or.kr/eindex.php

Proceedings of the 2nd Workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL 2011), pages 1–11,
Dublin, Ireland, October 6, 2011.

formated similarly to the Penn Treebank (Marcus et al., 1993). The Penn Treebank style constituent trees have been reliably converted to dependency trees using head-percolation rules and heuristics (Marneffe et al., 2006; Johansson and Nugues, 2007). By applying a similar conversion strategy to the Sejong Treebank, we can achieve a large set of training data for Korean dependency parsing.

Once we generate the dependency Treebank, any statistical dependency parsing approach can be applied (McDonald et al., 2005; Nivre, 2008). The challenging part is how to extract features from tokens consisting of multiple morphemes. For example, POS tags are typical features for dependency parsing under an assumption that each token consists of a single POS tag. This assumption is only partially true in Korean; a token can consist of a sequence of morphemes with different POS tags.[2]

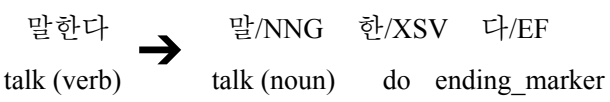


Figure 1: Morphological analysis of a verb *talk* in Korean. POS tags are described in Table 1.

In Figure 1, a verb *talk* in Korean consists of three morphemes, *talk* as a noun, *do* as a verb-derivational suffix, and a final ending marker, such that although it appears to be a single token, it is really a sequence of three individual morphemes where each morpheme has its own POS tag. It is not clear which combination of these morphemes yields the best representation of the token for dependency parsing. Moreover, deriving joined features from multiple tokens (e.g., a joined feature of POS tags between two tokens) can be problematic; considering all combinations of morphemes within multiple tokens can be cumbersome and generate very sparse features.

Obviously, having a good morphological analysis is very important for parsing. There are many automatic morphological analyzers available in Korean (Kang and Woo, 2001; Shim and Yang, 2002). Some of them use different kinds of morphologies better suited for their purposes. It is useful to have a fine-grained morphology; however, a more fine-grained morphology does not necessarily mean a better morphology for parsing. For instance, breaking a token into too many morphemes may cause a loss in the overall semantics of the token. Thus, it is worth comparing outputs from different morphological analyzers and seeing how much impact each morphology has on dependency parsing.

In this paper, we present head-percolation rules and heuristics to convert constituent trees in the Sejong Treebank to dependency trees. We then suggest a way of selecting important morphemes for dependency parsing. To get automatically generated morphemes, we use two existing morphological analyzers. All parsing models are built by a transition-based parsing algorithm. We evaluate our models on test sets in three different genres. Each test set is evaluated by using both gold-standard and automatic morphological analysis. We also compare the impact of fine-grained vs. coarse-grained morphologies on dependency parsing. To the best of our knowledge, this is the first time that Korean dependency parsing has been evaluated on labeled edges with such a large variety of data.

2 Related work

Marneffe et al. (2006) introduced a system for extracting typed dependencies from the Penn Treebank style constituent trees, known as the Stanford dependencies. Johansson and Nugues (2007) presented the LTH constituent-to-dependency converter that had been used to prepare English data for the CoNLL'08-09 shared tasks (Surdeanu et al., 2008; Hajič et al., 2009). Choi and Palmer (2010) later enhanced Johansson and Nugues' approach to handle new updates in the latest Penn Treebank format.

Besides the Sejong Treebank, there are few other Korean Treebanks available. The Penn Korean Treebank (Han et al., 2002) contains the Penn Treebank style constituent trees for newswire and military corpora (about 15K sentences combined). There is also the KAIST tree-annotated corpus (Lee, 1998) containing constituent trees annotated by different bracketing guidelines from the Penn Korean Treebank (about 30K sentences). We chose the Sejong Treebank because it is the largest and most recent Korean Treebank including several function tags that are useful for the dependency conversion.

[2]English words can consist of multiple morphemes as well (e.g., buying → buy/verb + ing/progressive_suffix), but such morphology is usually not used in parsing.

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

Table 1: POS tags in the Sejong Treebank (PM: predicate marker, CP: case particle, EM: ending marker, DS: derivational suffix, PR: particle, SF SP SS SE SO: different types of punctuation).

