Building Universal Dependency Treebanks in Korean

Jayeol Chun[1], Na-Rae Han[2], Jena D. Hwang[3], Jinho D. Choi[1]
Emory University[1], University of Pittsburgh[2], IHMC[3]
Atlanta GA 30322[1], Pittsburgh, PA 15260[2], Ocala, FL 32502[3]
che.yeol.chun@emory.edu, naraehan@pitt.edu, jhwang@ihmc.us, jinho.choi@emory.edu

Abstract

This paper presents three treebanks in Korean that consist of dependency trees derived from existing treebanks, the Google UD Treebank, the Penn Korean Treebank, and the KAIST Treebank, and pseudo-annotated by the latest guidelines from the Universal Dependencies (UD) project. The Korean portion of the Google UD Treebank is re-tokenized to match the morpheme-level annotation suggested by the other corpora, and systematically assessed for errors. Phrase structure trees in the Penn Korean Treebank and the KAIST Treebank are automatically converted into dependency trees using head-finding rules and linguistic heuristics. Additionally, part-of-speech tags in all treebanks are converted into the UD tagset. A total of 38K+ dependency trees are generated that comprise a coherent set of dependency relations for over a half million tokens. To the best of our knowledge, this is the first time that these Korean corpora are analyzed together and transformed into dependency trees following the latest UD guidelines, version 2.

Keywords: universal, dependency, conversion, korean, treebank

1. Introduction

The Universal Dependencies (UD) project has brought on an increasing momentum to the research community for finding morphological patterns and syntactic relations appropriate to multiple languages (Zeman et al., 2017). The UD project has facilitated collaborative work among several organizations for 70+ languages, and inspired computational linguists to further analyze both resource-rich and -poor languages by suggesting universal guidelines that help them create and augment treebanks in different languages. The UD project has also promoted research on cross-lingual learning that explores the possibility of adapting statistical parsing models from one language to another (McDonald et al., 2013).

Several treebanks had been introduced for Korean, all of which comprised annotation of morphemes and phrase structure trees (Choi et al., 1994; Han et al., 2002; Hong, 2009), each following its own set of guidelines. Phrase structure trees in these treebanks had been converted into dependency trees using head-finding rules and linguistically-motivated heuristics, and used to evaluate Korean dependency parsing performance (Choi and Palmer, 2011; Choi, 2013). The previous efforts did not, however, focus on the compatibility among dependency trees converted from different corpora, resulting in the generation of a distinct set of dependency relations for each treebank.

This paper presents three dependency treebanks in Korean, derived from existing corpora and pseudo-annotated by the latest UD guidelines, version 2. The motivation behind this study is to make a comprehensive analysis between these corpora and convert phrase structure trees across different treebanks into dependency trees with consistent relations, providing a large corpus of compatible dependency trees. The contributions of this work are as follows:

- The Google UD Korean Treebank is manually assessed and systematically corrected (Section 3.).
- Phrase structure trees in both the Penn Korean Treebank and the KAIST Treebank are converted into dependency trees using the UD guidelines (Sections 4. and 5.).
- Corpus analytics are provided that include statistics of the new dependency treebanks, and remaining issues with the current annotation (Section 6.).

To the best of our knowledge, this is the first time that these Korean corpora are analyzed together and transformed into dependency trees following the latest UD guidelines.

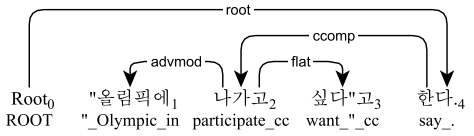
2. Related Work

Petrov et al. (2012) introduced the universal part-of-speech tagset and provided a mapping from 25 different treebank tagsets to this universal set. They showed that parsing performance using the universal part-of-speech tagset was comparable to the one using the original tagsets. McDonald et al. (2013) presented the universal dependency annotation and provided pseudo and manually annotated dependency treebanks for 6 languages. They showed promising results for cross-lingual parsing and initiated the effort for developing universally acceptable grammars. The official UD project started with a group of 10 languages (Nivre et al., 2015) and has expanded to over 70 languages. Recently, this project organized the CoNLL'17 shared task on multilingual parsing, involving over 40 languages (Zeman et al., 2017).

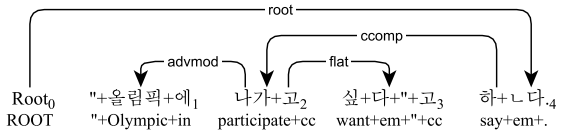
In addition, the Sejong Treebank, consisting of phrase structures trees for 60K sentences from 6 different genres of text released by Hong (2009), were converted into dependency trees by Choi and Palmer (2011). Despite of its large size, the Sejong Treebank is excluded from this work due to the license restriction. Hani corpora (Park, 2017) is also an effort annotated under UD guidelines; however, published exposition of this work has not yet been made available.

