Detecting and Correcting Learner Korean Particle Omission Errors

Ross Israel
Indiana University
raisrael@indiana.edu

Markus Dickinson
Indiana University
md7@indiana.edu

Sun-Hee Lee
Wellesley College
slee6@wellesley.edu

Abstract

We detect errors in Korean post-positional particle usage, focusing on optimizing omission detection, as omissions are the single-biggest factor in particle errors for learners of Korean. We also develop a system for predicting the correct choice of a particle. For omission detection, we model the task largely on English grammatical error detection, but employ Korean-specific features and filters; likewise, output analysis and the omission correction system illustrate how unique properties of Korean, such as the distinct types of particles used, need to be accounted for in adapting the system, thereby moving the field one step closer to robust multi-lingual methods.

1 Introduction

Grammatical error detection is useful to produce an improved final document for writing assistance, provide feedback to language learners, provide features for automatic essay scoring, and post-edit machine translation output (see references in Chodorow et al., 2012, sec. 2). Within this growing field, most of the work has focused on English, but there has been a small community of researchers working on other languages. We continue this trend by advancing the state-of-the-art in detecting errors in Korean particle usage.

Expanding to other languages and language families obviously presents new challenges, such as being able to handle word segmentation and greater morphological complexity (e.g., Basque (de Ilarraza et al., 2008), Korean (Lee et al., 2012), Hungarian (Dickinson and Ledbetter, 2012), Japanese (Mizumoto et al., 2011)); greater varieties of word order (Czech (Hana et al., 2010), German (Boyd, 2012)); case ending errors (Czech, German, Hungarian); differing definitions of function words (Korean, Japanese, Basque); and so forth. Investing in methods which apply across languages will make techniques more robust and applicable for even more languages.

An additional challenge for many of these languages is the lack of resources. Much previous work on detecting errors in Korean, for example, focused less on techniques and more on acquiring training data (Dickinson et al., 2011) and evaluation data (Lee et al., 2012). We thus desire techniques that work using smaller and/or unannotated data sets that may be less reliable than some of the corpora for better-resourced languages.

We focus on detecting errors in the presence or absence of Korean postpositional particles. Korean is a Less Commonly Taught Language (LCTL) needing proficient speakers and more pedagogical research (see Dickinson et al., 2008, sec. 2), making computational tools for Korean language learning important. Particles are used to mark properties akin to prepositions and also to case markers, as discussed in section 2. This makes our task applicable to similar languages like Japanese and more generally to agglutinative languages like Basque, Hungarian, and Turkish, as discussed in section 3 on related work. Particles are our focus because of the high prevalence of particle errors in learner data, accounting for 20–30% of learner errors (section 2).

One of the most frequent errors relating to particles is not using them when required (section 4); thus, simply detecting whether a particle is necessary can pinpoint nearly half the particle errors language learners make. In the interest of extending methods to new languages, we develop an omission error detection system rooted in work on

English preposition error detection (section 5), accounting for Korean-specific properties in the features and filtering of results (section 6). We then see the impact of such error detection on predicting the specific omitted particles (section 7).

We make the following contributions in this paper: 1) We present a functional Korean particle omission error detection system, adapted from previous English preposition work but tailored towards Korean in its morpheme-based approach and its novel features. 2) We outline system mistakes, highlighting unique properties of Korean, and point towards how to fix them. 3) We provide an error correction system—incorporating new discourse-based features and optimized separately from the first-stage classifier—which corrects a high percentage of omission errors. In so doing, we also discover that accounting for distinctions in types of Korean particles opens the door to further improvements. The overall lesson is that work from English can be adapted, but only if incorporating the nuances of the new language.

2 Korean Particles

Korean postpositional particles are units that appear after a nominal to indicate different linguistic functions, including grammatical functions, e.g., subject and object; semantic roles; and discourse functions. In (1), for instance, 가 (*ka*) marks the subject (function) and agent (semantic role).[1]

(1) 그래서 제**가** 한국말을 열심히 배우고
thus I-SBJ Korean-OBJ hard learn
싶어요
want
'Thus, I really want to learn Korean'

Similar to English prepositions, particles can also have modifier functions, adding meanings of time, location, instrument, possession, and so forth. For further discussion of Korean particles, see, e.g., chapter 3 of Yeon and Brown (2011).

