BUILDING A KOREAN PARTICLE ERROR DETECTION SYSTEM FROM THE GROUND UP

Ross Israel

Submitted to the faculty of the University Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of Philosophy

in the Department of Linguistics,

Indiana University

December 2014

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Doctoral Committee

Markus Dickinson, Ph.D.

Sandra Kübler, Ph.D.

Lawrence Moss, Ph.D.

Joel Tetreault, Ph.D.

April 11, 2014

Acknowledgements

As I begin thinking about who to list here that deserves my gratitude, the very first thought that occurs to me is that I will inevitably leave out some extremely important people. So, to all of those who deserve some thanks but do not see your name in the paragraphs that follow, take solace in the fact that you actually get this acknowledgement before anyone else: Thank you.

I would like to thank Markus Dickinson, who, aside from being just my academic advisor, has worn many other hats in my life since we first crossed paths including teacher, boss, advisor, basketball teammate (and opponent), research partner, friend, and even wedding officiant. I can safely say that there is no way that I can envision having gotten this far along in my studies had I not met Markus when I first arrived at IU in 2007. I would also like to thank my committee: Sandra Kübler, Larry Moss, and Joel Tetreault. I have had the great fortune to have learned from each of them since I started graduate school, whether through coursework, independent study, research assistantships, internships, or otherwise, and have always been indescribably grateful for each of those experiences. I would also like to thank Sun-Hee Lee, who co-authored much of the work that this dissertation is based upon, and has provided invaluable assistance in annotating and analyzing Korean data and providing translations whenever necessary.

I have had a wonderful experience at Indiana University in the Department of Linguistics and would like to thank all of the faculty and staff who have helped me along the way. I was lucky to hold various jobs throughout my studies that have provided funding while I carried out this work. While in Bloomington, I had the pleasure of meeting, befriending, and working with a lot of great people. In particular, Eric Baucom, Jeremy Casto, Stephanie Dickinson, Brian Flaherty, Yu-Yin Hsu, Charles Jochim, Sanja Kapidzic, Levi King, Nicole Miller, Marwa Ragheb, Alex Rudnick and Charese Smiley all made life in Bloomington and at IU much better than it might have been otherwise.

During my graduate studies, I had the amazing opportunities to work at Educational Testing Service and the Information Sciences Institute at USC. I had a great time at ETS and would especially like to thank Joel Tetreault and Martin Chodorow, who served as my mentors during my internship. Also the rest of the NLP team at ETS, especially Nitin Madnani, Michael Heilman, Chris Brew, Dan Blanchard, Diane Napolitano, and Derrick Higgins, was a pleasure to work with and learn from. I had the great honor to work with Jerry Hobbs and Ekaterina Ovchinnikova at ISI during the final stretch of this work and would like to thank them both for imparting immeasurable knowledge and experience along the way. Thanks also to Suzanne Wertheim, Vladimir Zaytsev, Fabrizio Morbini and Niloofar Montazeri.

I'd like to thank all of my friends who have stood by me through all of this. Phil Green in particular deserves my gratitude for all of the assistance he has provided in writing code that has helped in my coursework, as well as this dissertation. Thank you to everyone in my family who has been in my corner in every possible way since I started this endeavor. My parents and brother have been tremendously supportive of all of my academic pursuits for as long as I can remember. Thank you for always being there whether I needed encouragement, distraction, inspiration, or motivation. Finally, to Melissa Roemmele — completing this degree was the

second most important thing I did during my time at IU; meeting you was the first.

Thank you for everything.

Ross Israel

Building a Korean Particle Error Detection System from the Ground up

This dissertation describes an approach to automatically detecting and correcting grammatical errors in text produced by Korean language learners. Specifically, we focus on Korean particles, which have a range of functions including case marking and indicate properties similar to English prepositions. There are two main goals for this research: to establish reliable data sources that can serve as a foundation for Korean language learning research endeavors, and to develop an accurate error detection system. The machine learning-based system is built to detect errors of particle omission and substitution, then to select the best particle to produce grammatical output. The resources and results outlined in this work should prove useful in aiding other researchers working on Korean error detection and in moving the field one step closer to robust multi-lingual methods.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

If we knew what it was we were doing, it would not be called research, would it?'

— ALBERT EINSTEIN

This dissertation describes an approach to detecting and correcting particle errors in the writing of learners of Korean. 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]. The goal of this research is to automatically detect and correct particle errors in written learner language.

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

Grammatical error detection is an important, yet under-explored subfield of Natural Language Processing (NLP) (Leacock et al., 2010). Automatically finding errors in text is useful in a variety of contexts. Some of the most common applications include producing an improved final document for writing assistance (e.g., Chodorow et al., 2010; Hirst and Budanitsky, 2005; Kukich, 1992), providing feedback to language learners (e.g., Meurers, 2013; Heift and Schulze, 2007), providing features for automatic essay scoring(e.g., Yannakoudakis et al., 2011; Attali and Burstein, 2006; Burstein et al., 2003; Lonsdale and Strong-Krause, 2003) and post-editing machine translation output (e.g., Peng and Araki, 2005a; Knight and Chander, 1994). 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.

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), Hungarian (Dickinson

[1]Examples come from the learner corpus described in Chapter 4.

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 (Japanese, Basque); and so forth. An additional challenge for many of these languages is the lack of resources, thus requiring techniques that work using smaller and/or unannotated data sets that may be less reliable than some of the widely-available corpora for better-resourced languages. Performing Korean particle error detection involves dealing with all of the issues outlined above, making designing a system for particle error detection a substantial task on many fronts.

Before describing the system, it is important to establish the utility of such an endeavor. As such, we will begin here by first defining grammatical errors, especially how they pertain to error detection. We will then define and provide examples of what error detection and correction actually entail. Next we will provide some examples of common real-world applications of error detection. Finally in Section 1.3, we will provide an overview of the research described in this thesis.

1.1 Grammatical Error Detection

1.1.1 Grammatical Errors

Before one can begin to assess the usefulness of a grammatical error detection system, it is important to establish a definition of what a grammatical error is. Leacock et al. (2010) describe written grammatical errors as instances of either grammar, usage, or non-mechanical punctuation errors. This definition relies upon the distinction between usage and grammar in Fraser and Hodson (1978). Their description characterizes grammar as being the systemic ways that words and sentences are formed (i.e. morphology and syntax), whereas usage refers to particular ways of speaking and writing that acquire some social status and become the norm among dialect groups. Also relevant here is the canonical distinction between errors and mistakes, i.e. *mistakes* are so-called "performance errors" that both native and non-native speakers can make but are not indicative of corrupted learning of grammar, whereas *errors* are only made by language learners, are systematic, and result from a lack of knowledge about correct usage (Corder, 1967; Ellis, 2008).

While the working definition of grammatical errors laid out above is applicable to both children learning an L1 and those learning a second language, it is the language acquisition of the latter group, whom we will refer to as L2 learners, that is under examination in most cases of grammatical error detection in the field of NLP [2].

1.1.1.1 Error Detection and Correction

Now that we have established what denotes a grammatical error, at least in the confines of this dissertation, we can formulate a specific definition of the task under consideration. Grammatical error detection in general, and this work in particular, is concerned with automatically finding systematic errors in grammar or usage made in the writing of non-native speakers that indicate deficient learning, though mistakes are, by and large, detected at the same time as errors. Correction, then, is concerned only with rectifying such errors.

1.1.2 Applications for Error Detection

It is important to keep in mind that while "error detection" is commonly given as a task itself, the ultimate application of the system being described can be crucial

[2]L2 learners is used as a blanket term here for brevity's sake, though it could be a learner's third, fourth, etc. language that is being scrutinized.

in determining the optimal settings for a system. There are a variety of contexts in which error detection systems are applied; we outline some of the major ones next.

Automatic Grading One of the most commonly known applications of automatic grammatical error detection is for automatic grading systems (Burstein et al., 2003; Lonsdale and Strong-Krause, 2003; Yannakoudakis et al., 2011; Williamson et al., 2012). Automatic grading is the process of using a machine to make judgements about writing quality based on various aspects of grammar and word usage so that a holistic score can be given to a piece of writing (Chodorow et al., 2012). One good example of automatic grading comes from Educational Testing Service (ETS): e-rater® (Attali and Burstein, 2006). E-rater, operational at ETS since 1999, is the automatic scoring engine underlying a variety of ETS products including the online essay evaluation service, Criterion^{SM} (Burstein et al., 2003), as well as in high stakes testing assessments like the Graduate Record Exam (GRE), which includes essays by both native and non-native writers, and the Test of English as a Foreign Language (TOEFL), which is a test of English proficiency targeted towards non-native speakers. E-rater uses a variety of features to assign scores to essays, many of which are dependent upon detecting usage errors of grammatical elements such as prepositions, articles, punctuation, and agreement in addition to features that consider other facets of writing including, word length, vocabulary, organization, development, and so on. The rate of such errors are considered a strong indicator of overall writing proficiency.

Improving writing quality There have been a variety of tools developed for non-native English writers to help them write more fluently. In this context, grammatical error detection can be useful to target specific errors and point them out to the writer in hopes of producing better essays or other documents (Chodorow et al., 2010; Hirst and Budanitsky, 2005; Kukich, 1992). In the short term, the goal of providing this assistance is to improve the current document by highlighting errors and suggesting corrections. In the longer term, the goal is to help writers learn more about the language they are studying so that they can produce higher quality writing, as described next.

Tutoring Systems Next there is the case of intelligent language tutoring systems (ILTS), which are applications for assisting learners in their pursuit of second language study. ILTS have been proposed and/or developed for a variety of languages including German (Heift, 2010, E-tutor), Japanese (Nagata, 2009, Robo-Sensei), Portuguese (Amaral and Meurers, 2011, TAGARELA), Russian (Dickinson and Herring, 2008, Boltun), and Spanish (Hagen, 1999, Spanish for Business Professionals). These types of applications can be extremely useful when they make learners more aware of characteristics in their writing that indicate useful linguistic distinctions in their target language (Amaral and Meurers, 2006). In order to develop such systems, though, there is a need to deal with the inherent problems in analyzing language learner data so as to provide intelligent feedback (Meurers, 2013). A major step in this process is detecting and correcting grammatical errors. By targeting specific errors, a system can make learners more aware of non-native-like aspects in their use of language that require more attention. The feedback model can also take into account what errors are commonly made by the learner and adjust the feedback it provides accordingly.

Improving Machine Translation Grammatical error detection can also be a useful tool in correcting the output of natural language generation (NLG) systems for various applications. Most notably, it can be used to detect errors in machine translation (MT) output (e.g., Knight and Chander, 1994; Peng and Araki, 2005b). Error detection can also serve the MT community by being incorporated in quality metrics to produce translation scores that are as favorable or, in some cases, even better than popular metrics such as BLEU and METEOR when compared to human judgements of MT output (Parton et al., 2011). In these contexts, an NLG system is producing the input to the error detector rather than a writer, but the goal is similar, namely to produce more error-free language.

1.2 Korean as the Language of Focus

There are a number of reasons that Korean serves as the language of study for this dissertation. First of all, as mentioned above, the majority of work on grammatical error detection focuses on the writing of learners of English. Expanding to more languages will not only improve the language learning and research situations for the language under consideration, but will also provide insights about error detection in general that could prove useful to more languages.

Korean exhibits a myriad of features that are not present in Western languages that make it interesting from a linguistic perspective. First of all, Korean words are formed via a rich agglutinative morphology, distinguishing it from the more analytic system of English. Syntactically, Korean is a verb final language, but actually features relatively free word order, due in part to the use of case marking, i.e. words are specifically marked as to their function in the sentence. More detailed linguistic description of Korean is provided in Chapter 3.

Perhaps more important than the linguistic motivation is the fact that Korean is a Less Commonly Taught Language, which points to a need for better pedagogical research (Dickinson et al., 2008). Moreover, the US government has identified Korean as a critical language in need of more proficient speakers (National Virtual Translation Center, 2007). Developing corpora and computational tools will help both of these causes, as they will aid in furthering pedagogical research, language learning, and computational linguistics research.

1.3 The Current Approach

The end goal of this dissertation is to develop an automatic machine learning-based system for Korean particle error detection. This is a task that has not been given any serious attention in NLP literature to this point. As such, a number of preliminary steps have been required to ensure that this novel line of research is based on sound theory and practices.

One of these steps is to properly outline the scope of particle errors—what types of errors occur, what types of particles are involved in errors, how other words interact with particles, etc. To that end, we have been part of a Korean learner corpus particle error annotation project. This annotation process has informed much of the research that would follow. Korean particle error detection has not been investigated to this point, thus non-trivial issues such as what types of errors learners commonly make, what types of errors are most important to correct, how to define these errors, and how to handle the surrounding contexts can best be investigated by ensuring that all of these issues can be answered during the annotation process. Once completed, the annotated corpus serves as a crucial component to the error detection system described in this dissertation: an evaluation set. We put great care into ensuring the annotation's usefulness in an error detection system, while maintaining annotation methodology that allows the corpus to be used in other linguistic research settings. Given that other languages, such as Japanese and Arabic, face some of the same issues (e.g., Hanaoka et al., 2010; Abuhakema et al., 2008), fleshing them out for error annotation and machine learning is useful beyond this one situation.

Another requisite foundational measure for this approach also involves obtaining and processing native-like Korean data. Our approach is to procure a large scale corpus of native-like Korean from the web to serve as training data for classifiers. As we mentioned above, Korean does not benefit from the same amount of large-scale, readily available corpora that well-researched languages like English do. Utilizing the web provides an option that can work to attain large amounts of data for any language with a decently large presence on the internet. We explore using web-as-corpus techniques to build a corpus that is appropriate in terms of learner level, formality, and topic.

We also look into utilizing social network language learning data for error detection tasks. This data is collected from websites that encourage language learners to write essays in a language they are learning that will then be corrected by a native speaker of the language. Such data is valuable as it is produced on a large scale and contains some degree of markup that can be utilized to approximate annotations. With this data, one can build a large scale corpus of errors, attain parallel data for MT-based approaches, and determine confusion sets, e.g. what types of particles are often confused for another by learners.

After developing a base of useful corpora for the task, we can focus on establishing the actual task of particle error detection. Particles have a range of functions, including case marking and preposition-like functions. Like articles and preposition, they are a closed class of functional elements, so we can adapt techniques from

English for other closed class functional items to detect errors in usage. To that end, we develop machine learning-based approaches for particle error detection. Investing in methods which apply across languages will make techniques more robust and applicable for even more languages.

Because of the complex nature of Korean particles, it can be difficult or even foolish to paint the problem with a wide brush. Our approach is to break the task into a series of sub-tasks, in order to ask less involved questions of a classifier for each task. Essentially, we implement a pipeline approach that first determines whether there is an error involving a particle, and then if an erroneous particle is detected, a second classifier is employed to select the best particle for a given context, i.e. to correct the errors found at the previous step (cf. Gamon et al., 2008).

Crucially, the task of error detection is further broken down in the current methodology to treat different types of errors differently. That is, we employ separate classifiers for the tasks of particle omission detection and particle substitution detection. The main reason for utilizing differing classifiers is that the tasks are based on different research questions. In the case of particle omission, the question is whether or not an empty space should actually have a particle. For substitution detection, the question is whether or not an existing particle is correct. Once errors have been identified, we can move on to a separate classifier that attempts to pick the best particle for that context.

The rest of the dissertation is laid out as follows: Chapter 2 looks at relevant literature in the fields of learner corpus development and grammatical error detection. An overview of Korean language, especially particles, is given in Chapter 3. The error annotation scheme and the resultant corpus are described in Chapter 4. Chapter 5 details the methodology we employed to procure our web-based corpora, both the well-formed training corpus and the social networking language learning corpus.

In Chapter 6 we move on to describing the error detection system itself. Chapter 7 details our approach to detecting errors of particle omission. Substitution error detection is explained in Chapter 8. We then lay out our approach to error correction for both omissions and substitutions in Chapter 9. Finally, we conclude in Chapter 10.

Chapter 2

Review of Relevant Literature

Is there anyone so wise as to learn by the experience of others?

— VOLTAIRE

As mentioned in Chapter 1, error detection is a growing field. We begin this chapter by reviewing some of the major works in English error detection, which makes up a bulk of the research on error detection in general. We then move on to discuss non-English error detection, especially for Japanese, which is the most similar language to Korean for which there are various studies available. Finally, we will review relevant research that discusses learner corpora and annotation, as collecting and annotating learner data helps define the task and points out some of the difficulties associated with error detection research.

2.1 Error Detection for English

Much like the field of NLP in general, grammatical error detection has developed such that a bulk of the work has been done on English. There are a variety of reasons for this bias. First of all, English is commonly accepted as a global language with around 400 million native speakers, and estimates of as many as 1 billion non-native speakers (Guo and Beckett, 2007). Thus, there is a deserved amount of interest in developing linguistically aware tools to assist learners and teachers of English. Secondly, there are a vast amount of NLP tools available for processing English data such as part-of-speech (POS) taggers, tokenizers, parsers, etc., especially in regard to other languages (Leacock et al., 2010). Along with these tools, there are a number of learner corpora available for researchers to work with, e.g., ICLE (Granger, 2003b), CLC (Nicholls, 2003), NICT (Izumi et al., 2005), and NUCLE (Dahlmeier et al., 2013). These corpora are extremely valuable resources, as building and annotating corpora can be time and labor intensive endeavors.

All of these factors contribute to a flourishing community of researchers working on error detection in English. The field has developed over the last three decades, starting with early grammar checking tools such as the *Unix Writer's Workbench* (Macdonald et al., 1982), which was based mainly on string matching, and IBM's *Epistle* (Heidorn et al., 1982), which used parsers and linguistic grammars to analyze text. Such rule based approaches, while still in use in most publicly available grammar checking software, have given way, for the most part, to statistical approaches methods amongst the research community, mirroring the general trend in NLP.

In the early 1990s, statistical methods for word sense disambiguation began to take hold, based on the success of the approach of Gale et al. (1992) at AT&T. Soon, principles of using large scale corpora to inform models of regular language use were being applied to problems such as diacritic restoration (Yarowsky, 1994) and homophone spelling error detection (Golding and Roth, 1996). One issue with approaches to this point is that they train only for the contexts of specific confusion sets, so scalability comes into question. Knight and Chander (1994) developed a method for correcting article errors in machine translation output based on a decision tree classifier trained over lexical features to determine the best article for a given context. It was only a slight leap to go from correcting machine learning output to correcting language learner texts using similar methodologies.

Interest in grammatical error detection has grown to the point that there was a shared task focusing on preposition and article error correction for English in 2011 and 2012, featuring different approaches to the same tasks by a variety of research teams (Dale and Kilgarriff, 2010; Dale et al., 2012). More recently, there was a CoNLL shared task on grammatical error correction that evaluated performance on a wider range of errors including prepositions, determiners, noun number, verb form, and subject-verb agreement (Ng et al., 2013).

2.1.1 Preposition & Article Error Detection

Two of the more prominently dealt-with error types for English are prepositions and articles. Examples of preposition errors are given in Figure 2.1[1]. Each example shows an error and a correction, where the corrections deal only with preposition errors, leaving all other errors as is.

(2) a. Erroneous: Yes, I wait __ you.
    b. Corrected: Yes, I wait *for* you.

(3) a. Erroneous: So I go to home.
    b. Corrected: So I go home.

(4) a. Erroneous: Adult give money at New Years day.
    b. Corrected: Adult give money on New Years day.

Figure 2.1: Examples of Preposition Errors

In (2), there is an omission error, where the sentence is missing the preposition *for*. (3) is the opposite situation, where the sentence has an extraneous preposition, *to*. Then in (4), there is a substitution error, as *at* should be *on*. These error types can also be similarly characterized for articles (as well as other types of errors). The examples in Figure 2.1 illustrate some of the properties that make working with mal-formed data difficult. Namely, even if one targets a specific error, there is a significant possibility that the surrounding context will contain other errors. For example, in (4), there is an agreement error between the subject, *Adult*, and the verb, *give*, along with the preposition error. This means that any approach that utilizes surrounding context of a preposition to help make a decision as to whether or not an error is present could be affected, if it learns from well-formed data. Parsers and other NLP tools can also be impacted by such errors. For further dicussion of how

[1]These examples are taken from Han et al. (2010)

errors can affect *downstream* processing, see, e.g. King and Dickinson (2013).

Despite the difficulties present in the tasks, preposition and article error correction are two of the more approachable problems in the field of error correction, in part because the error typology is easy to define; the three error types in Figure 2.1 represent the most common ways that prepositions can be misused, though, spelling errors are a different issue that can concern prepositions and articles. More importantly, both prepositions and articles are closed sets of word classes, meaning that there is a finite set of types of each. Other word choice errors, verbs for example, are much more difficult to pin down due to the abundant and in fact growing set of verbs available. For approaches that use statistical methods, a large training corpus will have copious amounts of preposition and article contexts, as these function words play an important role in English grammar; any sentence more than a few words long is likely to have examples of both.

As prepositions and articles present a problem are among the most frequently occuring error types in many corpus studies (Leacock et al., 2010), and the tasks are relatively easier to define than for other errors, there has been a good deal of work done looking specifically at these error types. Prepositions and articles are very relevant to Korean particles, as they have a great deal of similarity. Particles are also a closed set, and perform some of the same grammatical functions (a detailed look at Korean particles is provided in Chapter 3). Given these similarities, we provide a review of relevant literature on English article and preposition error detection/correction. We focus especially on machine learning-based approaches, as that is the method which we will employ for detecting particle errors in this research (see Section 6). It is important to note here that while we will provide evaluation metric scores for many of these approaches, the research reported here was, by and large, performed on differing data sets, making comparison between systems on the same task difficult.

Izumi et al. (2004) use a maximum entropy classifier trained on a small set of error annotated learner data to detect errors in, among other things, prepositions and articles. The classifier is trained on words and POS information in five word window with two words on either side of a possible error. They achieve 68% precision on both articles and prepositions, with 29% recall for articles and 16% recall for prepositions. The use of a maximum entropy classifier as well as the feature set would serve as a starting point for many of the research that would come later that dealt with similar types of errors.

Han et al. (2004, 2006) describe an approach to detecting article errors using a maximum entropy model trained on well-formed text. The classifier is tasked with selecting from *a/an*, *the*, and *null* as the best article for a given NP. The system achieves 83% accuracy when tested on well-formed text, and about 90% precision and 40% recall on non-native learner essays. The system produced 85% agreement with human annotators for the binary task of whether or not there should be an article for the context.

Lee and Seneff (2006) describe a system for article prediction based on parse ranking, using a statistical parser. Training on over ten thousand transcripts of flight reservation data, they build a lattice where articles, prepositions, and auxiliaries are first removed, then allowed to be inserted between any two words. Testing on a small set of sentences from the same domain, they achieve 86% precision and 76% recall. It is worth keeping in mind that the system is not tested on actual learner data, however, the lattice approach did allow for nouns and verbs to be replaced by any inflectional variant to approximate noise in learner contexts.

Tetreault and Chodorow (2008, 2009) build upon earlier work in Chodorow et al. (2007) and adapt the approach from Han et al. (2006) to work specifically with prepositions, rather than articles. The authors extract 25 features to predict the correct preposition, including features capturing the lexical and grammatical context (e.g., the words and POS tags in a two-word window around the preposition) and features capturing various relevant selectional properties (e.g., the head verb and noun of the preceding VP and NP). They use a set of 34 prepositions, and have a large training corpus, around 7 million instances, made up of well-formed English from news sources. Instances with other errors in the context around the preposition are skipped, as classifier performance would be poor because the classifier is trained on well-formed text. The approach involves post-processing filters on top of the classifier that prevent certain contexts from being flagged as errors. In the case of errors of commission, they ignore the classifier and use only the heuristic filters, as these are far less frequent and it is difficult to build a single classifier with high precision for omissions, substitutions, and commissions. The system achieves about 84% precision and 19% recall on a test set of learner essays.

De Felice and Pulman (2009) train a maximum entropy model on British National Corpus (BNC) data to test on both well-formed text as well as English Language Learner (ELL) data from the Cambridge Learner Corpus (CLC) for preposition error detection on a set of nine common prepositions. Their approach is largely similar to that of Tetreault and Chodorow (2008), though they add syntactic information from CCG parses of the data, as well as named entity recognition. One of the main contributions of this work is their examination of the effectiveness of NLP preprocessing (POS tagging and parsing) on learner data and how it relates to the accuracy of an error detection system based on native-like data. They find that misspellings and grammatical malformations in the surrounding context of prepositions can lead to false positives (i.e. marking well-formed language as erroneous) by the classifier.

Lee and Knutsson (2008) use memory-based learning, with 10 million sentences in their training set to attempt to guess the best preposition for a given context, considering 10 common prepositions in English. Differing from a majority of the work done on grammatical error detection, the authors utilize a set of syntactic features obtained from a statistical parser. The testing is done only on altered Wall Street Journal text, rather than on learner data, so the surrounding contexts are always well-formed English. As such, the results are not really comparable to error detection efforts carried out on authentic data.

Gamon et al. (2008) describes an approach to detecting and correcting preposition and article errors in non-native text utilizing decision trees trained on native text and a language model (LM) trained on Gigaword, a large-scale corpus of English containing over four billion words. There are two classifiers for each phenomenon: a presence classifier to determine whether or not a preposition or article is necessary, and a choice classifier that selects the best one for a given context. The language model is then used to filter out unnecessary changes by requiring the output of the classifiers, known as the selection provider, to have a higher LM probability than the original input. Finally, an example provider queries the web for the chosen correction to provide learners with good examples of suggested output. Gamon (2010) extends this approach by combining the output of the SP and the LM as features for a meta-classifier trained on error annotated data. The meta-classifier improves on the previous results and achieves precision of about 77% and 85% for articles and prepositions, respectively.

Han et al. (2010) also work on detecting and correcting error in English prepositions in a learner corpus. They train a maximum entropy classifier on a large-scale error tagged corpus of learner English, distinguishing it from most previous statistical approaches that trained on well-formed text. They compare the results of the system training on the annotated learner data vs. using a well-formed corpus and find that the former achieves precision more than 10% higher than the latter, while recall is also slightly better for the error-trained version. The authors point out, though, that the training corpus is made up entirely of 10-16 year old native Korean speakers, and the test set is an unseen portion of the same data set. Thus, scalability and the performance of the classifier on learners with different language backgrounds are lingering issues.

Using article errors as a test platform, Rozovskaya and Roth (2010c) extend beyond the typical methodology of training on well-formed data by artificially inserting errors into otherwise well-formed text. They generate errors in number of ways, including randomly replacing correct articles with errors and by learning error patterns from language learner texts and altering random training instances to match the distributions. The results show favorable error reduction rates for the altered training data scenarios versus "clean" training scenarios for learners from various L1s, with the biggest gains coming for Chinese and Russian learners of English.

In work from the same year, Rozovskaya and Roth (2010a) consider preposition error detection and examine improving performance by restricting the confusion sets for each preposition. Rather than train a classifier that allows all possible prepositions under consideration in a study as a correct answer (cf., e.g. Tetreault and Chodorow, 2008), they enforce L1-dependent confusion sets for each preposition. The results show that restricting the confusion sets greatly improves system precision.

The same authors, Rozovskaya and Roth (2011) experiment on both articles and preposition errors and compare four linear learning models: Averaged Perceptron (AP), Language Model (LM), Naïve Bayes (NB), and a "counting method", SumLM. They test on the same corpus presented in Rozovskaya and Roth (2010b) and train using *WikiNYT*, a selection of Wikipedia and New York Times articles from the Gigaword corpus, and the Google Web1T 5-gram corpus. The results show that the AP generalizes the best of the four models. They also present an approach to adapt trained models to a language learner's L1 based on the probability of the error by learners of that language. This method allows for models trained on native data to be adapted for testing on data written by learners of any language for which error distributions are known.

Dahlmeier and Ng (2011) propose a method for preposition and article error detection and correction based on Alternating Structure Optimization (ASO). The approach involves combining information from native text with that of error-annotated data to form a unified model. ASO performs the target task, i.e. error correction, and an auxiliary task, i.e. article/preposition selection in native text, by minimizing errors on the two tasks during training. They evaluate on NUCLE, explained in detail in Section 2.3.1. The results show that ASO produces the best results overall, beating baseline experiments and two commercial grammar checkers for this task.

2.1.2 Other English Error Detection

While preposition and article error detection have accounted for a great deal of the research done for English error detection, there has still been a good deal of work done on other phenomena. We present here an overview of some of the recent work on less commonly researched errors.

2.1.2.1 Punctuation

There have been two system described recently designed to deal with specific punctuation error detection. Israel et al. (2012) describe a comma error detection system. Beginning with developing an annotation scheme and annotating a corpus of learner essays, they then used conditional random fields (CRFs) to build models of correct comma usage and apply that to learner data. The feature set includes a set of lexical and POS features in a five word window, as well as features that encode the distance of the target from the beginning and end of the sentence, and to the previous and following coordinating conjunction. Because comma errors are relatively uncommon, even in learner writing, with respect to correct usage, the authors implemented a high confidence filter on the system to mark any comma as erroneous. They achieved 94% precision and 32% recall on non-native data, and 85% precision and 20% recall on native student data. They also apply the system to the similar task of comma restoration (i.e. inserting commas into comma-less text) and achieved 85% precision and 66% recall on section 23 of the Wall Street Journal, beating established baselines (cf. Shieber and Tao, 2003; Gravano et al., 2009).

Cahill et al. (2013a) focus on hypen omission errors, while noting that hyphens, while less common than other types of errors, do occur in and affect the quality of learner essays. The authors used a maximum entropy classifier trained on a variety of different training corpora: 1) news text, 2) hyphen-error text automatically retrieved from Wikipedia edit history, and 3) a combination of both. The features were derived from local context of the position between two words (i.e. where a hypen could occur). They also utilized the Collins dictionary to indicate whether the two surrounding words occur as a hyphenated entry, and added a feature that encoded the proportion of the occurrences of the word pair as hyphenated or not in Wikipedia. The system achieves 99% precision and 41% recall for hyphen omission error detection/correction on a corpus of learner essays.

2.1.2.2 Tense-Aspect

A relatively understudied grammatical phenomenon in the context of automatic error correction is that of tense and aspect. Tajiri et al. (2012) take the problem on, working on finding such errors in the writing of Japanese learners of English. An example of a verb tense error is given in (5)[2], where the verb *go* is in the wrong tense in the second sentence. Correcting such errors is difficult for computational approaches because knowing that *went* is a better correction than *will go* is dependent upon knowing that the second sentence refers to a past event, which can only be understood based on the first sentence.

(5) I had a great time this summer. First I *go to Kaiyukan with my friends.

The authors set up a sequence labeling task using conditional random fields to label each verb phrase with tense/aspect based on the surrounding context. Testing on a corpus with 16,308 verb phrases, of which 1072 contain a tense/aspect error, the authors report precision-recall curves where precision never reaches above 40%, and recall stays below 20%. However, the CRF-based system does perform better than support vector machines (SVM) and maximum entropy-based systems using the same feature set. The system is able to find erroneous sentences with and F1-measure of about 70%.

[2]Example taken from the paper; Kaiyukan is an aquarium in Osaka, Japan.

2.1.2.3 Holistic views of Sentences

A handful of studies have opted to consider entire sentences as the focus for error detection/correction, rather than specifying some particular phenomena as the target for a system. Gamon (2011) is one such study that uses high order sequence models to detect the presence of any error in a sentence. In this approach, sentences are processed with a maximum entropy Markov model (MEMM) similar to one that would be used for POS tagging or named entity recognition (NER). The MEMM incorporates a variety of features including language model information, string-based features, and linguistic analysis features. The idea underlying the method is that ungrammatical sentences should produce relatively lower scores than grammatical ones.

Madnani et al. (2012) takes a different approach to holistically correcting erroneous sentences. They apply a "round-trip" machine translation (RTMT) technique to correcting entire sentences of learner English. RTMT typically involves translating from the target language to the learner's native language, then back to the target language to leverage the error correction built in to MT systems. The authors extend this technique by using multiple pivot languages, based on the idea that with several options to choose from, there should be a better chance of getting useful translations back. The system combines the RTMT for eight languages into a lattice and uses a combination of language model scores, weight edge scores, and other factors to produce a corrected sentence that may contain parts of multiple translations. The system improves 36% out of 200 sentences presented, and negatively impacts 31%.

2.1.2.4 Collocation Errors

Another interesting task in the field of grammatical error detection is that of finding errors in the use of collocations. Collocations are linguistic constructions made up of multiple words that function as a single idea. For an explanation of how difficult it can be to define exactly what constitutes a collocation see Leacock et al. (2010). For the purposes of this discussion, it is enough to say that collocations are combinations of words that convey a single meaning and that the meaning of the whole is often not the meaning of the parts. Chang et al. (2008) describe an approach to detecting and correcting errors in the English writings of L1 Chinese speakers. First, verb+noun collocations are extracted from the text and looked up in a list of over 630,000 collocations collected from the BNC. If a collocation is not found in the list, it is marked as an error. At this phase, the system produces 98% precision and 91% recall. To correct the collocations, the authors used a combination of bilingual dictionaries and parallel corpora to generate confusion sets and replace words in the collocation. The candidate corrections are ranked according to their occurrence in the collocation list. The correction list most often contains an acceptable correction in the first or second entry, with an mean reciprical rank value of 0.66. These results are quite good for this task and certainly when compared to other error detection scenarios. However, as the approach is dependent on building L1 specific resources, it faces major scalability problems.

2.1.2.5 Spelling

Spelling mistakes often make error detection of other types, e.g. articles and prepositions, more difficult, as any approach that involves looking at context is subjected to noise in the form of spelling errors. As such, language learner spell checking has developed as a specialized field because a) learners make different kinds of spelling mistakes than native speakers (Boyd, 2009), and b) spelling errors make other tasks including POS tagging, parsing, and detecting other grammatical errors more difficult (Sakaguchi et al., 2012).

Most spell checkers operate on a number of assumptions that are unsound for dealing with learner data. First of all, most spell checkers are built to aggressively search for simple mistakes, such that the correction usually is not greater than one edit (insertion, deletion, substitution, or transposition) away from the erroneous word, and do not expect to find >1-edit distances mistakes. Words that contain multiple edits are accordingly penalized in the ranking of suggestions for a misspelled word. This behavior is likely due to the commonly held assumption that 80-95% of mistakes are of the can be corrected by making only one edit, but research in second language studies has shown that this is often not the case with non-native writing (Damerau, 1964; Pollock and Zamora, 1984; Rimrott and Heift, 2005). However, Flor and Futagi (2012) report that their system for non-word spelling errors, ConSpel, though not designed specifically for L2 spell-checking, performs about the same on native and non-native text. This is likely due to the fact that the system was built to look for misspellings with an edit distance of more than one.