3. Google UD Korean Treebank

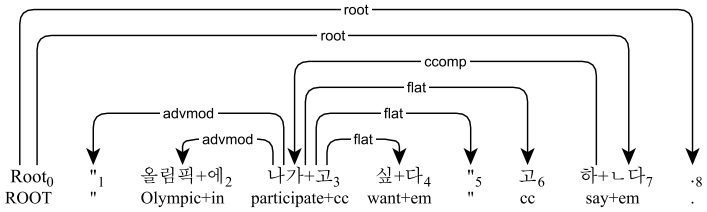
McDonald et al. (2013) provided the Google UD Treebanks comprising 6K sentences scraped from weblogs and newswire, annotated under the universal dependency guidelines for 6 languages including Korean. Because these treebanks were annotated before the official UD project started, the guidelines under which the Korean treebank was created



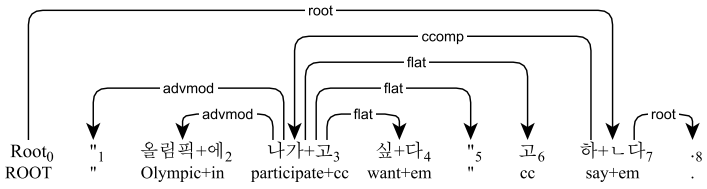
(a) A sample dependency tree from the original GKT.



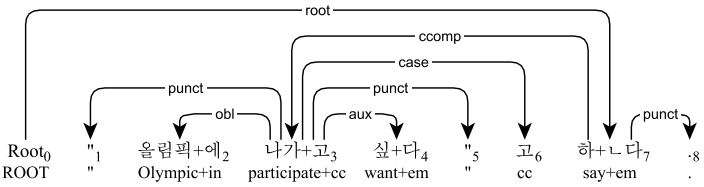
(b) After morphological analysis (Section 3.1.).



(c) After tokenization (Section 3.2.).



(d) After head ID remapping (Section 3.4.).



(e) After dependency relabeling (Section 3.5.).

Figure 1: Step-by-step illustration of our correction procedure of GKT (`cc`: coordinating conjunction, `em`: ending marker).

differed significantly from that of the version 2 of the UD (UDv2). The Google UD Korean Treebank (GKT) was automatically converted to follow the UDv2 guidelines, and distributed as a part of the CoNLL'17 shared task datasets. We perform a manual check over GKT to determine whether or not this automatic conversion generated sound dependency relations and carry out systematic correction.

3.1. Morphological Analysis

Korean is an agglutinative language with highly productive verbal and nominal suffixation, and limited prefixation. Without morphological analysis, then, any system that solely relies on surface forms must contend with the sparsity issue. As McDonald et al. (2013) points out, the automatic tokenization carried out for the original GKT was generally too coarse-grained; the suffixes or particles were left in with the tokens, indicating the necessity for future improvements through manual revision and annotation.

To help remedy this problem, we augment GKT with automatic morphological analysis obtained by the KOMA tagger, a general-purpose morphological analyzer for Korean (Lee and Rim, 2009) that produces the morpheme tagset defined by the Sejong Treebank (Hong, 2009). Figure 1(b) shows the morphological analysis of the original sentence in 1(a). The full morphological analysis is included for each token as the last column in our dataset.

3.2. Proper Tokenization

The tokenization in GKT does not split out the inflectional and derivational particle as separate tokens, nor are the punctuations tokenized. While a complete retokenization of particles in GKT is beyond the scope of this study, since improper tokenization of punctuation can lead to inappropriate dependency relations, we tackle the tokenization of symbols and punctuation marks for the proper configuration of the dependency relations. The morphological analysis from the KOMA tagger enables us to recognize symbols as well as particles so that they are split into separate tokens in our corpus. This is exemplified in Figure 1(c), where the two double quotes found in the 1st and 3rd tokens and the period in 4th token, are retokenized. Dependency labels for these new tokens are inferred from their morpheme tags. Over 9K tokens with embedded punctuation are revised, resulting in 3K additional tokens.

3.3. Part-of-speech Tags Relabeling

Once properly tokenized, measures are taken to assign appropriate parts-of-speech (POS) tags to separated tokens based on their morphemes. Note that the original GKT provides two POS tags for each token (columns 4 and 5), first of which is UDv2 compliant. Our relabeling focuses on replacing the first set of POS tag, and for the sake of consistency with other corpora, the secondary POS column is removed from our corpus.

3.4. Head ID Remapping

With tokenization and POS assignment complete, the head IDs of the separated tokens are redirected. In general, the word inherits the original head ID while the punctuation points to the previous token (i.e., token from which the punctuation was split) as seen in token 8 in Figure 1(d).

An exception is made for quotations or parenthetical phrases. Based on the observation that in general a quotation forms a sentence, a quotation (marked by quotation marks (e.g., “ ”) and seen in the 1st and 3rd token in Figure 1(b)) will feature its own sub-dependency tree where only its root will link to an element outside of the quotation. Therefore, the root of the sub-dependency tree is located by finding the link from within the quotation to an outside element. Punctuation points to the head of the quotation, as seen with 1st and 5th tokens in the Figure 1(d).