Learner Errors Particle errors are very frequent for Korean language learners, accounting for 28% of beginner errors in one corpus study (Ko et al., 2004). In (2a), for instance, a learner omitted a subject particle after the word 것 (*kes*,'thing'). The error has beencorrected in (2b).

(2) a. 각 곳에 여러 좋은 것 있어요
each place-AT many good thing exist

b. 각 곳에 여러 좋은 것**이**
each place-AT many good thing-SBJ
있어요
exist
'There are many good things'

[1]Examples come from the learner corpus; see section 4.2.

3 Related Work

While there is much related work on detecting preposition and article errors in English, e.g., the 2012 Helping Our Own (HOO) shared task (Dale et al., 2012), we will focus here on work on detecting errors in functional items in agglutinative languages (Korean, Japanese, Basque), as we most directly build from this. Roughly, agglutinative languages here are ones which "glue" syntactic categories, in the form of affixes, onto a word.

For Korean particle error detection, Dickinson and Lee (2009) train two parser models, one with particles included and one without, to compare mismatches. Their main purpose is to adapt treebank annotation to be more particle-aware, and they did not evaluate on real learner data.

We build more directly from Dickinson et al. (2011), who build web corpora of Korean in order to train machine learning models for particle prediction. They obtain 81.6% accuracy for particle presence. While the work is similar, comparing the current work to their results is problematic for a number of reasons. First, the work in Dickinson et al. (2011) was very preliminary, focusing on acquiring training data, and did not examine different levels of learners. Also, they used a different learner corpus with different annotation guidelines (see comparison in Lee et al., 2012), along with training data that was specifically tailored for the domains in the test corpus. Finally, for particle presence, they focus on overall system accuracy, rather than error detection, making direct comparison of results difficult (cf. Chodorow et al., 2012).

There has been more work in the comparable language of Japanese, which we review briefly. To begin with, Oyama (2010) uses a basic SVM model trained on well-formed Japanese to detect particle errors, focusing on eight different case particles and finding that the particle frequency distribution in the training corpus affects accuracy, ultimately evaluating on 200 learner particle instances of a single particle (*wo*).

Mizumoto et al. (2011) use statistical machine translation (SMT) techniques to detect and correct all errors within Japanese, using a "parallel" corpus of ill-formed and correctly-formed Japanese, based on correction logs from a collaborative language learning website. Our paradigm is much different, basing our method only on a correct model of the target language, given a relative lack of corrected data available in Korean and other lesser-resourced languages. We are, however, able to use some correction logs for building confusion sets (section 7.1). Imamura et al. (2012) correct Japanese particle errors using an approach similar to SMT ones, relying on a corpus of generated errors to learn a model of alignment to correct forms. We could explore generated errors in the future, but rely only on a model of correct Korean here.

Suzuki and Toutanova (2006) predict case markers in Japanese for an MT system, basing their techniques on semantic role labeling. They predict 18 case particles, a subset of all Japanese particles. They use a two-stage classifier, first identifying whether case is needed and then assigning the particular case ending, training the second classifier only on instances where a case marker was required. This breakdown and parts of their feature sets are similar to ours, but: a) they use (gold standard) parse features and treat the problem as one of predicting markers for *phrases*; and b) they correct machine errors, while we correct learner errors, allowing us to investigate methods such as using learner-based filters.

Turning to Basque, de Ilarraza et al. (2008) detect errors in five complex postpositions, where the postposition itself has a suffix, by developing 30 constraint grammar rules which use morphological, syntactic, and semantic information. While the rule-based system can work well, we pursue a strategy which incorporates different types of linguistic information through contextual features.

4 Data

4.1 Training Data: Collecting Web Data

In order to control the data for domain specificity, we follow the recommendations laid out in Dickinson et al. (2010) and extended in Dickinson et al. (2011). Namely, we use data collected from the web using search terms based on topics likely to be discussed in a learner corpus, in order to find semantically-relevant instances. This data is passed through an encoding filter to ensure that at least 90% of any document retrieved is written using Hangul (the Korean writing system). The resultant corpus is over 23 million words.

4.2 Testing Data: A Learner Korean Corpus

For testing data, we use a corpus of learner Korean (Lee et al., 2012, 2013) featuring 100 error-annotated essays from learners evenly split into four different categories: beginning (B) vs. intermediate (I) learners, and foreign (F) vs. heritage (H) learners, where *heritage* refers to learners who had Korean spoken at home.[2] We split the corpus into development and test sets by taking ≈20% of each subcorpus for development, and using the rest as testing. Table 1 gives the numbers of sentences, tokens, nouns, particles, total errors, and omission errors in the development and testing sets.