Automatic morphological analysis has been one of the most broadly explored topics in Korean NLP. Kang and Woo (2001) presented the KLT morphological analyzer, which has been widely used in Korea. The Sejong project distributed the Intelligent Morphological Analyzer (IMA), used to pre-process raw texts in the Sejong Treebank.[3] Shim and Yang (2002) introduced another morphological analyzer, called Mach, optimized for speed; it takes about 1 second to analyze 1.3M words yet performs very accurately. Han (2005) presented a finite-state transducer that had been used to check morphology in the Penn Korean Treebank. We use IMA and Mach to generate fine-grained (IMA) and coarse-grained (Mach) morphologies for our experiments.

Statistical dependency parsing has been explored relatively little in Korean. Chung (2004) presented a dependency parsing model using surface contextual information. This parser can be viewed as a probabilistic rule-based system that gathers probabilities from features like lexical items, POS tags, and distances. Oh and Cha (2008) presented another statistical dependency parsing model using cascaded chunking for parsing and conditional random fields for learning. Our work is distinguished from theirs in mainly two ways. First, we add labels to dependency edges during the conversion, so parsing performance can be evaluated on both labels and edges. Second, we selectively choose morphemes useful for dependency parsing, which prevents generating very sparse features. The morpheme selection is done automatically by applying our linguistically motivated rules (cf. Section 5.3).

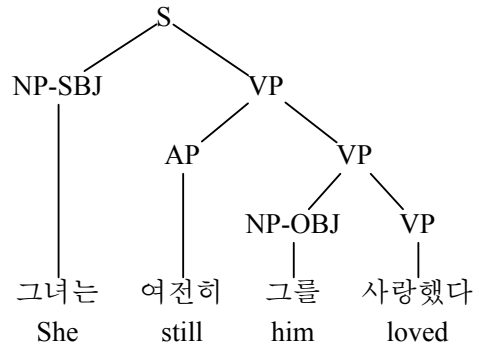
[3]The system is not publicly available but can be requested from the Sejong project (http://www.sejong.or.kr).

Han et al. (2000) presented an approach for handling structural divergence and recovering dropped arguments in a Korean-to-English machine translation system. In their approach, they used a Korean dependency parser for lexico-structural processing.

3 Constituent-to-dependency conversion

3.1 Constituent trees in the Sejong Treebank

The Sejong Treebank contains constituent trees similar to ones in the Penn Treebank.[4] Figure 2 shows a constituent tree and morphological analysis for a sentence, *She still loved him*, in Korean.



그녀는 → 그녀(she)/NP+는/JX

여전히 → 여전히(still)/MAG

그를 → 그(he)/NP+를/JKO

사랑했다 → 사랑(love)/NNG+하/XSV+았/EP+다/EF

Figure 2: A constituent tree and morphological analysis for a sentence, *She still loved him*, in Korean.

[4]The bracketing guidelines can be requested from the Sejong project, available only in Korean.

The tree consists of phrasal nodes as described in Table 2. Each token can be broken into several morphemes annotated with POS tags (Table 1). In the Sejong Treebank, tokens are separated mostly by white spaces; for instance, an item like 'A(B+C)D' is considered as a single token because it does not contain any white space in between. As a result, a token can be broken into as many as 21 individual morphemes (e.g., 'A', '(', 'B', '+', 'C', ')', 'D').

Notice that some phrases are annotated with function tags (Table 2). These function tags show dependency relations between the tagged phrases and their siblings, so can be used as dependency labels during the conversion. There are three other special types of phrase-level tags besides the ones in Table 2. `X` indicates phrases containing only case particles, ending markers, or punctuation. `L` and `R` indicate phrases containing only left and right brackets, respectively. These tags are also used to determine dependency relations during the conversion.

Phrase-level tags		Function tags	
`S`	Sentence	`SBJ`	Subject
`Q`	Quotative clause	`OBJ`	Object
`NP`	Noun phrase	`CMP`	Complement
`VP`	Verb phrase	`MOD`	Noun modifier
`VNP`	Copula phrase	`AJT`	Predicate modifier
`AP`	Adverb phrase	`CNJ`	Conjunctive
`DP`	Adnoun phrase	`INT`	Vocative
`IP`	Interjection phrase	`PRN`	Parenthetical

Table 2: Phrase-level tags (left) and function tags (right) in the Sejong Treebank.