In the case of parenthetical expressions involving (), <>, [], ‘’ and ≪≫, we found that in the vast majority of cases, the elements within the parenthetical symbols were supplementary phrases describing a preceding token. This being so, the head of the parenthetical phrase is assigned to the rightmost element[1]. When the parenthetical expression forms a single token with the preceding word as seen in Figure 2, the token preceding the parenthetical expression inherits the original head ID and becomes the head of the root of the parenthetical expression. If there are any case particles attached to the right of the parenthetical (see token 6 in the same figure), then the case markings are also made dependent on the token preceding the parenthetical expression.

[1]Note that Korean is a head-last language.

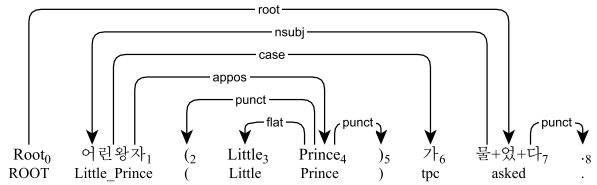


Figure 2: Example dependency tree with a parenthetical expression (tpc: topic marker).

3.5. Dependency Relabeling

Since the CoNLL'17 shared task, UDv2 has undergone changes that were not reflected in GKT. Thus, we apply morpheme-level rules to GKT and relabel all dependency relations to reflect the latest updates in UDv2. In Figure 1(e), the 2nd and 3rd tokens translate to *Olympics+in* and *participate*, respectively. Previous UDv2 considers *Olympics+in* an adverbial modifier (advmod) of *participate*, which is relabeled as an oblique (obl) in our corpus, as specified in the newest version of UDv2.

3.6. Lexical Correction

We manually assess the entire GKT for spelling errors. Social media is one of the main sources for GKT, which include a disproportionately large number of misspellings. Some are common incorrect spellings (e.g., 왠만하면 → 웬만하면) or deliberate non-standard forms known as 'netspeak' (e.g., 시른 → 싫은), while the rest are simple errors. Additionally, the HTML entity symbols are replaced with corresponding lexical symbols (e.g, & → &). The corrected spellings, 146 tokens in total, are provided in the lemma column.

4. Penn Korean Treebank

Han et al. (2006) created the Penn Korean Treebank (PKT) consisting of manual annotation of morphemes and phrase structure trees for 15K sentences from newswire in Korean. PKT is the only Korean treebank including annotation of empty categories, which enables to generate non-projective dependencies. The previous version of PKT (Han et al., 2002), which included phrase structure trees for 5k sentences from a military corpus—known as the Virginia corpus, is excluded from our conversion due to the lack of generality in its source, the military domain.

4.1. Empty Categories

Empty categories denote nominal units that point to the location of their antecedent syntactic elements found elsewhere in the sentence. In dependency structure, they serve to capture long-distance dependencies at the cost of introducing non-projective dependencies in the resultant tree. PKT features four empty categories exemplified in Figure 3: (1) trace *T* seen on line 3, (2) dropped subject *pro* seen on line 1, (3) empty operator *op* seen on line 0, and (4) ellipsis *?* seen on line 7.

4.1.1. Trace

An argument that precedes its subject leaves in its place a trace *T*. Given a terminal node that represents a trace like (NP-OBJ *T*-1) in line 3 in Figure 3, we find its

antecedent, (WHNP-1*op*) in line 1. Then we reorder the sentence in such a manner that the subtree with the non-terminal node as a root is extracted out of its position and inserted in place of the trace node, resulting in Figure 4.

```
0: (S (NP-SBJ (S (WHNP-1 *op*)
1:                (S (NP-SBJ *pro*)
2:          (VP (NP-ADV 어제/NNC)
3:              (VP (NP-OBJ *T*-1)
4:               사/VV+은/EAN))))
5:        (NP 아이폰/NPR+은/PAU))
6:     (ADJP (NP-COMP 어디/NPN+에/PAD)
7:           (VJ *?*))
8: ?/SFN)
```

Figure 3: Examples of 4 types of empty categories: *op*, *pro*, *T*, *?*.

```
0: (S (NP-SBJ (S (S (NP-SBJ  *pro*)
1:          (VP (NP-ADV 어제/NNC)
2:              (VP (NP-OBJ (WHNP-1 *op*))
3:                  사/VV+은/EAN))))
4:       (NP 아이폰/NPR+은/PAU))
5:     (ADJP (NP-COMP 어디/NPN+에/PAD)
6:           (VJ *?*))
7: ?/SFN)
```

Figure 4: The example in Figure 3 after trace mapping.

4.1.2. Empty Assignment and Empty Operator

Dropped arguments are represented by *pro* and relative clauses are represented by *op*. No explicit steps are taken to reorder sentence structures with these empty categories.