One of the major complicating issues for spell checking is dealing with "real-word" errors, that is, errors where a writer substitutes one real word for another (Islam and Inkpen, 2009). For non-word errors, detection usually involves checking a dictionary or corpus for the word, and if it is not there, the correction component comes in and generates a list of best suggestions, often based on Levenshtein Distance. For real word errors, detection and correction involve examining the context around a word, and essentially second guessing the author as to what word they meant to use (Hovermale and Mehay, 2009).

2.1.3 Parsing ill-formed input

One area of research that is an important part of the story of English grammatical error detection is that of parsing ill-formed input. Most parsers are built to handle native-like, grammatical input, so they often fail to produce accurate parses when presented with language learner input. Proofreading tools often involve using parsers that have some way of generating parses for sentences with grammatical flaws, such as subject-verb disagreement (Leacock et al., 2010). Along with being error-tolerant, these tools must also have some mechanism for indicating that a parse does not match standard grammatical constraints for the language. This line of research is often undertaken in the interest of providing an error analysis, rather than correcting a form. That is, it is more useful as a tool for diagnosing grammatical inaccuracies than rectifying them.

There are a variety of ways that one can go about adding grammatical error tolerance to computational grammars; we'll cover some of the most popular here. Dini and Malnati (1993) describe an approach that involves over-generating parse trees and ranking them in order of the least amount of grammatical constraints that must be violated to generate a parse. Heinecke et al. (1998) add rules that rank certain constraints, including some rules that may not be broken to limit the number of possible parses. Then there is the idea of adding "mal-rules" to grammars that allow parsing of specific errors, i.e. enabling a parser to spot specific grammatical errors such as missing articles (cf. Schneider and McCoy, 1998; Suppes et al., 2012). Another method is altering the parse unification process so that if two elements would not unify under normal constraints, unification can be allowed but with a penalized score for the parse (cf. Hagen, 1995; Fouvry, 2003). There is also a method of "parse-fitting" that calls for a top-down parser to run after a bottom-up parser has failed and attempts to find pieces of the parse to "fit" together (cf. Jensen et al., 1983).

2.2 Non-English Error Detection

Having discussed approaches to error detection in English, we turn now to non-English error detection. As stated in the Introduction, research on the task of Korean particle error detection has not been reported that we know of. However, there has been some similar work done on other languages. We review the most relevant of these here.

2.2.1 Japanese Error Detection

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 the current approach, but: a) they use (gold standard) parse features and treat the problem as one of predicting markers for *phrases*; b) because they can rely on parse features, the ignore potentially useful discourse features (cf. our current feature set in Section 8.3); and c) they correct machine errors, as opposed to learner errors, whereas we collect, annotate, and test on two distinct learner corpora (see Chapter 4 and Section 5.7). Working on learner language, which contains other errors, and without gold standard parses would likely mean a significant degradation in performance.

There has recently been a growing amount of error detection research for Japanese, which exhibits many similarities with Korean. Oyama (2010) uses a basic SVM model trained on well-formed Japanese to detect particle errors. The author focuses on eight different case particles and finds that the particle frequency distribution in the training corpus affects accuracy, i.e., the well-formed training corpus does not match the distribution of particles used by learners without some sampling. Running the system on unseen text from a newspaper corpus also used for training the system, the results were best for the particles used most frequently in the corpus in a particle selection task.The main learner corpus evaluation is done on 200 learner particle instances of a single particle (*wo*, the object marker).

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 social networking language learning website, Lang-8[3]. The translation model for the approach is built using the parallel corpus of erroneous and corrected sentences extracted from Lang-8, a monolingual corpus of correct Japanese is used to build the language model, and the task is to *translate* erroneous Japanese phrases into grammatical versions. The results show promise (though, by the authors' admission, there is room for improvement in their evaluation practices), however, this system looks to correct entire phrases, and therefore does not provide the granularity of looking at a specific phenomenon, e.g. particles. While correcting entire phrases or sentences may often correct particle errors themselves, without annotation for specific phenomena, one cannot be sure what types of errors the system handles best. This

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

lack of error knowledge in the system also inhibits the system's ability to provide meaningful feedback to learners about exactly what grammatical phenomena are the source of their errors.

Imamura et al. (2012) correct Japanese particle errors using an approach similar to SMT ones, correcting erroneous phrases and looking at resultant particle changes. They test on real learner data made by asking Chinese learners of Japanese to translate English Linux manuals into Japanese that was then annotated by native speakers. The approach relies on a corpus of generated psuedo-errors based on confusion matrices of each particle under consideration inserted into Japanese Linux manuals. The proposed system is based on domain adaptation where the psuedo-error corpus is the source domain and the real error corpus is the target domain; the feature set is optimized by weighting features that appear consistently between the source and target. The results show that the proposed system performs better than baseline experiments, but precision is, at best, around 85% and drops to around 60% when recall reaches 10%.

2.2.2 Basque Error Detection

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. They obtain 67% precision and 65% recall on all postpositions in their test data, but there were only 39 errors to be detected in their 2590 sentences. Despite the promise shown in the rule-based approaches in Eeg-Olofsson and Knutsson (2003) and de Ilarraza et al. (2008), the field has shifted towards statistical systems, a trend which we will follow with the current work.

2.2.3 Swedish Error Detection

Eeg-Olofsson and Knutsson (2003) report on a preposition error detector for Swedish. The authors annotate preposition errors in 140 essays written by learners of Swedish. Rather than the utilizing a data-driven approach like most of the error detection work that would follow it, the system is rule-based. Based on preposition POS tags assigned by a statistical trigram tagger, the authors composed 31 rules for specific preposition errors. For more on the usefulness of heuristic-based approaches for morphologically complex languages, see Bick (1998) for work on automatic analysis of Portuguese text.

2.2.4 German Error Detection

For German, Boyd (2012) detects errors of selection, agreement, and word order. The grammar checker described in the work, Fledgling, utilizes constraint-based dependency parsing and detects errors using contraint relaxation with a general-purpose conflict detection algorithm. The system is developed and tested on authentic German language learner data that is presented in the same work. Fledgling correctly judges grammaticality in 80% of sentences and is up to 91% accurate in diagnosing error types.

2.3 Annotating Learner Corpora

An obvious, yet non-trivial need for any error detection/correction research effort is an evaluation set. One option is to use well-formed data, but introduce errors by corrupting the correct text (see, e.g. Izumi et al., 2003; Wagner et al., 2007; Lee and Seneff, 2008; Rozovskaya and Roth, 2010c). This approach, while perhaps less costly to implement, is less desirable than using an actual corpus of learner language that contains real examples of errors in their natural context. The issue with using a learner corpus is that it must be annotated, and though there has been work done on automatically annotating data (Rosen et al., 2013), for the most part, this means that human annotators will need to annotate the data by hand. This is by no means a small task, and great care should be taken on the part of corpus developers to create annotations that will prove useful to the community at large and present information in a concise, meaningful way so that the corpus can be used effectively.

Annotation for a learner corpus is often much richer than that of other annotated corpora. A typical corpus of native language might include annotations for POS and some kind of syntactic scheme. Learner corpora that have been annotated for errors may feature rich annotation of linguistic phenomena that are often not noted in native language annotation schemes (Hana et al., 2010). While native annotation schemes are typically focused on POS tags and/or syntactic annotation, learner corpus annotation will usually look at specific phenomena, e.g. prepositions, articles, subject-verb agreement, etc. that can provide valuable linguistic insight into language use. As such, an annotated learner corpus can be an invaluable resource not only for NLP researchers working on error detection, but for second language acquisition (SLA) research as well. If an annotated learner corpus is large enough, it can also be a good resource for training NLP tools to better handle language learner data (Meurers, 2009). Learner corpora can be utilized to optimize language learning classes as well as resources for learners such as dictionaries and textbooks. They can also bring to light significant differences about learners of different backgrounds that might not be obvious otherwise.

One of the major contributions of this dissertation is the development of an annotation scheme for a corpus of learners of Korean (see Chapter 4). As such, a review of other work regarding annotating learner data is directly relevant here.

2.3.1 Annotated English Learner Corpora

The Cambridge Learner Corpus (CLC) described in Nicholls (2003) is, to our knowledge the largest annotated learner corpus for any language, with over 6 million essays annotated with a wide range of error types. The corpus handles approximately 80 error types with five main classifications of errors that can be applied to nine word types, along with a set of errors that do not relate directly to the main classifications. The annotation scheme is flat, forcing annotators to make a decision about the *best* correction for a given error. Unfortunately, most of the corpus is unavailable to the public. However, Yannakoudakis et al. (2011) released a small subset of the data, just over 1200 essays, as part of an effort to promote automatic assessment of English as second or other language essays.

Rozovskaya and Roth (2010b) annotate portions of two unannotated learner corpora: the International Corpus of Learner English (ICLE Granger, 2003b) and the Chinese Learner English Corpus (CLEC Gui and Yang, 2003). Both sets contain essays by learners of English. The resultant annotated corpus contains over 4000 preposition annotations, 8.4% of which are errors. The annotations focus on preposition confusions among the 10 most frequently used prepositions in English.

Dahlmeier et al. (2013) describes the National University of Singapore Corpus of Learner English (NUCLE). NUCLE contains over one million words with 27 categories of annotations for learner errors. The corpus is made up of take home assignments for English classes in undergraduate courses. The error rate is low — 3.82 per 100 words, suggesting that the writers of the essays are relatively high proficiency level learners. Annotators used an in house tool to highlight erroneous sections of the text and provide an error tag and suggested correction. The authors note that while annotators were given the option to provide multiple annotations for a single error, i.e. provide multiple tags/corrections, the occurrence of such annotations are infrequent. To our knowledge, as of now NUCLE is the largest freely available learner corpus, making it a valuable resource for English error detection and corpus studies.

The aforementioned English learner corpora have been the most often utilized in recent NLP work dealing with English grammatical error detection, at least among work done on publicly available learner data. For a more exhaustive list of corpora, see `http://www.uclouvain.be/en-cecl-lcworld.html`.

2.3.2 Non-English Learner Corpora

Moving away from English means that corpus developers will have a range of different challenges to deal with. From high-level issues such as complex morphology, phonotactics, freer word order, and other linguistic phenomena, to low-level issues such as character encoding and tokenization, non-English languages often come with a unique set of hills to climb in order to provide thorough annotation. A number of corpora have become available that take these challenges head on in order to provide useful learner data for a variety of languages.

Most directly relevant to the current research is Lee et al. (2009a), who describe a corpus of learner Korean annotated for particle errors. They use a hand-annotated corpus made up of college student essays that is divided according to student level: beginner or intermediate, and student background: heritage or non-heritage. Heritage learning is explained further in Chapter 4. The authors use six types of particle errors—omission, addition, replacement, malformation, paraphrasing, and spacing. The relavant statistics are provided for each type of error among different learner levels. This corpus provides a wealth of useful information that is crucial to this study, such as types and frequency of errors and differences among learner types. Their work, however, was not targeted specifically towards providing a platform for automatic systems. Accordingly, some modifications to their initial annotation scheme must be effected in order to implement a machine learning-based system for automatic error detection.

Lüdeling et al. (2005) discuss Falko, or *__F__ehler__a__nnotiertes __L__erner__ko__rpus* (“error-annotated learner corpus”), made up of German learner data. Falko utilizes multiple independent layers of annotation to address two key issues for learner annotation: 1) That annotations may contain multiple interpretations for correcting a single error, and 2) it should be possible to mark sequences of tokens as “error exponents”, i.e. errors need not be tied to a single token. Falko is annotated using EXMARaLDA (Schmidt, 2005), which allows for multiple independent layers to be encoded.

The Error-Annotated German Learner Corpus (EAGLE) corpus is described inBoyd (2010). The corpus is made up of workbook activities and essays by students in introductory college level German courses. The annotation scheme handles word form, selection, agreement, word order, and punctuation errors. The annotation scheme follows Lüdeling et al. (2005) in implementing a multi-layered approach using EXMARaLDA. The layers are: location, which marks which words are affected; description, which encodes the error type; and target, which provides the corrected version of the error. The annotations also include error *numbering*, which refers to the order in which each error occurs, as there are often multiple errors in a sentence, and therefore multiple corrections based on which error is handled first.

Hana et al. (2010) describe an error-annotated corpus of learner Czech. Czech is a rich language with a host of features that are often difficult for foreign learners such as inflection, agreement, derivation, and unique constituent ordering. The corpus is divided according to learner L1, covers a range of learner levels, and includes both spoken and written data. The complexity of the annotations necessary to capture all of the relevant information about errors in a highly inflectional language like Czech necessitates a multi-layered approach, similar to that in Lüdeling et al. (2005).

Dickinson and Ledbetter (2012) annotate a learner corpus of Hungarian. The corpus is made up of journal entries of college students across a range of learner levels. The annotation covers a variety of errors in four distinct categories: Character-level (phonological and spelling errors), Morpheme-level (agreement and derivation errors), Relation-level, and Sentence-level. Like Korean, Hungarian features a rich agglutinative morphological system. Thus, a multi-layered approach that focuses on morphemes, rather than words, is necessary to capture the subtleties that can arise from learner errors. The annotation scheme allows for the crucial difference between a learner error, and an adjustment to the final form that must be made to account for a correction of some other error.

Conclusion

In this chapter, we have provided an overview of the field of automatic error detection/correction. We also discussed various learner corpora and annotation schemes. While a great deal of the research in both areas has focused on English, we also discussed work in other languages such as Basque, Czech, Hungarian, German, Japanese, Korean, and Swedish. A lot of this research has served as a source of inspiration for the work described in the coming chapters. Before getting into the actual research carried out, though, we will, in Chapter 3, explain some of the intricacies of Korean language that distinguish it from languages like English, and provide new challenges for annotation and error detection.

Chapter 3

A Brief Overview of Korean Language and Particles

To handle a language skillfully is to practice a kind of evocative sorcery.

— Charles Baudelaire

The research described in this dissertation involves analysis of Korean language data. As such, a working knowledge of Korean is necessary in order to better understand much of the work that is described in the coming chapters. To this end, we present here a brief overview of the Korean language, explaining some of the nuances and subtleties that make working with Korean learner data a challenging and interesting endeavor and distinguish it from similar work done on English. We will especially examine the role and functionality of particles, as they are the focus of the research presented in this dissertation. Much of what follows will be facts about the language; Lee and Ramsey (2000), Sohn (2001), Nam (2005), and Yeon and Brown (2011) served as reference materials for this chapter.

3.1 Current Status and Classification

The Korean language is currently spoken by about 78 million native speakers. The majority of these speakers live in South (48 million) or North Korea (24 million), with smaller numbers living in China, Japan, the United States, and central Asia. It is the official language of both North and South Korea. The varieties of Korean spoken in North and South Korea have diverged somewhat since 1945; note that the descriptions of the language in this dissertation will assume a Seoul (South Korea) dialect, which is the variety of Korean commonly found in textbooks and taught in language classes.

Pinning down the exact origin of Korean, or what language family it belongs to, has proven to be a troublesome task, with no consensus among language historians. However, there are two main theories that prevail: 1) that Korean is a part of the Altaic language family, and 2) that it is closely related to Japanese, which has itself proven difficult to categorize. This difficulty has led some to classify the language as a language isolate, i.e. a language with no genetic ties to other known languages. The theory connecting Korean to Japanese is due in large part to their shared grammatical qualities, though it is worth noting that the langauges are not mutually intelligible (Sohn, 2001). In any case, the language has a number of features that set it apart from most other world languages, particularly from English and other Indo-European languages, as we will see in the rest of this chapter.

3.2 Linguistic Description of Korean

In this section, we will give a concise linguistic description of the Korean language, starting from the smallest units, phonemes, and working all the way up to full sentences.

3.2.1 Phonology

Modern Korean features a base of 19 consonants, many of which have distinct positional allophones at initial, medial, and final positions. There are eight "simple" vowels, and set of 12[1] diphthongs that consist of simple vowel and a glide, either /j/ or /w/. There is another vowel that is not a simple vowel, but distinguishes itself from the other diphthongs in that it consists of a glide from [ɨ] to [i]. There are a host of various assimilations and simplifications that can affect this base of phonemes depending on their positions.

Korean allows syllables with a maximal structure of /CGVC/, where C is a consonant, G is a semi-vowel (glide), and V is a simple vowel. Syllables must consist of one and only one simple vowel, which can then be preceded or followed by

[1]It is noted in Yeon and Brown (2011) that younger speakers have less distinctions, resulting in 10 or 11 of these diphthongs

the other elements. At the morphophonemic level, consonant clusters can appear word finally, but these are not expressed at the phonetic level. At a word boundary, or if the following syllable begins with a consonant, one of the consonants in the cluster must be dropped. If the following morpheme begins with a vowel, the final consonant from the cluster is moved to the onset of the next morpheme. These types of changes are clearly evident in the writing system, which we will discuss in Section 3.2.6.

3.2.2 Morphology

One of the major differences between Korean and more commonly researched languages like English lies in the morphological system. While English morphology tends towards the analytic side of the spectrum, with a fairly low word to morpheme ratio, Korean features a rich agglutinative system of word formation. Agglutinative languages, e.g. Japanese, Basque, Turkish, Korean, form words by stringing together long sequences of morphemes, the smallest meaningful unit of language (cf. e.g. Comrie, 1989). As such, no meaningful linguistic encoding is done by altering a root itself; rather, bound morphemes must be appended to the root if additional information is required. For example, consider (6), where 가시었겠습니 is a single word with five morphemes. The root, 가 (*ka*)m is the verb meaning "go", 시 (-*si*) is an honorific subject marker, 었 (-*ess*) is the past tense morpheme, 겠 (-*kyess*) is the presumptive modal, and finally 습니다 (-*supnita*) is an honorific sentence ending morpheme.

(6) 가-시-었-겠-습니다
ka-si-ess-kyess-supnita
go-HON-PAST-PRESUMPTIVE-HON. ENDING

'(a respectable person) may have gone'

Clearly, the agglutinative system of Korean allows for far more complex word units than languages like English. Furthermore, there are a large number of Korean morphemes that do not have any equivalent words or morphemes in English, such as the honorific addressee marker *-si*.

3.2.3 Parts-of-Speech

Korean classifies words differently from languages like English, which can lead to difficulty for some learners. Traditional Korean grammars often list nine classes of words: nouns, pronouns, verbs, adjectives, *prenouns* (cf. determiners), adverbs, interjections, and particles. It is worth noting that lexical category classification may vary among linguists (for more discussion, see Lee and Ramsey (2000)). Even if these word classes are accepted, there are some significant differences when comparing the definitions to Indo-European languages, particularly in the case of verbs and adjectives. We will now examine some of the most prominent difficulties associated with word classification.

Nouns, pronouns, and numerals, i.e. *substantives*, are often categorized together as they exhibit similar patterns and can be modified using similar morphemes. Significantly, substantives are the point of attachment for particles, the main area of focus for this work. Particles are discussed in detail in Section 3.2.4. Substantives generally pattern similarly to English nouns.

So-called *prenouns* are similar to English determiners for the most part. They are a small closed set of words that take no inflection and occur before nouns in a modifying role. The set includes words with meanings like *this*, *that*, and *other* that align well with English determiners, as well as a few words that are typically considered adjectives in most languages such as *old*, *new*, *ancient*, and *pure*.

One of the most interesting facets of Korean word classes is the similarity of verbs and adjectives. While the term adjective is used mainly to align Korean grammars with Indo-European grammars, what are referred to as adjectives pattern like verbs (see e.g., Kim, 2002, for more on the argument that Korean lacks true adjectives). Based on all of their similarities to verbs, Korean adjectives are sometimes referred to as *descriptive verbs*, with verbs referred to as *action verbs* or *processive verbs*. These parts-of-speech are often categorized together in Korean, as they may both act as predicates. Both processive and descriptive verbs must be combined with inflectional endings; they cannot stand alone.

One important distinction between processive and descriptive verbs is that processive verbs (like English verbs) may have a direct object, whereas descriptive verbs have a simpler argument structure that looks only for a subject or topic. Also, descriptive verbs can be conjugated and moved to directly modify nouns, thus functioning more like English adjectives. For example in (7a)[2], we see 예쁘 (*pretty*) at the end of the sentence acting as a verb marked with the indicative morpheme 다 ('ta'). In (7b), on the other hand, 예쁜 (*pretty*) is directly modifying the noun *woman*, and ㄴ ('n') has been appended, marking the word as an adjective.

(7) a. 저 여자가 예쁘다
that woman-SBJ pretty-IND
'That woman is pretty'

[2]This example borrowed from Kim (2002)

b.

저	예쁜	여자
That	pretty	woman

‘That pretty woman’

3.2.4 Particles

We turn now to particles, the main item under consideration in this dissertation. As we will show, these morphemes hold a position of extreme importance in the language. Korean postpositional particles are morphemes that attach to a preceding nominal to indicate a range of linguistic functions, including grammatical functions, e.g., subject and object; semantic roles; and discourse functions. In (8), for instance, *ka* marks the subject (function) and agent (semantic role). [3]

(8)

Sumi-**ka**	John-**eykey**	sunmwul-**ul**	cwu-pnita
Sumi-SBJ	John-TO	gift-OBJ	give-POLITE ENDING

‘Sumi gives a gift to John.’

Similar to English prepositions, particles can also have modifier functions, adding meanings of time, location, instrument, possession, and so forth, as in (9). In this example, *ul/lul* has multiple uses, marking an object, *ku* (*he*) and the time period, *i sikan* (*two hours*. Also, it is important to note here that in some cases there are multiple allomorphs of a single morpheme that differ phonologically and are selected based on the phonotactics of the surrounding context, e.g. *ul/lul*, *un/nun*, and *i/ka* are variants that depend on whether the preceding phoneme is a vowel or a consonant. In Section 4.3.11, we discuss how to categorize particles.

(9)

Sumi-**ka**	John-**uy**	cip-**eyse**	ku-**lul**	i	sikan-**ul**	kitaly-ess-ta.
Sumi-SBJ	John-GEN	house-LOC	he-OBJ	two	hours-OBJ	wait-PAST-END

‘Sumi waited for John for (the whole) two hours in his house.’

[3]We use the Yale Romanization scheme for writing Korean here to clearly delineate particle boundaries.

There are also particles associated with discourse meanings. For example, in (10) the topic marker *nun* is used to indicate old information or a discourse-salient entity, while the delimiter *to* implies that there is someone else Sumi likes. In this research, we focus on syntactic/semantic particle usage for nominals, planning to extend to other cases in the future, with the expectation that handling discourse particles will bring new complexities and may require a completely different approach.

(10) Sumi-**nun** John-**to** cohahay.
Sumi-TOP John-ALSO like
'Sumi likes John also.'

Due to their complex linguistic properties, particles are one of the most frequent error types among Korean language learners; Ko et al. (2004) report that particle errors were the second most frequent error in a study across different levels of Korean learners, and errors persist across levels (see also Lee et al., 2009a). In (11b), for instance, a learner might replace a subject particle with an object, based on the idea that in English, *book* would be the object of *need*.

(11) a. Sumi-*nun* chayk-*i* philyohay-yo
Sumi-TOP book-SBJ need-POLITE
'Sumi needs a book.'

b. *Sumi-nun chayk-**ul** philyohay-yo
Sumi-TOP book-OBJ need-POLITE

There are also rarer cases where particles follow adverbs and clausal or sentential units. In addition, some particles can be stacked together, in which case they follow a fixed order. In (12), for example, the particles *ey* and *nun* must be used in sequence to convey that there is something of interest *in* the photo, which is the topic of a sentence. A much more detailed overview of particle errors is provided in Chapter 4.

(12) sacin-ey-nun ...
photo-in-TOP ...
'in this photo'

3.2.5 Syntax

The structure of Korean sentences is typically SOV (subject-object-verb), as opposed to English, which is SVO. This means that the subject is usually the first element in the sentence, followed by the object, and finally the verb. Example (13)[4] provides an example of typical Korean SOV ordering, where the the subject, 민수 (*Minsu*) is followed by the object, 김치 (*apple*), and finally the verb.

(13) 민수가 김치를 먹었어요.
Minsu-SBJ apple-OBJ eat.
'Minsu is eating an apple'

Actually, while Korean is most often referred to as SOV, the only real constraint in most scenarios is that the verb occurs last. Moreover, in informal speech, some information may occur after the verb under certain conditions. Because particles mark nouns functions within the sentence, word order is actually somewhat free up to the verb. As can be seen in (14), the verb arguments may all move around each other while retaining grammaticality. While (14a) represents the standard SOV ordering, in (14b), the subject (*Minsu*) is presented after the recipient of the present (*Mina*); in (14b), the object (*present*) is presented first, followed by the subject and the recipient. Examples (14d) and (14e) show cases where the recipient or object can occur after the verb, in the case of informal, spoken Korean. The order is often dependent on discourse properties; older information comes first, and newer

[4]All examples in this subsection taken from Yeon and Brown (2011)

information occurs closer to the head. The writer (or speaker) can decide what information is most important, often dependent upon what is new information, and move that to the front of the sentence.

(14) a. 민수가 미나에게 선물을 주었어요.
Minsu-SBJ Mina-*to* present-OBJ give-PAST-POLITE ENDING.

b. 미나에게 민수가 선물을 주었어요.
Mina-*to* Minsu-SBJ present-OBJ give-PAST-POLITE ENDING.

c. 선물을 민수가 미나에게 주었어요.
present-OBJ Minsu-SBJ Mina-*to* give-PAST-POLITE ENDING.

d. 민수가 선물을 주었어요 미나에게.
Minsu-SBJ present-OBJ give-PAST-POLITE ENDING Mina-*to*.

e. 민수가 미나에게 주었어요 선물을.
Minsu-SBJ Mina-*to* give-PAST-POLITE ENDING present-OBJ.

'Minsu gave Mina a present'

Further complicating the syntactic ordering issue is the fact that arguments may often be dropped. For example (15) is an acceptable sentence made up of one word (*eaten*), which translates to, *I have eaten*. Here, the subject would be understood from surrounding context in the dialog. Objects may be similarly dropped in many cases. Here, the language again differs greatly from most Indo-European languages, where there might be some specific cases that warrant dropping, such as English imperatives where *you* is often dropped, but for the most part, every sentence has a subject, and an added object if the verb is transitive. Even in the case of pro-drop languages, such as Spanish or Italian, Korean differs in that the dropping is licensed due to the explicit marking of syntactic function of the non-dropped nouns in a sentence, rather than by agreement.

(15) 먹었어요
eaten
'(I) have eaten'

3.2.6 Writing System

The writing system most frequently used for Korean text is known as 한글 (*Hangul*). Hangul is categorized as an alphabetic syllabary (Taylor, 1980). It is alphabetic in that every phoneme in the language is assigned a unique character. The characters are then systematically put together in syllabic blocks, which are then used to write words, rather than written in a single horizontal string like in English and other Romance languages. The syllable 한 (*han*), for example, is three separate characters: ㅎ (*h*), ㅏ (*a*), and ㄴ (*n*) all put together. Hangul was developed during the reign of King Sejong, some suggest by the King himself, around 1443 C.E. The writing system was designed to be phonetically intuitive, with the shapes of characters referring to the place and/or manner of articulation. For example, ㄴ (*n* is meant to suggest the shape of the tongue moving towards the teeth to create an alveolar nasal, /n/.

In the past, Korean writing often featured a mix of both Hangul and Chinese characters. In the last few decades, though, the use of Chinese characters in Korea has waned to the point that a majority of publications contain none at all. Along with the decline of Chinese characters, there has been an increase in the amount of Roman characters infused into Korean writing. It is common to find acronyms and abbreviations, e.g. *TV*, as well as entire English words that have been borrowed into the language.

One important facet of the writing system is the use of white space. Unlike other prominent languages from the same region, Korean does utilize white space to delineate words, referred to as *ecels*. However, this was not always the case, as Hangul was originally written with no spaces. Even now, the use of spaces can be controversial, as there is not 100% agreement or consistency across text sources as to what constitutes a word, i.e. where white space should be inserted, and it is possible for less white space to be used in publications where total available space is an issue.

The use of Hangul is a significant issue in the context of this dissertation for two main reasons. First of all, as it is vastly different from most of the world's writing system, it is a potential area of difficulty for learners of Korean. As such, along with grammatical errors, one could expect to see a great deal of spelling, typographical, and spacing errors in a Korean learner corpus. Secondly, the indecisive grammatical conventions associated with spacing, as well as the non-ASCII character encoding, make dealing with Hangul computationally more difficult. Many POS taggers and parsers that are designed to be extensible to other languages given an appropriate training set are, in actuality, poorly equipped to handle the complex segmentation, word order flexibility, and possibility to omit significant grammatical elements that must be considered to accurately POS tag and parse Korean. Dedicated Korean NLP tools are scarce as well, due not only to the aforementioned complex grammatical issues, but also the lack of open source research efforts in comparison to English, so software options are limited.

Conclusion

This chapter has provided an overview of the Korean language, focusing especially on particles and the issues of the language that will prove most important for the research described in the coming chapters. With all of the necessary background material having been discussed, we move forward Chapter 4 to a discussion of KoLLA, a particle error-annotated corpus of leaner Korean.

Chapter 4

Test Data: Developing a Korean Learner Corpus

Beware of the man who works hard to learn something, learns it, and finds himself no wiser than before.

— KURT VONNEGUT, JR.

An obvious, though non-trivial, requirement for error correction is a corpus of errors on which one can test their system. For English there are a number of readily available annotated corpora made up of essays written by learners of the language, e.g. ICLE (Granger, 2004), NICT JLE (Izumi et al., 2005) HOO (Dale and Kilgarriff, 2010). There are even a growing number of annotated learner corpora that can be utilized for error correction work available for other languages, e.g. German (Boyd, 2010), Czech (Hana et al., 2010), and French (Granger, 2003a). This is unfortunately not the case for Korean yet. Thus, procuring a corpus and developing annotation that could be useful for a variety of tasks, especially error detection, has necessarily been a major component of this research. To this end, we have been a part of a larger effort among a group of researchers to build and annotate the KoLLA (Korean Learner Language Annotation) corpus[1].

4.1 A Previously Available Corpus

To our knowledge, the only previous foray into annotating a Korean learner corpus was described in Lee et al. (2009a). While this corpus provided a good starting point, and was used in initial experiments described in this research (i.e. the WaC evaluation experiments in Section 5.5.4), the annotations were not ideal for working with automatic NLP systems. The main issue is that it does not contain gold standard segmentation, thus requiring the user of the corpus to semi-automatically determine the particle boundaries.

In Example (16), taken from Lee et al. (2009a), the corpus marks 한그게서 (*hankukeyse*) as erroneous and 게 (*key*) as the correct particle—in addition to marking

[1]I was active in developing the annotation scheme, checking the annotation for errors, and assisting with the annotators with various technical support. The actual annotation was carried out by native Korean speakers.

this as a replacement error—but it does not indicate exactly where the correct particle goes to produce the correct form, 한그게 (*hankukey*) in this case. That is, the final correct form is not made explicit. Such a lack of transparency can lead to ambiguity for Korean as there are some particles which are more than one character, and in some cases, each of the individual characters can be a singleton particle as well. For example, 이나 (*ina*), 이 (*i*), and 나 (*na*) are all particles, so the segmentation of any word ending with 이나 is crucial. Moreover, there are some cases where the root form of a noun has a character that matches a particle. For example,먹이 (*meki*/'food for animals') could be confused with 먹 (*mek*/'calligraphy ink stick') that has been marked as a subject with 이.

(16) 한그-<u>게서</u> 온 후에 ...
Korea-FROM come-REL after ...
'After I came *from Korea ...'

a. Original: 게서

b. Corrected:게

Another issue with this corpus that leads to difficulties for automatic error detection is that only nominals with erroneous particles are annotated, i.e. nominals with correct particle usage (including no particle) are left unmarked. This decision limits the amount of information available, and in the case of error detection, makes some evaluations impossible. For example, without knowing the number of nominals that correctly have no particles, there is no way to calculate false positives and true negatives for a particle presence prediction task.

4.2 The KoLLA Corpus

In addressing the requirements of an annotated corpus for error correction testing, we worked as part of an annotation effort for a new corpus with similar properties as the one described in Lee et al. (2009a). The corpus consists of 100 essays from university-level learners from four institutions, including Wellesley College, Brigham Young University, the Korean Language Institute at Yonsei University, and the Georgia Institute of Technology. Importantly, the data is divided by level of learner and by heritage status, resulting in a four way split: 25 heritage beginners (HB), 25 heritage intermediates (HI), 25 foreign beginners(FB), and 25 foreign intermediates (FI) (cf. Lee et al., 2009a). The essays are the results of writing assignments from prompts about various topics related to everyday life of college students. The full listing of topics is given in Table 4.1. All of this data was collected by Sun-Hee Lee, an assistant professor in the Department of East Asian Languages and Literatures at Wellesley College.

The term heritage learner refers, in this case, to a language learner who has grown up in a Korean-speaking environment, although they do not necessarily speak the language. In general, their parents use Korean at home and communicate with their children in Korean. Heritage learners are capable of understanding Korean, but they may reply either in Korean or in English. Heritage learners in Korean classrooms typically do not know how to write Korean or have low-level writing skills. This split is interesting for both pedagogical and NLP applications of the corpus.

Recently, with the expanding number of 57 million heritage speakers of different languages in the U.S., there has been a remarkable increase in heritage language research from various parts of theoretical and applied linguistics (Polinsky

Topic	HB	HI	FB	FI
Introducing oneself	✓		✓	
School life	✓	✓	✓	
Hobbies	✓		✓	
Culture	✓			
Shopping	✓			
Korean food	✓			
People I like/Friends	✓	✓		✓
Travel	✓	✓		✓
Learning Korean		✓	✓	✓
Sense of values		✓		
Eliminating stress		✓		
Invasion of privacy		✓		
College entrance exams		✓		
The generation gap		✓		
Dreams		✓		
Happiness		✓		✓
Gatherings/parties		✓		
Life in Korea			✓	
Photographs			✓	
Life/Living				✓
Personality				✓
Traditional Korean clothing				✓
National Parks				✓

Table 4.1: Topics in KoLLA Corpus Essays

and Kagan, 2007; Montrul, 2010). However, there has not been much research that specifically investigates heritage language learning and especially Korean heritage language learning. From the NLP perspective, this means that we can expect different usages of particles in both type and frequency along this divide — giving researchers the choice of trying to build a single, robust system that can handle all users, or building separate systems that are optimized based on learner type or level.