3.2 Head-percolation rules

Table 3 gives the list of head-percolation rules (from now on, headrules), derived from analysis of each phrase type in the Sejong Treebank. Except for the quotative clause (`Q`), all other phrase types try to find their heads from the rightmost children, which aligns with the general concept of Korean being a head-final language. Note that these headrules do not involve the POS tags in Table 1; those POS tags are used only for morphemes within tokens (and each token is annotated with a phrase-level tag). It is possible to extend the headrules to token-level and find the head morpheme of each token; however, finding dependencies between different morphemes within a token is not especially interesting although there are some approaches that have treated each morpheme as an individual token to parse (Chung et al., 2010).[5]

`S`	`r`	`VP;VNP;S;NP|AP;Q;*`
`Q`	`l`	`S|VP|VNP|NP;Q;*`
`NP`	`r`	`NP;S;VP;VNP;AP;*`
`VP`	`r`	`VP;VNP;NP;S;IP;*`
`VNP`	`r`	`VNP;NP;S;*`
`AP`	`r`	`AP;VP;NP;S;*`
`DP`	`r`	`DP;VP;*`
`IP`	`r`	`IP;VNP;*`
`X|L|R`	`r`	`*`

Table 3: Head-percolation rules for the Sejong Treebank. `l`/`r` implies looking for the leftmost/rightmost constituent. `*` implies any phrase-level tag. `|` implies a logical OR and `;` is a delimiter between tags. Each rule gives higher precedence to the left (e.g., `S` takes the highest precedence in `VP`).

Once we have the headrules, it is pretty easy to generate dependency trees from constituent trees. For each phrase (or clause) in a constituent tree, we find the head of the phrase using its headrules and make all other nodes in the phrase dependents of the head. The procedure goes on recursively until every node in the tree finds its head (for more details, see Choi and Palmer (2010)). A dependency tree generated by this procedure is guaranteed to be well-formed (unique root, single head, connected, and acyclic); however, it does not include labels yet. Section 3.3 shows how to add dependency labels to these trees. In addition, Section 3.4 describes heuristics to resolve some of the special cases (e.g., coordinations, nested function tags).

It is worth mentioning that constituent trees in the Sejong Treebank do not include any empty categories. This implies that dependency trees generated by these headrules consist of only projective dependencies (non-crossing edges; Nivre and Nilsson (2005)). On the other hand, the Penn Korean Treebank contains empty categories representing long-distance dependencies. It will be interesting to see if we can train empty category insertion and resolution models on the Penn Korean Treebank, run the mod-

[5]Chung et al. (2010) also showed that recovering certain kinds of null elements improves PCFG parsing, which can be applied to dependency parsing as well.

els on the Sejong Treebank, and use the automatically inserted and linked empty categories to generate non-projective dependencies.

3.3 Dependency labels

Two types of dependency labels are derived from the constituent trees. The first type includes labels retained from the function tags. When any node annotated with a function tag is determined to be a dependent of some other node by our headrules, the function tag is taken as the dependency label to its head. Figure 3 shows a dependency tree converted from the constituent tree in Figure 2, using the function tags as dependency labels (SBJ and OBJ).

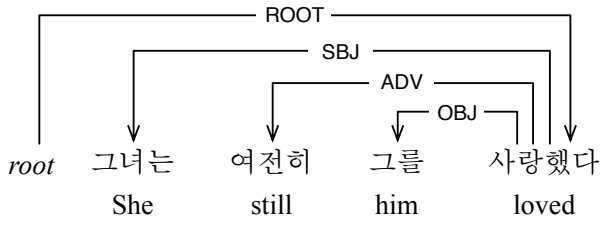


Figure 3: A dependency tree with labels, converted from the constituent tree in Figure 2.