	Sen.	Tok.	Noun	Part.	Err.	Om.
Dev	331	2673	955	849	103	51
Test	1079	8987	3266	2872	430	234
All	1410	11660	4221	3721	533	285

Table 1: Annotated corpus statistics (*sen*tences, *tok*ens, *noun*s, *part*icles, *err*ors, *om*issions)

Particle errors are marked as omissions, insertions (comissions), or substitutions, in a multi-layered framework. Spacing and spelling errors are corrected before the target form and correct segmentation are marked, segmentation being necessary since nouns and particles are written as a single orthographic unit. For our experiments, we use the correctly-spelled layer, mitigating the effect of spelling errors for testing an error detection system, as done for English (e.g., Tetreault and Chodorow, 2008; Chodorow et al., 2007).

All particles (erroneous or correct) are labeled as to their function (e.g., locative), allowing us to group particles into categories, to see how classifier performance differs. Figure 1 provides the four groups we consider (cf. tables 5 and 6). Additionally, some nominals require multiple particles in sequence (*Seq.*), and some of the annotations allow for particles from more than one category as a correct answer, i.e., a set of correct answers (*Set*).

4.3 Learner Error Analysis

Lee et al. (2009) annotate another corpus of learner Korean, divided using the same four-way split among learner level and type as the corpus described in section 4.2. We examine this corpus

[2]The corpus will be publicly released at: `http://cl.indiana.edu/~kolla/`.

Category	Example Functions
Structural Case	subject, object, genitive
Inherent Case	time, location, goal, etc.
Auxiliary	auxiliary, topic
Conjunction	conjunction

Figure 1: Particle Categories

to get a sense of the types of errors that learners of Korean make in essays. In this corpus, omission errors, i.e., instances where the learner has mistakenly omitted a particle, make up the biggest propotion of the errors (47.6%). The next most common are replacement errors, where the learner has used the wrong particle (44.6%). Comission errors—using a particle where none is necessary—make up the remainder of the errors (7.8%).

5 Approach

Particles have a range of functions, including case marking and preposition-like functions, but, since they are a closed class of functional elements, we can adapt techniques from English for other closed class functional items, namely prepositions and articles, to detect errors in usage.

We view the task of detecting and correcting errors as two steps (cf. Gamon et al., 2008). The first step is a binary choice that only involves determining whether or not a particle is required, a so-called *presence* (yes/no) classifier. The second classifier, the particle *choice* classifier, attempts to guess the best particle, once it has been established that a particle is needed. We actually treat the first step as a particle omission detection system because the expected rate of errors of comission is so low, and thus we specify that the classifier cannot reject a particle that is already present. Comission errors may require their own system.

We utilize the omission classifier as it nicely performs two functions. First, because it posits instances requiring particles, it also filters out instances that do not need a particle to be grammatical. Thus, the particle choice classifier does not need to include *NULL* as a possible class, cutting down on training size and complexity. Secondly, many errors can be found at this stage, as a lot of errors stem from learners omitting necessary particles (see section 4.3). Nearly half of the learner errors could be detected with an accurate omission particle detection system at this step. Thus, this classifier can provide useful feedback to learners, especially higher-level ones who may know the correct particle once its omission is highlighted.

6 Particle Omission Error Detection

We describe the particle presence classifier here, treating it as a task of particle omission detection. Any particle a learner uses is passed on, while we posit where a particle should have been used.

6.1 CRF Classifier

Conditional Random Fields (CRFs) have been utilized in a variety of NLP tasks in the last few years, and have been used recently for leaner error detection tasks, especially those which can be seen as sequence labeling tasks (e.g., Israel et al., 2012; Tajiri et al., 2012; Imamura et al., 2012). We use the comma error detection work in Israel et al. (2012) as a basis, and employ CRF++[3] to set up a binary classifier at this step based on 1.5 million instances from our web corpus. Here we consider all nominals, as annotated in the corpus, as possible candidates for particle insertion. When we derive features based on POS tags (section 6.2), however, we rely on an automatic POS tagger.