Thus, the KoLLA corpus includes a heritage learner group in order to make comparisons between heritage and non-heritage learners.

4.3 Defining particle error annotation

The main goal in this section is to properly define particle error annotation in a way that is directly relevant for automatic systems, taking into account the linguistic properties relevant for performing and evaluating the task. At the same time, the annotation itself can help define the task of particle error correction; the considerations taken to properly annotate Korean particles ultimately point toward the types errors that we can hope to correct with the system. The annotation must thus support the automatic task of predicting the correct particle (or no particle) in a given context (cf. work on English function words, e.g., Tetreault and Chodorow, 2008). However, as with any annotation effort, care must be taken to ensure that the corpus can also be used for other research aims, e.g. pedagogical research or statistical analysis (cf., e.g., Kim and Biber, 1994).

In order to lay a foundation for more robust evaluation of Korean particle error detection, the development of annotation for a new corpus which contains well-articulated information is required. The issues laid out in the following sections are important to deal with, not just to obtain appropriate annotation for Korean particle error detection, but also as a step in "developing best practices for annotation and evaluation" of learner data (Tetreault et al., 2010). These matters are especially pertinent to the annotation of erroneous functional elements in morphologically-rich languages, but many of them—e.g., the definition of grammaticality and handling multiple correct answers (section 4.3.6)—face any learner language annotation.

4.3.1 Defining the tokens

Korean words are generally formed by attaching suffixes to a stem. Regardless of the theoretical status of particles (see discussion in Chapter 3), they are written without spaces, meaning that it is critical to define the token boundaries. The issue of misspellings is also relevant to this discussion, as spelling mistakes can result in an obscuring of proper segmentation. In the next three sections, we will describe three layers of annotation, where the output of one layer is used as the input for the next.

4.3.2 Spacing errors

Given the differences in word formation and spacing conventions (e.g., noun and verb compounds are often written without spaces), spacing errors are common for learners of Korean (Ko et al., 2004; Lee et al., 2009a). Since particles are word-final entities, correcting spacing errors is necessary to define where a particle can be predicted. This case is similar to predicting a preposition between two words when those words have been merged. Consider (17), for instance. In order to see where the particle 를(*lul*) is to be inserted, as in (17b), the original merged form in (17a) must be split.[2]

(17) a. Original: 예-들-면
example-take-*if*

b. Corrected: 예-<u>를</u> 들-면
example-OBJ take-*if*

'if (we) take an example'

Words which have incorrectly been split are also corrected.

In (18a), for instance, the particle 한태 (*hanthay*) is incorrectly used as a separate unit. To clearly mark that this particle goes with this word, this spacing error should be corrected, as in (18). The complex morphological system of Korean can

[2]We will use *O* to refer to the original form and *C* to refer to its correction in examples like this.

explain the genesis for many of these errors. Analgous cases for more analytic languages like English where a morpheme is incorrectly split from a root, e.g. *John 's bicycle*, are difficult to conceive and probably not all that common in most learner corpora.

(18) a. Original: 저 한태 더 소중하-ㄴ 것
I to more precious-REL thing

b. Corrected: 저-한태 더 소중하-ㄴ 것
I-*to* more precious-REL thing

'the thing that is more precious to me'

Additionally, standard tokenization is handled on this layer, such as splitting words separated by hyphens or slashes, making the tokens compatible with POS taggers, as the tagger we use in this study (see Han and Palmer, 2004) does not handle tokenization itself. Of course, the original input is retained as part of the standoff annotation scheme.

4.3.3 Other modifications

There are other modifications made in mapping from the raw data to a tokenized form appropriate for particle error detection.

The first modification has to do with nominals occurring within quotation marks, as in example (19). In this case, the particle 를 (*lul*) is found outside the quotation marks, yet following standard tokenization practices, we generally separate quotation marks from words. If we continue that practice here, 를 (*lul*) will have to be a separate unit, as in (19b). This poses problems for automatic systems, as the particle is now separated from the nominal, yet particle-guessing is done on nominals. This problem is unique to languages and writing systems with agglutination; in languages like English, quotation marks are almost never used in the middle of a word.

(19) a. Original: "Jump"-를
"Jump"-OBJ

b. Annotation$_1$: " Jump " 를
" Jump " OBJ

c. Annotation$_2$: " Jump-를 "
" Jump-OBJ "

The use of quotation marks is often associated with an English word or words, as in example (19) either to make up for a gap in the student's lexicon or to talk about a proper noun or title. However, there are also similar usages of quotation marks with Korean words, as in example (20), where the student has enclosed a Korean movie title in quotation marks.

(20) "엽기적인 그녀"-를
"sassy girl"-OBJ
"'My Sassy Girl"'

In dealing with cases such as these, annotation developers are left with an unfortunate choice: do they retain the original ordering of the text with the understanding that the system will have trouble handling it, or do they alter the text so that the system has the best chance of performing well? In the end, either the tokenization, the error correction task, or the raw text must be violated during the annotation phase, leaving the annotator these three choices:

1. violate standard tokenization convention, by keeping quotation marks attached;
2. violate the particle error correction task, by tokenizing the particle as a separate unit;
3. violate the raw text, by transposing the quotation mark and the particle.

In the KoLLA corpus, the option illustrated in (19c), i.e. transposing the quotation mark and the particle (option 3 above), is employed. This annotation gives systems a better chance to correctly detect errors in these contexts. No information is lost, as the original text is retained in the first layer, and the changes could in principle be done as pre-processing in an actual tutoring system, so this solution works well for the purposes of error correction as well as learner corpus analysis research.

The next issue dealt with is parentheticals. In the KoLLA corpus, there are two main uses of parentheticals. The first is the obvious canonical purpose of parentheses, i.e. setting apart material that is non-essential, but relevant enough to the sentence as to be included. A second use that is more specific to learner corpora arises when students insert information not about the topic of the essay, but about their own use of the target language, Korean in this case. For example, if a student wants to use a word, but is unsure of the spelling, e.g. in the case of loan words, they might insert a parenthetical to clarify the meaning. A similar situation arises when a student is unsure of a word and provides a second guess at the word in parentheses.

Consider example (21), where the student attempts to use the Korean loanword for *Atlanta*, then provides the actual English word in parentheses and attaches a particle to the right parenthesis. It is worth pointing out here that there actually is a spelling error in the Korean word in (21a).

(21) a. Original: 앨랜다(Atlanta)-에서
Atlanta(Atlanta)-*from*

b. Annotation$_1$: 애틀랜타 (Atlanta) 에서
Atlanta (Atlanta) *from*

c. Annotation$_2$: 애틀랜타-에서
Atlanta-*from*

d. Annotation$_3$: 애틀랜타-에서 (Atlanta)
Atlanta-*from* (Atlanta)

There are multiple ways that one could annotate this phenomenon. I will discuss the ramifications of three of those here. First, one could follow standard tokenization practices as in (21b). This is easy in practice, and imposes no real change to the data provided by the learner. However, as was the case with the issue with quotation marks discussed above, this option is a detriment to the task of error correction, because the particle is not attached to any nominal, and will cause problems throughout an automatic NLP processing pipeline.

Removing the student's parenthetical, as in (21c) retains the meaning that they intended and rejoins the particle to its root. Moreover, one could even argue that removing the parenthetical in these instances actually helps because otherwise there are consecutive appearances of the word *Atlanta*, which is likely not the intended reading. The problem with this approach is that if it is applied in cases of true parentheticals, it involves removing a part of the writer's actual message. In (22), for example, the parenthetical actually adds information by specifying how long ago something happened, but the particle for the nominal 옛날 (*yeysnal*) is attached to the right parenthesis. So, if we make the correction in (22c) to rejoin the particle and nominal, we lose actual language data from the essay.

(22) a. Original: 옛날 (30-40년 전)에
old days (30-40years ago)-*in*

b. Annotation$_1$: 옛날 (30-40년 전) 에
old days (30-40years ago) *in*

c. Annotation$_2$: 옛날-에
old days-*in*

d. Annotation$_3$: 옛날-에 (30-40년 전)
old days-*in* (30-40years ago)

'In the old days (30-40 years ago)'

It might be tempting to employ the parenthetical deletion fix illustrated in (21c) to deal with the "editorial" parentheticals, but leaving meaningful ones like in (22) in the essay and dealing with them differently. However, treating these cases differently is not a viable option for a full-scale automatic system, as the decision as to which parentheticals would be difficult to assess without the intervention of a human annotator. Automatically identifying what type of parenthetical has been used by a language learner could certainly be a useful enterprise, but is outside of the scope of this work and is thus left for future work.

In order to legitimately treat all cases of parentheticals with particle spacing errors in the KoLLA corpus, the annotation illustrated in (21d) and (22d) is used. That is, the entire parenthetical is transposed with the following particle. While this might seem drastic on the surface, it is somewhat comparable to the quotation mark transposition described earlier, albeit with a longer string. In both cases, the entirety of the information provided by the student is passed to the system, but the ordering is changed as to be more favorable for automatic processing of particles.

4.3.4 Spelling errors

For work in English grammatical error detection, researchers often work only with pieces of data that do not include spacing, punctuation, and spelling errors, with the idea that a full system will handle such errors (e.g., Tetreault and Chodorow, 2008). To support error detection systems under similar assumptions, the KoLLA corpus includes fixes for spelling errors, in a second tier of annotation that assumes the correct spacing and tokenization established in the previous layer.

As with spacing errors, when spelling errors are not corrected, the correct particle cannot always be defined. Correct particles rely on correct segmentation, which misspellings can mask. In (23), for instance, the erroneously inserted character 기 (*ki*) makes it difficult to determine the boundary between the stem (맛) and suffix (으로), as it could be a mistake on either of the morphemes. Thus, all spelling errors that directly affect particle bearing nominals are corrected in KoLLA.

(23) a. Original: 갈비 맛<u>기</u>로는
rib ???

b. Corrected: 갈비 맛-으로-는
rib taste-AUX-TOP

'as for rib taste'

4.3.5 Segmentation

After describing corrections for spacing and spelling errors, we turn to word segmentation. To know whether a particle should be used, the *position* where it could occur be must be properly defined, leading to a need for the correct segmentation of particle-bearing words (i.e., nominals). Again, we see a contrast with similarly themed English error detection work where words such as particles and articles can be inserted into any white space, rather than needing to occur within a word. This annotation layer builds upon the previous two: it is often impossible to determine segmentation of learner forms, so correctly spelled forms serve as the basis for segmentation (cf. (23)). Also, this layer assumes correct particle forms, so all contextual spacing and spelling errors are corrected in the segmentation annotations.

Because it uses the corrected forms as input, the segmentation layer is fundamentally different from the previously described spacing and spelling layers in that no corrections need to be done here. Rather, this layer allows for better research of the language itself, in that it facilitates comparison to the output of other processes done on the corpus, e.g. POS-tagging, morphological analysis, parsing, etc. Most of these types of tools expect unsegmented forms as input, and then use their own rules and templates to split ecels into morphemes before tagging. Having this layer means that at the evaluation phase of the error correction task, one can check if the system had any chance of guessing a particle correctly based on the segmented form that came from the pipeline.

With segmentation, one can propose evaluating: 1) against the full correct form, or 2) against the correct particle. In the error correction work described in this dissertation, we will be using the correct particle for comparison. Note also that the important segmentation is of nominals, as we are only interested in particle error detection at the moment. Segmentation of other words, while a nice additional feature, is not required right now, and is left for future work on the corpus.

One item of interest that is dealt with here is the notion of sequential particles. This refers to a situation when multiple particles, e.g., 만-을 (*man-ul*), should be attached to a single noun for grammaticality. These are segmented into their individual units in this layer of annotation. This segmentation allows for different treatments of sequential particles; for machine learning, one could combine them into a single class, for example.

After these three layers of annotation defining the tokens, we turn now to the annotation of interest: defining what the target particle form should be. What is needed, then, is clarity on assigning the correct particle, i.e., the *target form*.

4.3.6 Defining the Target Form

A major, non-trivial step in designing particle error annotation for a learner corpus of Korean is deciding what exactly constitutes grammaticality, i.e., the target form, which sets the standard for the language. For English, a missing preposition in a context where there could be one almost always denotes a grammatical error, but for languages such as Korean (and e.g. Japanese) this is not so. This complication arises because particles can be dropped in spoken and even written Korean. Because some particles can be dropped, we have to ask whether the annotation marks obligatory or possible particles, as this may result in a large discrepancy in the annotation.

The annotation requires particles which are obligatory within a very specific definition of *grammaticality*. Namely, they are particles which beginning learners are taught to use. Future work may want to divide correct particles into obligatory and optional categories, but this decision captures the minimum needed for particle prediction systems and is consistent with the fact that particles are usually not dropped in formal Korean (Lee and Song, 2011).

The annotation guidelines follow the principle of "minimal interaction," (e.g., Hana et al., 2010): the corrected text does not have to be perfect; it is enough to be grammatical (at least for particles and the particles to which they attach —see Section 4.3.10). Within this definition of grammaticality, it is worth noting the particles that are not annotated. Firstly, in some cases, although a particle is possible, no one is required to use it. For example, in noun-noun compounds, one could use a genitive case marker between the nominals, but it is not mandatory, even for beginners. Thus, if such use of a genitive case marker is not used by a learner, it is not added to the annotation, as the sentence without the genitive is still grammatical according to this definition. Likewise, there is an object marker which can be placed between verbs, but which learners are not taught, and so it is not annotated, either. In both cases, the key criterion is whether the form is mandatory for learners. Any particles that do not fit this strict definition are not added to the annotation in KoLLA.

A final issue of grammaticality that must be dealt with is the overlap in distribution of topic particles, e.g., 은/는 (*ul*/*lul*) with structural case markers, most often the subject marker 이/가 (*i*/*ka*). There are many situations where either one of these particles could correctly be used in Korean, and the result would be perfectly acceptable even for native speakers. For example, consider (24) where the same sentence is provided with a subject marker in the first case, and a topic marker in the second. The difference in these situations stems from pragmatics and what the speaker or writer wants to emphasize. In the KoLLA corpus, though in general, the annotation ignores pragmatics, if a sentence initial noun is ambiguous as to whether the subject or topic marker is more appropriate, both are marked as acceptable. Otherwise, e.g., if the sentence already has a different noun marked as a subject or topic, only the most appropriate particle is listed in the annotation.

(24) a. Subject: 그래서 제-**가** 한국말-을 열심히 배우고 싶어요 .
so I-SBJ Korean-OBJ passionately learn want

b. Topic: 그래서 제-**는** 한국말-을 열심히 배우고 싶어요 .
so I-TOP Korean-OBJ passionately learn want

'So, I passionately want to learn Korean.'

4.3.7 Error Types

Before getting into the annotation of the correct particle, it is important to establish the conventions for annotating the errors themselves. Obviously, there can be great benefit for all kinds of research in knowing the frequencies and contexts for different types of errors. We will describe a pipeline approach to error correction in Chapter 6 that identifies whether or not a nominal should be followed by a particle at all or if there is a particle, if it is correct, before deciding what the actual particle should be. In order to evaluate such an approach, it is important to know in what way the target is erroneous. If there is an error of omission, for example, it can at least be detected with a binary classifier that guesses the presence of a particle after a nominal.

Three categories of errors marked up in the corpus described in Lee et al. (2009a)—omission, replacement, and commission are retained in the KoLLA corpus, along with one additional category, ordering.

Errors are marked as omissions when a learner neglects to use a required particle. In (25a), for instance, a learner omitted a subject particle after the word 것 (*kes*). While dropping a subject particle here might be acceptable in some registers of Korean, it is not within the definition of grammaticality used in this corpus. The error has been corrected in (25b). (26a) illustrates an omission error where the learner has correctly used one particle, but two particles are needed for the sentence to be grammatical, as in the corrected version in (26b). Specifically, the learner has only used the topic marker, 은 (*un*) after 사진 ("photo") in (26a), but the sequence of *in*+topic marker is required, as in (26b). Errors of this type, while extremely rare in this corpus, do occur and are marked as omissions, as each particle is considered a distinct unit. However, researchers can also choose to consider these as a single unit, thus treating the error as a substitution, by simply checking the original and correct particle annotation layers to see if more than one particle was used/needed.

(25) a. Original: 각 곳에 여러 좋은 것 있어요
each place-*at* many good thing exist

b. Corrected: 각 곳에 여러 좋은 것이 있어요
each place-*at* many good thing-SBJ exist

'There are many good things'

(26) a. Original: 이 사진은 버브 대이란이 있습니다
this photo-TOP Bob Dylan-SBJ exist

b. Corrected: 이 사진에는 버브 대이란이 있습니다
this photo-*in*-TOP Bob Dylan-SBJ exist

'Bob Dylan is in this photo'

Substitution errors refer to instances where the student uses a particle, but it is not the correct particle for a given context. In (27a), the learner has mistakenly used the subject particle, 이 (*i*), with the first word in the sentence, 사신 (*sasin*), likely on the assumption that every sentence must have a subject. However, in this case, the subject is the implied first person, and 사신 is actually a direct object of the verb 찍 (*ccik*). The correct formation of the sentence is given in (27b).

(27) a. Original: 사진-이 찍-고 책-을 샀어요.
photo-SBJ take-*and* book-OBJ bought.

b. Corrected: 사진-을 찍-고 책-을 샀어요.
photo-OBJ take-*and* book-OBJ bought.

'(I) took a photo and bought a book.'

Errors of particle commission are, not surprisingly, the opposite of omission errors. That is, the learner incorrectly uses a particle, when none should actually be there. An example is given in (28a), in which the learner has attached the particle 에 (*ey*), which can be used to denote when something happens (translated as 'at' in this example), to the word 보통 (*pothong*). This construction is not grammatical in Korean, and would thus be marked as an error of commission. The corrected sentence is provided in (28b) [3].

(28) a. Original: 보통-<u>에</u> 쉬-고 있-으면 음악-을 듣습니다.
normally-*at* rest-PROG-*if* music-OBJ listen

b. Corrected: 보통 쉬-고 있-으면 음악-을 듣습니다.
normally rest-PROG-*if* music-OBJ listen

'Normally when (I) am resting, (I) listen to music.'

Commission errors can also show up in cases where the learner uses a sequence of particles, but only a single particle is needed, as in (29a). Here, the learner has redundantly inserted 에 (*ey*) after 집 (*cip*), along with the necessary particle, 까지 (*kkaci*). The corrected sentence is given in (29b).

(29) a. Original: 웰슬리 대학-에서 제 집-에-까지 삼십 분-걸려요.
Wellesley College-*from* my house-*at*-*to* thirty minutes-take

b. Corrected: 웰슬리 대학-에서 제 집-까지 삼십 분 걸려요.
Wellesley College-*from* my house-*to* thirty minutes take

'It takes thirty minutes from Wellesley College to my house.'

There are a few instances of the last error type, particle ordering errors. This error type can only occur in sequential particles (discussed in Section 4.3.5). For example, in (30a), the learner concatenated 나 (*na*) and 한테 (*hanthey*), when they should have concatenated them in the opposite order, as in (30b).

(30) a. Original: 호텔-이나 길-에서 아무-나-한테 인사하세요.
hotel-*or* street-*in* anyone-AUX-*to* greet

b. Corrected: 호텔-이나 길-에서 아무-한테-나 인사하세요.
hotel-*or* street-*in* anyone-*to*-AUX greet

'Please say hi to anyone in a hotel or the street.'

[3]Note that there is the interesting construction 쉬-고 있-으면 in this sentence where two separate tokens convey a single notion (rest-progressive-if) when translated into English

The error type could simply be ascertained by examining and comparing the original and corrected fields. However, the *Error Type* annotation field, while redundant, does make the task of quickly identifying and/or collecting errors automatically a simpler process. Moreover, it is useful as a safeguard against annotation errors, as illustrated in Section 4.4.2.

4.3.8 Determining the Correct Particle

Once a particle has been marked as erroneous, the obvious next step is knowing what the correct particle or particles should be for the given context. The choice of best particle can be a complicated one, relying on a number of factors. First of all, there are hundreds of particles in the Korean language (Kang, 2002). With so many choices, it is not surprising that there can be disagreement, even among native speakers as to which particle is best for a specific context. The phonotactics of Korean are another factor in particle choice; whether the preceding nominal ends in a consonant or vowel can affect the allomorph of the particle. Segmentation also presents a challenge in choosing the best particle to annotate, as it interacts with the aforementioned phonotactics issue so that some nominals combine with particles to form entirely new units.

As with English prepositions and articles, there are situations where more than one particle could be correct. In these cases, the annotations list all reasonable alternates, allowing for a fair evaluation, where a system can evaluate against a set of correct particles. A particle which makes the sentence grammatical will end up in the set. It may seem like there is usually one particle which is the best answer, but there are no clear criteria for selecting one, and, in fact, there is a low inter-annotator agreement in a pilot experiment run while constructing this corpus, as opposed to the higher agreement for a set of particles (see Section 4.4.1). Thus, annotators are not forced to mark a single-best particle in a set.

The next issue is that many particles change based on simple phonological criteria—e.g., 을 (*ul*) is used for nouns ending in a consonant and 를 (*lul*) for those ending with vowels. While this distinction may not make the annotation task more difficult, it is worth pointing out as it could lead to confusion or disagreement among researchers. To be precise, the exact particle which is required in a given context is annotated in the KoLLA corpus. Users of the annotation could easily map one variant to another if desired.

The final factor we discuss here is that the particle form is sometimes non-trivial to annotate, as particles contract or merge with the noun, resulting in a surface realization which is not a true particle. In (31a), for example, the genitive 의(*uy*) merges with the preceding pronoun 저 (*ce*) to become 제 (*cey*). We could thus segment the target particle as either ㅣor 의.

(31) a. <u>제</u> 미래-에 도움-이
my future-AUX help-SBJ

b. <u>저+의</u> 미래+에 도움+이
my-GEN future-AUX help-SBJ
'(It will be) a help for my future.'

In these types of cases, the full particle (e.g., 의) is marked as the target particle, as in (31b), for two reasons. 1) The form ㅣ is not a full particle, only a contracted form, and annotating full particles is preferable, to be consistent across different examples. 2) This is the segmentation that taggers often provide (e.g., Han and Palmer, 2004).

4.3.9 Handling multi-token particles

Example (32) illustrates a case of, not just a single particle, but two units which go together. Namely, 을 (*ul*) is best changed to the complex form 에 대해서 (*ey tayhayse*), rather than just 에. These cases are fixed forms, with forms like 대해서 often originally deriving from verbs. Theoretical linguists (e.g., Nam, 1993; Ryu, 2007) disagree on whether these forms are single particles or not; with the whitespace, it is easy to reconstruct different analyses. In the KoLLA annotation scheme, such cases are treated like a single unit. That is, 대해서 (*tayhayse*) is not segmented and treated like a separate word, rather it is included in the same field of annotation as the original particle, 을, in this case. This treatment allows for the simplest annotation of making a single correction to the original text, a treatment similar to that of sequence particles.

(32) a. Original: 으막-<u>을</u> 생가카다
music-OBJ think

b. Corrected: 으막-<u>에 대해서</u> 생가카다
music-ABOUT think
'think about music'

4.3.10 The influence of surrounding errors

Another issue affecting the annotation of particle errors is how to handle surrounding forms. The issue of suroundinng errors aside from the target error type is relevant for corpus annotation and learner error detection for all languages and adds a significant amount of difficulty to tasks within the field. While many learner errors do not affect particle errors, some interact with particles, either directly or indirectly. For example, (33) illustrates a learner error that is actually tied to the selectional restriction of, the verb, 의지한다 (*uycihanta* - 'lean on'), in regards to animacy of its object. Here, 의지한다 is the wrong choice because it requires an animate object and 시험 (*sihem* - 'exam') is inanimate. If we correct the verb to 달려있다 (*tallyeissta* - 'depend'), as in (33b), the correct particle is 에 (*ey*). If we do not correct the verb, then the learner's particle, 을 (*ul*), is syntactically appropriate for the verb, even if the verb's selectional restrictions are not followed.

(33) a. Original: 내 인생-이 이 시험-을 의지한다
my life-SBJ this exam-OBJ lean-on

b. Corrected: 내 인생-이 이 시험-에 달려있다
my life-SBJ this exam-ON depend

'My life depends on this exam'

It is important to clearly designate *at what point* in the process the particle is correct. The current annotation does not deal with word choice and related semantic issues, so the emphasis is on the correct particle at the point before any such errors are corrected. In cases like the one in (33), there is no correction to (33b). Instead, in these cases the verbs and other lexical errors are left as is and the focus is on annotating the particle based on the current verb. This makes the particle-selection task for machine learning more attainable and is easily extendible with multi-layered annotation. If, for example, the annotation required the correct particle, even with the wrong verb, a machine learner would be tasked with essentially guessing the wrong particle for a given verbal context. Changing the verb and particle before passing along the data to the error correction system would be making the somewhat large assumption that some automatic process will have already corrected verb choice errors.

Note that this is a significant question to address, as correction is a dynamic process. Because of the interrelated nature of different errors, different researches have dealt with context errors in a variety of ways: Rozovskaya and Roth (2010b) correct a set of related errors, and Gamon (2010) "eliminated sentences containing nested errors and immediately adjacent errors when they involve pertinent (preposition/article) errors," eliminating 30% of the sentences. Hana et al. (2010) annotate multiple layers of Czech learner language in order to capture the dynamic process of correction (cf. also Boyd, 2010), and with the multi-layered annotation in the KoLLA corpus, one could follow suit here, although the relation to system evaluation would be more complex. In the future, there is potential for further exploring the interrelated errors of verb and particle choice in this corpus.

4.3.11 Particle Categories

The last piece of information that is provided in the annotation is a tag that gives a description of what the particle's function is. For example, the particle 를 (*lul*) is given the *O* tag, specifying it as an object marker. There are several different types of particles all of which behave differently and may reflect different error patterns. Although boundaries can be fuzzy, it can be helpful to distinguish some broad categories (Nam and Ko, 2005; Lee, 2004): 1) structural case markers; 2) inherent case markers (cf. prepositions); 3) auxiliary particles, i.e., those which change lexical properties or a word's semantics, such as topic markers (cf. articles); and 4) conjunctions. Traditional grammars (e.g., Nam and Ko, 2005) make a 3-way distinction, not separating structural and inherent case. However, these subtypes pattern differently: structural case markers exhibit more optionality and are less tied to specific predicates, among other differences (see discussion in Lee, 2004). The four-way distinction is utilized throughout this research. This categorization helps because learners can make different kinds of mistakes with different kinds of particles, and systems can be developed, evaluated, or optimized with respect to a particular kind of particle. As a classification of particles is not without controversy, especially for, e.g., datives (see Lee, 2004), other researchers may prefer a different meta-categorization—which the annotation of individual functions in this corpus also supports.

As mentioned earlier, there are hundreds of particles in Korean. Depending on genre, Kang (2002) reports that 107 particles appeared in academic abstracts while 245 were found in literature. However, many of these are not used often, and certainly not by learners—e.g., nine particles cover a set of thesis abstracts and 32 cover 95% in Kang's study. Taking into account that this corpus is composed of essays by beginning learners who are writing about everyday topics and are likely at a lower level than those who are writing theses, one can assume that the vast majority of particles belongs to an even smaller subset.

A list of the 40 most frequent particles for first and second year learners was gleaned after consulting two textbooks, *Integrated Korean* (Beginner and Intermediate) and *Yonsei Korean* (1 & 2) (Park et al., 2003). This list is used as a basis to show the four-way categorization we use in Figure 4.1. Note that the count includes all unique orthographic forms, so each allomorph as distinct. For example, 가 (*ka*) and 이 *i*) are both included in the count even though they are allomorphs of a single morpheme. Also, the 40 particles include some overlapping of categories due to ambiguity of forms, e.g., 에 (*ey*) can be a time, locative, or goal adverbial. These 40 particles do not represent every particle in the corpus. Most of the other particles in the corpus are either used by advanced intermediate learners, or result from learners using predicates requiring particles they have not been taught or heritage speakers going beyond their level.

Structural Case:

SBJ: 이/가
SBJ_HON: 께서
OBJ: 을/를
GEN: 의

Auxiliary:

TOP: 은/는
AUX: 도 ('also')
만 ('only')
마다 ('each')
바께 ('only')
처럼 ('like')
만큼 ('as much as')
대로 ('as')

Conjunction:

CONJ: 와/과 ('and')
하고 ('and')
이나/나 ('or')
이든지/든지('or')
이라든지/라든지 ('or')
이랑/랑 ('and')

Inherent Case:

DAT: 에/에게 ('to')
한테 ('to')
DAT_HON: 께 ('to')
COMP: 이/가
TIME: 에 ('in, at')
LOC: 에 ('to')
에서 ('from')
INST: 으로/로 ('with')
DIR: 으로/로 ('to, as')
SRC: 에서 ('from')
에게(서) ('from')
한테(서) ('from')
부터('from')
SRC_HON: 께 ('from')
GOAL: 에('to')
까지 ('to')
WITH: 와/과 ('with')
하고('with')
이든지/든지 ('with')
VOC: 아/야
COMPAR: 보다

Figure 4.1: 40 Particles Taught to Learners of Korean

It is not surprising that learners have trouble with particles, just based on a cursory glance at Figure 4.1. There are, for example, no less than seven different forms of particles that can all translate to the English word *to*. Conversely, the particle 에 (*ey*) can be translated as, at least, *to*, *in*, and *at*. Such instances of hyponymy/hypernymy are likely sources of errors as language learners tend to rely on their L1 when assigning semantic relationships in their L2 (Ijaz, 1986). Moreover, these 40 particle forms make up less than 20% of all Korean particles. For further discussion of particle see Section 3.2.4.

4.4 Annotating KoLLA

The previous discussion outlines the type of annotation needed for evaluating Korean particle errors made by learners. The annotation was carried out by two native speakers of Korean: one of whom is a professor of Korean, with a Ph.D. in linguistics, the other was a graduate student in linguistics and a Korean Language teacher's assistant. To put all of the annotation together, the annotators used the Partitur Editor of EXMaRALDA [4]. This tool is ideal for these annotation conventions as it easily supports the layered approach, clearly linking rows of annotation with columns of words. An example of full annotation is given in Figure 4.2, for the sentence in example (34)[5].

(34) a. Original: 물론 뉴욕-에서 태어-났-기 때문-에 영어 바께
of course New York-*in* born-PAST-NML reason-*for* English only
할 수 있-겠-죠
speak way exist-FUT-END

'Of course, since (I) was born in New York, I was able to speak only in English. '

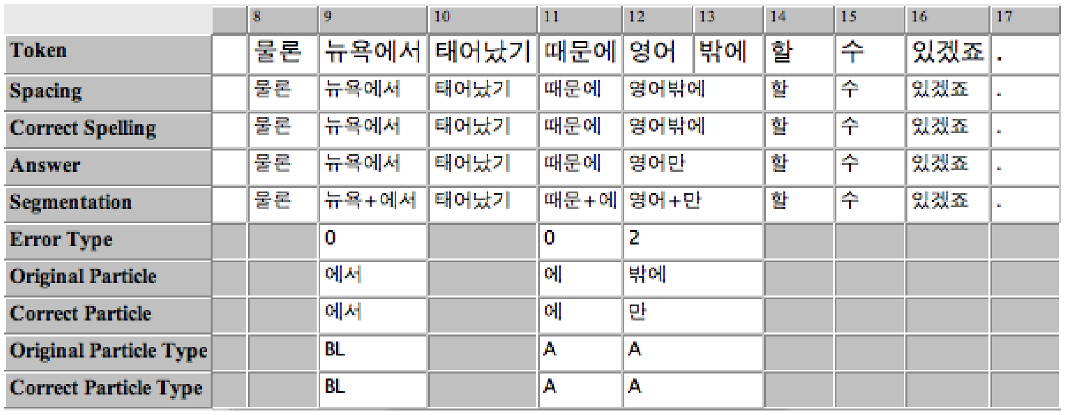
b. Corrected: 물론 뉴욕-에서 태어-났-기 때문-에 영어-만
of course New York-*in* born-PAST-NML reason-*for* English-*only*
할 수 있-겠-죠
speak way exist-FUT-END

As can be seen in the figure, positions 12 and 13 are merged in the *Spacing* layer, in order to correct the spelling, as the particle 바께 (*pakkey*) was originally written as a separate token. In this instance, there is no spelling error, so the *Spelling* layer is identical to *Spacing*. The *Answer* layer changes from the erroneous to the correct particle for position 12; 바께 is changed to 만 (*man*. This change is encoded as a substitution error (represented with the '2' on) the *Error Type* layer. The *Segmentation* layer clearly defines the morpheme boundaries for nominals by inserting a + between each morpheme in the ecel. The next two layers represent the *Original Particle* and *Correct Particle*, where only the particle is given. Additionally, both the original and correct particles are encoded as auxiliary particles ('A') in the *Original*

[4]http://www.exmaralda.org/en$_i$ndex.html

[5]NML is a nominalizing morpheme that can cause a verb to act like a noun

Particle Type and *Correct Particle Type* layers. In this particular sentence, we also see two correctly-used particles at positions 9 and 11, encoded as error type '0' (i.e., no error), with one a locative adverbial ('BL').



		8	9	10	11	12	13	14	15	16	17
Token		물론	뉴욕에서	태어났기	때문에	영어	밖에	할	수	있겠죠	.
Spacing		물론	뉴욕에서	태어났기	때문에	영어밖에		할	수	있겠죠	.
Correct Spelling		물론	뉴욕에서	태어났기	때문에	영어밖에		할	수	있겠죠	.
Answer		물론	뉴욕에서	태어났기	때문에	영어만		할	수	있겠죠	.
Segmentation		물론	뉴욕+에서	태어났기	때문+에	영어+만		할	수	있겠죠	.
Error Type			0		0	2					
Original Particle			에서		에	밖에					
Correct Particle			에서		에	만					
Original Particle Type			BL		A	A					
Correct Particle Type			BL		A	A					

Figure 4.2: Corpus annotation for (34), using the PartiturEditor of EXMARaLDA (Schmidt, 2010)

4.4.1 Inter-annotator agreement

To gauge the reliability of the annotation, two experienced annotators annotated the correct particle and the error type (addition, replacement, omission)[6] on the heritage intermediate subcorpus. We report the agreement on both tasks here. Given the high number of times they both gave no particle to a word (in 1774 ecels), these cases were removed when calculating agreement, so as not to overly inflate the values. Also, when either annotator used more than one particle for an instance (occurring 9 times), only full agreement counts.