The other labels (ROOT and ADV) belong to the second type; these are labels inferred from constituent relations. Algorithm 1 shows how to infer these labels given constituents c and p, where c is a dependent of p according to our headrules. ROOT is the dependency label of the root node (*root* in Figure 3). ADV is for adverbials, and (A|D|N|V)MOD are for (adverb, adnoun, noun, verb) modifiers, respectively. DEP is the default label used when no condition is met. These inferred labels are used when c is not annotated with a function tag.

There are two other special types of labels. A dependency label P is assigned to all punctuation regardless of function tags or constituent relations. Furthermore, when c is a phrase type X, a label X is conjoined with its genetic dependency label (e.g., CMP → X_CMP). This is because the phrase type X usually consists of morphemes detached from their head phrases (e.g., case particles, ending markers), so should have distinguished dependency relations from other standalone tokens. Table 4 shows the distribution of all labels in the Sejong Treebank (in percentile).

input : (c, p), where c is a dependent of p.
output: A dependency label l as $c \overset{l}{\leftarrow} p$.
begin
  if p = *root* **then** ROOT → l
  elif c.pos = AP **then** ADV → l
  elif p.pos = AP **then** AMOD → l
  elif p.pos = DP **then** DMOD → l
  elif p.pos = NP **then** NMOD → l
  elif p.pos = VP|VNP|IP **then** VMOD → l
  else DEP → l
end

Algorithm 1: Getting inferred labels.

AJT	11.70	MOD	18.71	X	0.01
CMP	1.49	AMOD	0.13	X_AJT	0.08
CNJ	2.47	DMOD	0.02	X_CMP	0.11
INT	0.09	NMOD	13.10	X_CNJ	0.02
OBJ	8.95	VMOD	20.26	X_MOD	0.07
PRN	0.15	ROOT	8.31	X_OBJ	0.07
SBJ	11.74	P	2.42	X_SBJ	0.09

Table 4: Distribution of all dependency labels (in %).

3.4 Coordination and nested function tags

Because of the conjunctive function tag CNJ, identifying coordination structures is relatively easy. Figure 4 shows constituent and dependency trees for a sentence, *I and he and she left home*, in Korean.

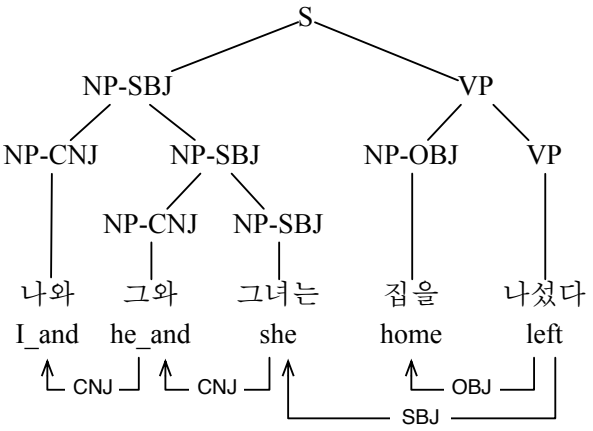


Figure 4: Constituent (top) and dependency (bottom) trees for, *I and he and she left home*, in Korean.

According to our headrules, *she* becomes the head of both *I* and *he*, which is completely acceptable but can cause long-distance dependencies when the coordination chain becomes long. Instead, we make

NN	→	NNG|NNP|SL|SH	VX	→	VX (verb)	SN	→	XSN
NX	→	NNB	AX	→	VX (adjective)	SV	→	XSV
NP	→	NP	DT	→	MM	SJ	→	XSA
NU	→	NR|SN	AD	→	MA⋆	IJ	→	IC
VI	→	VV (intransitive)	JO	→	J⋆	NR	→	NF
VT	→	VV (transitive)	EP	→	EP	UK	→	NA|NV|XR
AJ	→	VA|VCN	EM	→	EF|EC|ET⋆	SY	→	SF|SP|SS|SE|SO|SW
CP	→	VCP	PF	→	XPN			

Table 5: Mappings between POS tags generated by Mach and IMA. In each column, the left-hand and right-hand sides show POS tags generated by Mach and IMA, respectively.

each previous conjunct a dependent of its following conjunct.