4.1.3. Ellipsis

Elided elements are indicated with *?* in PKT, which can result from a dropped predicate in a matrix clause (Figure 3) or when two clauses are coordinated with an implicitly shared predicate (Figure 5). In the first case, resolving the predicate will involve contextual information and therefore is outside of our project's scope. In the second scenario, mapping ellipsis must be performed intra-sententially: the first step is locating the predicate that has been 'deleted', and point to it as a head. PKT however does not provide an index that links the ellipsis token and its antecedent like it does with empty operators, presumably due to the fact that not all ellipses have in-sentence antecedents. To remedy this, we represent this relationship as a fixed conjunct, as seen with the 3rd and the 7th token in Figure 5. The relationship is established through simple heuristics of matching constituency tags at phrasal and morpheme level as well as functions tags if they exist.

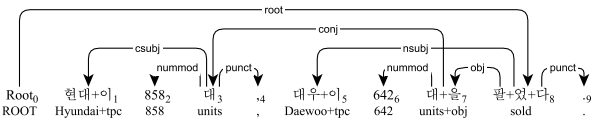


Figure 5: Example dependency tree with Ellipsis (obj: objective case particle).

4.2. Coordination

Following the guideline of Choi and Palmer (2011), each conjunct points to its right sibling as its head so that the rightmost conjunct becomes the head of the phrase. Because PKT does not offer the conjunctive function tag, our conversion discovers coordination structure by applying a set of heuristics[2]. An example of the coordination structure is shown in Figure 4.2., where 호박 (*pumpkin*) is the head of its left sibling 양파와 (*Onion+tpc*), and 오이가 (*Cucumber+tpc*) is made the head of the entire noun phrase involving the coordinated structure.

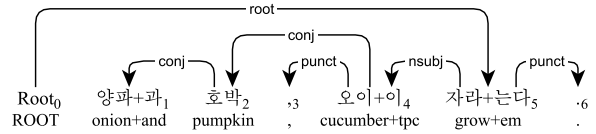


Figure 6: Sample PKT dependency tree with coordination.

4.3. Part-of-speech Tags

The POS tags are manually mapped from PKT to UDv2;[3] for the most part, this mapping is categorical. One exception is DAN, determiner-adnominal, which encompasses two semantically distinct subgroups: (1)demonstrative prenominals (e.g., 이 (*this*), 그 (*the*), 저 (*it*)) and (2)attribute adjectives that lack predicative counterparts (e.g., 새 (*new*), 헌 (*old*)). The former is mapped to DET (determiner); the latter to ADJ (adjective). Additionally, we identify nominal and verbal particles whose function are to encode conjunction and assigned them to the appropriate UDv2 POS tags. PCJ (conjunctive post-position) is singled out and assigned to CCONJ (conjunction), while the remaining post-position categories (PCA, PAD, PAU) are mapped to ADP (adposition). The ECS (coordinate, subordinate, adverbial) verbal endings require additional attention to context: they are categorized as CCONJ when they are considered coordinating verbs or verb phrases, and as SCONJ when considered coordinating clauses. All remaining verbal endings are categorized as PART (particle) along with copula (CO) and suffixes (X*).

4.4. Dependency Relations

The establishment of dependency relations starts with handling empty categories, discussed in Section 4.1. Then each node is assigned its head with head-percolation rules based on Table 1. The dependency relationship between the node and its head is inferred by investigating the function tags, phrasal tags and morphemes.

5. Kaist Treebank

Choi et al. (1994) created the KAIST Treebank (KTB) containing phrase structure trees for 31K sentences from various sources including literature, newswire, and academic manuscripts. Trees in this corpus were converted into dependency trees and used as a part of the shared task on parsing

[2]A simpler version of the heuristics used for PKT is exemplified by Algorithm 1, that is, coordination heuristics for KAIST.

[3]The mappings between the POS tagsets from PKT, KTB, and UDv2 can be found from our project site.

morphologically rich languages (Choi, 2013). Unlike PKB, KTB does not include empty categories and function tags, which renders the dependency conversion more challenging.

5.1. Coordination

Coordination in KTB is discovered and handled by Algorithm 1, which calls Algorithm 2 to check whether a given phrase or a sentence contains a coordination. However, the lack of empty categories in KTB, and hence the lack of representation of verb ellipsis, is the most notable difference between the two corpora. As it was for PKT, the rightmost conjunct becomes the head of the coordination.