The agreement rate was 94.0% for the error type (Cohen's kappa=79.1), and 92.9% (kappa=92.3) for specific particles. These values are extremely high, which

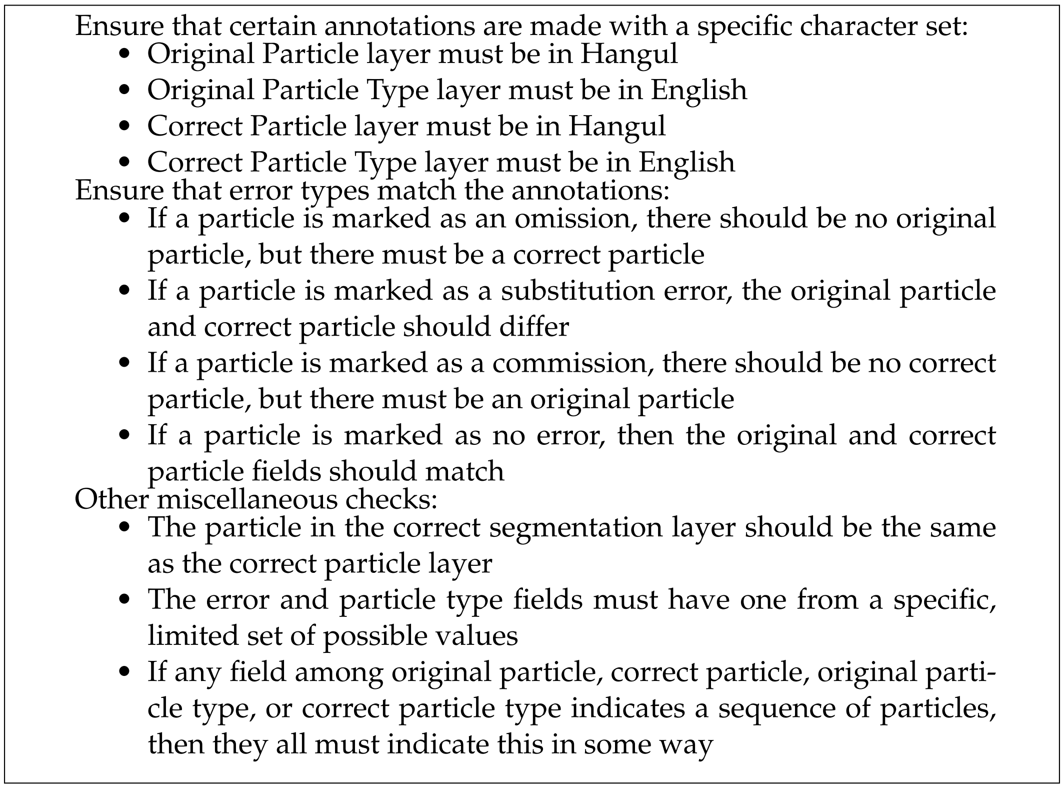
[6]This experiment was done before adding ordering errors to the annotation guidelines; after revising the annotation, it was also the case that no ordering errors needed to be annotated in the subcorpus used for this experiment.

can be explained by the fact that these annotators were highly-trained and were using a relatively stable set of guidelines which had been under development for some time (based initially on Lee et al. (2009a)). Kappa for particle agreement is high because of the fact that there are over 30 particles, with no overwhelming majority categories, so it is unlikely for annotators to agree by chance. Previous work (Lee et al. (2009a)), which did not allow multiple particles per position, had a lower agreement rate (e.g., kappa for particle value = 62%), likely due to less well-articulated guidelines.

Multiple particles To gauge how difficult it is, in general, to assign more than one particle, an additional experiment was carried out, examining cases where more than one particle may be possible. The experiment included 30 verbs that license more than two particles for a nominal argument. Using these verbs, hand-constructed sentences with missing particles were presented to two annotators who were asked to fill in the missing particles in the order of preference. Although the agreement rate of sets of particles was 87.8%, the agreement of the "best" particle was only 60%. This supports the decision in section 4.3.8 to annotate sets of particles.

4.4.2 Annotation Error Checking

As with most annotation efforts, the resultant corpus was not without some human errors, most of which are simply mistakes and/or inconsistencies introduced by the annotator while manipulating the data (Dickinson, 2005). Fortunately though, the robust, layered annotation of the corpus allows for automatic checks of the data that can flag possible annotation mistakes in a variety of ways. Figure 4.3 lists the checks that we implemented to ensure that various aspects of the annotation are well-formed. There are checks for character-specific encodings of certain layers, checks that ensure if an error has been marked (or not) in the *Error Type* layer, that the other layers show the same type of error, as well as some consistency checks.


Ensure that certain annotations are made with a specific character set:

- Original Particle layer must be in Hangul
- Original Particle Type layer must be in English
- Correct Particle layer must be in Hangul
- Correct Particle Type layer must be in English

Ensure that error types match the annotations:

- If a particle is marked as an omission, there should be no original particle, but there must be a correct particle
- If a particle is marked as a substitution error, the original particle and correct particle should differ
- If a particle is marked as a commission, there should be no correct particle, but there must be an original particle
- If a particle is marked as no error, then the original and correct particle fields should match

Other miscellaneous checks:

- The particle in the correct segmentation layer should be the same as the correct particle layer
- The error and particle type fields must have one from a specific, limited set of possible values
- If any field among original particle, correct particle, original particle type, or correct particle type indicates a sequence of particles, then they all must indicate this in some way

Figure 4.3: Annotation Error Checks

Clearly, a test set that is without errors is best, in order for evaluation to be fair. While these checks may not catch all possible annotation errors, they have served to flag a variety of errors very quickly. The other option, i.e. human hand-checking of the entire corpus, would be a time-expensive endeavor that is, like the annotation effort itself, prone to human fatigue and inconsistency.

4.4.3 Relevant Statistics

Finally, we provide here a breakdown of the statistics that can be gleaned from the fully annotated corpus. Before moving forward, I would like to stress that while we present the numbers for the complete corpus here, it is only in the interest of providing statistical details about the corpus and each of its subcorpora. When developing methodology and tuning machine learning optimizations, only the facts that can be gleaned from the development set were considered.

Table 4.2 provides the numbers for sentences, tokens, average tokens per sentence (*TpS*), nouns, particles, and the percentage of nouns that should have particles (*N-P%*) for both the *Raw* and *Annotated* corpora, whenever possible. Here, *Tokens* represents the number of Korean *ecels*, which are roughly equivalent to English words, i.e. strings of text surrounded by white space, but can be made up of a root and a number of morphemes. The missing cells in the *Raw* section of the table denote that the information is not available. In the case of nouns, the annotation scheme marks them only after spacing has been corrected, thus there is no way to accurately count nouns in the raw data and obviously no way to calculate nouns that should have particles. Within both the *Raw* and *Annotated* sections of the corpus, we give the counts of the phenomena for each of the subcorpora where *F* denotes *foreign*, *H* denotes *Heritage*, *B* denotes *beginner* and *I* denotes *intermediate*, as well as for the completed corpus (*Combined*).

		Sentences	Tokens	TpS	Nouns	Particles	N-P%
Raw	FB	360	1965	5.5	-	589	-
	FI	371	3700	9.9	-	1128	-
	HB	367	2645	7.2	-	833	-
	HI	299	2822	9.4	-	910	-
	All	1406	11132	7.9	-	3460	-
Annotated	FB	361	1997	5.5	714	654	91.6
	FI	374	3917	10.5	1412	1206	85.4
	HB	376	2715	7.2	961	881	91.7
	HI	299	3031	10.1	1134	980	86.4
	All	1410	11660	8.3	4221	3721	88.2

Table 4.2: Basic KoLLA Corpus Statistics Before and After Annotation

There are a number of noteworthy facts about the corpus that can be seen in Table 4.2. First of all, the FB section is significantly smaller than the other sections in terms of tokens, despite having a fairly typical number of sentences, with an average of 5.5 tokens per sentence. The HI corpus, on the other hand, has a lower number of sentences than the other subcorpora, yet still has more tokens than either of the beginners' subsections, for an average of closer to 10 tokens per sentence. In general, the TpS shows that the beginners' data contain fewer tokens per sentence on average than the intermediates. Thus, the beginners' subcorpora have shorter, likely simpler sentence structures, while the intermediates' subcorpora feature longer, more complex sentences. The implications for automatically processing the corpus that can be gleaned from these numbers are that the FB corpus might be more easily correctly POS tagged than some of the other subsections, but the language probably does not match what we would expect to find in natively written documents on the web, i.e. the language used in the training corpus (c.f. Chapter 5).

In Table 4.3, we provide the type/token ratio for each subcorpus of KOLLA, the entire corpus, and a comparably-sized part of the Wall Street Journal section of the Penn Treebank. First of all, when comparing the KoLLA corpus to the WSJ, the type/token ratio is drastically higher for the Korean learners. While this result matches expectations regarding Korean corpora due to its morphological complexity (Han et al., 2002), it underlines the importance of developing methodology for Korean that accounts for more extreme data sparsity, rather than relying solely on error detection practices that work well for English. Comparing each of the subcorpora, one can see that heritage learners utilize a more diverse lexicon than foreign learners, and the same holds true for intermediates with respect to beginners.

Table 4.4 shows the number of each particle category (c.f. Section 4.3.11) af-

	FB	FI	HB	HI	All	WSJ-PTB
Types	923	2015	1277	1594	5809	3139
Tokens	1997	3917	2715	3031	11660	11670
Ratio	.462	.514	.471	.525	.498	.269

Table 4.3: Type/Token Ratio by KoLLA Sub-corpus

ter corrections have been made to the corpus. The first two columns, structural and inherent case, are the most prevalent types of particles in the corpus, accounting for 43.6% and 24.2% of the particles in the corpus, respectively. The structural category includes subject and object markers, as well as genitive case, so it's not surprising that they are so prevalent. Inherent case markers function similar to English prepositions. Auxiliary particles are broken down into two subcategories here, topic (*Top.*) and all others. Topic markers are clearly used more frequently than other auxiliary particles, making up 13.2% of the total particles, compared to 5.2% for other auxiliaries, meaning that all auxiliaries make up about 18.4%. The next category in the table is conjunctive (*Conj.*), which are the least frequent of the four main particle categories, accounting for only 5.5% of the total particles in the corpus.

The next column Table 4.4 is *Set* which refers to instances where a single particle is needed, but there are multiple options that come from different categories. There 207 (5.6% of total particles) of these, which can then be broken down into two subcategories: *S/T*, which refers to instances where either a subject or topic particle works equally well, and all other cases. I show the count for *S/T* because these are instances that pattern similarly to Structural Case or topic markers, and could thus be included in counts for those types of particles. Finally, we provide the number for instances where a sequence (*Seq.*) of particles, i.e. multiple particles for a single noun, is necessary for grammaticality. These cases are extremely infrequent in this corpus, accounting for only 2.6% of all particle usage.

	Structural Case	Inherent Case	Auxiliary Top.	Auxiliary Other	Conj.	Set S/T	Set Other	Seq.	Total
FB	245	157	133	20	30	29	38	2	654
FI	563	292	126	65	77	30	15	38	1206
HB	352	233	128	39	46	20	33	30	881
HI	461	220	107	69	52	25	17	29	980
All	1621	902	494	193	205	104	103	99	3721

Table 4.4: Particles by Category

Next, we turn to error type distribution among the KoLLA subcorpora in Table 4.5. As can be seen in the *All* row, omissions and substitutions make up the vast majority of the errors in the corpus, accounting for 53.5% (285/533) and 39.4% (210/533) of the total errors (nearly 93% together), respectively. These findings remain consistent throughout the subcorpora, though there are differences regarding the prevalence of omissions vs. substitutions, ranging from an equal 1:1 ratio in the FB (least native-like) subcorpus, to 1:0.59 in the HI (most native-like) subcorpus, suggesting that as students progress, if they use a particle at all, the chances of it being correct go up.

	Particles	Errors Total	Errors Omission	Errors Substitution	Errors Commission	Errors Ordering
FB	654	134	63	63	8	0
FI	1206	166	86	65	14	1
HB	881	108	60	37	9	2
HI	980	125	76	45	4	0
All	3721	533	285	210	35	3

Table 4.5: Number of Error Types Annotated in the KoLLA Corpus

In Section 4.1, we discussed the annotated corpus of learner Korean described in Lee et al. (2009a). This corpus was divided using the same four-way split among learner level and type as the corpus described in section 4.2. An examination of this corpus reveals a similar distribution of error types to that of the KoLLA corpus. Table 4.6 provides the counts for omissions, substitutions, and commissions (ordering errors were treated as substitutions) found in this corpus. As can be seen in the table, omission errors make up the biggest proportion of the errors (47.6%), followed by substitution errors, and finally commission errors (7.8%).

	Total	Omission	Substitution	Commission
FB	218	93	108	17
HB	262	122	119	21
FI	270	134	111	25
HI	157	83	67	7
All	907	432	405	70

Table 4.6: Number of Error Types Annotated in the Lee et al. (2009b) Corpus

Conclusion

In this chapter we have thoroughly described an essential component for testing the particle error correction system that is the focus of the dissertation: the evaluation data. Along with serving as a sound test set for the current work, the KoLLA corpus provides both the pedagogical and corpus linguistics research fields a valuable resource, as there are few, if any, other annotated corpora of learner Korean available. The corpus features a multi-layered annotation scheme that provides a step by step look at how the errors are annotated. The annotation scheme adheres to the principal of minimal interaction and remains as theory neutral as possible while still providing robust information about each particle in the corpus.

With the evaluation data established, we will turn next to the other set of essential data for the current approach: the training data for machine learning. In Chapter 5, we will review our methodology for gathering and utilizing appropriate data from the Web to serve as training data for the machine learning experiments described in Chapters 7, 8, and 9.

Chapter 5

Procuring Korean Data From the Web

To write it, it took three months; to conceive it three minutes; to collect the data in it all my life.

— F. SCOTT FITZGERALD

The web is a resource that has grown in popularity among NLP research over the past decade. It provides a wealth of data that can prove extremely useful, especially for languages like Korean that lack large-scale procured data resources. With that in mind, we explore two ways that the web can serve as a useful source of data for Korean error detection in this chapter. One way is to provide well-formed text to serve as training for a machine learning system, cf. the approach in Tetreault and Chodorow (2008). Secondly, we explore using social networking language-learning data to construct a large scale corpus of annotated Korean particle errors.

Because the error detection system described in this research relies heavily on machine learning, a reliable training corpus is an obvious requirement. However, there are a number of factors that make obtaining quality training data a significant problem for learner language, especially in the case of Korean. First of all there are far fewer available resources for Korean than for more canonically researched languages like English. Ideally, we could use the annotated learner corpus described in Chapter 4, but data driven methods like machine learning require corpora that are much larger than what we have for Korean so far. Second, Korean features rich, agglutinative morphology, so any corpus of Korean typically has a very high type/token ratio, making data sparsity an issue (Han et al., 2002). Another issue arises from the disparate linguistic patterns that appear in different registers and genres Korean, i.e. more formal Korean is vastly different from casual Korean (Kim and Biber, 1994).

5.1 Training Data for Machine Learning

When developing a machine learning system for learner error detection, there are basically two distinct options for training data: 1) use native-like data to build an error free model of what language *should* look like as a comparison to test data (c.f., Tetreault and Chodorow, 2008), or 2) construct data that contains instances of errors along with correct usage (c.f., Rozovskaya and Roth, 2010c; Dahlmeier and Ng, 2011). It should be noted that these options lead to different machine learning paradigms; unique methodologies must be employed depending on the type of training data available.

For approaches that rely on native corpora, researchers can use a prebuilt corpus that is publicly available, or turn to less traditional means of corpus building, such as utilizing web-scraping methodology. While the option to utilize annotated learner data is attractive, it is also often unrealistic because, as discussed in Chapter 4, annotated learner corpora are a rare commodity for Korean, and even for English, finding an annotated learner corpus large enough to serve as training data for a machine learning-based system on its own is not a viable option at this point. As such, there are two viable options for procuring a Korean corpus large enough for corpus driven efforts. One is to utilize unannotated corpora to produce a model of well-formed Korean; the other is to artificially insert errors into data to approximate learner data. Even when this decision has been made, the issue of procuring a corpus appropriate for the task at hand reamains. In Section 5.2, we will describe some native Korean corpora that are currently available and examine their utility in training a machine learner for detecting errors in learner data.

5.2 Existing Korean Corpora

There are a number of native Korean corpora that have been developed and used for various research purposes. Here we will discuss the details of some of the more established projects that have information publicly available.

There was a large interest in developing annotation guidelines and corpora in Korea in the late 1990s (cf. Yoon and Choi, 1999a,b; ETRI, 1999). However, these corpora prove less than ideal for many applications as they are made up of largely formal text and are difficult to procure. Moreover, most or all of the documentation for these resources is only available in Korean, making for a significant challenge for non-Korean researchers.

One of the prominent Korean corpus efforts is the 21st Century Sejong Project (a.k.a., The Korean National Corpus, Kang and Kim, 2004; Kim, 2005; Kim et al., 2007). The project has spanned over a decade, beginning in 1996 and continuing through 2007 in an effort to collect and, in some cases, annotate corpora of different kinds of Korean. The corpora cover a broad overview of Korean language, including Modern Korean in South Korea, North Korean and Korean abroad, Old Korean, and Korean-English and Korean-Japanese parallel corpora. The Modern Korean corpus contains both written and spoken varieties. The written section has a raw corpus of 62 million words, 15 million morphologically analyzed (cf. POS tagged) words, a morpheme-sense tagged corpus of 12.5 million words, and a treebank with 800,000 words. Figure 5.1 details the sources used for constructing the 62 million word written corpus of Modern Korean. As can be seen in the figure, a majority of the data comes from formal registers such as news and nonfiction writing.

Book — nonfiction	33%
Newspaper	29%
Book — fiction	17%
Magazine	13%
Scripted spoken	5%
Other	3%

Figure 5.1: Data Types in Sejong Corpus of Modern Korean

Another effort for annotating Korean comes from the Penn Korean Treebank (PKTB) (Han et al., 2002). The treebank is a smaller corpus with only 54,000 words. However, it does feature fully bracketed Penn Treebank (PTB) style syntactic annotation, along with morphological analysis (POS tagging). The data comes from Korean military training manuals that contain information about various military procedures. Version 2.0 of the PKTB contains another 132, 040 words of newswire text taken from the Korean Newswire Corpus[1] (Han and Ryu, 2005). In theory, having such a resource of expertly annotated syntactic structure could be useful in an error detection project, the smaller size of these corpora likely prohibits their utility in practice in a data-driven approach.

For a variety of reasons, these corpora are not ideal for the current approach. First, not every language has such resources, and one goal for this research is to work towards a language-independent platform of data acquisition. Secondly, the domain of student writing is generally different from news-heavy corpora (Gamon, 2010), so even the resources that are available may not be the best fit for tasks that examine learner data. News texts are typically written more formally than learner writing, and exhibit different linguistic tendencies than casual or learner data. In the case of particle usage in general, this could actually be a useful feature of news texts, however, Korean has an extremely complex set of rules regarding register and formality, and other considerations such as sentence length, honorifics, verb tense, and the use of copular verbs can vary according to genre of writing (Kim and Biber, 1994).

We are interested in situations where the task is constant—here, detecting grammatical errors in particles—but the domain might fluctuate. This is the case when

[1] `http://catalog.ldc.upenn.edu/LDC2000T45`

a learner is asked to write an essay on a prompt (e.g., "What do you hope to do in life?"), and the prompts may vary by student, by semester, by instructor, etc. As shown in Chapter 4, Section 4.2, the topics covered in the test corpus are made explicit, so utilizing a methodology that can exploit that may lead to better results.

5.3 Utilizing the World Wide Web

In the absence of a large-scale publicly available corpus of well formed Korean to serve as training data, we turn to the World Wide Web as the next best option for gathering useful data for error detection. Fortunately, there is a fairly large community of researchers that focus on developing methodology for producing web-based corpora. The general idea here is to scrape Korean webpages to build a customized corpus. These Web as Corpus (WaC) techniques present a viable option to build a corpus that suits the needs of the current approach. Although the World Wide Web might not be a useable source for *every* language, it still provides data for many languages that, at present, do not have pre-built resources available. Thus it fulfills, to some extent, the goal of developing language-independent methodology. Also, building a corpus this way provides a great deal of control over what type of content is represented in the data. By isolating a particular domain, we can hope for greater degrees of accuracy in error detection; see, for example, the high accuracies for domain-specific grammar correction in Lee and Seneff (2006). We will lay out the methodology and results of our own endeavor to produce a suitable Korean web corpus in Section 5.5.

Along with building web corpora, the World Wide Web contains data that can be helpful in other facets of this research. In particular, there have been recent efforts made to utilize online language learning websites that rely on user submitted annotations of language learners, e.g. Lang-8[2]. In Section 5.7, we will discuss how this type of data is utilized in the current research.

5.4 Web as Corpus Research

Before moving forward, we present here a brief historical overview of the use of corpora in NLP and more specifically, using the Web as a corpus, as well as a review of WaC research and methodologies.

Controlled, static corpora have been extremely useful and have a rich history. The earliest computer-based corpus is generally considered to be the Brown corpus, which is a corpus of about one million words of English compiled in the 1960s (Francis and Kucera, 1979). The next few decades would see more corpora developed, e.g. the COBUILD Bank of English (Sinclair, 1987), the British National Corpus (Leech, 1992), though the original intent of most of these corpora were not associated necessarily with NLP. It was not until the Church and Mercer (1993) introduction to the special issue of *Computational Linguistics* that corpora started to be recognized as beneficial to NLP (Kilgariff and Grefenstette, 2003). These types of corpora have been, and continue to be, a mainstay for many tasks in NLP. But, as discussed earlier in Section 5.2, we are more interested in more dynamic corpora. The Web is a resource that offers researchers a way to build maleable corpora that can be built to address very specific needs.

Though it could be difficult to say exactly when NLP researchers started to use the Web as a resource, one decent measuring stick is to look at the first use of the Web at an Association for Computational Linguistics (ACL) meeting. In this case, Mihalcea and Moldovan (1999) used web counts to rank word sense frequencies.

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

Then in 2000, Jones and Ghani (2000) presented work on building language specific corpora from the Web, and Fujii and Ishikawa (2000) used the web to extract definitions of technical terms from the Web. Since then, WaC has caught on to the point that there is a workshop devoted entirely to the field (SIGWAC[3]), and a great deal of research has been done specifically on how to best utilize the Web to build corpora.

Baroni and Bernardini (2004) introduced BootCaT [4], a suite of perl scripts that implement an iterative procedure to bootstrap specialized corpora from the web. The process is iterative in the sense that after an initial pass using a defined set of seed terms to build queries, the resultant webpages can be analyzed and new seed terms extracted to make new queries. The authors used the scripts to build English and Italian corpora that show that the corpora are well-built and useful by evaluating the informativeness of the webpages returned, extracting n-grams from the corpora and ensuring their relevance to the intended topic of the corpora, and comparing the corpora to the original texts from which the initial seed terms were extracted. I will go over the procedure of BootCaT in detail in Section 5.5.

Baroni and Ueyama (2004) lay out methodology for using BootCaT to construct a corpus of Japanese. This work is important to the current approach, as Japanese and Korean share many properties that could be potential stumbling blocks for a suite of scripts designed to work on European languages. Most significantly, the encoding of the webpages differs significantly, and both languages have a fair amount of loan words that are often written in a foreign character set (for Japanese and Korean, the Latin alphabet is a foreign character set, for example). Crucially, the

[3]`http://sigwac.org.uk`

[4]`http://bootcat.sslmit.unibo.it`

authors explain the methodology for altering the original BootCaT scripts to work with Japanese. The resultant Japanese corpus is made up of mostly relevant documents, meaning that with a large enough corpus, one can expect to have a fairly useful and topic- focused corpus using this methodology.

Ueyama and Baroni (2005) further develop the methodology they established with their 2004 effort in order to produce large-scale corpora of Japanese. Using 100 seed terms comprised of basic Japanese vocabulary, they constructed 100 random 3-tuples to query Google. They take the top 10 hits for each query and discard duplicate urls. The Japanese writing system does not use white space between words, so they tokenized the text with ChaSen (Matsumoto et al., 1999), resulting in two corpora of 3.5 and 4.5 million tokens.

Sharoff (2006) builds large-scale BNC-like corpora for Chinese, English, German, Romanian, Ukrainian and Russian, and compares the resultant corpora to existing traditional corpora where possible. Sharoff uses similar methodology to that laid out in Baroni and Bernardini (2004) to produce web corpora, but on a much larger scale to generate much bigger corpora. That is, he uses 500 seed terms and 5000-8000 queries. The resultant corpora are over 100 million tokens, demonstrating that using the web to build corpora is a viable option for producing extremely large scale corpora.

Combining the threads seen in Ueyama and Baroni (2005) and Sharoff (2006), Erjavec et al. (2008) use the web to build an extremely large corpus of Japanese for use with The Sketch Engine[5], a large scale corpus query tool. (Kilgarriff et al., 2004). The authors present JpWaC (Japanese Web as Corpus), which contains data from nearly 50,000 URLs, resulting in over 400 million tokens. They use Japanese

[5]`http://www.sketchengine.co.uk`

translations of the top 500 most frequent non-function words from the BNC as seed terms and randomly generate 4-tuples to query Google.

Fantinuoli (2006) provides a different look at building web corpora by utilizing the BootCaT methodology to extract smaller, focused, topic-specific corpora for English, German, and Italian. These corpora are much smaller than those in Sharoff (2006) and Erjavec et al. (2008), ranging from 400,000 tokens (English) to 1.5 million tokens (Italian), and rather than being built as a general corpus as in those efforts, these corpora are intended to contain documents about Leukemia. For the task of term extraction from a corpus, i.e., finding terms that are indicative of a particular subject, the web corpora produce better or equivalent results to hand-built corpora in the tests performed and in a fraction of the time.

5.5 Using BootCaT

We chose to work with BootCaT (Baroni and Bernardini, 2004) to build the web-corpora used in this research. BootCaT is an attractive option because it works well out of the box, and at the time that we collected data, it allowed for large-scale queries using Yahoo! as a search engine. It should be noted that Yahoo! has since changed its policies and that BootCaT now uses Bing and limits the number of queries per month that a single user can make, which may limit its usefulness in developing large scale corpora like the ones presented here. However, the methodologies for collecting relevant corpora are not tied to a particular search engine or to BootCaT itself and could, in theory, work just as well in other web-scraping endeavors.

The BootCaT process is an iterative algorithm to bootstrap corpora, starting with various seed terms. The procedure is as follows:

1. Select initial seeds (terms).
2. Combine seeds randomly.
3. Run search engine queries using the output from Step 2.
4. Retrieve the corpus from the urls returned in Step 3.
5. Extract new seeds via corpus comparison.
6. Repeat steps #2-#5.

For our purposes, the final two steps are unnecessary, as they would abstract away from the specific topics contained in the test corpus, so they are not used.

To use BootCaT with non-ASCII languages, one needs to check the encoding of webpages in order to convert the text into UTF-8 for output, as has been done for, e.g., Japanese (Erjavec et al., 2008; Baroni and Ueyama, 2004). Using a UTF-8 version of BootCaT, we modified the system by using a simple Perl module (`Encode::Guess`) to look for the EUC-KR encoding of most Korean webpages and switch it to UTF-8. The pages already in UTF-8 do not need to be changed. BootCaT also removes boilerplates and cleans the HTML files with various filters.

In gathering training data from the web for Korean particle error correction, we face the challenge of obtaining data which is appropriate both for: a) the topic the learners are writing about, and b) the linguistic construction of interest, i.e., containing enough relevant instances. In the ideal case, one could build a corpus directly for the types of learner data to analyze. Luckily, using the web as a data source can provide such specialized corpora (see, e.g. Baroni and Bernardini, 2004; Fantinuoli, 2006), in addition to larger, more general corpora (see, e.g. Sharoff, 2006; Erjavec et al., 2008). A crucial question, though, is how one goes about designing the right web corpus for analyzing learner language (see, e.g., Sharoff, 2006, for other contexts). Obtaining data is important in the general case, as non-English languages tend to lack resources.

As an initial foray into collecting web corpora for the purpose of generating a well-formed model of Korean for a machine-learning system, we built a variety of corpora while manipulating different conditions and parameters in BootCaT. Along with testing various sets of seed terms aimed at gathering general and focused corpora, we mainly examined three parameters in the initial experiments: number of seeds, tuple-length, and number of queries.

5.5.1 Seed Selection for Korean Web-based Corpora

A crucial first step in constructing a web corpus is the selection of appropriate seed terms for constructing the corpus (e.g., Sharoff, 2006; Ueyama, 2006). In our particular case, this begins the question of how one builds a corpus which models native Korean and which provides appropriate data for the task of particle error detection. The data should be genre-appropriate and contain enough instances of the particles learners know and used in ways they are expected to use them (e.g., as temporal modifiers). A large corpus will likely satisfy these criteria, but has the potential to contain distracting information. In Korean, for example, less formal writing often omits particles, thereby biasing a machine learner towards under-guessing particles. Likewise, a topic with different typical arguments than the one in question may mislead the machine.

Hence there are two basic ways of going about seed selection to build a corpus for our purposes: 1) Use a large set of general terms to build as big of a corpus as possible, ignoring the issue of topic sensitivity, or 2) use a controlled set of seed terms aimed at building a smaller topic-specific corpus. We compare the effectiveness of corpora built with different seeds in training a machine learner in Section 5.5.4.

5.5.1.1 A general corpus

To construct a general corpus, we used a list of words provided by a native Korean speaking Linguist and instructor of Korean that are likely to be in a learner's lexicon. The list is 50 nouns for beginning Korean students for seeds; it includes basic vocabulary entries like the words for *mother, father, cat, dog, student, teacher*, etc.

5.5.1.2 Focused corpus #1

In many tasks, such as those dealing essays written from a prompt, written answers to specific questions, etc., researchers often know what domain, i.e. the subject of a discourse, learner essays are written about. Based on this fact, we experimented with selecting seeds for a more topic-appropriate corpus, with the understanding that the system would be less robust when examining writing that does not relate to the selection of topics represented in the training corpus. Theoretically, with sound methodology and tools for developing web-based corpora, corpora could be built with little effort for dealing with new topics. Such task-specific knowledge is fairly common for many second language writing scenarios, so this approach should be useful for other corpora and languages. We used a smaller set of 10 seed terms based on the range of topics covered in the foreign intermediate (FI) corpus from Lee et al. (2009a) (see Section 4.1 in Chapter 4), shown in Figure 5.2. These terms are, like the aforementioned general corpus seeds, level-appropriate for learners of Korean.

한국 (*hankwuk*) 'Korea'	사람 (*salam*) 'person(s)'
한국어 (*hankwuke*) 'Korean (lg.)'	친구(*chinkwu*) 'friend'
계절 (*kyeycel*) 'season'	가족 (*kacok*) 'family'
행복 (*hayngpok)* 'happiness'	운동 (*wuntong*) 'exercise'
여행 (*yehayng)* 'travel'	모임 (*moim*) 'gathering'

Figure 5.2: Seed terms for the focused corpus

In general, even when working with extremely large web corpora, it is important to gauge the quality of the collected data from a more qualitative perspective. After examining a small corpus built with these seeds, a number of problems were discovered with the corpus collected using these seeds. From initial observations, the difficulty stems in part from the simplicity of the seed terms in Figure 5.2.

First, there are issues concerning collecting data which is not pure Korean. Along with useful sites, there is data extracted from Chinese travel sites, where there is a mixture of non-standard foreign words and unnatural-sounding translated words in Korean. Ironically, I also found actual Korean learner data, as in Figure 5.3. The figure shows a screen shot from one of the pages returned using the seeds in Figure 5.2; the problem here is that the author of the Korean text is clearly a non-native speaker, thus, the language that he uses is likely not suitable for building a corpus of well-formed Korean for training a machine learner.

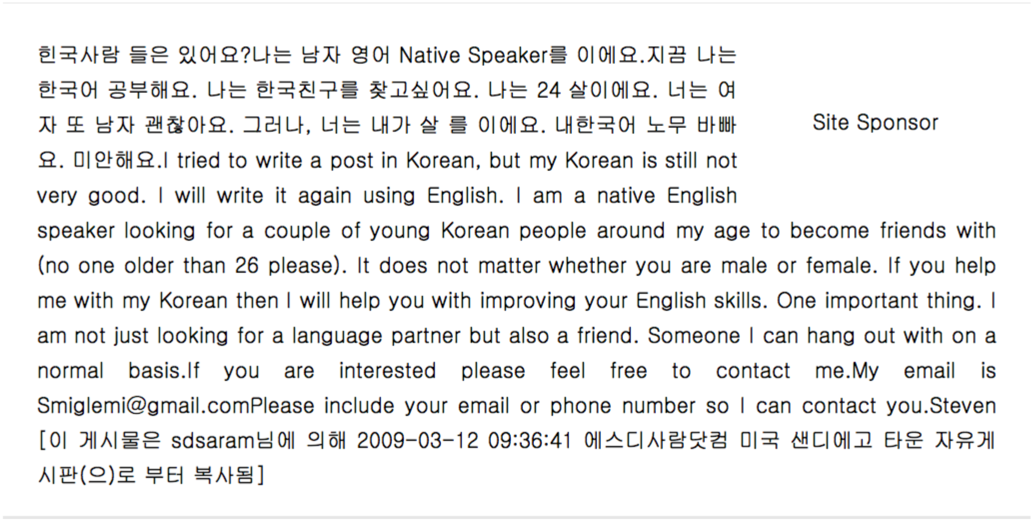

힌국사람 들은 있어요?나는 남자 영어 Native Speaker를 이에요.지끔 나는 한국어 공부해요. 나는 한국친구를 찾고싶어요. 나는 24 살이에요. 너는 여자 또 남자 괜찮아요. 그러나, 너는 내가 살 를 이에요. 내한국어 노무 바빠요. 미안해요.I tried to write a post in Korean, but my Korean is still not very good. I will write it again using English. I am a native English speaker looking for a couple of young Korean people around my age to become friends with (no one older than 26 please). It does not matter whether you are male or female. If you help me with my Korean then I will help you with improving your English skills. One important thing. I am not just looking for a language partner but also a friend. Someone I can hang out with on a normal basis.If you are interested please feel free to contact me.My email is Smiglemi@gmail.comPlease include your email or phone number so I can contact you.Steven [이 게시물은 sdsaram님에 의해 2009-03-12 09:36:41 에스디사람닷컴 미국 샌디에고 타운 자유게시판(으)로 부터 복사됨]

Figure 5.3: Screen Shot of actual webpage in Focused Corpus #1

Secondly, there are topics which, while exhibiting valid forms of Korean, are too far afield from what one would expect learners to know, including religious sites with rare expressions; poems, which commonly drop particles; gambling sites; and so forth. Finally, there are cases of ungrammatical uses of Korean, which are used in specific contexts not appropriate for the purposes of this corpus. These include newspaper titles, lists of personal names and addresses, and incomplete phrases from advertisements and chats. Particle usage is far less frequent in these types of webpages.