Notice that the function tag SBJ is nested twice. In the Sejong Treebank, a nested function tag indicates the head of the phrase, for which we do not need headrules. Thus, whenever a node with a nested function tag is encountered, we take the node as the head of the phrase, regardless of the headrules.

4 Morphological analysis

To generate automatic morphemes and POS tags for input to our dependency parsing model, we run two systems. One is called the Intelligent Morphological Analyzer (IMA), which generates the POS tagset described in Table 1.[6] The other is called Mach that gives a more coarse-grained POS tagset (Shim and Yang, 2002). Table 5 shows mappings between POS tags generated by these systems. Note that these mappings are derived manually by comparing outputs of the two systems.

Both systems are dictionary-based analyzers using vocabularies collected from various sources, including Korean dictionaries. Given a token, these analyzers first generate possible sequences of morphemes, then rank the sequences by adding POS tags with probabilities measured from their training data. Neither of these systems gave the option of retraining their probabilistic models, so we had to verify if their models were not biased to our test data. IMA was distributed before the Sejong Treebank project, so was not trained on the Treebank. We got in touch with the author of Mach and made sure their training model was not biased to our test data.

[6]지능형 형태소 분석기: http://www.sejong.or.kr/

The reason we use outputs from two different systems is to compare the impact of fine vs. coarse-grained morphologies on dependency parsing in Korean. IMA gives not only richer POS tags but also more fine-grained (segmented) morphemes than Mach. We hypothesize that a richer morphology does not necessarily provide better features for dependency parsing. We evaluate our hypothesis by comparing parsing models trained on morphemes and POS tags generated by these two systems.

5 Dependency parsing

5.1 Parsing algorithm

To build statistical parsing models, we use Choi and Palmer (2011)'s transition-based dependency parsing approach, which has shown state-of-the-art performance in English and Czech. The key idea of this approach is to combine transitions from projective and non-projective dependency parsing algorithms so it can perform projective and non-projective parsing accordingly. As a result, it shows an expected linear time parsing speed for generating both projective and non-projective dependency trees.

Our algorithm uses three lists: λ_1, λ_2, and β. $\lambda_{1,2}$ contain tokens that have been processed and β contains tokens that have not been processed by the algorithm. For each parsing state, the algorithm performs one of the five transitions: LEFT-POP, LEFT-ARC, RIGHT-ARC, NO-ARC, and SHIFT. Transitions are determined by comparing the last token in λ_1, say w_i, with the first token in β, say w_j. After a transition is performed, w_i either moves into λ_2 or gets removed from λ_1, depending on whether or not the oracle predicts the token is needed in later parsing states. w_j is then compared with the last token in

λ_1, that is w_{i-1}. When the oracle predicts there is no token in λ_1 that has a dependency relation with w_j, w_j is removed from β and added to λ_1 along with all other tokens in λ_2. The procedure is repeated with the first token in β, that is w_{j+1}. The algorithm terminates when there is no token left in β.

5.2 Machine learning algorithm

We use Liblinear L2-regularized L1-loss SVM for learning (Hsieh et al., 2008), applying $c = 0.1$ (cost), $e = 0.1$ (termination criterion), $B = 0$ (bias).

5.3 Feature extraction

As mentioned in Section 3.1, each token in our corpora consists of one or many morphemes annotated with different POS tags. This morphology makes it difficult to extract features for dependency parsing. In English, when two tokens, w_i and w_j, are compared for a dependency relation, we extract features like POS tags of w_i and w_j (w_i.pos, w_j.pos), or a joined feature of POS tags between two tokens (w_i.pos+w_j.pos). Since each token is annotated with a single POS tag in English, it is trivial to extract these features. In Korean, each token is annotated with a sequence of POS tags, depending on how morphemes are segmented. It is possible to join all POS tags within a token and treat that as a single tag (e.g., NNP+NNG+JX for the first token in Figure 5); however, these tags usually cause very sparse vectors when used as features.

낙랑공주는	호동왕자를	사랑했다.
*Nakrang*_Princess	*Hodong*_Prince	loved.