6.2 Features

The feature set for particle omission detection is mainly composed of words and POS tags in the surrounding context, where tags are derived from a POS tagger (Han and Palmer, 2004). We use a five-word sliding window, processing each token in the document, although only nominals are possible candidates for particle insertion. The five-word window includes the target word and two words on either side for context; the feature set, with examples, is given in table 2.

We break all words into their root and a string of affixes, each with its own POS tag (or tags, for multiple affixes) to better handle the morphological complexity of Korean and avoid sparsity issues. Particles are removed when extracting affixes, so as not to include what we are trying to guess. For the text and POS of the root, we use unigram, bigram, and trigram features, as shown in the table; for the affixes, we use only unigrams. We also have a feature (*combo*) for each root that combines the text and POS into a single string.

In addition to these adjacency-based features, we also encode the previous and following nouns

[3]`http://crfpp.googlecode.com/svn/trunk/doc/index.html`

position	Unigrams					Next		Prev		# of nouns passed	# of nouns remain
	text		POS		combo	Noun	Pred	Noun	Pred		
	Root	Affix	Root	Affix							
Target	것	NONE	NNX	NONE	것_NNX	NONE	있	곳	좋	1	0
Word$_{-1}$	좋	은	VJ	EAN	좋_VJ	NONE	NONE	것	좋	1	1
Word$_{-2}$	여러	NONE	DAN	NONE	여러_DAN	것	좋	곳	NONE	1	1
Word$_{+1}$	있	어요	VJ	EFN	있_VJ	NONE	NONE	것	좋	2	0
Word$_{+2}$	.	NONE	SFN	NONE	._SFN	NONE	NONE	것	있	2	0

Bigrams - text				Bigrams - POS			
W$_{-2}$+W$_{-1}$	W$_{-1}$+T	T+W$_{+1}$	W$_{+1}$+W$_{+2}$	W$_{-2}$+W$_{-1}$	W$_{-1}$+T	T+W$_{+1}$	W$_{+1}$+W$_{+2}$
여러+좋	좋+것	것+있	있+.	DAN+VJ	VJ+NNX	NNX+VJ	VJ+SFN

Trigrams - text			Trigrams - POS		
W$_{-2}$+W$_{-1}$+T	W$_{-1}$+T+W$_{+1}$	T+W$_{+1}$+W$_{+2}$	W$_{-2}$+W$_{-1}$+T	W$_{-1}$+T+W$_{+1}$	T+W$_{+1}$+W$_{+2}$
여러+좋+것	좋+것+있	것+있+.	DAN+VJ+NNX	VJ+NNX+VJ	NNX+VJ+SFN

Table 2: Features and examples for 것 in (2b) - '각 곳에 여러 좋은 것이 있어요'

and predicates, to approximate syntactic parent features. The *predicates* can be verbs, adjectives that function like verbs in Korean, and auxiliary verbs. Finally, we use two features to encode the amount of nouns that have already occured in the sentence, as well as how many still remain. The usage of topic particles, for instance, relies in part on knowing where in the sentence a noun occurs, with respect to other nouns.

6.3 Filtering

Because learners are more often correct than erroneous in their usage of particles, we want to ensure that the output of classifier does not predict errors in too many instances. To this end, we have built a filter into the classifier. For these errors of omission, we check how confident the classifier is in its answer and only posit omission errors if the classifier's confidence is above a certain threshold. Tuning on the development corpus (section 6.4), we tried a variety of thresholds, in a hill-climbing approach, and found 85% to be the best.

6.4 Results

For all results in this paper, we follow the recommendations from Chodorow et al. (2012). We evaluate by comparing the writer, annotator, and system's answer for each instance; true positives (TP), for example, are cases where the annotator (gold standard) and system agree, but the writer (learner) disagrees. In our case, positives are cases where the system posits a particle while the learner did not. We count only instances of nominals without particles in the writer's data, as these are the only ones which could have omission errors. Along with precision (P), recall (R), and an F-score ($F_{0.5}$), we provide the number of errors (n), true positives (TP), true negatives (TN), false postives (FP), and false negatives (FN), for the sake of clarity and future comparison. As a baseline, we use the majority class, i.e., guessing a particle for every nominal in the corpus.