5.5.1.3 Focused corpus #2

To avoid some of this noise, the next attempt involved a second set of seed terms, representing relevant words in the same domains, but of a more advanced nature (as judged by a native speaker), i.e., topic-appropriate words that may be outside of a typical learner's lexicon. The hypothesis is that this approach is more likely to lead to native-like, quality Korean. Two advanced words for each topic were selected, as provided in figure 5.4.

kyo-sa 'teacher'	*in-kan* 'human'
phyung-ka 'evaluation'	*cik-cang* 'workplace'
pen-yuk 'translation'	*wu-ceng* 'friendship'
mwun-hak 'literature'	*sin-loy* 'trust'
ci-kwu 'earth'	*cwu-min* 'resident'
swun-hwan 'circulation'	*kwan-kye* 'relation'
myeng-sang 'meditation'	*co-cik* 'organization'
phyeng-hwa 'peace'	*sik-i-yo-pep* 'diet'
tham-hem 'exploration'	*yen-mal* 'end of a year'
cwun-pi 'preparation'	*hayng-sa* 'event'

Figure 5.4: Seed terms for the second focused corpus

The seeds in Figure 5.4 lead to a corpus with a greater percentage of well-formed data, namely data from news articles, encyclopedic texts, and blogs about more serious topics such as politics, literature, and economics. While some of this data might be above learners' heads, it is, for the most part, well-formed native-like Korean. Also, the inclusion of learner data has been dramatically reduced. However, some problems persist, namely the inclusion of poetry, newspaper titles, religious text, and non-Korean data. We will outline our methodology for alleviating some of these problems in Section 5.5.6.2.

5.5.2 WaC Parameters

Next, we will discuss each of the parameters that can be set in BootCaT to control the results of the queries performed to build a web-corpus, namely number of seeds, tuple-length, and number of queries. Each of these can have a decided affect on the quality and size of the resultant corpus.

5.5.2.1 Number of seeds

The first way to vary the type and size of corpus obtained is by varying the number of seed terms. The exact words given to BootCaT affect the domain of the resulting corpus, and utilizing a larger set of seeds leads to increased potential to create a larger corpus. Of course, the number of seeds can also drastically affect the size of the corpus; using more seeds means that more searches can be performed, and thus more pages returned.

For the general (G) corpus, we used: G1) all 50 seed terms, G2) 5 sets of 10 seeds, the result of splitting the 50 seeds randomly into 5 buckets, and G3) 5 sets of 20 seeds, which expand the 10-seed sets in G2 by randomly selecting 10 other terms from the remaining 40 seeds. This breakdown into 11 sets (1 G1 + 5 G2 + 5 G3) allows for the examination of the effect of using different amounts of general terms and facilitates easy comparison with the first focused corpus, which has only 10 seed terms.

As a first pass, for the first focused (F_1) corpus, we used: $F_1$1) the 10 seed terms, and $F_1 2$) five sets of 20 seeds, obtained by combining $F_1 1$ with each seed set from G2. This second group provides an opportunity to examine what happens when augmenting the focused seeds with more general terms; as such, this is a first step towards larger corpora which retain some focus. For the second focused corpus (F_2), I simply used the set of 20 seeds. Thus, there are seven sets of focused seed terms ($1\ F_1 + 5\ F_1 2 + 1\ F_2$). Combining the general and focused sets then, there are 18 different sets of seeds for testing.

5.5.2.2 Tuple length

One can also experiment with tuple length in BootCaT. The shorter the tuple, the more webpages that can potentially be returned, as short tuples are likely to occur in several pages (e.g., compare the number of pages that all of *person happiness season* occur in vs. *person happiness season exercise travel*). On the other hand, longer tuples are more likely truly relevant to the type of data of interest, and thus more likely to lead to well-formed language. We experimented with tuples of different lengths, namely 3 and 5. With 2 different tuple lengths and 18 seed sets, there are 36 different conditions for testing, so far.

5.5.2.3 Number of queries

The final parameter that we explore is the number of queries, i.e. how many web searches one actually runs with the tuples that BootCaT creates. While this parameter may seem trivial at first glance, the maximum number of searches that can be executed is directly connected to the number of seeds and tuples. For 3-word tuples with 10 seed terms, for instance, there are 10 items to choose 3 objects from: $\binom{10}{3} = \frac{10!}{3!(10-3)!} = 120$ possibilities. Using all combinations is feasible for small seed sets, but becomes infeasible for larger seed sets, e.g., with 5-word tuples made up of 50 seed terms, the number explodes to $\binom{50}{5} = 2,118,760$ possibilities. To keep the number of queries feasible, we opted for the following: for 3-word tuples, generate 120 queries for all cases and 240 queries for the conditions with 20 and 50 seeds. Similarly, for 5-word tuples, generate the maximum 252 queries with 10 seeds, and both 252 and 504 for the other conditions. With the previous 36 sets (12 of which have 10 seed terms), evenly split between 3 and 5-word tuples, there are 60 total corpus permutations, as in Table 5.1.

		# of seeds					
tuple	# of	General			F_1		F_2
len.	queries	10	20	50	10	20	20
3	120	5	5	1	1	5	1
	240	n/a	5	1	n/a	5	1
5	252	5	5	1	1	5	1
	504	n/a	5	1	n/a	5	1

Table 5.1: Number of corpora based on parameters

5.5.2.4 Other possibilities

There are other ways to increase the size of a web corpus using BootCaT. First, one can increase the number of returned pages for a particular query. With more pages returned, obviously there is potential to gather a larger corpus. However, as Yahoo! returns the most relevant pages first, pages further down on the list may not contain all of the words in the query, and with longer tuples, one is more likely to get more duplicate documents, which are removed via filter regardless. Hence, as with many other decisions in general for web scraping, there is a trade-off between corpus size and focus. In order to maintain a single methodology for general and focused corpora, we left the number of returned pages per query fixed at 20. regardless of all other parameter settings.

Secondly, one can perform iterations of searching, extracting new seed terms with every iteration. Again, the concern is that by iterating away from the initial seeds, a corpus could begin to lose focus. This option, then, would not boost the quality of a corpus for the purposes of this study.

5.5.3 Quantitative Evaluation

					Ecel		Particles		Nominals	
Corp.	Seed	Len.	Qs	URLs	Total	Avg.	Total	Avg.	Total	Avg.
Gen.	10	3	120	1096.2	1140394	1044.8	363145	331.5	915025	838.7
		5	252	1388.2	2430346	1779.9	839005	618.9	1929266	1415.3
	20	3	120	1375.2	1671549	1222.1	540918	394.9	1350976	988.6
		3	240	2492.4	2735201	1099.4	889089	357.3	2195703	882.4
		5	252	1989.6	4533642	2356	1359137	724.5	3180560	1701.5
		5	504	3487	7463776	2193.5	2515235	741.6	5795455	1709.7
	50	3	120	1533	1720261	1122.1	584065	380.9	1339308	873.6
		3	240	2868	3170043	1105.3	1049975	366.1	2506995	874.1
		5	252	2500	4461889	1784.8	1478894	591.6	3630533	1452.2
		5	504	4735	8412322	1776.6	2654176	560.5	6227603	1315.2
F_1	10	3	120	1315	628819	478.1	172415	131.1	510620	388.3
		5	252	1577	1364885	865.4	436985	277.1	1069898	678.4
	20	3	120	1462.6	1093772	747.7	331457	226.8	885157	604.9
			240	2637.2	1962741	745.2	595570	226.1	1585730	602.1
		5	252	2762	2660463	992.3	915913	341.6	2020172	753.5
			504	4760.2	4832548	1018.6	1652811	347.5	3587007	754.7
F_2	20	3	120	1417	1054925	744.5	358297	252.9	829416	585.3
			240	2769	1898383	685.6	655757	236.8	1469623	530.7
		5	252	1727	4510742	2611.9	1348240	780.7	2790667	1615.9
			504	2680	6916574	2580.8	2077171	775.1	4380571	1634.5

Table 5.2: Basic statistics of different web corpora

To gauge the properties of size, genre, and degree of particle usage in the corpora, independent of application, basic statistics of the different web corpora are given in Table 5.2, where the average is given over multiple corpora for conditions with 5 corpora. We provide the number of seeds (Seed), the length of the queries (Len.), the number of queries (Qs), the average number of URLs returned, the total number of *ecels* (cf. words delineated by white space) in a corpus, the average (Avg.) number of ecels per document, the total number of particles in a corpus, the average (Avg.) number of particles per document, the total number of nominals in a corpus, and the average (Avg.) number of nominals per document. Nominals are determined using a hybrid (trigram + rule-based) morphological tagger for Korean (Han and Palmer, 2004), which utilizes the tagset from the Penn Korean Treebank (Han et al., 2002).

Some significant trends emerge when comparing the corpora in the table. First of all, longer queries (length 5) returned not only more unique webpages, but also longer webpages on average than shorter queries (length 3). This effect is most dramatic for the F_2 corpora. The F_2 corpora also exhibit a higher ratio of particles to nominals than the other web corpora, which means there will be more positive examples in the training data for the machine learner based on the F_2 corpora. This aspect proves to be extremely significant when considering the high rate of particle usage in the learner corpus described in Chapter 4.

5.5.4 Testing the Utility of Web-based Corpora for Machine Learning

The purpose of the web-corpus for this research is to serve as training data for a machine learner. As such, along with any quantitative and qualitative assessment of the language used in the corpora that have been gathered, actually implementing a machine learning task allows for direct comparison of the corpora in an experimental setting. To this end, we set up an initial machine learning experiment to measure the utility of the corpora as training data in a particle presence prediction task. To avoid optimizing initial experiments on the test data utilized in Chapters 7, 8, and 9, we use the same FI corpus described in Lee et al. (2009a) that was used to develop seed terms for testing.

For particle presence prediction, the machine learner is simply trying to predict whether or not there should be a particle in a given context. Note that this is not a pure error detection task, as we will be looking at all nominals in the test corpora. The particle presence prediction task serves as a good measuring stick because it is a simple binary decision, and has broad application over a test corpus. Error detection, by comparison, would have far fewer test instances because there are far fewer errors than nominals in a given learner corpus of Korean.

The experimental setup here is similar to that of the error detection experiments described in Chapter 8, but with a reduced feature set. The text is segmented and POS tagged using a hybrid (trigram + rule-based) morphological tagger for Korean (Han and Palmer, 2004). Each ecel is broken down into: a) its stem and b) its combined affixes (excluding particles), and each of these components has its own POS, possibly a combined tag (e.g., EPF+EFN), with tags from the Penn Korean Treebank (Han et al., 2002). The feature vector uses a five ecel window that includes the target and two ecels on either side for context. Each ecel is presented as four separate features: stem, affixes, stem_POS, and affixes_POS. These features provide lexical and grammatical context for the instance. There are also features for the preceding as well as the following noun and verb, which are a naive attempt at approximating selectional properties. For the noun/verb features, only the stem is used, as this is largely a semantically-based property. Additionally, particles were removed from the context affixes, so as not to rely on surrounding learner particles. The class is then either *Yes* or *No* depending on whether or not there should be a particle for the given context.

An example of the features for these experiments is given in (35), where the system is predicting particle presence for the word *Yenge* ('English') in (35a). We generate the features in (35b). The first five lines refer to the previous two words, the target word, and the following two words, each split into stem and suffixes along with their POS tags, and with particles removed. The sixth line contains the stems of the preceding and following noun and verb, and finally, there is the class (YES/NO).

(35) a. Mikwuk-*eyse* sal-*myense* Yenge-*man-ul* cip-*eyse* ss-*ess-eyo*.
America-in live-while English-only-OBJ home-at use-Past-Decl
'While living in America, (I/she/he) used only English at home.'

b.

Position	ROOT	POS	AFFIX	POS
Word $_{-2}$	Mikwuk	NPR	NONE	NONE
Word $_{-1}$	sal	VV	myense	ECS
Target	Yenge	NPR	NONE	NONE
Word$_{+1}$	cip	NNC	NONE	NONE
Word$_{+2}$	ss	VV	ess+eyo	EPF+EFN
PrevVB	PrevNN	NextVB	NextNN	CLASS
sal	Mikwuk	ss	cip	YES

5.5.5 Web as Corpus Parameters

The tests were all run with the default settings for TiMBL for the results in table 5.3. A baseline of always guessing a particle actually produces a high F-score of 81.7%, but this is driven by the obviously high (100%) recall, at the cost of precision, which is only 69%. The best experimental results were achieved when training on the 5-tuple F_2 corpora, leading to F-scores of 82.37% and 82.67% for the 252 query and 504 query corpora, respectively. Moreover, the precision when using these corpora are around 84%, which beats the baseline by 15%. This finding reinforces the hypothesis that more advanced seed terms result in more reliable Korean data, while staying within the domain of the test corpus. Both longer tuple lengths and greater amounts of queries have an effect on the reliability of the resulting corpora. Specifically, 5-tuple corpora produce better results than similar 3-tuple corpora, and corpora with double the amount of queries of n-length perform better than smaller

	Seeds	Len.	Quer.	P	R	F
Gen.	10	3	120	81.54%	76.21%	78.77%
		5	252	82.98%	77.77%	80.28%
	20	3	120	81.56%	77.26%	79.33%
		3	240	82.89%	78.37%	80.55%
		5	252	83.79%	78.17%	80.87%
		5	504	**84.30%**	79.44%	81.79%
	50	3	120	82.97%	77.97%	80.39%
		3	240	83.62%	80.46%	82.00%
		5	252	83.62%	77.17%	80.25%
		5	504	82.72%	79.66%	81.15%
F_1	10	3	120	81.41%	74.67%	77.88%
		5	252	83.82%	77.09%	80.30%
	20	3	120	82.23%	76.40%	79.20%
			240	82.57%	77.19%	79.78%
		5	252	84.41%	78.54%	81.36%
			504	84.77%	78.42%	81.46%
F_2	20	3	120	81.63%	76.44%	78.93%
			240	82.57%	78.45%	80.44%
		5	252	84.21%	80.62%	82.37%
			504	83.87%	**81.51%**	**82.67%**

Table 5.3: Step 1 (particle presence) results, with various training corpora on the FI corpus

comparable corpora. Although larger corpora tend to do better, it is important to note that there is not an absolute correlation, though larger focused corpora perform slightly better than smaller focused corpora. The general 50/5/252 corpus, for instance, is similarly-sized to the F_2 focused 20/5/252 corpus, with over 4 million ecels (see table 5.2). The focused corpus—based on fewer yet more relevant seed terms—has 2% better F-score. Thus, we used the best settings, i.e. 5 tuples, 504 queries, as the basis for all subsequent web-based training corpora.

5.5.6 WaC Improvements Beyond BootCaT

After reviewing all of the results outlined in Table 5.3 it becomes clear that the methodology that has been utilized up to this point is sound. Furthermore, the focused corpora are, for the most part, in the desired domains, and produce favorable particle prediction results compared with more general corpora (cf. Table 5.3).

With a clear advantage for focused corpora, we move on now to further improving the methodology for collecting topic-specific corpora in at least three ways: 1) further control the topics/content contained in the returned pages such that they are more relevant to the learner data; 2) restrict the amount of non-Korean language returned; and 3) attain a more favorable distribution of particles that more closely matches the frequency of usage in a learner corpus. Each of these improvements are done outside of the actual web scraping process of BootCaT.

The first step, collecting more relevant content, is a pre-processing issue; we present a more robust methodology for seed selection in the following section. The next two steps are both post-processing. To ensure that the corpora exhibit mainly Korean data, we use an encoding filter on returned pages once BootCaT is done. Finally, after collecting the appropriate corpora, we use instance sampling to reduce the number of negative instances, i.e. nominals that lack particles. Using the best parameters from the previous approach, i.e. 504 queries of 5-tuples (see table 5.3, we employ each of the improvements outlined above in building a new web corpus based on the topics present in both the Lee et al. (2009a) FI corpus, as well as a completely different training corpus for the heritage intermediate (HI) corpus described in the same paper.

5.5.6.1 Using sub-corpora

The focused corpora do a better job than the general corpora, so there is some benefit to choosing the best seed terms to suit a given test corpus. To that end, we more thoroughly investigate domain-appropriate seed term selection here. A potential problem with the seed selections described in Sections 5.5.1.2 and 5.5.1.3 is that because all of the terms are put into a single set and then randomly combined into tuples for web querying, the potential to get unwanted topics remains. For example, if the corpus *Health Management* and *The Generation Gap* as topics, words such as *pyengwen* ('hospital') and *kaltung* ('conflict') are possible seed terms. This becomes problematic when these words occur together in a query, as pages returned could be about, e.g. war, a topic which fits neither category, and is not represented in the learner corpus at all.

Rather than use all of the seed terms to create a single corpus, a better approach is to divide the seed terms into separate sets, based on the individual topics from the learner corpus. Then sub-corpora can be combined to create a cohesive corpus covering all the topics. For example, use 10 *Travel* words to build a subcorpus, 10 *Learning Korean* words for a different subcorpus, and so forth. This means that terms appropriate for one topic are not mixed with terms for a different topic, ensuring more coherent web documents.

In this case, we use the 8 topics in the FI-inspired training corpus, with 10 terms each, for a total of 80 seeds. Likewise, we use 13 topics, with 10 terms each, for a total of 130 seeds for the HI-inspired training corpus. The full sets of seed terms for each training corpus are given in appendices A and B for the FI and HI versions, respectively. The seed terms can be easily changed for other data.

5.5.6.2 Filtering

One difficulty with the previously built corpora is that some of them have large amounts of other languages along with Korean. The keywords are in the corpora, but there is additional text, often in Chinese, English, or Japanese. These types of pages are unreliable for the purposes of building native-like Korean corpora, as they may not exhibit natural Korean. While there is an option available in BootCaT to specify the particular language of interest, this setting alone, at least for Korean, does not do a great job of removing non-Korean text. By using a simple encoding filter, we can check whether a majority of the characters in a webpage are indeed from the Korean writing system, and remove pages beneath a certain threshold. The pages are removed, rather that retaining what remains after removing the offending sentences to ensure that the quality of the data remains high; we can not be certain that these pages were written by native Korean writers.

Setting the threshold is non-trivial. First of all, it is plausible and fairly common for native speakers of Korean to use other languages, and therefore other character sets, in their own writing. Indeed, Chinese characters are permissible in Korean. Secondly, one must consider the size of the corpus. With the filter set too high, the corpus size dwindles to such a small number that it becomes unusable for training a machine learner. We experiment with various thresholds in Table 5.4.

For the discussion that follows, all experiments are done using maximum entropy (MaxEnt) learning. The feature set is the same as the one used in Table 5.3. Table 5.4 shows the results of the MaxEnt system for step 1, using training data built utilizing the seed selection approach described in Section 5.5.6.1 on the FI corpus with filter thresholds of 50%, 70%, 90%, and 100%—i.e., requiring that percentage of Korean characters—as well as the unfiltered corpus. The best F-score is with the filter set at 90%, despite the size of the filtered corpus being smaller than the full corpus. Accordingly, we use the 90% filter on the training corpus for the experiments described below.

Threshold	100%	90%	70%	50%	Full
Ecel	67k	9.6m	10.3m	11.1m	12.7m
Instances	37k	5.8m	6.3m	7.1m	8.4m
Accuracy	74.75%	81.11%	74.64%	80.29	80.46%
Precision	80.03%	86.14%	79.65%	85.41%	85.56%
Recall	84.50%	86.55%	84.97%	86.15%	86.23%
F-score	82.20%	**86.34%**	82.22%	85.78%	85.89%

Table 5.4: Step 1 (particle presence) results with filters on the FI corpus

5.5.6.3 Instance Sampling

As mentioned in Section 5.4, It is not always the case that the webpages contain the same ratio of particles as learners are expected to use. To alleviate the over-weighting of having no particle attached to a noun, one solution is to downsample the corpora for the machine learning experiments, by removing a randomly-selected proportion of (negative) instances. Instance sampling has been effective for other NLP tasks, e.g., anaphora resolution (Wunsch et al., 2009), when the number of negative instances is much greater than the positive ones. In the web-based corpora, nouns have a greater than 50% chance of having no particle; we thus down-sample to varying amounts of negative instances from about 45% to as little as 10% of the total corpus. For the sake of comparison, about 70% of the nominals in the Lee et al. (2009a) corpus should have a particle.

One could explore matching the distribution of particular particles to an appropriate learner distribution. For the comparable English case, this would involve, for example, ensuring that the training distribution of *for* is approximately the same as in the learner data. We do not explore this, as it is likely that such ratios are not consistent across learner data.

The results for instance sampling are given in table 5.5. We experiment with positive to negative sampling ratios of 1.3/1 (≈43% negative instances), 2/1 (≈33%), 4/1 (≈20%), and 10/1 (≈10%). The 90% filter and 1.3/1 downsampling settings are applied to the training corpus (Section 5.5.5) for all experiments carried out in the rest of the dissertation.

P/N ratio	10/1	4/1	2/1	1.3/1	1/1.05
Instances	3.1m	3.5m	4.3m	5m	5.8m
Accuracy	74.75	77.85	80.23	81.59	81.11
Precision	73.38	76.72	80.75	84.26	86.14
Recall	99.53	97.48	93.71	90.17	86.55
F-score	84.47	85.86	86.74	**87.12**	86.34

Table 5.5: Step 1 (presence) results with instance sampling

One goal has been to improve the web as corpus corpus methodology for training a machine learning system. The results in tables 5.4 and 5.5 reinforce the earlier finding that size is not necessarily the most important variable in determining the usefulness or overall quality of data collected from the web for NLP tasks. Indeed, the corpus producing best results (90% filter, 1.3:1 downsampling) is more than three million instances smaller than the unfiltered, unsampled corpus. For more discussion on smaller, controlled corpora producing optimal results, see Khan et al. (2013).

5.6 Utilizing Online Language Learning Data

With the rise of social networking service (SNS) sites such as Twitter, Facebook, MySpace, etc have come some specialized SNS. One area of focus is language learning. The idea behind these sites is that users will write essays in a language they are trying to learn hoping that users of the site that speak that language fluently will edit the document to point out mistakes and make corrections. Essentially, the hope is that with a large user base, there will be a great deal of overlap of users who speak language X and want to learn language Y with users who speak language Y and want to learn some language, and that users will provide useful feedback for one another.

Some popular social networking sites for language learning include Livemocha, smart.fm, and Lang-8, all of which have slightly different designs and features. Livemocha[6] has been looked at in a variety of recent studies of social networking service for language learning (SNSLL) (cf., Jee and Park, 2009; Liaw, 2011; Harrison, 2013). The site currently boasts 16 million users from nearly 200 countries and supports over 35 languages. There are three key learning areas: lesson content, the global community, and a motivational system (Jee and Park, 2009). Users can participate in interactive settings, improving speaking and listening skills, and submit written essays both of which can be reviewed by peers. The site also features language lessons, some of which are similar to those found in Rosetta Stone, available in free and paid subscriptions (Liaw, 2011). Users are also invited to join peer groups and find partners for speaking and/or writing practice. It is worth noting, however, that students are not allowed to write about any topic that they wish on Livemocha (Mizumoto et al., 2011), but rather must choose from a set number of topics.

Lang-8 is a similar site that currently has over 700,000 users that focuses on allowing users to write freely on any topic in their language of study and then having native speakers of that language immediately begin giving feedback. Learners are always encouraged to make corrections to other members' writings, facilitating a "language exchange". This exchange is the sole purpose of Lang-8, i.e. there are no online language lessons available through the site. Thus, the data that can be found on Lang-8 is a potentially more natural form of SNS for language learning, as all

[6] `livemocha.com`

the learning is done through interactions, and writers have the freedom to write about whatever they choose.

For many of these sites, errors in written essays are marked up by annotators and stored in revision logs. While obtaining the logs themselves might not always be easy, or even possible, they can be an invaluable resource for the error correction community. The revision logs provide an option for gathering annotations on a large scale at very little cost to researchers. However, the data is vulnerable to a fair amount of noise; corrections may not be correct, using POS tools can introduce problems, etc. As such, developing methodology for using data from these types of sources should prove extremely useful for all languages, but especially for languages like Korean, which have a growing number of learners but a small number of available learner corpora. To this end, we explore extracting an annotated corpus of particle errors from the Lang-8 website revision logs.

5.7 Lang-8 as a Source for Learner Data

Mizumoto et al. (2011) describe an approach to Japanese error correction that utilizes error-revision logs from Lang-8 (see also Cahill et al. (2013b) for extracting English preposition errors from lang-8). Using the HTML from the website, they develop a simple metric to determine annotations based on corrections to the sentences. The tag `<span class="sline">` creates a strikethrough and denotes that the text between the tags should be removed, for example, *strikethrough* would be rendered ~~strikethrough~~ if surrounded by the `sline` tags. `<span class="red">` and `<span class="f_blue">` indicate red and blue coloration of a word, respectively. These are used somewhat arbitrarily by annotators on the site, but often indicate some sort of change or insertion. The Lang-8 website does not require, or even suggest, specific uses for each tag, but does encourage editors to "use the correction tools in a way that is easy for the recipient to understand your meaning.[7]" It should be noted that the HTML tags referred to above are the ones that are reported in Mizumoto et al. (2011). In the version of the Lang-8 data used for the research described in this dissertation, the tags are `[f-sline]`, `[f-red]`, and `[f-blue]`.

In Mizumoto et al. (2011), the authors are trying to construct corrected sentences, so they remove anything inside the `sline` tags and then remove all HTML tags, assuming that everything left is corrected text. The approach described in the paper involves statistical machine translation to correct entire sentences, rather than focusing on a specific error type. For this dissertation, on the other hand, we focus on particles,

The Lang-8 revision logs[8] are saved as json files such that each user-submitted writing (referred to in this work as essays) is given its own entry that consists of: 1) a unique ID string, 2) the language of study, 3) an array that contains each sentence from the original text as an item, and 4) an array of arrays where each outer array corresponds to a sentence from the array in (3) and each inner array is a list of suggested corrections for that sentence. Note that there can be multiple corrections suggested for any sentence, as multiple users might have different ideas for what would be the best correction for a given sentence.

We developed an algorithm to automatically extract corrected sentences from the Lang-8 revision logs and provide particle annotation similar to that described in Chapter 4 to create a large-scale particle error annotated corpus of Korean. Figure 5.5 provides the steps that we take in creating this corpus.

[7]As of 2/2/2014, as indicated at `http://blog.lang-8.com/post/19885729150/how-can-i-correct-an-entry`

[8]Data kindly provided by Mamoru Komachi.

1. Remove all text between `sline` tags, and remove all other tags from all candidate corrections (c.f., Mizumoto et al. (2011))
2. Compare original text to each of the candidate corrections in order to determine which correction requires the least amount of change from the original to produce grammatical output
3. Split both original and corrected text into white-space delimited tokens
4. Segment and tag both the original and corrected text
   - We use Han and Palmer (2004)'s morphological tagger
5. Find single morpheme differences in tagged output of original and corrected text
6. If a difference occurs in an ecel where the root is tagged as a noun:
   - Consider the particle(s) attached to the root noun from the corrected text as the correct particle(s) (CP)
   - If there is no particle, CP is assigned a value of "NONE"
   - Find the corresponding noun in the original text and consider the particle(s) attached to the root as the original particle(s) (OP)
7. If there is no difference for a given noun, assign the particle(s) to both the CP and OP for that noun.
8. Determine error type (null, omission,substitution, or commission, cf. Section 4.3.7) by comparing CP to OP.

Figure 5.5: Algorithm for Gathering Data from Lang-8 Data

In step 1, the methodology described in Mizumoto et al. (2011) for extracting learner-annotator sentence pairs is applied to each json object. This is all that is necessary to capture the changes in sentences, i.e. all that is necessary for an SMT approach. The steps that we describe below involve much more processing of the Lang-8 data. These steps are necessary for creating an annotated corpus that provides more detailed information about a grammatical feature, particles in this case.

In step 2, we use the difflib module from Python, specifically the `ratio()` function from `SequenceMatcher`, to get a numeric representation of overall similarity between each learner-annotator sentence pair. This value is a float between 0 and 1.0, where 1.0 denotes an exact match, and 0 denotes no overlap between the sentences. We restrict the choice of the sentence to be the one that is closest to 1.0 and that it must be above 0.5. There are a number of reasons for these restrictions. First of all, we want to get sentences that require as few edits as possible to get back to the original sentence. If there are multiple possible corrections for a given sentence, the hope is that the selected version will reflect the least amount of work necessary to produce a grammatical version of the learner sentence. Allowing for more edits opens up the possibility that the annotator has changed the sentence far beyond a recognizable version of the original. Secondly, there are instances in the error logs where the annotator will add editorial remarks or clarifications to the original sentence that can be difficult to recognize automatically. Given these cases, the less similar the sentence (i.e., the closer to 0 according to the `ratio()` score), the more likely that the annotator has added information to the sentence that is not necessary for the purposes of this corpus. Any sentences for which no corrections exist that meet these criteria are discarded, as the data might not be reliable enough for this task.

Figure 5.6 provides an example of step 2 from Figure 5.5. Here, Annotator A and Annotator B both make changes to the sentence provided by the learner. Annotator B's corrections produce a higher `ratio` score, meaning that this sentence is more like the original than the corrections suggested by Annotator A. Thus, the Annotator B's version of the sentence is retained as the "correct" version of the sentence.

Next in steps 3 and 4, the data is prepared and tagged using the Han and Palmer (2004) morphological tagger. The tagged output is used in making decisions in the coming steps. Using the tagger here raises a couple of issues worth addressing at this point. First of all, the tagger preforms the necessary task of identifying mor-

Learner: 새로운 시스템이 설치된 후에 곧바로 다시 비밀번호를 못 설치한 것은 너의 실수였다~
Annotator A: 새로운 시스템이 설치된 후에 곧바로 다시 비밀번호를 설정하지 못한 것이 나의 실수였다.
difflib ratio(Learner,AnnotatorA) = 0.58333
Annotator B: 새로운 시스템이 설치된 후에 곧바로 다시 비밀번호를 설치 안했던 것이 나의 실수였다~
difflib ratio(Learner,AnnotatorB) = 0.66667

Figure 5.6: Example of Lang-8 extraction step 2

pheme boundaries. Finding these boundaries is a non-trivial step in this process as Korean features a rich agglutinative morphological system, and the particles that are under examination are bound morphemes that can be difficult to identify (see Chapter 3). However, the tagger is not always accurate, especially with noisy learner data, so some inaccuracies are likely added at this step. Figure 5.7 provides an example of the tagged output produced in these steps for the *Learner* sentence as well as the *Corrected* version.

Learner: 새롭/VJ+은/EAN 시스템/NNC+이/PCA 설치되/VV+ㄴ/EAN 후/NNC+에/PAD 곧바/NNC+로/PAD 다시/ADV 비밀번호/NNC+를/PCA 못/ADV 설치하/VV+ㄴ/EAN **것/NNX+은/PAU** 너/NPN+의/PCA 실수였다~/NNC ./SFN
Corrected: 새롭/VJ+은/EAN 시스템/NNC+이/PCA 설치되/VV+ㄴ/EAN 후/NNC+에/PAD 곧바/NNC+로/PAD 다시/ADV 비밀번호/NNC+를/PCA 설치/NNC 안하/VV+었/EPF+던/EAN **것/NNX+이/PCA** 나/NPN+의/PCA 실수였다 /NNC

Figure 5.7: Example of Lang-8 extraction steps 3 and 4

In step 5, Python's difflib module is employed again, utilizing the `compare()` function from `Differ`. The function accepts two arrays, the correct tagged sentence and the learner tagged sentence split on white space in this case, and returns a list that provides each element of the first list with how it relates to the second in terms of equality, addition (+), or subtraction (−). Substitutions are treated as a subtraction and an addition. Using `compare()` on the arrays of words from each version of the sentence allows us to isolate the changes made by the annotator. In Figure 5.8, partial output from the `compare()` function is given for the same sentences in Figures 5.6 and 5.7. A - or + indicates that the text in question is removed or added, respectively, from the *Learner* sentence to create the *Corrected* sentence. In the case of a substitution, a - line is immediately followed by a + line, as shown in the second and third lines in Figure 5.8, where 를 (an object marker) has replaced 가 (a subject marker) after the noun 자료 ("material").

```
    모두/ADV
-   자료/NNC+가/PCA
+   자료/NNC+를/PCA
-   다/ADV
    되찾/NNC+을/PCA
    수/NNC+가/PCA
    없다/NNC
    ./SFN
```

Figure 5.8: Example of Lang-8 extraction steps 5

In steps 6 and 7, we check for changes made to words tagged as nouns and identify what particles, if any, were used by both the learner and annotator. Considering again the noun 자료 in Figure 5.8, we see that the learner used the subject marker, and the annotator corrected it to an object marker. Finally in step 8, with the knowledge gained in steps 6 and 7, we assign the corresponding error types to the nouns in question. In the case of the situation in Figure 5.8, the particle would be marked as a substitution error. Note that in the cases of changes made to other words in the sentence, the corrected version is always accepted (this means that the resultant corpus features corrected forms of words and sentences).

The restrictions put on the extraction process cut a fair amount of data out of the original Lang-8 dump; the original json object had 21,779 essays, only 16,643 remained after processing. This means that for 5136 of the essays there were either no annotations provided, or the annotations were not of the quality that the algorithm for extraction required for inclusion.

The resultant corpus, referred to henceforth as Lang-8-Ko, is a large, manually annotated, automatically collected corpus of learner Korean with error markup that is very similar to the scheme described in Chapter 4. The final corpus has 818,342 tokens in 75,362 sentences. This corpus is considerably larger than the KoLLA corpus, which has 11,660 tokens in 1410 sentences. The tokens per sentence ratio is 10.85, higher than all four of the sub-corpora in KoLLA. Whatever the underlying reason, this data has some significant differences from the KoLLA corpus.

5.7.1 Generating Confusion Sets from Lang-8 Data

Aside from serving as an annotated learner corpus, there is another useful purpose that the Lang-8 data can serve. That is, it can be used to refine existing data. While all of the work described in Section 5.5 results in a web corpus that will serve well as a starting point for training machine learners for particle error detection, there is still at least one major hurdle left in optimizing training data for the task. The issue here is what particles should be allowed to be guessed by the classifier.