낙랑공주는 → 낙랑/NNP+공주/NNG+는/JX
Nakrang + Princess + JX
호동왕자를 → 호동/NNP+왕자/NNG+를/JKO
Hodong + Prince + JKO
사랑했다. → 사랑/NNG+하/XSV+았/EP+다/EF+./SF
Love + XSV + EP + EF + .

Figure 5: Morphological analysis for a sentence, *Princess Nakrang loved Prince Hodong.*, in Korean.

An alternative is to extract the POS tag of only the head morpheme for each token. This prevents the sparsity issue, but we discover that no matter how we choose the head morpheme, it prunes out too many other morphemes helpful for parsing. Thus, as a compromise, we decide to select certain types of morphemes and use only these as features. Table 6 shows the types of morphemes used to extract features for our parsing models.

FS	The first morpheme
LS	The last morpheme before JO|DS|EM
JK	Particles (J⋆ in Table 1)
DS	Derivational suffixes (XS⋆ in Table 1)
EM	Ending markers (E⋆ in Table 1)
PY	The last punctuation, only if there is no other morpheme followed by the punctuation

Table 6: Types of morphemes in each token used to extract features for our parsing models.

Figure 6 shows morphemes extracted from the tokens in Figure 5. For unigrams, these morphemes can be used either individually (e.g., the POS tag of JK for the 1st token is JX) or jointly (e.g., a joined feature of POS tags between LS and JK for the 1st token is NNG+JX) to generate features. From our experiments, features extracted from the JK and EM morphemes are found to be the most useful.

FS	LS	JK	DS	EM	PY
낙랑/NNP	공주/NNG	는/JX	-	-	-
호동/NNP	왕자/NNG	를/JKO	-	-	-
사랑/NNG	-	-	하/XSV	다/EF	./SF

Figure 6: Morphemes extracted from the tokens in Figure 5 with respect to the types in Table 6.

For n-grams where $n > 1$, it is not obvious which combinations of these morphemes across different tokens yield useful features for dependency parsing. Trying out every possible combination is not practical; thus, we restrict our search to only joined features of two morphemes between w_i and w_j, where each morpheme is taken from a different token. It is possible to extend these features to another level, which we will explore in the future.

Table 7 shows a set of individual features extracted from unigrams. w_i is the last token in λ_1 and w_j is the first token in β as described in Sec-

tion 5.1 (they also represent the i'th and j'th tokens in a sentence, respectively). 'm' and 'p' indicate the form and POS tag of the corresponding morpheme.

	FS	LS	JK	DS	EM	PY
w_i	m,p	m,p	m,p		m,p	m
w_j	m,p	m,p	m,p		m,p	m
w_{i-1}		m	m,p	p	m,p	m
w_{i+1}		m			m,p	m
w_{j-1}	m	p	m,p		m,p	m
w_{j+1}	m,p	m	m	p	p	m

Table 7: Individual features from unigrams.

Table 8 shows a set of joined features extracted from unigrams, w_i and w_j. For instance, a joined feature of forms between FS and LS for the first token in Figure 5 is *Nakrang+Princess*.

	FS	LS	JK
JK	m+m	m+m	
EM	m+m	m+m	m+m

Table 8: Joined features from unigrams, w_i and w_j.

Finally, Table 9 shows a set of joined features extracted from bigrams, w_i and w_j. Each column and row represents a morpheme in w_i and w_j, respectively. 'x' represents a joined feature of POS tags between w_i and w_j. 'y' represents a joined feature between a form of w_i's morpheme and a POS tag of w_j's morpheme. 'z' represents a joined feature between a POS tag of w_i's morpheme and a form of w_j's morpheme. '*' and '+' indicate features used only for fine-grained and coarse-grained morphologies, respectively.

	FS	LS	JK	DS	EM
FS	x	y,z	x^*,y^+,z	z	z
LS	x	x, z	x^*,y^+	x,z	x
JK	x^*,y,z^+	x,y	x^*,y^+,z^+	x,y^+	x^*,y^+
DS	x,z	y	x	x	x, z
EM	x	z	y, z	z	x, z^+

Table 9: Joined features from bigrams, w_i and w_j.