Algorithm 1: find_coordination(C, R)

Input : A constituent C; the headrule R of C
$child$, $head \leftarrow$ `null`, `null`;
$children \leftarrow C$'s children list;
$type \leftarrow$ contains_coordination(C, $children$);
switch $type$ **do**
  case *0*
    return $false$;
  case *1*
    foreach $c \in children$ **do**
      if $child$ = `null` **then**
        $child$, $head \leftarrow c$, c;
      else
        if *c is* `sp` **then**
          c.set_head($child$, `punct`);
        else if *c ends with* `jcj` **then**
          $child$.set_head(c, `conj`);
          $child$, $head \leftarrow c$, c;
        else if *c is* `maj` **then**
          c.set_head(*C's right sibling*, `cc`);
        else
          $child$.set_head(c, `conj`);
          $child$, $head \leftarrow c$, c;
  case *2*
    foreach c *in* $children$ **do**
      if *child is null* **then**
        $child$, $head \leftarrow c$, c;
      else
        $child$.set_head(c, `conj`);
        $child$, $head \leftarrow c$, c;
if $type > 0$ **then**
  C.update_head(head);

5.2. Part-of-speech Tags

Similarly to PKT, the KTB POS tag mapping, for the most part, is categorical; exhibiting many-to-one mappings from KTB to UDv2. In some cases, KTB and UDv2 take a different slice through the semantics of what these tags represent. For example, while the KTB's case particles generally map to the UDv2's adpositions (`ADP`), the conjunctive case particles (`jcj`) in KTB functionally align with the UDv2's conjunctions (`CONJ`). Much like PKT, the ending particles (`x*`) in KTB are analyzed on the basis of semantic context: adveribial derivational suffixes (`xsa`) signal assignments to the UDv2's adverbs (`ADV`), while the rest of the ending particles in KTB are considered `PART` in UDv2.

Algorithm 2: contains_coordination(C, N)

Input : A constituent C;
  An ordered list N of child constituents of C
Output : The conjunct-flag, either 0, 1, 2 or 3
if C *is* `NP` **then**
  foreach c *in* N **do**
    if c is `maj` or `sp` **then return** 1
    if c ends with `ecc` or `jcj` **then return** 1
if C *is* `VP` *or* `ADJP` **then**
  foreach c *in* `N` **do**
    if c ends with `ecc` **then return** 2
return 0;

5.3. Dependency Relations

KTB dependency conversion follows the procedure outlined for PKT where the head of nodes is located with head-percolation rules based on Table 2.

While the dependency label inference benefits from the rich morphological analysis of KTB, the small number of phrasal tags and the absence of function tags has led to complications such as mapping of noun phrases ending with *jxt* to `dislocated`. Similarly to PKT, where *-SBJ* function tag denotes a subject node, KTB offers three morpheme tags for the same purpose: *jcs*, *jcc*, and *jxt*. However, while *jcs* and *jcc* roughly correspond to `nsubj` and `csubj`, `jxt` suggests that the phrase is the topic of the phrase or clause, but offers nothing informative in distinguishing whether it is in fact a subject (which it frequently is) and, if so, whether it is a clausal or nominative subject. Although UDv2 offers `dislocated` for topical elements ubiquitous in languages like Korean and Japanese, KTB offers no systematic way of distinguishing `dislocated` from its subject counterparts in `nsubj` or `csubj`.

Phrase	D	Headrules
S	r	VP;ADJP;S;NP;ADVP;*
VP	r	VP;ADJP;VV|VJ;CV;LV;V*;NP;S;*
NP	r	N*;S;N*;VP;ADJP|ADVP;*
DANP	r	DANP|DAN;VP;*
ADVP	r	ADVP;ADV;-ADV;VP;NP;S;*
ADJP	r	ADJP;VJ;LV;*
ADCP	r	ADC;VP;NP|S;*
ADV	r	VJ;NNC;*
VX	r	V*; NNX;*
VV	r	VV;NNC;VJ;*
VJ	r	VJ;NNC;*
PRN	r	NPR;N*|NP|VP|S|ADJP|ADVP;*
CV	r	VV;*
LV	r	VV;J;*
INTJ	r	INTJ;IJ;VP;*
LST	r	NNU;*
X	r	*

Table 1: Headrules for PKT. **Phrase** lists all phrasal tags in PKT. **D** denotes the search direction, `r` denotes searching for rightmost constituent, `*` denotes any tag headed by what follows, and `|` denotes logical `or`. Each **Headrule** gives higher precedence to the left tag on the list.

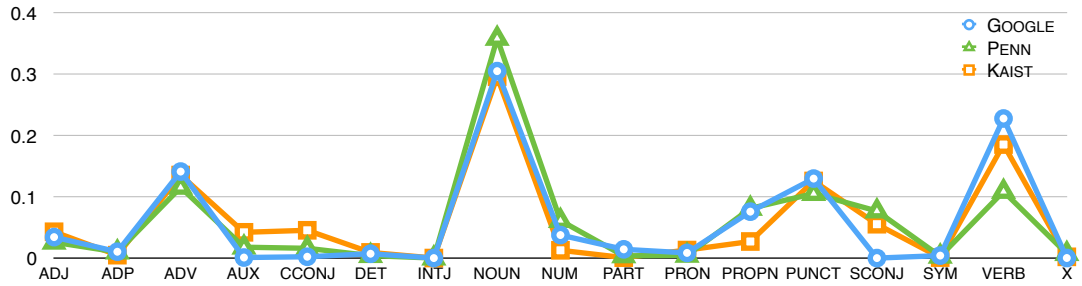


Figure 7: Distributions of part-of-speech tags for all three treebanks.