There are hundreds of particles in the Korean language, but many of these are not used often, and certainly not by learners in an academic setting — 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, to optimize the system, the training data should only include particles which can reasonably be expected to appear in a given context. Hence, the task of determining what particles are licensed and/or likely becomes relevant, as it could drastically reduce the number of particles from which the classifier needs to choose, which can lead to overall system improvement (see, e.g. Rozovskaya and Roth, 2010a).

Ideally, one could simply generate confusion matrices from a hand-annotated learner corpus like KoLLA. This approach might be possible for languages like English that have such resources available. However, as detailed in Chapter 4, Korean does not have a large community of researchers working on annotating particles. Unfortunately, the size and scope of the KoLLA corpus is limited, meaning that confusion sets generated from it alone might have gaps that would lead to poor performance for classifiers. The Lang-8-Ko corpus, while not an example of carefully annotated data, can still serve as a great resource for error detection. In the case of constructing confusion sets, this data is more ideal than a smaller corpus like KoLLA simply due to the fact that it is much larger, and thus less affected by idiosyncrasies and noise.

Drawing on the idea of reduced training sets in Rozovskaya and Roth (2010a), we built confusion sets from the Lang-8 data described in Section 5.7 that provide the set of particles that are confused for a given particle, along with their counts.

It should be noted that some particles are allomorphs of a single morpheme, with the surface form being controlled by regular phonological rules. These are reduced to the form that would follow a vowel to lessen the effect of data sparsity, as it is trivial to produce the correct variant. These reductions are given in Figure 5.9. i and ka are both mapped to ka, etc.

Notably, some of the members of a confusion matrix built in such a way might not actually belong there. In the case of using an automatic tagger with Korean learner data, there is potential for things to be tagged as particles that should not be tagged as such, and hence, should not be part of a particle confusion set. Moreover,

이	⇒	가
은	⇒	는
을	⇒	를
으로	⇒	로
와	⇒	과

Figure 5.9: Allomorph Reduction of Particles

some actual particles that could be used as replacements are probably very low in frequency and could confuse the system. Such noise should be relatively low given a large enough data set, but should still be removed when building the final confusion sets.

After compiling the list of all possible insertion candidates, it is pruned such that the frequency of any particle in the list must be at least 10% of the most frequent particle. For example, if 가 (*ka*) appears 100 times as the most frequently inserted particle, then any particle appearing less than 10 times would be removed. Figure 5.10 provides the resultant confusion sets for the most common source particles. *Source* is the particle used by the writer, and the confusion set is the set of particles most often used to correct them. We present only the sets for omissions (*null*) and five of the most commonly mistakenly substituted particles in Lee et al. (2009a), as well as the set of particles inserted in the case of missing particles. The five source particles in the table account for over 80% of the substitution errors in that corpus. The confusion sets will be utilized in Chapters 7, 8, and 9.

Source particle	Confusion Set
null	가, 는, 를, 에, 의
가	는, 를, 에, 의, 도
는	가, 를, 에
를	가, 는, 에, 로, 도
에	에서, 가, 는, 를, 로, 의
에서	에, 를, 로

Figure 5.10: Confusion Sets for Particle Errors

Conclusion

This chapter has outlined the way that we constructed and optimized training data for the machine learning components of the error detection and correction systems that we use in the current approach. Along the way, we also discussed the reasoning for utilizing web-based corpora rather than relying on prebuilt Korean corpora by pointing out shortcomings (as pertaining to the current task) of the corpora that are currently available. We also outlined a methodology for building a large scale particle error-annotated corpus of Korean from the language learning website Lang-8. Combined with the learner corpus described in Chapter 4, we have all of the data necessary to move on to running error detection and correction experiments.

Chapter 6

An Overview of the Error Detection/Correction System

Everyone has a plan 'till they get punched in the mouth.

— MIKE TYSON

In Chapters 4 and 5, we laid out the methodology for procuring and, where possible and/or necessary, annotating Korean data so that a particle error detection/correction system could be constructed and tested. In this chapter, we will give a bird's eye view of the system itself, before going into more detail in Chapters 7, 8, and 9 about how we handle omission error detection, substitution error detection, and particle selection (i.e. correction), respectively.

6.1 Identifying the Issues for Particle Error Detection

The first step in building a system to detect and correct grammatical errors is to know exactly what types of errors exist, as well as how the expected prevalence of each type of error. In Section 4.3.7, we identified and defined four types of particle errors: omissions (leaving out a necessary particle), substitutions (use of the wrong particle), commissions (adding an unnecessary particle), and ordering errors (having the correct particles attached to a nominal, but in the wrong order). Considering the numbers of errors presented in Table 4.6, we found that in the Lee et al. (2009a) corpus, omissions accounted for 48% (432/907) of the total errors and substitutions accounted for another 45% (405/907), with commission errors accounting for the remaining 7%. Note that Lee et al. (2009a) did not consider ordering errors as a separate error type, as they are extremely infrequent and could be considered a substitution error, depending on how segmentation factors into one's annotation scheme (for more on this, see Section 4.3.5). Using all of this information, we can work on designing a system that will be of the most benefit for users while also optimizing performance.

Because omission errors and substitution errors make up over 90% of the errors in the corpus presented in Lee et al. (2009a), we decided to focus specifically on building error detection for these types of errors. Commission errors are uncommon among error types, and ordering errors are even less common. Moreover, given the annotation scheme utilized for the KoLLA corpus, commission and ordering errors could be recognized by a substitution error classifier.

6.2 Development and Test Data

Three of the major contributions for this work involve data: the annotation learner corpus described Chapter 4 (KoLLA), the acquisition and optimization of a Korean web corpus to serve as training data for machine learners, and the use of Lang-8 revision history to develop both a particle-annotated corpus and confusion sets for learner particle errors (the web corpus and Lang-8 are both described in Chapter 5). All of these data sets will play important roles in the error detection and correction experiments that follow.

The KoLLA corpus was developed in large part to serve as a test set for error detection/correction efforts (though it is also a useful resource for other endeavors), and that is the role that it will fill in this work. However, as it is a relatively smaller corpus, under 12,000 tokens, significant questions arise in how to best utilize it for testing. Splitting the data into development and testing sets will mean that the resultant splits may be too small to adequately deal with some phenomena. This issue is dealt with in detail in Section 6.2.1. Another option, then, is to use the Lang-8-Ko corpus described in Section 5.7 as development data, thus leaving all of KoLLA to serve as unseen test data. This option brings its own set of potential difficulties, as explained in Section 6.2.2.

The issue of the size of the development set is strongly tied to the nature of the errors under consideration. Omission error detection can be defined as a binary classification task of guessing whether or not a particle should be present. That is, the system is only concerned with finding spaces where any particle, as opposed to some specific type of particle, is necessary; it is a fairly homogenous set of errors. Thus, developing on a subset of the KoLLA corpus should be sufficient. In the case of substitutions, on the other hand, every particle has a somewhat unique confusion set (cf. Figure 5.10). Also, the task is somewhat more difficult than finding omission errors, as the distinguishing factors that necessitate one particle rather than another are often subtle. As such, to generalize from development to testing, it is likely necessary to have a large base of instances for each confusion set.

6.2.1 Splitting the Corpus for Experimentation

There are a number of ways to go about splitting the KoLLA corpus into development and test sets. We will discuss the merits of two methodologies here. The first, and easiest, way is to select one of the four natural sub-corpora (i.e. Foreign Beginners, Foreign Intermediates, Heritage Beginners, and Heritage Intermediates) to use as the development set and test on the other three. The other way to handle the problem is to randomly split the data into development and test sets.

The first method, i.e. using one of the predefined sub-corpora as a development set, was used in a couple of pilot studies, mainly because the corpus had been a work in progress, and only one set was completely annotated. However, there are a number of drawbacks to this method. Chiefly among them is the question of which corpus to use as the development set. We can expect Foreign Beginners and Heritage Intermediates to be at opposite ends of the spectrum in terms of how fluent their language will be. Choosing one of these corpora would mean either: A) optimizing the system to work with the least native-like Korean and hoping it does well on more advanced language, or B) optimizing on more native-like constructions and hoping to also be able to handle less fluent Korean. In other words, all of the learner groups can be expected to show different linguistic tendencies based on their level of familiarity and comfort with the language, so a system that works extremely well for one, may perform much worse on a different level of learners.

Turning now to a more traditional random split of the data into development and test sets, there are a number of questions with this approach, especially when dealing with a corpus of this size. First of all, how much data should be put into each set? With a relatively smaller corpus like KoLLA, this question is non-trivial; too large a development set and there's not enough testing data to make strong judgements about system performance, too small a development set and the chance that there are instances that are different from what the system has been developed to handle increases. The next question is, should the split be done according to the number of essays, sentences, tokens, or errors in the data? Again, because the resultant sets will not be very large, this decision must be given due attention. If splitting based on the number of essays, sentences, or tokens it is possible that the error rates in the development and test sets will be far apart, however, splitting to attain equivalent error rates could lead to one corpus being much larger based on total words than the other. Finally, there is the question of taking random sentences or essays, or taking an equal amount of sentences or essays from each learner-type.

We opted to split the data into random development and test sets. The data is split according to essays, rather than sentences, as some of the features for machine learning are discourse dependent, and thus could not be extracted from single sentences. Around 20% of the data was put into the development set, with the remaining 80% in testing. The split was made with an eye towards getting similar error rates across both corpora, while maintaining the 20%-80% data split overall.

Still, despite taking care to try and maintain a reasonable amount of both development and test data for error detection, the size of the development corpus is prohibitive. The resultant development set has 48 omission errors, and only 38 substitution errors (plus 14 commission errors and three omission errors that could be caught by the substitution error detector). We use the development and test sets for the omission error detection experiments in Chapter 7, as omission errors can be expected to pattern fairly similarly to one another. On the other hand, considering that a substitution error can occur for any given particle, this is not likely to be enough development data to generalize to a larger, or any, test set. Thus, we will present only optimal results for the entire corpus. In order to provide a fair assessment of how the system performs when optimized and applied to unseen data, we explore using the Lang-8-Ko corpus (cf. Section 5.7) as a second evaluation corpus.

6.2.2 Lang-8 as Development Data

As outlined in Section 6.2.1, the relatively small size of the KoLLA corpus makes utilizing one part of it for development and another for testing unreliable for some tasks. Lang-8-Ko, on the other hand, is certainly a large corpus, but has questions regarding its overall quality in comparison to KoLLA.

The main issue here is that while KoLLA was hand annotated by Korean language experts, Lang-8-Ko was developed by utilizing markups provided by speakers of unknown expertise levels and producing KoLLA-style annotations by means of an algorithm. Because the Lang-8 contributors do not have a standard rubric to follow, the annotations may not present a strong view of the definition of grammaticality that we used when designing the annotation scheme for KoLLA (cf. Section 4.3.6). Thus, one cannot help but wonder how reliable a single annotation may be. The hope is that given the size of the corpus, over 818,000 tokens, a somewhat stable form of grammatical Korean can be gleaned from generalizations made over the entire corpus.

Another issue that causes concern here is the authors of the Lang-8 essays. In the case of KoLLA, we know the learner level and heritage status of each learner who contributed to the corpus. We have no such information about the Lang-8 learners. We do know that the overall tokens-per-sentence ratio is higher in Lang-8-Ko (10.9) than for any subcorpus in KoLLA (ranging from 5.5 to 10.5). This could mean that the sentence structures used in the majority of Lang-8-Ko will not compare well to those in KoLLA, especially in the case of the two beginner subcorpora.

Finally, in KoLLA we know every topic that was available for students to write about, as the essays were conducted in Korean learning classes. Lang-8 has no such restrictions, thus neither does Lang-8-Ko, obviously. This could prove problematic as the training data was developed specifically for use with KoLLA, after establishing that a smaller, controlled corpus could serve as ideal training data for machine learning experiments in Section 5.5.4. One possible solution to this problem would be to utilize topic modeling techniques to identify the texts in Lang-8-Ko that are most similar to the topics in KoLLA, however, this could theoretically greatly reduce the size of the corpus, lessening its utility for the current research.

Despite these data quality questions, we utilize Lang-8-Ko as development data for the substitution error detection experiments in Chapter 8. We split the data into ten random, roughly equally sized splits, so that we can use nine for development while leaving the tenth as unseen test data.

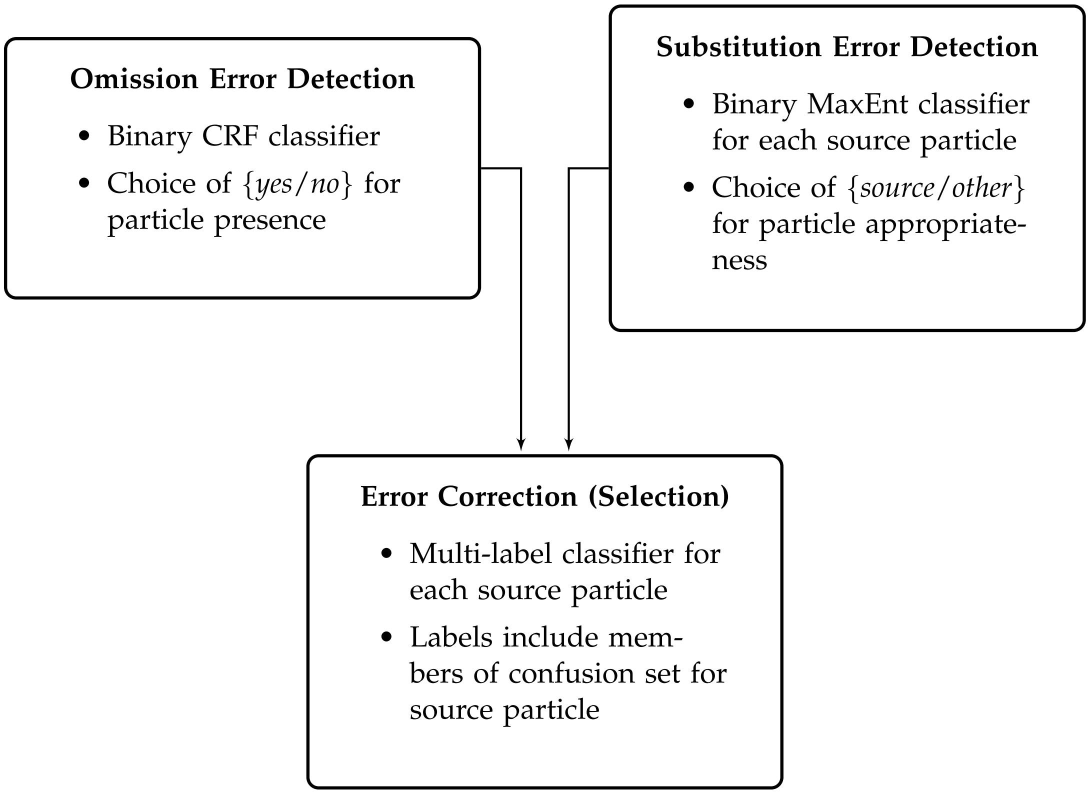

Omission Error Detection
• Binary CRF classifier
• Choice of {yes/no} for particle presence
Substitution Error Detection
• Binary MaxEnt classifier for each source particle
• Choice of {source/other} for particle appropriateness
Error Correction (Selection)
• Multi-label classifier for each source particle
• Labels include members of confusion set for source particle

Figure 6.1: Korean Particle Error Detection/Correction Pipeline

6.3 An Error Correction Pipeline

We view the task of detecting and correcting errors as two separate steps, each with its own classification (cf. Gamon et al., 2008), rather than using a single selection classifier to perform both tasks. Further, we develop different systems for omission and substitution error detection. Figure 6.1 provides the three main components in the pipeline approach that we use.

As can be seen in Figure 6.1, we set up both the omission and substitution error detection tasks as binary decisions. For omissions, this is straightforward, as the classifier examines any nominal with no particle, and simply decides if any particle should be there, with simply *yes* or *no* labels. For substitutions, the binary task is perhaps a little bit less clear. We examine any nominal with a particle and employ a classifier that has been trained such that the labels are either the existing particle, or *other*, where the *other* instances are a made up from all members of a confusion set made by checking what particles are used to correct the target particle in Lang-8 (see Section 5.7.1 for discussion on construction of the confusion sets). Note that the error detection classifiers do not have to be performed in a specific sequence, as they are focusing on non-overlapping phenomena. Chapters 7 and 8 provide more details and results for the omission and substitution error detection systems, respectively.

The correction component of the pipeline is a similar classifier to the one employed for substitution error detection, except for one critical aspect: the set of labels includes all actual members of the confusion set. That is, rather than being a binary choice, this classifier is designed to actually select the best candidate. Selection of particles is a very difficult task given the wide range of functions that particles can perform in Korean (cf. Section 3.2.4). In restricting the difficult, multi-label classification task to the final step of the pipeline, we need only run selection when we have established that the learner was incorrect, which is a relatively small number of times for most learner essays given the large number of particles used in formal Korean. The selection classifier is detailed along with results in Chapter 9.

6.4 POS Tagging

Another issue that can be addressed in regards to the entire system is that of POS tagging the data sets (KoLLA and Lang-8-Ko) in order to extract POS features for the classifiers. We use the same morphological tagger for Korean that is utilized for the training data (see Section 5.5.3). The tagger is run on the data assuming correct spacing of nouns and particle in order to give it the best possible chance at accuracy. However, for a task focusing on particles, accuracy is indeed a concern.

The numbers for nouns and particles in KoLLA, as according both to the annotation and the tagger are given in Table 6.1, as well as the number of tokens.

Sub-corpus	Tokens	Nouns		Particles	
		Anno.	Tagged	Anno.	Tagged
FB	1997	714	1073	654	705
FI	3917	1412	1989	1206	1274
HB	2715	961	1350	881	956
HI	3031	1134	1537	980	1018
Combined	11660	4221	5949	3721	3953

Table 6.1: KoLLA Corpus Noun and Particle POS Tags

Looking at the entire corpus, the tagger appears to be drastically over-guessing noun and particle tags. There are 4221 words marked as nouns in the annotated corpus, but the tagger identifies 5949, over 40% more than are actually in the corpus. The problem is not as bad for particles, with 3721 in the annotations, and 3953 (6% more) in the tagger output. Still, the tagger does not have high accuracy on this corpus, which could prove to be problematic for classifiers using POS features. We chose to utilize such features, nonetheless, as the high type-token ratio for Korean combined with a smaller amount of training data means that features based on raw text will likely be prone to sparsity issues. While we do not have gold-standard versions of Lang-8-Ko or the web-scraped training corpus to check the accuracy the tagger on those data sets, we assume that some amount of tagger error is present in both.

6.5 Shared Features

The experiments run in Chapters 7, 8, and 9 all rely on a shared base of features that are mainly composed of words and POS tags in the surrounding context of each possible insertion point, where tags are derived from a POS tagger (Han and

Unigrams										
	W_{-2}		W_{-1}		TARGET		W_{+1}		W_{+2}	
	Root	Aff	Root	Aff	Root	Aff	Root	Aff	Root	Aff
Text	여러	NONE	좋	은	것	NONE	있	어요	.	NONE
POS	DAN	NONE	VJ	EAN	NNX	NONE	VJ	EFN	SFN	NONE
Combo	여러_DAN		좋_VJ		것_NNX		있_VJ		._SFN	

Syntactic Approximations			
Previous Predicate	Previous Noun	Next Predicate	Next Noun
좋	곳	있	NONE

Trigrams			
	W_{-2}+W_{-1}+W_0	W_{-1}+W_0+W_{+1}	W_0+W_{+1}+W_{+2}
Text	여러+좋+것	좋+것+있	것+있+.
POS	DAN+VJ+NNX	VJ+NNX+VJ	NNX+VJ+SFN

Bigrams				
	W_{-2}+W_{-1}	W_{-1}+W_0	W_0+W_{+1}	W_{+1}+W_{+2}
Text	여러+좋	좋+것	것+있	있+.
POS	DAN+VJ	VJ+NNX	NNX+VJ	VJ+SFN

Table 6.2: Features and examples for 것 in **'각 곳에 여러 좋은 것이 있어요'**

Palmer, 2004). The features are drawn from a five-word sliding window (target ± 2), processing each token in a document. Only nominals are possible candidates for the omission error classifier, only nominals with particles are possible candidates for substitutions, and only instances where a particle omission or substitution error have been flagged are candidates for the selection classifier. 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 6.2 for the example in (36a) where 것 (*kes*) is the target. Note that the omitted particle 이 (*i*) should be inserted, as in (36b), though the example would still hold if the learner had used some other particle, e.g., 를 (*lul*) erroneously. The base set of 43 features is provided here, though not all features are used by each classifier; details are provided in the appropriate chapters.

(36) a. Original: 각 곳에 여러 좋은 것 있어요
each place-*at* many good thing exist

b. Corrected: 각 곳에 여러 좋은 것이 있어요
each place-*at* many good thing-SBJ exist

'There are many good things'

All words are broken down 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. There is also a (*combo*) feature for each root that combines its text and POS into a single string. These features are based largely on the feature sets in works done on English preposition error detection work, (e.g. Tetreault and Chodorow, 2008).

In addition to these adjacency-based features, we also encode the previous and following nouns and predicates, to approximate syntactic parent features. The *predicates* can be verbs, adjectives that function like verbs in Korean (cf. Section 3.2.3), and auxiliary verbs. Finally, two features to encode the amount of nouns that have already occurred in the sentence, as well as how many still remain. The usage of subject particles, for instance, relies in part on knowing where in the sentence a noun occurs, with respect to other nouns.

These word and POS based features are all that are used for the omission error detection experiments described in Chapter 7. For the substitution error detection and selection experiments, though, more detailed information is given to the classifier. In both cases, discourse information is extracted from the text, based on the knowledge that some decisions as to which particle is best for a given context are dependent on information that cannot be captured by the features described in this section. The discourse features are detailed in Section 8.3.

6.5.1 Best Practices for Evaluating Error Detection

One of the goals that has been at the forefront of this dissertation, is to follow (whenever possible) or establish (whenever necessary) best practices for all facets of the research. This becomes relevant in the following chapters as we begin to present the results of error detection/correction experiments. In particular, this will involve ensuring that evaluation methods are valid and that they provide as much useful information to the reader as possible. With this in mind, we will adhere, as much as possible, to the recommendations laid out in Chodorow et al. (2012) when presenting error detection results. That paper scrutinizes the evaluation methods that have been used across the field of automatic grammatical error detection/correction and determines the strengths and weaknesses of different evaluation schemes.

One important finding is that the nature of error detection in the writing of non-native learners means that the data will be skewed, that is, the amount of errors in comparison to non-errors is extremely low in most situations. This skew in the data means that some traditional evaluations will be less informative. Thus the recommendation is to report raw measurements of true negatives (TN), true positives (TP), false negatives (FN), and false positives (FP) whenever possible, and to clearly define what scenarios result in each contingency. Another useful piece of information from the paper is the establishment of the written-annotated-system (WAS) evaluation scheme, which arises from the fact that error detection has three sources of information that must be compared: the written input, the annotated input, and the system input. The scheme is laid out in Figure 6.2, X, Y, and Z represent answers from each input for the same instance. The final row is marked only with a * because the definition changes depending on whether error detection or correction is under consideration; for error detection it is a TP, as the system has recognized that X is not the best choice. All references to TNs, TPs, FNs, and FPs in the current work will assume these definitions.

	Written	Annotated	System
TN	X	X	X
FP	X	X	Y
FN	X	Y	X
TP	X	Y	Y
$*$	X	Y	Z

Figure 6.2: *WAS* evaluation scheme

It is important to choose an evaluation metric that provides the most useful information for the task at hand. Some of the more popular metrics are presented in Figure 6.3. This research is focused on building a tool to help learners of Korean, but the system could also be used for other applications (essay scoring, for example). In building tools for learners, it is important to emphasize precision, even at the cost of recall. As such, the F_1 is not necessarily the most informative metric for how well the system performs. Because F_1 weights R and P equally, it is entirely possible to have a system that finds 100% of the errors in a learner essay, but will have extremely low precision and end up with a strong looking F_1. However, it is also important to consider R to some extent; a system with 100% P but only 2% R is doing very little. In such a context, $F_{0.5}$ provides a good option - P is weighted more heavily than R. Thus, in reporting error detection results, we will often present P, R, and $F_{0.5}$.

Accuracy (A) $= \frac{TP+TN}{TP+TN+FP+FN}$
Precision (P)$= \frac{TP}{TP+FP}$
Recall (R)$= \frac{TP}{TP+FN}$
True Negative Rate (TNR) $= \frac{TN}{TN+FP}$
F-measure (F$_{1}$) $= 2 \cdot \frac{P \cdot R}{P+R}$
F$_{0.5}$-measure (F$_{0.5}$) $= 1.5 \cdot \frac{P \cdot R}{1.5 \cdot P+R}$

Figure 6.3: Evaluation metrics

Conclusion

This chapter has provided an overview of the entire particle error detection pipeline that we designed for the experiments in Chapters 7, 8, and 9. Here, we focused on the overall design of the pipeline approach and the shared aspects among the individual components. The chapters that follow will provide greater detail about the classifiers and feature sets used to perform the task for which they were designed. We will also provide system results in those chapters.

Chapter 7

Particle Omission Error Detection

Where can I find a man who has forgotten words so I can have a word with him?

— ZHUANGZU

In Chapter 6, we explained that the current approach breaks particle error detection into different tasks — here, we describe in detail the task of detecting errors of omission. As can be seen in Tables 4.5 and 4.6 in Chapter 4, omission errors constitute the majority of the particle errors (around 54% and 48%, in the respective tables) in a given learner corpus. Thus developing an error detection system that can reliably identify omission errors would pinpoint nearly half the particle errors language learners make. In this chapter, we describe a system designed to do just that.

7.1 Approach

To detect omission errors in the testing data, we train a CRF classifier (Lafferty et al., 2001) on just over one and a half million words taken from the training data. The classifier is essentially a particle presence predictor — particles are removed from target instances in the training data and the class is a binary representation of whether or not there should be a particle (YES/NO). At the testing phase, only instances where the learner did not attach a particle to the target nominal are examined. If the system returns *YES*, the instance is considered an error. Here we consider all nominals, as annotated in the corpus, as possible candidates for particle insertion.

While such a system could theoretically work to identify commission errors as well, the extremely low rate of commissions in comparison to instances where some particle should be used makes the task very difficult. In the KoLLA development set, there are 849 particles, and only 14 (1.6%) of those are errors of commission. This definition of the task also nicely separates instances where the writer did not use particle from instances the writer did use a particle. Under this framework, commission errors are actually more similar to substitution errors than omission errors (see Chapter 8).

Another effect of defining errors for this classifier as only those where no particle is used by the learner is that some errors that are marked as omissions will actually not be handled at this step. As mentioned in Chapter 4, there are instances where a *sequence* of particles are actually required to be attached to a single noun to produce grammatical output. In the cases where the learner only used one particle, but the annotator judged that there should have been two, the instance is marked as an omission error. This analysis makes perfect sense from a language learning standpoint, as the learner has simply omitted a particle. From the standpoint of this machine learning task though, this analysis is incongruous with the definition of the system. Because the system only considers instances in which *no* particle at all was used as possible sites of omission, omissions where one out of a sequence of particles cannot be handled by this classifier.

Again, these instances are more like substitution errors — in this case, the sequence particles are analyzed as a single unit. Such an analysis can be argued from a linguistic standpoint as well; one could take the view that segmenting the stem and particles as distinct units would mean that any error in the particles unit is a substitution error, rather than an omission. The ability for researchers to define such instances either as omissions or substitutions highlights the usefulness of the layered, single morpheme annotation approach. If the annotation was flat and only marked erroneous words without providing the segmentation, the original particle, the annotated particle, and the error type, it would be more difficult to separate these types of instances. The segmentation in particular is key to providing rich annotation with flexibility to define error types.

7.2 CRF Classifier

Conditional Random Fields (CRFs) have been shown to be a good tool for sequence labeling tasks. Lafferty et al. (2001) introduce the framework and compare their usefulness to that of Hidden Markov models (HMMs) and Maximum Entropy Markov Models (MEMMs). CRFs utilize the joint probability for an entire sequence of labels for a sequence, rather than using per-state models at each state given the following states, as in MEMMs. That is, where an algorithm like MEMMs normalize locally at each state, CRFs normalize globally, allowing for the best overall sequence to be preferred. This difference means that CRFs lack the undesirable bias of other algorithms towards states with fewer transitions.

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 can define particle presence to be a sequence labeling task by setting up the data so that for every word, we let the classifier decide if a particle should be present or not. Particles can only attach to nominals, so we simply ignore decisions for other parts of speech. We use the comma error detection work in Israel et al. (2012) (see description of this work in Chapter 1) as a basis, and employ CRF++[1] to set up a binary classifier at this step based on 1.5 million instances from our web corpus. While that might seem like a small amount of training data, it is important to keep in mind: 1) the task here is guessing simply *yes* or *no* about particle presence, so a wide variety of particle coverage is not required, and 2) the algorithm underlying CRF classification allows for models to be built with less training data than other classifiers (Lafferty et al., 2001).

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

7.3 Features

The features for this step are simply those found in Section 6.5. That is, the full set of word and POS based features provided in Table 6.2

7.4 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 add a filter after the classifier to ensure that the classifier does not label too many correct instances as errors. For 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 7.5), we tried a variety of thresholds, in a hill-climbing approach, and found 85% to be the best.

7.5 Results

	n	TN	TP	FP	FN	Precision	Recall	$F_{0.5}$
Dev baseline	48	0	48	98	0	32.88	100.00	37.97
Test baseline	220	0	220	391	0	36.01	100.00	41.29
Dev no filter	48	64	40	34	8	54.05	83.33	58.14
Dev 85% filter	48	90	26	8	22	76.47	54.17	70.65
Test 85% filter	220	373	96	18	124	84.21	43.64	71.01

Table 7.1: Particle omission error detection results

Following the recommendations in Chodorow et al. (2012) (discussed in Chapter 1), I 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 this case, positives

	n	TN	TP	FP	FN	Precision	Recall	$F_{0.5}$
FB	47	45	18	5	29	78.26	38.30	64.75
HB	51	46	19	4	32	82.61	37.25	66.43
FI	63	153	32	6	31	84.21	50.79	74.42
HI	59	129	27	3	32	90.00	45.76	75.42

Table 7.2: Particle omission error detection results by learner type (test data)

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 positives (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. It is important to keep in mind that the baseline is evaluated by the same criteria as the full system, i.e. rather than caring about overall accuracy (which should be quite high, given the amount of nominals that require a particle), we are interested in its utility in finding omission errors.

Table 7.1 provides the results for particle omission detection on both the development and test corpus. Here we present the baseline, the results based only on the classifier's decision (no filter), and the results for the best filter. As the 85% threshold results in the best $F_{0.5}$, we use this system on the test data. Table 7.2 provides the results for particle omission detection broken down by subcorpus.

7.6 Quantitative Analysis

The baseline system, i.e. always inserting a particle, clearly marks all omitted particle instances as errors, resulting in 100% recall. However, as this is such a simplistic approach, the precision suffers greatly, and at only 32 - 36% (Dev and Test, respectively) is not suitable for use in a system designed to assist learners, for automatic grading, or most other applications of error detection.

Looking next at the result of the classifier on the development set with no filter (Dev no filter), one can see a significant improvement over the baseline; there is a 21% improvement of $F_{0.5}$ for the development set. With no filter, the classifier suggests the most likely class, i.e. whichever of the two classes receives over 50% confidence. As errors are relatively infrequent, however, the system should not flag learner errors too often. Table 7.3, provides the number of total particles, particle errors, and omission errors in the development set. As seen in the table, about 11% (106/955) of nominals are left without a particle by the learner. Of these 106 cases, only 45.3% (48) are annotated as needing a particle . The CRF with the default $>$ 50% confidence setting ("Dev no filter" in Table 7.1) inserts 74 particles (40 TP + 34 FP), i.e. it inserts particles in 69% of the cases of bare nominals. Consider the following hypothetical situation in regards to the "Dev no filter" system: If the classifier posited the same number of errors (74) and found all 48 errors, the recall would be 100% (48 TPs). However, that would mean that there would still be 26 FPs in the best case scenario because the system posited 74 errors, so precision would be about 65% ($\frac{48}{26+48}$). As such, there is still room for improvement over the CRF system.

Using the optimal 85% confidence filter, the system performs much better. The the system achieves a modest, though potentially useful, 76.5% precision and $F_{0.5}$ 70.7% on the development set. It actually does even better on the Test set, at 84.2% precision, and 71% $F_{0.5}$ on the test set. The improvement in testing over development is likely a function of having such a small development corpus for this task. Looking at the results in Table 7.2, some interesting patterns emerge. In general, the system performs better on essays by heritage learners than on those by foreign learners in their peer groups. Likewise, the system performs better on essays written by intermediate learners than on essays written by beginners, regardless of heritage status. Both of these trends make sense, as it is to be expected that heritage learners and intermediate learners would produce data that is more native-like, i.e. more similar to the writing represented in the training corpus of web data than foreign learners and beginning learners, respectively. Combining these facts, the system performance on the FB and HI sub-corpora, i.e. those which are most likely to produce the least and most native-like Korean, sits at opposite ends of the spectrum, with FB producing the worst results and HI the best. These results point towards building different systems for different learner types that can be tuned for the types/frequency of errors common to each subset of learners.

Total Nominals	Total Particles	Bare Nominals	Omission Errors
955	849	106	48 (45.3%)

Table 7.3: Statistics for Omission Errors in the KoLLA Development Set

7.7 Qualitative Analysis

In order to get a sense of what types of cases give a system trouble, looking over the FPs, i.e., cases where the system predicted a particle not in the gold standard, provides a lot of insight. Consider first the example given in (37), in which 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). In any case, marking this instance as an error is not the right interpretation.

(37) 특히 외국 **사람들** 눈에는 더욱 그렇습니다.
particularly foreign **people** eye more is-so.
'In particular, it is thus for the eyes of foreign people.'

A more involved annotation and evaluation scheme might point to a more lenient system that could be useful in some applications; pointing out to a learner that a genitive particle here is not wrong, could certainly be useful to learners of a certain level. Also, note that a metric like error reduction, in which the writer's input is compared to the system's output in terms of the amount of errors present, would not be penalized if these annotations were present. Fully teasing apart particle licensing from particle requirement requires more thorough discussion of when particle dropping is permitted. Moreover, licensing gets away from the strict definition of grammaticality used in the annotation scheme described in Chapter 4.