A few other features such as dependency labels of [w_j, the rightmost dependent of w_i, and the leftmost dependent of w_j] are also used. Note that we are considering far fewer tokens than most other dependency parsing approaches (only w_i, w_j, $w_{i\pm1}$, $w_{j\pm1}$). We expect to achieve higher parsing accuracy by considering more tokens; however, this small span of features still gives promising results.

6 Experiments

6.1 Corpora

The Sejong Treebank contains 20 different corpora covering various topics. For our experiments, we group these corpora into 6 genres: Newspaper (NP), Magazine (MZ), Fiction (FI), Memoir (ME), Informative Book (IB), and Educational Cartoon (EC). NP contains newspapers from five different sources talking about world, politics, opinion, etc. MZ contains two magazines about movies and educations. FI contains four fiction texts, and ME contains two memoirs. IB contains six books about science, philosophy, psychology, etc. EC contains one cartoon discussing world history.

Table 10 shows how these corpora are divided into training, development, and evaluation sets. For the development and evaluation sets, we pick one newspaper about art, one fiction text, and one informative book about trans-nationalism, and use each of the first half for development and the second half for evaluation. Note that these development and evaluation sets are very diverse compared to the training data. Testing on such evaluation sets ensures the robustness of our parsing model, which is very important for our purpose because we are hoping to use this model to parse various texts on the web.

	NP	MZ	FI	ME	IB	EC
T	8,060	6,713	15,646	5,053	7,983	1,548
D	2,048	-	2,174	-	1,307	-
E	2,048	-	2,175	-	1,308	-

Table 10: Number of sentences in training (T), development (D), and evaluation (E) sets for each genre.

6.2 Evaluations

To set an upper bound, we first build a parsing model based on gold-standard morphology from the Sejong Treebank, which is considered fine-grained morphology. To compare the impact of fine-grained vs. coarse-grained morphologies, we train two other

	Gold, fine-grained			Auto, fine-grained			Auto, coarse-grained		
	LAS	UAS	LS	LAS	UAS	LS	LAS	UAS	LS
NP	82.58	84.32	94.05	79.61	82.35	91.49	79.00	81.68	91.50
FI	84.78	87.04	93.70	81.54	85.04	90.95	80.11	83.96	90.24
IB	84.21	85.50	95.82	80.45	82.14	92.73	81.43	83.38	93.89
Avg.	**83.74**	**85.47**	**94.57**	**80.43**	**83.01**	**91.77**	**80.14**	**82.89**	**91.99**

Table 11: Parsing accuracies achieved by three models (in %). LAS - labeled attachment score, UAS - unlabeled attachment score, LS - label accuracy score

parsing models, based on the output of IMA, [auto, fine-grained], and the output of Mach, [auto, coarse-grained]. All parsing accuracies achieved by these three models are provided in Table 11.

The [gold, fine-grained] model shows over three points improvement on the average LAS compared to the [auto, fine-grained] model. The [auto, fine-grained] morphology gives an F1-score of 89.59% for morpheme segmentation, and a POS tagging accuracy of 94.66% on correctly segmented morphemes; our parsing model is expected to perform better as the automatic morphological analysis improves. On the other hand, the differences between the [auto, fine-grained] and [auto, coarse-grained] models are small. More specifically, the difference between the average LAS achieved by these two models is statistically significant (McNemar, $p = 0.01$); however, the difference in the average UAS is not statistically significant, and the average LS is actually higher for the [auto, coarse-grained] model.

These results seem to confirm our hypothesis, "a more fine-grained morphology is not necessarily a better morphology for dependency parsing"; however, more careful studies need to be done to verify this. Furthermore, it is only fair to mention that the [auto, fine-grained] model uses a smaller set of features than the [auto, coarse-grained] model (Table 9) because many lexical features can be replaced with POS tag features without compromising accuracy for the [auto, fine-grained] model, but not for the [auto, coarse-grained] model.

It is interesting to see that the numbers are usually high for LS, which shows that our models successfully learn labeling information from morphemes such as case particles or ending markers. All three models show robust results across different genres although the accuracies for NP are significantly lower than the others. We are currently working on the error analysis of why our models perform worse on NP and how to improve accuracy for this genre while keeping the same robustness across the others.