Table 3 provides the results for particle omission detection on our development corpus. Here we present the baseline, the results based only on the classifier's decision (no filter), and the results for the best filter. We use precision-weighted $F_{0.5}$ rather than the traditional F_1 because precision is more important than recall for most error detection applications. As the 85% threshold results in the best $F_{0.5}$, we use this system on the test data.

Table 4 provides the results for particle omission detection broken down by subcorpus. The FB (foreign beginner) subcorpus has the worst performance, most likely due to their language being most distant from the well-formed Korean of the training corpus, as well as the most distant from the development set. Overall, however, the system has a solid 84.9% precision on all test subcorpora.

6.5 Analysis

In looking over some FPs, i.e., cases where the system predicted a particle not in the gold standard, we discovered that some of these cases involved the optionality of particles. For example, in (3), the system posits a particle after 사람들 (*salamtul*, 'people'). This is a case of a nominal being used in a genitive fashion, and so a genitive particle could be used here, but it is not required. In some sense, the system rightly points to particle usage being *licensed* in this setting. However, the corpus annotation only marks particles that are *necessary* for grammaticality (Lee et al., 2013). Fully teasing apart particle licensing from particle

	n	TN	TP	FP	FN	Precision	Recall	$F_{0.5}$
Dev baseline	51	0	51	98	0	34.23	100.00	39.41
Test baseline	233	0	233	391	0	37.34	100.00	42.69
Dev no filter	51	64	43	34	8	55.84	84.31	59.89
Dev 85% filter	51	90	27	8	24	77.14	52.94	70.68
Test 85% filter	233	373	101	18	132	84.87	43.35	71.23

Table 3: Particle omission error detection results

	n	TN	TP	FP	FN	Precision	Recall	$F_{0.5}$
FI	66	153	33	6	33	84.62	50.00	74.32
FB	51	45	18	5	33	78.26	35.29	62.94
HB	53	46	20	4	33	83.33	37.74	67.11
HI	63	129	30	3	33	90.91	47.62	76.92

Table 4: Particle omission error detection results by learner type (test data)

requirement requires more thorough discussion of when particle dropping is permitted.

(3) 특히 외국 **사람들** 눈에는 더욱
particularly foreign **people** eye more
그렇습니다.
is-so.

'In particular, it is thus for the eyes of foreign people.'

Other cases do not license particles, but the nominals still have particle-like functions. In (4), for instance, the nominal phrase 이 때 (*i ttay*, 'this time') carries a temporal meaning—much like that conveyed in the temporal particle 에 (*ey*), but no particle is allowed here, because the function is more like an adverb (cf. *today* in English).

(4) **이 때** 너무 감정에 치우치지
this time too feeling-at give-way-to
않도록 주의하어야 해.
don't pay-attention-to must.

'This time, you must pay attention to not giving way to feeling.'

Regarding false negatives, i.e., cases where we do not posit a particle when we should, one major problem we observe involves noun-verb and noun-noun sequences. If a learner views a noun and a following word like a compound, it conceals the fact that the noun requires a particle. For instance, in (5) (learner-omitted particles in curly brackets), the word 성격 (*sengkyek*, 'personality') needs a subject particle, but it forms a compound with 좋 (*choh*, 'good'), obscuring the noun's role.

(5) 성격{**이**} 좋은 아{가}
personality{**-SBJ**} good-REL kid{-SBJ}
태어날 때 환경이 나쁘다면 ...
born time environment-SBJ bad-if ...

'When a child who has good personality is born, if the environment is bad ...'

Another complication is the variability of particle requirements due to minor changes in the amount of information presented, for example, the addition of one prepositional phrase changing whether a particle is necessary or not. Combined with misclassifications resulting from segmentation errors from the POS tagger, it seems like the false negative set can be reduced with better linguistic preprocessing fed into the system.

7 Particle Choice for Omission Errors

Once we have established that there is a missing particle, the next step is to select the best particle to be placed in the given context. Thus, we send all instances classified as missing a particle to a second classifier that makes this selection.

7.1 Confusion Set for Particle Omission

The scope of the training data selected, i.e. what particles should be allowed to be guessed by the classifier, is a significant decision at this stage. There are hundreds of particles in the Korean language, but many of these are not used often, e.g., 9 particles cover 70% of particle use in a data set of thesis abstracts and 32 cover 95% in a study by Kang (2002). Thus, the training data should only include particles which can reasonably be expected to appear when the learner has omitted one. Utilizing similar methodology to Mizumoto et al. (2011), we build a confusion set from data collected from the language learing and social networking website, Lang-8.[4]

To build the set, we searched the user-edited versions of the essays for any word corrected by appending text resembling a particle. Due to the somewhat ambiguous nature of particles with respect to other morphemes and root endings, we cannot be certain that all of these edits are in fact particles, but can be confident that a majority are.