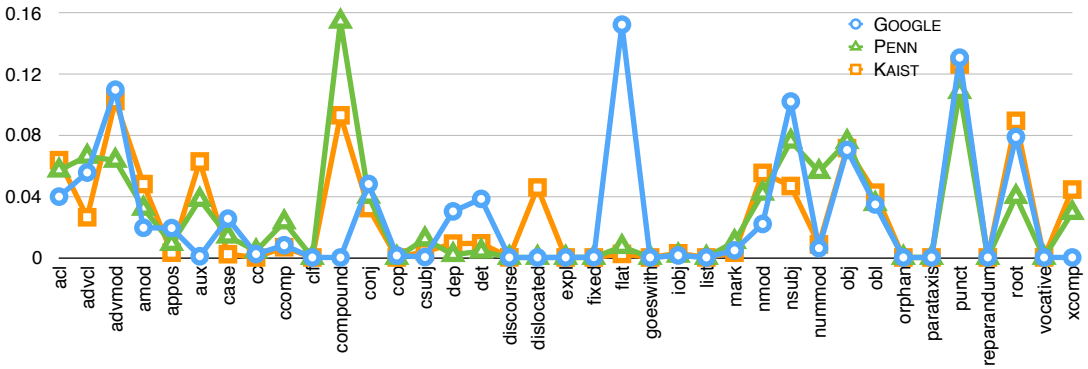


Figure 8: Distributions of the dependency labels for all three treebanks.

Phrase	D	Headrules
S	r	VP;ADJP;S;NP;ADVP;*
VP	r	pv*|pa*|n*|VP|NP;ADJP;S;*
NP	r	n*|f|NP|S|pv*|VP|pa*|ADJP;ADVP|MODP;*
ADJP	r	ADJP|pa*|n*;ADVP;VP;NP;S;*
ADVP	r	ADVP;VP;ma*;NP;S;*
AUXP	r	AUXP;NP;p*;n*;px;*
MODP	r	mm*;VP;ADJP;NP;*
IP	r	ii;p*;n*;ADVP;m*;*
X	r	*

Table 2: Headrules for KTB (see Table 1 for tabular details).

6. Corpus Analytics

6.1. Statistics of the New Dependency Treebanks

At approximately 26 dependency nodes per sentence, PKT includes on average the longest and complex sentences among the three corpora. This is likely reflective of the news domain PKT represents. KTB is by far the largest corpus in this study with its sentence complexity comparable to that of GKT at approximately 12 dependency nodes per sentence.

Number	GKT	PKT	KTB	Total
Sentences	6,339	5,010	27,363	38,712
Nodes	80,392	132,041	350,090	562,523

The frequencies of the POS tags in the three corpora are shown in Table 3. The three corpora shared NOUN, VERB, ADV and PUNCT as the top parts-of-speech (Figure 7). Beyond these four, no other POS reaches double-digit %, and the relative rankings start to diverge. In both PKT and GKT, PROPN (proper noun) is the fifth-highest ranking POS, while it is seen ranking much lower in KAIST, which instead has ADJ (adjective) taking the spot. NUM (number) is prominent in PKT which is likely a reflection of its news domain. Absence of the SCONJ in GKT is due to the tokenization that does not analyze particles as separate tokens. Notably, AUX (auxiliary)[4] and PART (particle)[5], which were entirely lacking in the original GKT , were partially introduced into the revised GKT as the result of tokenization of symbols and punctuation marks as discussed in Section 3.2..

The frequencies of dependency labels in the three corpora are shown in Table 4. The distributions of the dependency labels display intriguing trends across all treebanks (Figure 8). PKT and KTB appear consistent except in `compound`, `nummod`, `dislocated` and `nsubj`. As briefly mentioned, `compound` and `nummod` are likely domain-specific particularities. As for `dislocated` and `nsubj`, the discussion of 5.3. likely explains the discrepancy. GKT's abundant annotation of `flat` is a remnant of coarse tokenization that led to embedded tokens labeled `flat` as a whole.

6.2. Discussion

GKT While a number of salient errors has been handled in this work, our analysis show that there are a number of remaining issues with GKT that we strongly recommend be addressed in a future release of the data. The errors include structural problems, incorrect argument attachment, and

[4] All verbs were uniformly categorized as VERB in the original GKT. Given that auxiliary verb is a well-established category in Korean grammar, we find this a rather puzzling design decision.

[5] Particles were not tokenized in the original GKT.