The system also makes mistakes in some cases that do not license particles, but the nominals still have particle-like functions. In (38), 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). In this case, it would appear that the system and human learners are likely having similar problems in differentiating between where a temporal particle is needed and a temporal word may stand alone.

(38) **이 때** 너무 감정에 치우치지 않도록 주의하어야 해.
this time too feeling-at give-way-to 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 multi-word compound, it conceals the fact that the noun requires a particle. For instance, in (39) (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.

(39)

성격{**이**}	좋은	아가	태어날	때	환경이	나쁘다면
personality-{**SBJ**}	good-REL	kid-SBJ	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. In (40a) for example, the writer has omitted a necessary locative particle, 에 (*ey*, 'ey'), after "October", and the system failed to recognize a missing particle as well. Now consider (40b), which is identical other than the inclusion of the prepositional phrase, 서울-에 (*Seoul-ey*, 'in Seoul'). When this intervening phrase is added, the 에 (*ey*, 'ey') after "October" becomes optional, as indicated by the parentheses.

(40) a.

저는	2007년	10월{**에**}	가아	보았+습니다
I-TOP	2007year	10month-{*in*}	go	try-PAST-END

'I went (there) in October 2007'

b.

저-는	2007년	10월(-에)	서울-에	가	보-았-습니다.
I-TOP	2007year	10month(-*in*)	Seoul-*in*	go	try-PAST-END

'I went to Seoul in October 2007'

False negatives are, in general, less problematic for an error detection system than false positives. Still, knowing exactly what types of problems are causing both essay writers and the error detection system to miss cases where particles should be used is useful to know. It is conceivable that with a large enough set of false positives, rule-based filters could be added to the system to look for specific scenarios, e.g. likely multi-word compounds, and relax the confidence filter to allow the system to posit a particle with less confidence in these instances.

Conclusion

In the case of determining where omission errors are, the system presented performs the task at levels that could be useful to a language tutoring system, as is. Moreover, there is likely room for improvement; with more data, it is possible that more learner-type specific training data and system filters could be developed, as the current system handles some learner types significantly better than others (see, e.g. the divergence in performance on FB vs. HI data in Table 7.2).

At this point, though, we have only described how to detect errors of omission. For some applications, e.g. a language tutoring system, providing more feedback can prove extremely valuable. In the case of errors, providing the user with a correction to the error is often desirable. In Chapter 9, we will provide explanation of a classifier designed to select the best particle once an error has been found. Before that, though, we will remain on the task of error detection, this time focusing on substitution errors in Chapter 8.

Chapter 8

Particle Substitution Error Detection

The only man who never makes a mistake is the man who never does anything.

— Theodore Roosevelt

In Chapter 7, we described a CRF-classifier based approach to detecting cases in which learners did not use a particle where one was required. In this chapter, we will provide the details of another error detection system, this time focusing on cases where there is an erroneous particle present. In essence, this system is a particle *presence* error detector. It is important here to note that by and large, the errors that will be dealt with by this component are substitution errors, that is, one particle should be replaced by another. However, this part of the system can also find errors in which the learner has used one particle, but two particles in a sequence are needed for grammaticality, errors of commission (using a particle when none is needed), and errors of particle form (allomorphic or spelling mistakes). Put simply, this system checks each existing particle and is tasked with determining whether or not that particle is correct for the given context.

8.1 Approach

Whereas the omission classifier was only applied to nominals that did not feature a particle, this step involves a classifier that *only* looks at nominals with a particle. To do this, we employ maximum entropy classifiers targeting a specific set of particles.

8.1.1 Restricting the Error Search Space

As described above, the task here is relatively straightforward — the question that needs answering at this step is this: is a particle that has been posited by a writer correct in this context? Thus, every nominal where a learner has used a particle is a possible site for error detection. However, it is important to keep in mind the error rate of learners and restrict the task to ensure that system precision remains high. In this case, we want to only look at cases where the particle used is often confused for another particle. This decision mirrors that of much of the work done on English preposition error detection that focuses on a subset of the most commonly used preposition, setting aside the problem of finding errors in prepositions that occur less often (De Felice and Pulman, 2008; Gamon et al., 2008; Han et al., 2010, see, e.g.).

In order to get a sense of where errors occur most often in learner essays of Korean, we examined the substitution errors found in the corpus described in Lee et al. (2009a). In that corpus, the results were clear: 가 (*ka*), 에 (*ey*), 를 (*lul*), 는 (*nun*), and 에서 (*eyse*) accounted for a vast majority of the particles that were used erroneously by the writers of the essays; these five particles were the source of 329 (81%) of the 405 substitution errors annotated in the corpus. However, as 는 (*nun* is used only as a discourse marker, that is, it marks the *topic* of a sentence, we do not include this particle in the set of candidates for substitution. The use of topic markers rather than subject markers is a problematic issue for learners of Korean, as the two particles are often, though not always, interchangeable. The difference is often difficult to describe for language learners and even native speakers (Lee and Ramsey, 2000; Yeon and Brown, 2011). As such, it would be expecting too much of an automatic system to capture the subtle distinctions necessary to perform the task accurately. Still, the four particles that we focus on make up nearly 70% of all errors.

While the KoLLA corpus contains fewer error annotations than the Lee et al. (2009a) offering, one can reasonably expect to see similar distributions across these corpora, as the essays come from writers of similar learner levels and backgrounds. Thus, we will look only at instances in which the learner used one of these four source particles as possible error sites: 가 (*ka*), 에 (*ey*), 를 (*lul*), and 에서 (*eyse*) .

While this will obviously mean that the coverage will not be perfect, it will mean that the system not need to be concerned with instances which occur less often in the data, which should lead to higher precision for the instances it does cover.

8.1.2 Establishing a Binary Distinction

While one way of detecting substitution errors via a classifier would be to train the classifier on all candidates for insertion and consider any mismatch between the system and the writer to be an error flag. This methodology has often been used for error detection/correction tasks in English (see e.g., Tetreault and Chodorow, 2008; Han et al., 2010). Setting up the task in such a way essentially combines detection and correction of errors into a single classifier. However, this approach requires that the system consider each available label as a possibility at each decision point. As the set of labels grows, even closed sets such as prepositions and particles can be quite large, the classifier must distribute some probability to each option.

We choose to lighten the load at the error detection phase by making the task binary. Based on the idea that particle omission errors can be treated by considering whether or not blank space (i.e. no particle) is the best choice for the nominal in question, we set up a similar task here. That is, for each nominal that has one of the four particles outlined above as an affix, the classifier guesses if that particle or some other one is best. To that end, we reduce the training set for each classifier such that it is target vs. *OTHER*. For example, in the 가 (*ka*) training set, all of the possible target (correct) particles are reduced to *OTHER*. Then for each instance where the learner has used 가, there is a binary decision whether 가 or some other particle is best for that nominal in that context.

8.2 Maximum Entropy Classifier

To detect substitution errors, there are a wide variety of particles that must be considered, meaning that the classifier should have broad coverage, and work well with larger training corpora. Thus, we use maximum entropy for the task at hand; specifically, the Maximum Entropy Toolkit[1].

Maximum entropy has been in use in NLP since at least the mid 1990s in a variety of tasks including language modeling, machine translation, sentence boundary detection, POS tagging, parsing, and many other tasks (see e.g., Rosenfeld, 1994; Berger et al., 1996; Ratnaparkhi, 1996, 1998; Malouf, 2002). Most directly relevant to the task at hand, maximum entropy classification has been used in similar error detection tasks such as preposition error detection (Tetreault and Chodorow, 2008), and performed well. Based on the fact that the algorithm has proven effective at selecting the best preposition from a controlled set of outcomes, we model the task of substitution error detection on that work.

8.3 Features

The feature set for particle substitution detection is again mainly composed of words and POS tags in the surrounding context of each possible insertion point, as described in Section 6.5. Again using a five-word window (target ± 2), and only nominals with one of the four particles under consideration are examined as possible substitution errors. Along with the lexical and POS features, we add a handful of features designed to exploit discourse information in the text, which we describe next.

[1] `http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html`

8.3.1 Discourse Features

As the choice for what particle is often dependent on complex information that goes beyond the scope of lexical items and POS tags, we developed features to provide the classifier with some discourse information. First of all, there are a series of binary topic/subject/object tracking features that are positive if the target word was used as a topic, subject, or object in the previous sentence, or if it was used at all in the previous sentence. These tracking features inform the classifier if the target has been important recently in the discourse.

Next, there are features that inform the classifier if a topic, subject, or object particle has been used in the current sentence yet. These features get at the idea that, in most sentences, a subject, topic, and/or an object will be necessary. Moreover, it is often the case that if you have already seen a subject particle, you will not see a second, likewise with topic markers, although this rule is not ironclad.

Finally, three numeric features have been added: one that provides a percentage of how far along in the sentence the current target word is (position/sentence length), one for the number of nouns the left of the target, and one for the number of nouns to the right of the target. The idea with the distance feature is that some particles are more likely towards the beginning of the sentence (e.g., topics and subjects), and some are more likely later in the sentence. The number of nouns to the left and right get at the idea that most or all sentences will have at least one or two particles, so for short sentences, it can be useful to know how many nouns could possibly still take a particle.

8.4 Experiments

As described earlier in this work, we have collected three distinct corpora: the KoLLA learner corpus (see Chapter 4), the web-scraped corpus of well-formed Korean (referred to as WaC-Ko henceforth), and the Lang-8-Ko corpus of online language learning essays (for WaC-Ko and Lang-8-Ko, see Chapter 5). We will utilize each of these for the substitution error detection experiments described in this chapter.

8.4.1 Experimental Setup

As pointed out in previous similar research on English error detection (e.g. Dahlmeier and Ng, 2011; Han et al., 2010), there are (at least) two scenarios that can be used for training classifiers for grammatical error detection tasks. One method involves training on well-formed text and flagging an error if the classifier disagrees with a learner on the answer for a given context (cf. Tetreault and Chodorow, 2008, for example). The second method involves training on annotated learner data so that the classifier is informed of not only the correct label, but the answer provided by the learner (cf. Dahlmeier and Ng, 2011, for example). That is, the feature set for the classifier in the second scenario includes the particle originally written by learner, some of which are errors. We employ both methods here.

In running experiments where we train only on well-formed instances, we can train on WaC-Ko and test on both Lang-8-Ko and KoLLA. To include learner errors in the training set, we have to rely only on Lang-8-Ko for training, as we have no error information in WaC-Ko. Both Lang-8-Ko and KoLLA are annotated for errors, so we can test using both sets. Moving forward we will refer to data sets that include learner errors as features with a +, i.e. KoLLA+ and Lang-8-Ko+. Thus, we

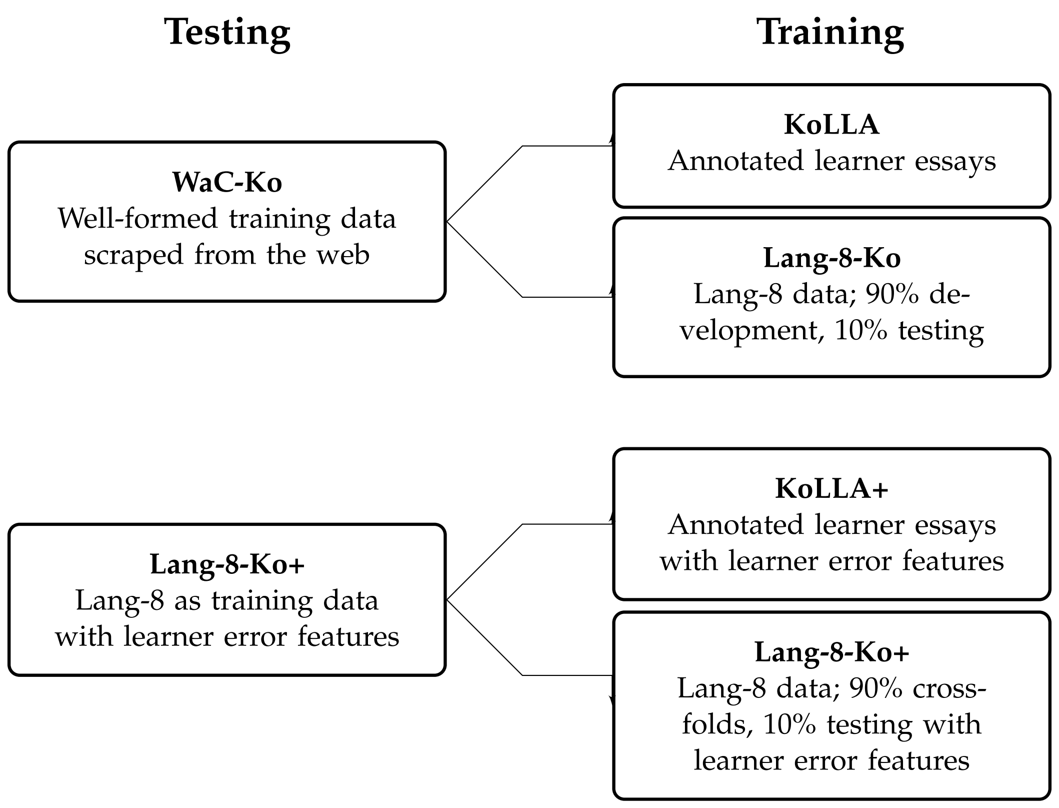

Testing
Training
WaC-Ko
Well-formed training data scraped from the web
KoLLA
Annotated learner essays
Lang-8-Ko
Lang-8 data; 90% development, 10% testing
Lang-8-Ko+
Lang-8 as training data with learner error features
KoLLA+
Annotated learner essays with learner error features
Lang-8-Ko+
Lang-8 data; 90% cross-folds, 10% testing with learner error features

Figure 8.1: Training and Testing combinations

have five sets of data to work with, Wac-Ko, Lang-8-Ko, KoLLA, Lang-8-Ko+, and KoLLA+. Figure 8.1 depicts which training sets will be used to build classifiers for each test set. As can be seen, we run four experiments: WaC-Ko serves as training for classifiers to handle KoLLA and Lang-8-Ko test data; Lang-8-Ko+ is used to build classifiers for KoLLA+ and Lang-8-Ko+.

When testing on KoLLA and KoLLA+, rather than develop on a subset of the corpus and test on the rest, we present optimal performance for each sub-corpus, i.e. Foreign Beginner (FB), Heritage Beginner (HB), Foreign Intermediate (FI), and Heritage Intermediate (HI). The main reason behind this decision is that there simply are not enough instances of errors for each particle to draw useful conclusions from development. For example, there are only 33 errors involving 를 (*lul*), the object marker, and those are spread across the four sub-corpora, with as few as five in the FI sub-corpus. Clearly, developing on one — four instances and testing on whatever is left over will not yield useful results. A more worthwhile and interesting use of KoLLA, then, is to look at each sub-corpus individually and examine the difference between learner types.

For Lang-8-Ko and Lang-8-Ko+, we split the data into 10 random folds. When training on Wac-Ko and testing on Lang-8-Ko, we develop on 90% (nine folds) of the data, and test on the remaining 10%. For the Lang-8-Ko+ experiments, we run nine-fold cross validation and test on the same unseen 10%.

8.4.2 Allomorph Choice Errors

As mentioned in Section 4.3.8, for some particles there are more than one allomorph for a single morpheme, the choice of which depends upon the phonotactics of the word. Essentially, if the preceding morpheme ends in a vowel, the particle must begin with a consonant, and vice versa. Two such cases are the subject marker and the object marker. The subject marker can occur as 이 (*i* or 가 (*ka*); the object marker as 을 (*ul*) or 를 (*lul*). As such, before sending instances in which the writer used either the subject or object marker to the classifier, we can catch any errors of allomorph choice with a simple string-based check. This check can both detect and correct these errors.

In both corpora, these allomorphic errors are extremely infrequent. Looking first at their frequency in KoLLA, subject marker allomorph choice errors occur only three times, with 49 total subject marker errors, so about 6% of the errors come from incorrect allomorph use. There are two object marker allomorph choice errors in KoLLA, with 35 total errors, just under 6% as well. In Lang-8-Ko, there are 121 allomorph choice errors for the subject marker, or about 3% of the 3284 errors.

Finally, in Lang-8-Ko, there are 188 object allomorph choice errors, meaning they account for about 7% of the 2720 total errors involving object markers.

8.4.3 Optimizing Training Data

In Chapter 5, we discussed using data from the language learning SNS website, Lang-8, to build confusion sets for particle substitutions to inform training data selection. These sets are utilized here (as well as in Chapter 9) to restrict the set of particles that are considered as possible targets (i.e. corrections) for the source (erroneous) particles in the learner data. With hundreds of possible Korean particles, very few of those are likely to be frequently confused for each other, so we want to focus on the most likely errors. The training corpora, both WaC-Ko and Lang-8-Ko+, are divided into four sets, one for each of the four target particles.

Each training set contains a source (writer error) particle and the set of particles that are most often used to correct the errors. One implication here is that the five training sets are different sizes. More importantly, though, is that each set has a different ratio of source to target particle instances. Table 8.1 gives the number of instances for each source, its set of target confusions, the total number of instances, and the ratio of source to target instances. As can be seen in the table, the amount of source particles varies greatly across the different particle training sets; 를 (*lul*) is fairly common in comparison to the members of its confusion set, with a ratio of 1:2.3, whereas 에서 (*eyse*) is uncommon both in the corpus in general (only 174,962 instances) and in relation to its confusion set, with a ratio of 1:32.3. In order to control for this, we down sample so that the ratio is 1:2 for each particle set. As a final step, any lexical n-gram feature values that occur less than five times are replaced with a dummy value, to reduce the effect of data sparsity.

Source	Source Count	Target-set Count	Total Instances	Ratio
를 (*lul*)	1,396,877	3,197,236	4,594,133	1:2.3
가 (*ka*)	1,086,640	4,046,221	5,132,861	1:3.7
에 (*ey*)	553,607	4,842,929	5,396,536	1:8.7
에서 (*eyse*)	174,962	5,640,219	5,815,181	1:32.3

Table 8.1: Training Set Size Statistics for Substitution Error Detection

8.4.4 Filtering

As was the case with omission error detection, the proportion of substitution errors in comparison to correctly used particles is quite low. Thus, we again require the system to have higher than the default 50% confidence to ensure that there are not too many errors posited in the data. As described earlier, there are four distinct classifiers at this stage (one each for four different source particles). Rather than set a single threshold for all classifiers, we set a filter for each classifier independently of the others. That is, we find the best threshold for each particle and each sub-corpus, using development data in the case of Lang-8-Ko.

8.5 Results

We present the results for each of our four sets of experiments here. The experiments are divided first in terms of training data, then in terms of testing data. Each table presents a separate source particle in each row, with the confidence filter, the number of errors (*n*), the number of true negatives (TN), true positives (TP), false positives (FP), and false negatives (FN). In terms of evaluation metrics, we provide precision (P), recall (R), precision weighted F-measure ($F_{0.5}$), and overall accuracy for finding errors (Acc). As there is no specific purpose for the system in mind, the question of what results are “optimal” is debatable. In general, though, as most applications for error detection involve improving or assessing learner language, precision should be given preference. In the following results, we optimize by trying to keep precision above 75% whenever possible. $F_{0.5}$ serves as another good indicator for producing optimal output, as it gives some weight to recall while preferring precision.

Note that we provide only the quantitative results here. Discussion and analysis of the results follow in Section 8.6.

8.5.1 WaC-Ko Models for KoLLA Test Data

First, we present results for tests in which WaC-Ko served as training data for classifiers with KoLLA test data. Table 8.2 provides the results for the FB sub-corpus. The results for the testing on the FI, HB, and HI sub-corpora are given in Tables 8.3, 8.4, and 8.5, respectively.

Source	Filter	*n*	TN	TP	FP	FN	P	R	$F_{0.5}$	Acc
ey	.83	28	72	9	3	19	75.00	32.14	59.21	78.64
eyse	.9	5	37	1	0	4	100.00	20.00	55.56	90.48
lul	.97	10	90	2	0	8	100.00	20.00	55.56	92.00
ka	.89	15	89	5	0	10	100.00	33.33	71.43	90.38
Total	-	58	288	17	3	41	85.00	29.31	61.59	89.61

Table 8.2: Error Detection: WaC-Ko Training — KoLLA FB Test

Source	Filter	*n*	TN	TP	FP	FN	P	R	$F_{0.5}$	Acc
ey	.89	19	126	3	0	16	100.00	15.79	48.39	88.97
eyse	.89	7	55	1	1	6	50.00	14.29	33.33	88.89
lul	.97	6	188	2	1	4	66.67	50.00	62.50	97.43
ka	.87	8	97	2	0	6	100.00	25.00	62.50	94.29
Total	-	40	467	8	2	32	80.00	20.00	50.00	93.32

Table 8.3: Error Detection: WaC-Ko Training — KoLLA HB Test

Source	Filter	n	TN	TP	FP	FN	P	R	$F_{0.5}$	Acc
ey	.9	12	125	1	2	11	33.33	8.33	20.83	90.65
eyse	.94	6	45	1	2	5	33.33	16.67	27.78	86.79
lul	.93	10	184	4	0	6	100.00	40.00	76.92	96.91
ka	.94	16	248	3	2	13	60.00	18.75	41.67	94.36
Total	-	44	602	9	6	35	60.00	20.45	43.27	93.71

Table 8.4: Error Detection: WaC-Ko Training — KoLLA FI Test

Source	Filter	n	TN	TP	FP	FN	P	R	$F_{0.5}$	Acc
ey	.84	11	103	4	0	7	100.00	36.36	74.07	93.86
eyse	.8	0	33	0	0	0	-	-	-	100.00
lul	.9	9	187	6	4	3	60.00	66.67	61.22	96.50
ka	.95	10	178	2	1	8	66.67	20.00	45.45	95.23
Total	-	30	501	12	5	18	70.05	40.00	60.90	95.89

Table 8.5: Error Detection: WaC-Ko Training — KoLLA HI Test

8.5.2 WaC-Ko models for Lang-8-Ko Test Data

Keeping with the WaC-Ko training data, we present the results for testing on the Lang-8-Ko corpus. Table 8.6 provides the results for the development corpus. We used the optimal settings for each particle in the development set to produce the results in Table 8.7 on the unseen Lang-8-Ko fold.

Source	Filter	n	TN	TP	FP	FN	P	R	$F_{0.5}$	Acc
ey	.92	2055	19131	218	150	1837	59.24	10.61	30.90	90.69
eyse	.92	659	4919	69	61	590	53.08	10.47	29.26	88.46
lul	.995	2442	32606	326	25	2116	92.88	13.35	42.38	93.90
ka	.96	2976	28341	243	66	2733	78.64	8.17	28.85	91.08
Total	-	8132	84997	856	302	7276	73.93	10.53	33.54	91.89

Table 8.6: Error Detection: WaC-Ko Training — Lang-8-Ko Development

Source	Filter	n	TN	TP	FP	FN	P	R	$F_{0.5}$	Acc
ey	.92	232	2157	28	12	204	70.00	12.07	35.71	91.00
eyse	.92	70	542	6	5	64	54.55	8.57	26.32	88.82
lul	.995	278	3613	28	4	250	87.50	10.07	34.48	93.48
ka	.96	308	3095	28	9	280	75.68	9.09	30.70	91.53
Total	-	888	9407	90	30	798	75.00	10.13	32.80	91.98

Table 8.7: Error Detection: WaC-Ko Training — Lang-8-Ko Test

8.5.3 Lang-8-Ko+ models for KoLLA+ Test Data

Turning now to the classifiers that utilize Lang-8-Ko+ as training data, we first present the results for testing on KoLLA+. Tables 8.8, 8.9, 8.10, and 8.11 provide the results on the KoLLA+ FB, HB, FI, and HI sub-corpora, respectively.

Source	Filter	*n*	TN	TP	FP	FN	P	R	$F_{0.5}$	Acc
ey	.8	28	74	6	1	22	85.71	21.43	53.57	77.67
eyse	.75	5	35	3	2	2	60.00	60.00	60.00	90.48
lul	.88	10	90	3	0	7	100.00	30.00	68.18	93.00
ka	.86	15	90	7	0	8	100.00	46.67	81.40	92.38
Total	-	57	288	19	3	37	86.36	32.75	65.06	88.47

Table 8.8: Error Detection: Lang-8-Ko+ Training — KoLLA FB Test

Source	Filter	*n*	TN	TP	FP	FN	P	R	$F_{0.5}$	Acc
ey	.83	19	126	4	0	15	100.00	21.05	57.14	89.66
eyse	.81	7	54	1	2	6	33.33	14.29	26.32	87.30
lul	.96	6	188	2	1	4	66.67	33.33	55.56	97.44
ka	.96	8	97	2	0	6	100.00	25.00	62.50	94.29
Total	-	40	465	9	3	31	75.00	22.50	51.14	93.31

Table 8.9: Error Detection: Lang-8-Ko+ Training — KoLLA HB Test

Source	Filter	*n*	TN	TP	FP	FN	P	R	$F_{0.5}$	Acc
ey	.85	12	123	2	4	10	33.33	16.67	27.78	89.93
eyse	.8	6	43	1	4	5	20.00	16.67	19.23	83.02
lul	.9	10	182	3	2	7	60.00	30.00	50.00	95.36
ka	.85	16	237	6	13	10	31.58	37.50	32.61	91.35
Total	-	44	585	12	23	32	34.29	27.27	32.61	91.56

Table 8.10: Error Detection: Lang-8-Ko+ Training — KoLLA FI Test

Source	Filter	*n*	TN	TP	FP	FN	P	R	$F_{0.5}$	Acc
ey	.95	11	103	2	0	9	100.00	18.18	52.63	92.11
eyse	.9	0	33	0	0	0	-	-	-	100.00
lul	.92	9	191	3	0	6	100.00	33.33	71.43	97.00
ka	.68	10	147	5	32	5	13.51	50.00	15.82	80.42
Total	-	30	474	10	32	20	23.81	33.33	25.25	90.29

Table 8.11: Error Detection: Lang-8-Ko+ Training — KoLLA HI Test

8.5.4 Lang-8-Ko+ models for Lang-8-Ko+ Test Data

Finally, we get to the results for training with Lang-8-Ko+ and testing on Lang-8-Ko+. As previously stated, we employ 9-fold cross validation to learn optimal settings; these results are in Table 8.12. Table 8.13 provides the results for applying these settings to the unseen fold of Lang-8-Ko+.

8.6 Analysis

Obviously, with the results in Section 9.3 spread over 12 tables, there are a lot of possibilities for analysis. We will examine some specific facets of the results here.

KoLLA Learner Level We see a definite trend when comparing the results of the KoLLA sub-corpora in both training scenarios. That is, the system handles beginners' data much better than it does that of intermediates'. In terms of precision, performance on beginners' data never drops below 70%, and peaks at about 86% considering all particles combined. For intermediates, the results are significantly

Source	Filter	*n*	TN	TP	FP	FN	P	R	$F_{0.5}$	Acc
ey	.7	2055	19141	176	140	1879	55.70	8.56	26.51	90.54
eyse	.6	659	4913	56	67	603	45.53	8.50	24.33	88.12
lul	.97	2442	32610	250	21	2192	92.25	10.24	35.45	93.69
ka	.9	2976	28376	181	31	2795	85.38	6.08	23.67	91.00
Total	-	8132	85040	663	259	7469	71.91	8.15	28.04	91.73

Table 8.12: Error Detection: Lang-8-Ko+ Training — Lang-8-Ko+ Development

Source	Filter	*n*	TN	TP	FP	FN	P	R	$F_{0.5}$	Acc
ey	.7	232	2156	20	13	212	60.61	8.62	27.47	90.63
eyse	.6	70	538	7	9	63	43.75	10.00	26.12	88.33
lul	.7	278	3616	23	1	255	95.83	8.27	30.75	93.43
ka	.9	308	3098	17	6	291	73.91	5.52	21.25	91.30
Total	-	888	9408	67	29	821	69.79	7.55	32.80	91.77

Table 8.13: Error Detection: Lang-8-Ko+ Training — Lang-8-Ko+ Test

worse, with a high mark of about 69% for the HI sub-corpus with a WaC-Ko trained classifier, to a low of about 22% for the same sub-corpus, but with the Lang-8-Ko+ training. Interestingly, this result is in direct contrast with the results for omission error detection, where we see better performance among intermediate learners than beginners (cf. tab:omission-test). Finally, we see that the Lang-8-Ko corpus patterns similarly to the KoLLA beginners' sub-corpora, especially when using Lang-8-Ko+ as training.

Source Particle We can group the source particles into two categories here: Inherent case (*ey* and *eyse*) and structural case (*lul* and *ka*). Because the amounts of errors are relatively lower in the KoLLA corpus, the results are erratic with each particle producing precision of 100% in at least one training scenario for a single sub-corpus, but also dropping to 50% or lower in other tests for the same particle. A better source for providing insight here is the Lang-8-Ko corpus. In both the WaC-Ko and Lang-8-Ko+ tests, we see a clear divide between results for the inherent and structural case particles, where structural case is handled much better by our classifiers. In particular, the results for *lul*, the object marker, hover around 90% precision, whereas *ka*, the subject marker, stays closer to 80%. For inherent case, on the other hand, we never see precision above 70% for ey, or above 55% for eyse.

Training Type One of the more intriguing results relates to the difference between training only on well-formed data, i.e. WaC-Ko, and training on data with learner errors, i.e. Lang-8-Ko+. While the WaC-Ko training scenario works better overall, the difference is not that great in most equivalent experiments, in the case of KoLLA beginning learner sub-corpora and Lang-8-Ko test data. In fact, it even improves slightly for the KoLLA FB corpus when comparing the results for all particles combined. This similarity in performance is surprising and encouraging when we consider the size of the two training sets. Before dividing the corpora by confusion set, WaC-Ko contains over six-million instances. Lang-8-Ko+ contains only 247,270. That is, Lang-8-Ko+ is only about four percent of the size of WaC-Ko, yet still produces similar results. In the case of KoLLA intermediate learners, Lang-8-Ko+ proves to be a poor data set for building a model with overall precision at 34% and 22% for foreign and heritage learners, respectively. This finding, combined with the fact that Lang-8-Ko data patterns similarly to KoLLA beginners in most testing scenarios suggests that the Lang-8 data is produced by less experienced learners of Korean.

This finding highlights the potential utility of incorporating erroneous learner particles as features when training the machine learner. With a greater amount of annotated learner data, we are hopeful that even better results can be achieved.

Conclusion

This chapter described our approach to Korean particle substitution error detection. We found that, in general, the system performs fairly well on beginners' data, but not as well when essays by more advanced students are under consideration. We also found that when testing on Lang-8-Ko data, the results were similar to that of the KoLLA beginners, signaling that perhaps the learners that used the Lang-8 website at the time of the release of this data were, by and large, beginning learners. There is still more work to be done to produce classifiers that are better equipped at handling intermediate learner data. We also compared to testing scenarios and found that the Lang-8-Ko+ corpus, despite being considerably smaller than the WaC-Ko corpus produces similar precision when applied to the same data. This result highlights the utility of annotated training data, as the main difference between these data sets is the incorporation of learner particles as a feature for machine learning.

In the next chapter, we lay out the final task for the line of research presented in this dissertation. That is, now that errors have been detected (correctly or not) the next logical step is to correct them. We will describe a system that chooses the best particle for a given context, based on the information from the error detection classifiers that the writer's choice is incorrect.

Chapter 9

Particle Error Correction

See everything, overlook a great deal, correct a little.

— POPE JOHN XXIII

In Chapters 7 and 8, we described our approach to detecting errors of particle omission and substitution, respectively. Here we describe the final module in our pipeline approach (see Section 6.3): the error corrector.

9.1 Approach

The job of the omission and substitution error detection modules is to identify the errors in a given text. So, at this phase, we are only applying the classifier in the case of known errors. We will generate corrections for erroneous instances where the source particle comes from the following set: the subject marker, 가 (*ka*), the object marker, 를 (*lul*), the preposition-like particles 에 (*ey*) and 에서 (*eyse*), and *null*, i.e. errors of omission. The error detection modules both dealt with binary distinctions; whether or not there should be some particle for the omission module, and whether or not an existing particle is appropriate for the substitution module. For correction, the task is naturally multi-class, as the classifier must consider a range of options at each error site depending on likely confusion sets.

9.1.1 Classifier and Feature Set

Here we use the same classifier and feature set as in the substitution error detection experiments described in Chapter 8. That is, we use the Maximum Entropy Toolkit for classification, the full set of word and POS-based features described in Section 6.5, and the discourse features described in Section 8.3.1.

9.2 Experiments

In Chapter 8, we established that among the training scenarios we tested, using the well-formed WaC-Ko corpus produced the best results for testing on both KoLLA and Lang-8-Ko (see Section 9.3). As such, we will use the same source (WaC-Ko) to train the classifiers we employ here for correcting errors. We will present results for testing on the errors by the error detection modules, which we will refer to as *pipeline*, as well as results for all errors (i.e. including false negatives from the error detection modules), which we will refer to as *gold*.

9.2.1 Allomorph Choice Errors

As described in Chapter 8, errors of allomorph choice can be easily caught with a string-based check. Because the allomorphic distinctions are binary, we can not only detect these errors with 100% precision and recall, we can also correct them with 100% accuracy. The results presented here include these corrections. To reiterate, there are three subject marker errors and two object marker errors in KoLLA, and 121 subject marker errors and 188 object marker errors in Lang-8-Ko that can be resolved with this preprocessing check.

9.2.2 Adding Weight to Classifier Confidence

One issue with the WaC-Ko corpus that must be considered at this point is that of particle distribution. In Table 8.1, we can see that particles are not equally represented in the WaC-Ko corpus. For example, 를 (*lul*) occurs about 1.4 million times, whereas 에서 (*eyse*) occurs about 175,000 times. This discrepancy is to be expected, as the corpus is built to model typical, native-like Korean, and some particles naturally occur more frequently than others. However, this natural distribution is not necessarily ideal for an error correction classifier.

The issue stems from the fact that some source particles are more likely to be confused for certain targets than others. In the case of 에 (*ey*), for example, it can be

Source Particle	Weighted Confusion Set
ey	에서*10, 를, 가, 는
eyse	에*10, 를, 로
lul	가*1.5, 에, 는, 도
ka	를*2.5, 는*2.8, 에, 의, 도

Figure 9.1: Weighted Confusion Sets for Selection Experiments

expected that 에서 (*eyse*) is the best option for correcting the particle because they perform similar functions and are known to be confused for one another in practice by learners of Korean, as we can confirm by examining confusion matrices based on Lang-8-Ko and the Lee et al. (2009a) learner corpus (see Figure 5.10). However, the confusion matrix for *ey* should also include the subject marker and object marker, among other things, even though they are less likely to be the best correction for a given error. Rather than sample the corpus here, we add weight to the classifier confidence assigned to each candidate when selecting the best option.