7 Conclusion and future work

There has been much work done on automatic morphological analysis but relatively less work done on dependency parsing in Korean. One major reason is the lack of training data in dependency structure. Here, we present head-percolation rules and heuristics to convert constituent trees in the Sejong Treebank to dependency trees. We believe these head-rules and heuristics can be beneficial for those who want to build statistical dependency parsing models of their own.

As a pioneer of using this dependency Treebank, we focus our effort on feature extraction. Because of rich morphology in Korean, it is not intuitive how to extract features from each token that will be useful for dependency parsing. We suggest a rule-based way of selecting important morphemes and use only these as features to build dependency parsing models. Even with a small span of features, we achieve promising results although there is still much room for improvement.

We will continuously develop our parsing model by testing more features, treating morphemes as individual tokens, adding deterministic rules, etc. We are planning to use our parsing model to improve other NLP tasks in Korean such as machine translation and sentiment analysis.

Acknowledgments

Special thanks are due to Professor Kong Joo Lee of Chungnam National University and Kwangseob Shim of Shungshin Women's University. We grate-

fully acknowledge the support of the National Science Foundation Grants CISE-IIS-RI-0910992, Richer Representations for Machine Translation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Jinho D. Choi and Martha Palmer. 2010. Robust constituent-to-dependency conversion for multiple corpora in english. In *Proceedings of the 9th International Workshop on Treebanks and Linguistic Theories*, TLT'9.

Jinho D. Choi and Martha Palmer. 2011. Getting the most out of transition-based dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, ACL:HLT'11, pages 687–692.

Tagyoung Chung, Matt Post, and Daniel Gildea. 2010. Factors affecting the accuracy of korean parsing. In *In Proceedings of NAACL Workshop on Statistical Parsing of Morphologically Rich Languages*, SPMRL'10, pages 49–57.

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

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning: Shared Task*, CoNLL'09, pages 1–18.

Chung-hye Han, Benoit Lavoie, Martha Palmer, Owen Rambow, Richard Kittredgc, Tanya Korelsky, Nari Kim, and Myunghee Kim. 2000. Handling structural divergences and recovering dropped arguments in a korean/english machine translation system. In *Proceedings of the Association for Machine Translation in the Arnericas*, AMTA'00.

Chung-hye Han, Na-Rae Han, Eon-Suk Ko, Martha Palmer, and Heejong Yi. 2002. Penn korean treebank: Development and evaluation. In *In Proceedings of the 16th Pacific Asia Conference on Language, Information and Computation*, PACLIC'02.

Na-Rae Han. 2005. Klex: A finite-state transducer lexicon of korean. In *In Proceedings of the 5th International Workshop on Finite-State Methods and Natural Language Processing*.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. 2008. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning (ICML'08)*, pages 408–415.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for english. In *Proceedings of the 16th Nordic Conference of Computational Linguistics*, NODALIDA'07.

Mark Johnson and Ahmet Engin Ural. 2010. Reranking the berkeley and brown parsers. In *Proceedings of the North American Chapter of the Association for Computational Linguistics Conference*, NAACL'10, pages 665–668.

Seung-Shik Kang and Chong-Woo Woo. 2001. Automatic segmentation of words using syllable bigram statistics. In *In Proceedings of the 6th Natural Language Processing Pacific Rim Symposium*, NLPRS'01, pages 729–732.

Kong Joo Lee. 1998. *Probabilistic Parsing of Korean based on Language-Specific Properties*. Ph.D. thesis, Korea Advanced Institute of Science and Technology.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, LREC'06, pages 449–454.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL:HLT'08)*, pages 950–958.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, ACL'05, pages 99–106.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Jin-young Oh and Jeong-Won Cha. 2008. Robust korean dependency analysis based on crfs. In *Proceedings of the 20th Annual Conference on Human and Cognitive Language Technology*, HCLT'08, pages 23–28.

Kwangseob Shim and Jaehyung Yang. 2002. Mach : A supersonic korean morphological analyzer. In *Proceedings of the 19th International Conference on Computational Linguistics*, COLING'02, pages 939–945.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning: Shared Task*, CoNLL'08, pages 59–177.