After compiling all possible insertion candidates, we prune the list by requiring a particle's frequency to be at least 10% of the most frequent particle. For example, if 가 appears 100 times as the most frequently inserted particle, any particle appearing less than 10 times would be removed.

7.2 TiMBL

For this task, we use memory-based learning, namely TiMBL (Daelemans and van den Bosch, 2005). The nearest neighbor algorithm is desirable as training data is sparse, and there are a variety of possible classes to choose from. After filtering the web-corpus to only include instances based on the confusion set extracted from the Lang-8 data, we have 5.7 million instances for training.

7.3 Features

For the particle selection system, we build upon the particle omission detection features (cf. section 6): we use unigrams, bigrams, and trigrams of the words and POS tags, a combination word+POS unigram, the previous and following verbs and nouns, and the count of nouns passed and remaining in the sentence. We only use nominals as targets for instances, using a five-word window for context. Some of the n-gram features with high numbers of possible values are less helpful, and we remove them, namely the unigram features for the two words farthest from the target, as well as the bigrams that do not include the target.

We then extend this information by adding features, some of which provide discourse information. 1) Knowing if there is already a subject, object, or topic particle in the sentence often means that there should not be another of the same type used; thus, we add binary features encoding if any of these have occured yet. 2) We also add binary features relating to the usage of the target word in the previous sentence, encoding if the target was marked as the topic, subject, or object, or if it was in the previous sentence at all. 3) A numeric feature is used that tracks how far along we are in the sentence, based on the idea that certain particles, e.g. subjects, are more likely to occur earlier in the sentence, whereas others, e.g. objects, occur later. 4) Finally, we include the previous particle used by the learner, again because some particles are not likely to be reused in a sentence.

7.4 Results and Analysis

Here we present the results for the selection classifier in terms of the accuracy of the classifier on choosing the best particle for an instance already defined as erroneous. By the definition of this task—selecting the correct particle for an *error*—there are no FNs or TNs. Thus, recall is rather meaningless, and accuracy and precision reduce to the same metric ($\frac{TP}{TP+FP}$). Additionally, as mentioned in section 4.2, the particles in the test corpus can be grouped into different categories, and we provide results broken down by category and subcorpus. Instances that require a sequence of multiple particles to be correct (*Seq.*) are not currently handled, but we leave them in the results for clarity and completeness. FPs from the error detection step are also included, although the system clearly cannot select a correct particle for them.

Table 5 shows the performance of the selection classifier on the instances identified as omission errors by the binary classifier (i.e., TPs and FPs identified by the *pipeline*). Overall, this classifier selects the correct particle 52.9% (63/119) of the time in the test data when presented with instances from the previous classifier.

	Dev	Test	FI	FB	HB	HI
Str.	17/20	56/80	16/28	11/15	14/15	15/22
Inh.	1/2	5/8	1/2	1/1	1/2	2/3
Aux.	0/1	1/3	0/1	0/0	0/1	1/1
Cnj.	0/1	0/0	0/0	0/0	0/0	0/0
Set	0/2	1/4	0/1	0/1	1/1	0/1
Seq.	0/1	0/6	0/1	0/1	0/1	0/3
FPs	0/8	0/18	0/6	0/5	0/4	0/3
Total	18/35	63/119	17/39	12/23	16/24	18/33
%	51.4	52.9	43.6	52.2	66.7	54.5

Table 5: Results for particle selection on instances from binary omission classifier (*pipeline*)

Table 6 provides the results for testing on all instances with omission errors (based on the *gold*

[4] `http://lang-8.com`

standard), i.e., including the FN instances from the binary omission classifier mistakenly marked as correct, but not FPs. For all corpora combined, the classifier selects the best particle 58.4% (136/233) of the time in the test data. The overall accuracy gleaned from Tables 5 and 6 is encouraging as we move forward, as it means that the classifier performs reasonably well on cases where it has a chance of selecting the best particle in both the *pipeline* and *gold* experimental environments.