Tag	Description	GKT	PKT	KTB
ADJ	Adjective	2,760	3,431	14,223
ADP	Adposition	1,791	1,251	1,498
ADV	Adverb	11,361	15,174	49,204
AUX	Auxiliary	74	2,263	12,906
CCONJ	Coordinating Conjunction	223	2,453	19,368
DET	Determiner	573	685	4,824
INTJ	Interjection	3	0	56
NOUN	Noun	32,345	46,866	105,193
NUM	Numeral	847	7,931	4,848
PART	Particle	31	464	268
PRON	Pronoun	682	857	7,712
PROPN	Proper Noun	490	12,257	12,366
PUNCT	Punctuation	10,440	13,428	38,925
SCONJ	Subordinating Conjunction	0	9,780	18,466
SYM	Symbol	328	376	260
VERB	Verb	18,431	13,855	59,273
X	Other	13	970	700
Total		80,392	132,041	350,090

Table 3: Frequencies of part-of-speech tags in the final resulting corpora.

incorrect dependency labelling.[6] Additionally, GKT shows a (mostly) consistent tendency to go with a head-first analysis in cases of conjunction (i.e., *talking* is the direct dependent of *reading* for conjunction *talking and reading*) and noun-noun compounds[7], both of which represent inconsistent treatments of a verb-final language.

Additionally, the GKT currently contains duplicates in the dataset, many of which are fairly complex sentences. Out of the 195 duplicates present in the data (out of total 6,339 sentence tokens), 113 duplicates appear verbatim in both the training and test sets (represents over 11% of the test data) and 28 duplicates cross over training and development sets (represents 3% of the development set), which indicates a flawed data sampling process.

PKT and KTB The conversion and error-analysis for PKT has undergone various iterations and the UDv2 compliant PKT data is now complete. PKT has been praised for its strong annotation consistency; that coupled with well-publicized documentation has enabled a quick and reliable implementation of the targeted conversion strategies.

KTB, our newest converted treebank, is near completion, however, there are still a few lingering issues that require attention. One issue that often came up was the treatment of grammaticalized multi-word expressions such as -ㄹ 것이다 (*-l kesita*) and -ㄹ 수 있다 (*-l swu issta*). On the face of it, they involve dependent nouns 것 (*kes*, 'thing') and 수 (*swu*, 'way') respectively to literal translations of '... will be a thing' and 'there is a way to ...'. On the whole, however, they are grammaticalized forms that encode future/irrealis and epistemic modality, respectively: PKT acknowledges this and marks them as multi-word auxiliaries in annotation which facilitated our conversion process. In KTB, these forms had to be individually and lexically targeted to ensure parallel treatment. The Google Treebank, however, does not make such provision; as a matter of fact, it lacks AUX as a POS category altogether, which means this corpus remains disparate on this issue. This illustrates difficulty in achieving uniformity across multiple corpus resources by way of automatic and semi-automatic conversion.

7. Conclusion

We present the manual assessment and revision process for the GKT, and the phrase-structure to UD conversion of Penn Korean and KAIST treebanks, discussing some of the statistics and the current issues relating the three presented treebanks. To the best of our knowledge, this is the first time that these three Korean corpora are converted together into dependency trees following the latest UD guidelines, resulting in a total of 38K+ dependency trees.

It is our expectation that the compilation of these treebanks will help facilitate further research in dependency parsing in Korean, where the lack of training data has remained an obstacle. Furthermore, we expect that the conversion methodologies described in this paper will serve as helpful resources to those wishing to carry out phrase-structure to dependency conversion for other corpora.

Future directions include further enhancements to the quality of treebanks established in this study and the development of parsers based on this dataset to aid further research in Korean NLP. All our resources including source codes and links to the corpora are provided at: https://github.com/emorynlp/ud-korean.

[6] We suspect these errors were present in the original annotation of the corpus and propagated to the current distribution of CoNLL'17 shared task data.

[7] This is true even in a noun-noun compound where one of the noun explicitly case marked such as "**샐러드 바-를** 먹을 수 있다" (tr. **salad bar-obj** can eat), where *salad* is assigned the head even though *bar* is marked with the accusative case.

Tag	Description	GKT	PKT	KTB
acl	Clausal Modifier of Noun	3,198	1,488	21,468
advcl	Adverbial Clause Modifier	4,515	11,636	20,487
advmod	Adverbial Modifier	8,810	2,964	19,102
amod	Adjectival Modifier	1,566	1,595	16,584
appos	Appositional Modifier	1,544	1,182	1,059
aux	Auxiliary	64	4,807	18,935
case	Case Marking	1,624	1,548	1,343
cc	Coordinating Conjunction	223	785	5,234
ccomp	Clausal Complement	651	9,858	15,655
clf	Classifier	0	0	1
compound	Compound	0	28,908	24,696
conj	Conjunct	3,863	9,960	20,774
cop	Copula	102	418	303
csubj	Clausal Subject	21	8,014	1,202
dep	Unspecified Dependency	2,437	609	3,019
det	Determiner	3,077	685	4,824
discourse	Discourse Element	0	0	47
dislocated	Dislocated Elements	0	0	20,964
expl	Expletive	0	0	0
fixed	Fixed Multiword Expression	13	528	3,186
flat	Flat Multiword Expression	12,252	18	803
goeswith	Goes With	0	0	0
iobj	Indirect Object	108	222	967
list	List	0	0	0
mark	Marker	372	1,003	799
nmod	Nominal Modifier	1,761	5,555	22,045
nsubj	Nominal Subject	8,290	4,012	17,444
nummod	Numeric Modifier	489	154	3,295
obj	Object	5,801	9,823	23,605
obl	Oblique Nominal	2,784	3,357	11,577
orphan	Orphan	0	0	0
parataxis	Parataxis	0	0	0
punct	Punctuation	10,494	13,073	39,016
reparandum	Overridden Disfluency	0	0	0
root	Root	6,332	5,036	27,363
vocative	Vocative	0	0	15
xcomp	Open Clausal Complement	1	4,803	4,278
Total		80,392	132,041	350,090

Table 4: Frequencies of dependency labels in the final resulting corpora.

8. Bibliographical References

Choi, J. D. and Palmer, M. (2011). Statistical Dependency Parsing in Korean: From Corpus Generation To Automatic Parsing. In *Proceedings of IWPT workshop on Statistical Parsing of Morphologically Rich Languages*, SPMRL'11, pages 1–11.

Choi, K.-S., Han, Y. S., Han, Y. G., and Kwon, O. W. (1994). KAIST Tree Bank Project for Korean: Present and Future Development. In *In Proceedings of the International Workshop on Sharable Natural Language Resources*, pages 7–14.

Choi, J. D. (2013). Preparing Korean Data for the Shared Task on Parsing Morphologically Rich Languages. Technical Report 1309.1649, ArXiv.

Han, C.-H., Han, N.-R., Ko, E.-S., Palmer, M., and Yi, H. (2002). Penn Korean Treebank: Development and Evaluation. In *In Proceedings of the 16th Pacific Asia Conference on Language, Information and Computation*, PACLIC'02.

Han, N.-R., Ryu, S., Chae, S.-H., yun Yang, S., Lee, S., and Palmer, M. (2006). Korean Treebank Annotations Version 2.0. https://catalog.ldc.upenn.edu/LDC2006T09.

Hong, Y. (2009). 21st Century Sejong Project Results and Tasks (21세기 세종 계획 사업 성과 및 과제). In *New Korean Life (새국어생활)*. National Institute of Korean Language.

Lee, D.-G. and Rim, H.-C. (2009). Probabilistic Modeling of Korean Morphology. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(5):945–955, July.

McDonald, R., Nivre, J., Quirmbach-Brundage, Y., Goldberg, Y., Das, D., Ganchev, K., Hall, K., Petrov, S., Zhang, H., Täckström, O., Bedini, C., Bertomeu Castelló, N., and Lee, J. (2013). Universal Dependency Annotation for Multilingual Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL'13, pages 92–97.

Nivre, J., Bosco, C., Choi, J., de Marneffe, M.-C., Dozat, T., Farkas, R., Foster, J., Ginter, F., Goldberg, Y., Hajič, J., Kanerva, J., Laippala, V., Lenci, A., Lynn, T., Manning, C., McDonald, R., Missilä, A., Montemagni, S., Petrov, S., Pyysalo, S., Silveira, N., Simi, M., Smith, A., Tsarfaty, R., Vincze, V., and Zeman, D. (2015). Universal Dependencies 1.0. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Park, J. (2017). Universal dependencies for korean: Hani (ver1.0) [data set].

Petrov, S., Das, D., and McDonald, R. (2012). A Universal Part-of-Speech Tagset. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, LREC'12, pages 2089–2096.

Zeman, D., Popel, M., Straka, M., Hajic, J., Nivre, J., Ginter, F., Luotolahti, J., Pyysalo, S., Petrov, S., Potthast, M., Tyers, F., Badmaeva, E., Gokirmak, M., Nedoluzhko, A., Cinkova, S., Hajic jr., J., Hlavacova, J., Kettnerová, V., Uresova, Z., Kanerva, J., Ojala, S., Missilä, A., Manning, C. D., Schuster, S., Reddy, S., Taji, D., Habash, N., Leung, H., de Marneffe, M.-C., Sanguinetti, M., Simi, M., Kanayama, H., dePaiva, V., Droganova, K., Martínez Alonso, H., Çöltekin, c., Sulubacak, U., Uszkoreit, H., Macketanz, V., Burchardt, A., Harris, K., Marheinecke, K., Rehm, G., Kayadelen, T., Attia, M., Elkahky, A., Yu, Z., Pitler, E., Lertpradit, S., Mandl, M., Kirchner, J., Alcalde, H. F., Strnadová, J., Banerjee, E., Manurung, R., Stella, A., Shimada, A., Kwak, S., Mendonca, G., Lando, T., Nitisaroj, R., and Li, J. (2017). CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, CoNLL'17, pages 1–19.