	Dev	Test	FI	FB	HB	HI
Str.	23/29	112/164	34/51	17/31	29/36	32/46
Inh.	3/8	11/30	1/7	2/7	3/6	5/10
Aux.	3/6	10/16	1/3	2/4	5/7	2/2
Cnj.	0/1	0/0	0/0	0/0	0/0	0/0
Set	0/3	3/7	1/2	0/1	2/2	0/2
Seq.	0/4	0/16	0/3	0/8	0/2	0/3
Total	29/51	136/233	37/66	21/51	39/53	39/63
%	56.9	58.4	56.1	41.1	73.6	61.9

Table 6: Results for particle selection on all instances of particle omission (*gold*)

7.5 Further Restricting the Task

It is clear from the number of errors for each particle category that structural particles are the type most often omitted by learners of Korean, accounting for 68% (193/284) of omission errors in all subcorpora combined. Based on this finding, we ran a set of experiments in which we trained a classifier to only insert structural case particles.

	pipeline		*gold*	
	Dev	Test	Dev	Test
Str.	17/20	67/80	23/29	133/164
%	85	83.8	79.3	81.1
Total	17/35	67/119	23/51	133/233
%	48.6	56.3	45.1	57.1

Table 7: Results for particle selection using a structural case-only classifier

This classifier actually performs better than when restricting selection to the particles from the confusion set for the *pipeline* experiment setting (cf. Table 5, 56% > 52%), though there is a slight drop in performance as compared to the confusion set classifier in the *gold* experiments (cf. Table 6, 57% < 58%). In both cases, however, there are significant gains made when only examining structural particles; this classifier correctly identifies the best particle over 80% of the time in both the *pipeline* and *gold* test settings. These results show the potential in handling specific linguistic types of particles in Korean differently.

8 Conclusion and Outlook

We have presented a system for detecting and correcting learner Korean particle omission errors. We used a two-stage pipeline utilizing CRFs to make a binary decision as to whether or not a nominal without a particle should be followed by a particle, followed by a memory-based learner to select the best particle in the case of an omission. The binary classifier performs with 85% precision and 44% recall in the testing data, for an $F_{0.5}$-score of 71%; these results could lead to a useful error detection tool for learners and/or teachers. The selection classifier is also fairly accurate, choosing the best particle close to 60% of the time to correct omission errors. These results compare favorably with English preposition and determiner error correction work (cf. Dale et al., 2012), though those results involve all error types, not just omissions.

Our experiments for the selection task using specific particle types indicate that constraining the set of particles for a given context helps greatly. We saw improvement in choice accuracy by using only structural case particles to train a classifier for selecting structural case. This encouraging result can help direct research moving forward. One could build a classifier to identify what category of particle is most likely for a given context after determining a particle is missing and before sending it to a final selection classifier.

Finally, as we improve the omission detection/correction pipeline, the next logical step for building a tool for more robust grammatical error detection is to take on errors of substitution and commission. The lessons learned here from particle choice, using a feature set that incorporates dialog-based features and constraining the set of particles that can be selected for a given context, should prove particularly useful for the substitution task. Just as we have seen that structural case particles are the most likely to be dropped, we may be able to find patterns for what types of particles can be substituted or over-used by learners. Confusion sets for the types of errors made by learners (cf., e.g., Rozovskaya and Roth, 2010) should be even more useful for substitution errors.

References

Adriane Boyd. 2012. *Detecting and Diagnosing Grammatical Errors for Beginning Learners of German: From Learner Corpus Annotation to Constraint Satisfaction Problems*. Ph.D. thesis, The Ohio State University.

Martin Chodorow, Markus Dickinson, Ross Israel, and Joel Tetreault. 2012. Problems in evaluating grammatical error detection systems. In *Proceedings of COLING-12*.

Martin Chodorow, Joel Tetreault, and Na-Rae Han. 2007. Detection of grammatical errors involving prepositions. In *Proceedings of the 4th ACL-SIGSEM Workshop on Prepositions*, pages 25–30. Prague.

Walter Daelemans and Antal van den Bosch. 2005. *Memory-Based Language Processing*. Studies in Natural Language Processing. Cambridge University Press, Cambridge, UK.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 54–62. Montréal.

Arantza Díaz de Ilarraza, Koldo Gojenola, and Maite Oronoz. 2008. Detecting erroneous uses of complex postpositions in an agglutinative language. In *Proceedings of COLING-08*. Manchester.

Markus Dickinson, Soojeong Eom, Yunkyoung Kang, Chong Min Lee, and Rebecca Sachs. 2008. A balancing act: How can intelligent computer-generated feedback be provided in learner-to-learner interactions. *Computer Assisted Language Learning*, 21(5):369–382.

Markus Dickinson, Ross Israel, and Sun-Hee Lee. 2010. Building a Korean web corpus for analyzing learner language. In *Proceedings of the 6th Workshop on the Web as Corpus (WAC-6)*. Los Angeles.

Markus Dickinson, Ross Israel, and Sun-Hee Lee. 2011. Developing methodology for Korean particle error detection. In *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 81–86. Portland, OR.

Markus Dickinson and Scott Ledbetter. 2012. Annotating errors in a Hungarian learner corpus. In *Proceedings of LREC 2012*. Istanbul.

Markus Dickinson and Chong Min Lee. 2009. Modifying corpus annotation to support the analysis of learner language. *CALICO Journal*, 26(3).

Michael Gamon, Jianfeng Gao, Chris Brockett, Alexander Klementiev, William Dolan, Dmitriy Belenko, and Lucy Vanderwende. 2008. Using contextual speller techniques and language modeling for esl error correction. In *Proceedings of IJCNLP-08*. Hyderabad, India.

Chung-Hye Han and Martha Palmer. 2004. A morphological tagger for korean: Statistical tagging combined with corpus-based morphological rule application. *Machine Translation*, 18(4):275–297.

Jirka Hana, Alexandr Rosen, Svatava Škodová, and Barbora Štindlová. 2010. Error-tagged learner corpus of Czech. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 11–19. Uppsala, Sweden.

Kenji Imamura, Kuniko Saito, Kugatsu Sadamitsu, and Hitoshi Nishikawa. 2012. Grammar error correction using pseudo-error sentences and domain adaptation. In *Proceedings of ACL-12 - Volume 2*, ACL '12, pages 388–392. Stroudsburg, PA, USA.

Ross Israel, Joel Tetreault, and Martin Chodorow. 2012. Correcting comma errors in learner essays, and restoring commas in newswire text. In *Proceedings of NAACL-HLT 2012*.

Seung-Shik Kang. 2002. *Korean Morpheme Analysis and Information Retrieval (in Korean)*. Hongrung Publishing Company.

S. Ko, M. Kim, J. Kim, S. Seo, H. Chung, and S. Han. 2004. *An analysis of Korean learner corpora and errors*. Hanguk Publishing Co.

Sun-Hee Lee, Markus Dickinson, and Ross Israel. 2012. Developing learner corpus annotation for Korean particle errors. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 129–133. Jeju, Republic of Korea.

Sun-Hee Lee, Markus Dickinson, and Ross Israel. 2013. Challenges in annotating korean particle errors. In *20 years of learner corpus research: looking back, moving ahead (LCR 2011)*.

Sun-Hee Lee, Seok Bae Jang, and Jae-Young Song. 2009. Particle errors in an annotated Korean learner corpus - a comparative analysis of heritage learners & non-heritage learners-. In *Proceedings of Annual Conference of American Association of Teachers of Korean*. Seattle, USA.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining Revision Log of Language Learning SNS for Automated Japanese Error Correction of Second Language Learners. In *Proceedings of IJCNLP-12*, pages 147–155. Chiang Mai, Thailand.

Hiromi Oyama. 2010. Automatic error detection method for japanese particles. *Polyglossia*, 18.

Alla Rozovskaya and Dan Roth. 2010. Generating confusion sets for context-sensitive error correction. In *Proceedings of EMNLP-10*, pages 961–970. Cambridge, MA.

Hisami Suzuki and Kristina Toutanova. 2006. Learning to predict case markers in japanese. In *Proceedings of COLING-ACL-06*, pages 1049–1056. Sydney.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for esl learners using global context. In *Proceedings of ACL-12: Short Papers - Volume 2*, ACL '12, pages 198–202. Stroudsburg, PA, USA.

Joel Tetreault and Martin Chodorow. 2008. The ups and downs of preposition error detection in ESL writing. In *Proceedings of COLING-08*. Manchester.

Jaehoon Yeon and Lucien Brown. 2011. *Korean: A Comprehensive Grammar*. Routledge, New York.