We use the *gold* experiments on the development section of Lang-8-Ko to set the weights for the source particles 가 (*ka*), 에 (*ey*),를 (*lul*), and 에서 (*eyse*) and apply these weights in all other experiments. For omission errors, we rely on the original confidence provided by the classifier, as we have not run omission error detection experiments on Lang-8-Ko. The assigned weights are given in Figure 9.1, if no weight is assigned to a particle, its weight is simply 1, or the confidence from the classifier.

For the inherent case particles 에 (*ey*) and 에서 (*eyse*), they are most likely to be confused with one another, and these particles occur far less frequently than some of the members of their confusion sets, e.g. 를 (*lul*) and 가 (*ka*). Thus, we multiply the classifier's confidence in 에서 *eyse* by ten for instances where 에 (*ey*) is the source particle, and vice versa. In the case of the object marker, 를 (*lul*), we add weight to the subject marker 가 (*ka*), as learner errors often involve using an object when a subject is necessary. Finally, for 가 (*ka*), we add weight to the object marker, as well as the topic marker, 는 (*nun*). The weights applied for *lul* and *ka* source errors are lower, as the most likely substitution candidates are some of the most frequently used particles.

9.3 Results

We present the results for each of our four experiments here. Each table presents a distinct source error in each row followed by the number of errors (*n*), the number instances that the system gets correct, and system's accuracy (Acc). For these experiments, accuracy is the most useful metric, as the precision and recall for detection have already been established and will not be altered at this phase. Discussion and analysis of the results follow in Section 9.4.

9.3.1 KoLLA Test Data: *Pipeline* results

First, we present results for *pipeline* (errors found by the detection modules) experiments on the KoLLA test data. Table 9.1 provides the results for the FB sub-corpus. Likewise, Tables 9.2, 9.3, and 9.4 provide the best results for testing on the HB, FI, and HI sub-corpora, respectively. These tables include rows for both *Total* and *All TPs*. For the HB and FI sub-corpora, these are identical, however, there are a greater number of total errors for the FB and HI sub-corpora in the *All TPs* row. The difference here is in what is counted for evaluation; the *Total* row excludes errors of commission that were identified by the substitution error detection module, as the training sets for correction do no include a *null* option; in essence, this row provides only the count of errors we have some chance of getting correct. The *All TPs*

Source	*n*	# correct	Acc
ey	7	2	28.57
eyse	1	1	100
lul	6	4	66.67
ka	4	4	100
null	26	11	42.31
Total	44	22	50.00
All TPs	47	22	46.81

Table 9.1: KoLLA FB — *Pipeline*

Source	*n*	# correct	Acc
ey	3	1	33.33
eyse	1	1	100
lul	2	2	100
ka	2	2	100
null	23	13	56.52
Total	31	19	61.29
All TPs	31	19	61.29

Table 9.2: KoLLA HB — *Pipeline*

Source	*n*	# correct	Acc
ey	1	1	100
eyse	1	1	100
lul	4	4	100
ka	3	2	66.67
null	44	22	50.00
Total	53	30	56.61
All TPs	53	30	56.61

Table 9.3: KoLLA FI — *Pipeline*

Source	*n*	# correct	Acc
ey	4	1	25.00
eyse	-	-	-
lul	5	4	80.00
ka	2	2	100
null	35	16	45.71
Total	46	23	50.00
All TPs	47	23	48.93

Table 9.4: KoLLA HI — *Pipeline*

row includes all substitution and commission errors, to give a sense of exactly how many errors we can correct.

9.3.2 KoLLA Test Data: *Gold* results

Here, we present results for *gold* (assuming the detection models achieved 100% precision and recall) experiments on the KoLLA test data. Table 9.5 provides the results for the FB sub-corpus. Tables 9.6, 9.7, and 9.8 provide the best results for testing on the HB, FI, and HI sub-corpora. For this set of experiments, we examine only instances of substitution errors, i.e. no commission errors are included. Because these experiments assume gold standard substitution detection, we consider only and all substitution errors, allowing for the possibility that a separate module could perform commission error detection/correction.

Source	*n*	# correct	Acc
ey	25	16	64.00
eyse	5	5	100
lul	10	8	80.00
ka	14	11	78.57
null	63	35	55.56
Total	117	75	64.11

Table 9.5: KoLLA FB - *Gold*

Source	*n*	# correct	Acc
ey	16	10	62.50
eyse	7	4	57.14
lul	5	3	60.00
ka	7	5	71.43
null	95	66	69.47
Total	130	88	67.69

Table 9.6: KoLLA HB - *Gold*

Source	*n*	# correct	Acc
ey	12	8	66.67
eyse	6	3	50.00
lul	8	7	87.50
ka	15	9	50.00
null	85	65	76.47
Total	126	92	73.02

Table 9.7: KoLLA FI - *Gold*

Source	*n*	# correct	Acc
ey	11	5	45.45
eyse	-	-	-
lul	8	6	75.00
ka	10	5	50.00
null	76	56	73.68
Total	105	72	68.57

Table 9.8: KoLLA HI - *Gold*

9.3.3 Lang-8-Ko Test Data: *Pipeline* Results

Here, we present the results for testing on the Lang-8-Ko corpus, again in the *pipeline* scenario. Table 9.9 provides the results for the development (cross-fold) section of the corpus. Table 9.10 shows the results on the unseen Lang-8-Ko fold.

Source	*n*	# correct	Acc
ey	183	90	49.18
eyse	64	51	79.69
lul	326	306	93.87
ka	243	226	93.01
Total	816	673	82.48

Table 9.9: Lang-8-Ko Dev. — *Pipeline*

Source	*n*	# correct	Acc
ey	28	11	39.28
eyse	6	6	100
lul	28	27	96.43
ka	28	28	100
Total	90	72	80.00

Table 9.10: Lang-8-Ko Test — *Pipeline*

9.3.4 Lang-8-Ko Test Data: *Gold* Results

Here, we present the results for testing on the Lang-8-Ko corpus, again in the *gold* scenario. Table 9.11 provides the results for the development (cross-fold) section of the corpus. Table 9.12 shows the results on the unseen Lang-8-Ko fold.

Source	*n*	# correct	Acc
ey	1667	629	37.73
eyse	619	282	45.58
lul	2016	1281	63.54
ka	2541	1868	73.51
Total	6843	4060	59.33

Table 9.11: Lang-8-Ko Dev. — *Gold*

Source	*n*	# correct	Acc
ey	186	65	34.95
eyse	67	22	32.83
lul	231	137	59.31
ka	272	211	77.57
Total	756	435	57.54

Table 9.12: Lang-8-Ko Test — *Gold*

9.4 Analysis

KoLLA Learner Level For substitution error detection, we saw a distinct difference between the performance of the system on texts written beginner and intermediate level students, where the system appeared to be better equipped to handle beginner data. This trend does not hold here, as performance is similar across learner levels in the *pipeline* experiments (Tables 9.1, 9.2, 9.3, and 9.4), and a slight increase in performance on intermediate data in the *gold* experiments (Tables 9.5, 9.6, 9.7, and 9.8).

Source Particle There are three distinct categories of source errors that this module deals with: Inherent case (*ey* and *eyse*), structural case (*lul* and *ka*), and omissions (*null*). The KoLLA corpus contains relatively few errors of inherent and structural case particle errors, so Lang-8-Ko is probably a more stable source for examining the differences. As was the case with substitution error detection, we see that the system does a better job at selecting the best option to correct a structural case particle error than for inherent case. In Table 9.9, for example, the accuracy for correcting *lul* and *ka* source errors are both over 90%. The results drop in the Lang-8-Ko *gold* scenario overall (cf. Table 9.11), but the increased performance among structural case particles holds. Looking at (Tables 9.5, 9.6, 9.7, and 9.8), one can see that over a large proportion of omission errors, the system can reliably select the best particle to correct the error about 70% of the time (222/319, for all sub-corpora combined), regardless of learner level/type.

***Pipeline* vs. *Gold* Experiments** A contrastive look at the performance of the system in the *Pipeline* and *Gold* scenarios yields some interesting findings. First of all, considering the KoLLA *pipeline* experiments, it is obvious that the number of instances for each particle type is too low to draw useful conclusions about the different source particles. Overall though, there are 174 instances, of which the system gets 94 correct, or about 54%. We see much higher accuracy in the *gold* results, with performance the FB sub-corpus the worst at 64.11% (*Total*), and up to 73.02% in the FI sub-corpus.

For the Lang-8-Ko testing experiments, on the other hand, we see the exact opposite result; the *pipeline* scenario results are markedly better than for the *gold* scenario. However, given the expectation that Lang-8-Ko likely contains a non-trivial amount of noise in the annotations due to annotator accuracy, POS tagging and other pre-processing steps, the *gold* results may be unfair. It could be that the errors that we did not detect with the substitution module are not errors, or that the sentences are just too far divergent from native-like Korean to allow for the system to process them accurately. It is also worth noting that these results represent the low end of how well the system performs on this data, as the annotations only mark a single particle as acceptable for each instance. The KoLLA annotations allow for sets of particles in some cases (see Section 4.3.8). Altering the Lang-8 annotation extraction process could allow for sets of particles in this test data as well, which would likely lead to better results.

Conclusion

This chapter described our final system module: the error correction system. This module is designed to select the best particle to correct an error, and can do so as accurately as 93% of the time over a large sample of structural case particle source errors (cf. Table 9.9). The system is less accurate when dealing with inherent case and omission source errors, but still produces accurate corrections over 50% of the time in most experiments. While there is still work to be done to optimize over different source errors and for different leaner levels and types, we feel that this is a good foundation for how such a module could be built.

Chapter 10

Conclusion

Often when you think you're at the end of something, you're at the beginning of something else.

— Fred Rogers

When we began the research that would eventually become the basis for this dissertation, the main goal was to develop a reliable, accurate system for detecting and correcting particle errors in the writing of Korean language learners. Along the way, several more goals became necessary steps to allow completion of the error detection system. First of all, we had to procure, annotate, and optimize several corpora to serve as training and/or testing data for a machine learning system. Secondly, given the dearth of automatic grammatical error detection research being carried out on languages other than English, we wanted to establish methodologies that could work for other languages as well. Along the same lines, we were interested to see what methods of English error detection could be adapted to work for similar problems in other languages.

10.1 Data Development

In Chapters 4 and 5 we presented three distinct corpora: The KoLLA corpus of expert-annotated learner Korean, the WaC-Ko corpus of web-scraped, well-formed Korean, and the Lang-8-Ko corpus of peer-annotated learner Korean automatically extracted from the social networking language learning website, Lang-8. Each of these corpora, and the process by which they were procured, are valuable contributions to the field of error detection.

10.1.1 KoLLA

The KoLLA corpus features an annotation scheme that was designed specifically to provide all of the necessary information to serve as a test corpus for an automatic error detection system, while also serving as a valuable resource for research on language pedagogy and corpus based studies of Korean learner language. The annotations provide information about each particle in the corpus, noting whether or not there is an error, the type of error, a suggested correction, the function of both the erroneous and the correct particle, as well as segmentation, spelling, and spacing problems in the surrounding context.

The corpus is divided evenly among heritage leaners and foreign learners, and among beginner and intermediate learners, allowing for detailed analysis of both system performance and language learner tendencies among different types of learners. The corpus itself is freely available[1] and should prove useful to any researchers working on problems relating to the role of particles in Korean language learning. The description of the annotation scheme is useful itself, as great care was taken in designing informative, yet flexible error annotations. The steps described in annotating KoLLA should prove useful to researchers designing annotation schemes for similar issues in other languages.

10.1.2 WaC-Ko

The WaC-Ko corpus is a large-scale corpus of Korean text that was scraped from the web for the purpose of serving as training data for building machine learning models for Korean error detection tasks. The process of collecting reliable Korean data from the web proved to be non-trivial, as encoding issues, the presence of other languages along with Korean in many webpages, and the question of whether or not returned web-pages contained native-like Korean were all issues that had to be dealt with along the way.

We ran a series of experiments to establish methodologies for gathering high quality Korean data from the web, that should apply to any language that can be

[1] `http://cl.indiana.edu/~kolla/`

found on the internet, as long as there is a decent representation of the language to be found. The experiments showed that smaller, topic-focused corpora can produce better quality training data for machine learning than larger, less focused corpora. We hope that WaC-Ko and the methodologies defined in building it will prove useful to other researchers working on Korean, as a major hurdle for many projects focusing on Korean language is the scarcity of freely available, large-scale corpora; a problem which surely affects other languages as well. Utilizing the internet to build them offers one low-cost, practical solution to the problem.

10.1.3 Lang-8-Ko

The only freely available, annotated, large-scale corpus of learner language that we know of are collections of English learner texts, e.g., NUCLE (Dahlmeier et al., 2013) and ICLE v2 (Granger et al., 2009). Thus, we set out to find some way to gather a large-scale corpus of learner Korean. Following methodologies laid out in Mizumoto et al. (2011) and Cahill et al. (2013a), we utilize data pulled from the social networking language learning website Lang-8. By processing the revision logs from essays written by learners and corrected by native speakers, we were able to produce a corpus of over 800,000 words of learner Korean with particle annotations similar to those found in KoLLA.

These social networking language learning websites offer a unique approach to building annotated learner corpora as the annotation is already provided. Such data sources have only recently become available, and require more investigation to ascertain how useful they may be to the research community. The methodology laid out in this dissertation offers one solution to building corpora with annotation for specific bound morpheme errors. There are still issues to be addressed involved in verifying the validity of the annotations and filtering noise from the corpus, but overall, data collected from these websites could prove to be invaluable to researchers who focus on automatic grammatical error detection for languages other than English.

10.2 Error Detection and Correction

Having gathered the necessary data for error detection experiments, we moved on to actually establishing the error detection and correction tasks themselves in Chapters 7, 8, and 9, dealing with omission errors, substitution errors, and error correction individually. Omission error detection and substitution error detection are both binary tasks that send detected errors on to the multi-class error correction module.

10.2.1 Omission Error Detection

We define the task of omission error detection as a binary sequence labeling task where a classifier decides whether or not nominals with no particle should actually have one. The omission error detection module works well right now, with nearly 85% precision and 44% recall. The system is extremely reliable at detecting errors in advanced learner language, achieving 90% precision and 45% recall for the KoLLA heritage intermediate sub-corpus, though it is less reliable for the foreign beginner corpus, with precision of 78%. While there is always room for improvement, the omission error detection module described here works well as is, and with more focused training sets, could likely be lead to even better results for different types of learners. The way the task is defined, i.e. particle insertion, is extendible to a variety of error types and other languages; any grammatical feature for which some percentage of errors can be encoded as a binary distinction of presence or absence, e.g. prepositions, articles, particles, punctuation, etc., is a good candidate for this methodology.

10.2.2 Substitution Error Detection

For errors that involve existing particles, we implement a second module built on a maximum entropy classifier that is tasked with determining whether or not a particle is appropriate for its context. Again, the task is binary as each particle is considered independently as appropriate or not. Most of the errors dealt with at this phase are errors of substitution, although errors of commission and omissions of particles in a sequence can be caught with this module as well. Here we ran a variety of experiments training on both WaC-Ko and Lang-8-Ko, and testing on KoLLA and Lang-8-Ko. We also examined the utility of adding learner particles as a feature to training the classifier.

Our results show that, while this module is not perfect, there are some scenarios where it functions quite well. For beginning learners, the system has fairly high precision; 85% for all source particles in the KoLLA foreign beginner sub-corpus. The module performs particularly well on cases where the source of the error are subject or object markers with nearly 80% precision for subjects and over 90% for objects over a large sample of Lang-8-Ko test data. The spread of the results indicates that building specific classifiers for different learner types and source errors is likely a good idea moving forward.

10.2.3 Particle Error Correction

The final module in the system handles correcting the errors that have been detected by the previous modules. The task here is multi-label, but restricted to cases where we are fairly certain an error has occurred. We restrict the search space to look only at candidates from each source particle's confusion set in order to ensure that unlikely candidates do not exacerbate the difficult task of selecting the best particle for a given context.

The results stay above 50% accuracy when considering all source particles, though the system is again better equipped at dealing with subject and object markers, as opposed to inherent case markers (cf. English prepositions), where the selection of the best particle appears to be more difficult. Still, for subjects and objects, the system is able to suggest the best particle for correction over 90% of the time in the Lang-8-Ko corpus, which seems to be composed mostly of beginning level learners of Korean.

The optimized system features omission error detection precision of about 85% and recall over 44%. For substitutions, we know that the system can detect object and subject errors with between 80 and 90% precision. Finally the system correct subject and object errors 90% of the time and other errors with at least 50% accuracy. Given all of this, we are confident that this error detection and correction system could prove useful to learners and teachers of Korean. We hope that the methodologies that have proven useful here will serve as inspiration for other researchers to explore building error detection systems for a wide range of languages and grammatical features.

10.3 Future Work

10.3.1 KoLLA Annotation

As mentioned previously, the current size of the KoLLA corpus (less than 1500 sentences) leads to some difficulties in development and calls into question the scalability of results presented for the corpus. An obvious solution, then, is to add more data. While the initial annotation of the 100 essays in the corpus was a long process, the majority of time was actually spent developing and refining the annotation scheme. With the annotation scheme finalized and methodology for utilizing EXMARaLDA already in place, adding more essays and increasing the size of the corpus quickly is a realistic expectation. A reasonable goal would be to acquire and annotate enough data to have 1000 instances of each of the major particle error types. The 100 essays that we have now include 285 and 210 omission and substitution errors, respectively. Thus, quadrupling the size of the corpus should get us close to that mark, and take the sentence count closer to 6000.

10.3.2 Optional Particle Annotation

One issue that was touched upon in Section 4.3.6 is the idea of including whether each particle is optional or obligatory. That is, some particles can often be dropped (most often in informal Korean, but sometimes in more formal registers) without compromising grammaticality. For example, in (41), we see a sentence with topic, subject, and object markers in (41a). Any one or even all of these particles can be dropped in this case, without loss of grammaticality, resulting in sentence (41b). This issue has many subtle nuances that must be sussed out before taking on optional/obligatory particles in the context of grammatical error correction. Lee and Song (2011) make a distinction between particle ellipsis(i.e. dropping a required particle) and non-occurence (i.e. not using a non-required particle), and identify three contructions for non-occurence: genetives in certain syntactic contexts, in light verb constructions, and with bound nouns, which must be preceeded by a demonstrattive and only occur in specific situations. Using these guidelines could be a good start for annotating optionality in KoLLA.

(41) a. Full: 오늘은 학생이 책을 읽어요.
Today-TOP student-SBJ book-OBJ read

b. Dropped: 오늘 학생 책 읽어요.
Today student book read

'The student is reading a book today.'

The current version of our annotation scheme requires that all particles must be present, but one can see the potential benefit of marking some particles as optional for an error detection system, depending on the use case. An obvious way to obtain these annotations would be to ask trained annotators to mark each particle in a corpus as optional or obligatory as part of the annotation scheme. But for something like Lang-8, which has over 800,000 words, this is not realistic. This methodology also allows for bias from annotators. Crowd-sourcing, i.e. using a service such as Amazon's Mechanical Turk where workers are paid to provide feedback, presents a potential solution for adding optional/obligatory annotations on a large scale and capturing wide-spread opinion for each example. Workers could be given a task of saying whether a given particle is necessary, optional, or wrong for a sentence. With such information about each particle, results could be provided for the corpus, where optional particles are discounted in the case of omission, for example. With more robust results, the system then could be tuned for formal or casual writing scenarios to provide better feedback to learners.

10.3.3 Improving Lang-8 Annotations

The current annotations that we automatically generate for Lang-8 represent a large step forward for Korean leaner corpus studies. However, there is still room for improvement in terms of adding more content to the annotations. One piece of information that is currently discarded from the annotations is when annotators disagree as to the best particle for a given context. As it stands, the annotation generation algorithm only considers one correction from a set of submitted corrections. If all members of that set were analyzed and one nominal was given different particles amongst the different corrections, we could use that information to build a set of appropriate particles for the nominal in question, as we have for some nominals in KoLLA. For example, in many sentences subject and topic markers are interchangeable in Korean for the same word; adding that knowledge to the corpus could be beneficial in terms of allowing more robust results for error correction and for providing a better look at what constitutes grammatical Korean.

Another issue that comes from the current practice of keeping only one annotation is that we cannot be sure that we are keeping the best one. Here again, crowdsourcing could be a useful tool. If each sentence where there are multiple corrections was provided to workers, they could select which corrections were most suitable for a given sentence. This would help to improve the quality of the corpus by disregarding bad sentences and also allow us to build better sets of particles, as discussed in the previous paragraph. Also, on a large enough scale, i.e. with enough annotations, one could hope to learn patterns that would allow for automatically selecting the best annotations for a given sentence.

10.3.4 Procuring More Annotated Data

As the Lang-8-Ko corpus proves to be such a good resource for this work, gathering more similarly constructed data for helping with grammatical error detection is obviously desirable. Though there has been a rise in the prevalence of SNSLL websites, many of these, unfortunately, do not freely provide their data for research purposes. Another option, though, is the use of Wikipedia revision logs. Cahill et al. (2013b) utilized Wikipedia revisions to generate a corpus of English preposition errors, using an approach similar to the one described in Section 5.7. Wikipedia can be freely edited by users from across the world, so the situation where an ungrammatical edit is made by a non-native speaker and subsequently edited for grammar, but not content, by a native speaker. This approach works well for English, which has such a large presence on Wikipedia, and which is widely used by non-native speakers. Whether or not the same would hold true for Korean which has decidedly less non-native speakers is not as certain, but is worth investigating moving forward. It is also worth noting the using Wikipedia logs is actually a good option for getting more data to build statistics to inform decisions, rather than getting learner data itself.

10.3.5 Experiments

Because this dissertation represents the first major work, as far as we know, dealing with Korean particle error detection, there are a number of directions to go with research and new experiments. We will describe some of these that we hope to pursue in the future here.

10.3.5.1 Identifying Particle Categories

One major area for improvement upon the results reported in Chapter 9 is that of correcting, i.e. selecting the best particle in the case of, omission errors. For *Pipeline* experiments, the current system achieves only 48% accuracy on omission errors across corpora; though for *Gold* experiments the accuracy is much better at 69%. In any case, we would like to improve these accuracies. In order to improve these results, we propose adding an intermediate classifier to identify what category of particle (e.g. inherent case, structural case, auxiliary, or conjunction) is the best fit to correct a given context. The motivation here is that the classifier tasked with selecting the best particle for omissions simply has too many options, and that accuracy could be improved if we could use a classifier that only included particles of a specific category. Thus, an accurate intermediate classifier tasked with selecting the best category is necessary for this approach. The hope is that high accuracy for such a classifier is attainable given that particle categories are a small, closed set. For example in (42a) the word 것 (*kes*, "photo") should be followed by a subject marker, 이 (*i*), as it is in (42a)[2]. The proposed approach would be to: 1) identify the omission error, 2) determine that a structural case marker is necessary, and 3) choose 이 from the set of case markers.

(42) a. Original: 각 곳에 여러 좋은 것 있어요
each place-*at* many good thing exist

b. Corrected: 각 곳에 여러 좋은 것이 있어요
each place-*at* many good thing-SBJ exist

'There are many good things'

[2]This example is also used in Section 5.6

10.3.5.2 Consistency Across Datasets & Algorithms

The research for this dissertation examines three different tasks: omission error detection, substitution error detection, and error correction. All three tasks were considered separately and the experiments were designed and implemented over a long period of time. As such, some of the decisions regarding what algorithms to use and what corpora to utilize lack consistency. Moving forward, we would like to provide uniformity among these experiments.

Omission Errors in Lang-8-Ko One example of an inconsistency is that we do not presently have results for omission error detection or correction utilizing Lang-8-Ko. We had not yet developed the Lang-8-Ko corpus at the time that these experiments were done, and the results for omission error detection were considered strong and reliable. However, having more data to verify the results is certainly desirable, especially in the case of correcting the errors.

Algorithm Selection Another example of inconsistency in the current work is the change from CRF for omission error detection to maximum entropy classifiers for substitution error detection. The motivation for using CRF was that the presence or absence of a particle is a simple binary choice much like the presence or absence of a comma in English, a task for which CRFs are well suited (cf. Israel et al., 2012), working well with a small training set. We switched to maximum entropy for substitution errors because deciding if a particle is correct for a given context is more nuanced than knowing that a particle is missing, and maximum entropy has been used for similar tasks in English (cf. Tetreault and Chodorow, 2008). While it is a fair point that utilizing a similar approach for both, or even comparing the two to show that one is significantly better than the other for either task would be ideal, it could also be argued that the difference would likely be negligible. CRFs are an implementation of maximum entropy, after all, and side by side comparisons of classifiers on the same task often yield similar enough results that we do not expect a drastic change performance for either task using the other algorithm(cf. Liu et al., 2005). In any case, these experiments should be run to confirm our hypotheses.

10.3.5.3 Using Pseudo Errors

Recently one approach that has been gaining traction in the field of grammatical error correction is that of generating pseudo errors and inserting them in native training data. As described in Chapter 2, Rozovskaya and Roth (2010c) utilized this approach and compared it to training on well-formed data and found that the classifier trained on the corpus with pseudo-errors achieved favorable results. Similarly, the ASO model presented in Dahlmeier and Ng (2011) that combines information from native text with error annotated data performs better than classifiers trained on only native data. In Chapter 8, we found that the classifiers trained on the error annotated Lang-8-Ko corpus performed nearly as well as the classifiers trained on well-formed Korean, despite the fact that Lang-8-Ko is only about four percent of the size of the well-formed WaC-Ko corpus.

Given all of the above, a promising approach could be to use the distribution of particle errors that can be gleaned from Lang-8-Ko to generate pseudo errors in WaC-Ko to serve as training data for classifiers. We could then test and compare classifiers trained on well-formed data, error-annotated data, pseudo-error data, and implement methodology similar to the ASO model.

10.3.5.4 Errors in Topic Markers

A major issue that requires investigation moving forward is that of errors in topic markers, i.e. 는/은 (*nun/un*). In Section 4.3.6, we touched on the fact that topic markers and subject markers are often interchangeable such that either can be used in the same context without compromising grammaticality. Then in Section 8.1.1, we mentioned that the distinction is a tough one even for native speakers to characterize, so we did not include topic markers in the errors we attempt to detect. Some exploratory tests on the KoLLA corpus using the same methodology we used to find other substitution errors applied to the topic marker produced highly unfavorable results; F-score did not go above 33.3% for any sub-corpus. As topic markers are a significant source of learner errors, future work should focus on developing methods for finding these errors. The challenge will be in developing new methodology that focuses on pragmatics and discourse features to determine whether or not a topic marker is appropriate for a given context.

10.3.5.5 Utilizing Previous Decisions

One potential feature for machine-learning based particle error correction that we looked into briefly but ultimately did not pursue for this work was using previous decisions by the system to inform the current decision. Especially in the case of discourse-informed particle choices, such as subject and object particles, knowing what the previously used particle in the sentence is can be very useful. For the experiments presented in this dissertation, we relied on the particles that were provided by the writer to generate these features, accepting that the writer's particle could potentially be incorrect. One could setup a sequential classifier that would keep track of previous decisions so we could know if a previously used particle in a sentence was incorrect, and what the system’s best guess for a particle would be in that context.

Bibliography

Abuhakema, Ghazi, Reem Farajand Anna Feldman and Eileen Fitzpatrick (2008). Annotating an Arabic Learner Corpus for Error. In *Proceedings of the sixth international conference on Language Resources and Evaluation (LREC 2008)*. Marrakech, Morocco.

Amaral, Luiz and Detmar Meurers (2006). Where does ICALL Fit into Foreign Language Teaching? Talk given at CALICO Conference. May 19, 2006. University of Hawaii.

Amaral, Luiz and Detmar Meurers (2011). On using intelligent computer-assisted language learning in real-life foreign language teaching and learning. *ReCALL* 23(1), 4–24.

Attali, Yigal and Jill Burstein (2006). Automated Essay Scoring with E-rater v.2. *Journal of Technology, Learning, and Assessment* 4(3).

Baroni, Marco and Silvia Bernardini (2004). BootCaT: Bootstrapping Corpora and Terms from the Web. In *Proceedings of LREC 2004*. pp. 1313–1316.

Baroni, Marco and Motoko Ueyama (2004). Retrieving Japanese specialized terms and corpora from the World Wide Web. In *Proceedings of KONVENS 2004*.

Berger, Adam L., Vincent J. Della Pietra and Stephen A. Della Pietra (1996). A maximum entropy approach to natural language processing. *Comput. Linguist.* 22(1), 39–71.

Bick, Eckhard (1998). Structural lexical heuristics in the automatic analysis of Portuguese. *Proceedings of the 11th Nordic Conference on Computational Linguistics* .

Boyd, Adriane (2009). Pronunciation Modeling in Spelling Correction for Writers of English as a Foreign Language. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium*. Boulder, Colorado: Association for Computational Linguistics, pp. 31–36.

Boyd, Adriane (2010). EAGLE: an Error-Annotated Corpus of Beginning Learner German. In *Proceedings of LREC-10*. Malta.

Boyd, Adriane (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.

Burstein, Jill, Martin Chodorow and Claudia Leacock (2003). Criterion Online Essay Evaluation: An application for automated evaluation of student essays. In *Proceedings of the Fifteenth Annual Conference on Innovative Applications of Artificial Intelligence*.

Cahill, Aoife, Martin Chodorow, Susanne Wolff and Nitin Madnani (2013a). Detecting Missing Hyphens in Learner Text. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*. Atlanta, Georgia: Association for Computational Linguistics, pp. 300–305.

Cahill, Aoife, Nitin Madnani, Joel Tetreault and Diane Napolitano (2013b). Robust Systems for Preposition Error Correction Using Wikipedia Revisions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pp. 507–517.

Chang, Yu-Chia, Jason S. Chang, Hao-Jan Chen and Hsien-Chin Liou (2008). An automatic collocation writing assistant for Taiwanese EFL learners: A case of corpus-based NLP technology. *Computer Assisted Language Learning* 21(3), 283–299.

Chodorow, Martin, Markus Dickinson, Ross Israel and Joel Tetreault (2012). Problems in Evaluating Grammatical Error Detection Systems. In *Proceedings of COLING 2012*. Mumbai, India: The COLING 2012 Organizing Committee, pp. 611–628.

Chodorow, Martin, Michael Gamon and Joel Tetreault (2010). The Utility of Article and Preposition Error Correction Systems for English Language Learners: Feedback and Assessment. *Language Testing* 27(3), 419–436.

Chodorow, Martin, Joel Tetreault and Na-Rae Han (2007). Detection of Grammatical Errors Involving Prepositions. In *Proceedings of the 4th ACL-SIGSEM Workshop on Prepositions*. Prague, pp. 25–30.

Church, Kenneth W. and Robert L. Mercer (1993). Introduction to the special issue on computational linguistics using large corpora. *Computational Linguistics* 19(1).

Comrie, Bernard (1989). *Language universals and linguistic typology: Syntax and morphology*. University of Chicago press.

Corder, S.P. (1967). The significance of learners' errors. *International Review of Applied Linguistics* 5, 160–170.

Dahlmeier, Daniel and Hwee Tou Ng (2011). Grammatical error correction with alternating structure optimization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. Association for Computational Linguistics.

Dahlmeier, Daniel, Hwee Tou Ng and Siew Mei Wu (2013). Building a Large Annotated Corpus of Learner English: The NUS Corpus of Learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*. Atlanta, Georgia: Association for Computational Linguistics, pp. 22–31.

Dale, Robert, 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*. Montréal, pp. 54–62.

Dale, Robert and Adam Kilgarriff (2010). Helping Our Own: Text Massaging for Computational Linguistics as a New Shared Task. In *International Conference on Natural Language Generation*.

Damerau, Fred J. (1964). A technique for computer detection and correction of spelling errors. *Commun. ACM* 7(3), 171–176.

De Felice, Rachele and Stephen Pulman (2008). A classifier-based approach to preposition and determiner error correction in L2 English. In *Proceedings of COLING-08*. Manchester.

De Felice, Rachele and Stephen Pulman (2009). Automatic detection of preposition errors in learner writing. In *CALICO Journal 26(3) (Special Issue of the 2008 CALICO Workshop on Automatic Analysis of Learner Language)*.

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

Dickinson, Markus (2005). Error detection and correction in annotated corpora. Ph.D. thesis, The Ohio State University.

Dickinson, Markus, 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.

Dickinson, Markus and Joshua Herring (2008). Developing Online ICALL Exercises for Russian. In *The 3rd Workshop on Innovative Use of NLP for Building Educational Applications*. Columbus, OH.

Dickinson, Markus and Scott Ledbetter (2012). Annotating Errors in a Hungarian Learner Corpus. In *Proceedings of LREC 2012*. Istanbul.

Dini, Luca and Giocanni Malnati (1993). Weak Constraints and Preference Rules. In Paul Bennet and Patrizia Paggio (eds.), *Preference in Eurotra*, Commission of the European Communities.

Eeg-Olofsson, Jens and Ola Knutsson (2003). Automatic Grammar Checking for Second Language Learners - the Use of Prepositions. In *Proceedings of Nodalida'03*. Reykjavik, Iceland.

Ellis, Rod (2008). *The Study of Second Language Acquisition*. Oxford University Press, second edn.

Erjavec, Irena Srdanoviȁ, Tomaz Erjavec and Adam Kilgarriff (2008). A Web Corpus and Word Sketches for Japanese. *Information and Media Technologies* 3(3), 529–551.

ETRI (1999). *POS tag guidelines*. Tech. rep., ETRI.

Fantinuoli, Claudio (2006). Specialized corpora from the Web and term extraction for simultaneous interpreters. In *WACKY! WORKING PAPERS ON THE WEB AS CORPUS. GEDIT*. CEDIB.

Flor, Michael and Yoko Futagi (2012). On Using Context for Automatic Correction of Non-word Misspellings in Student Essays. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. Stroudsburg, PA, USA:

Association for Computational Linguistics, pp. 105–115.

Fouvry, Frederik (2003). Constraint Relaxation With Weighted Feature Structures. In *In Proceedings of the 8th International Workshop on Parsing Technologies*. pp. 23–25.

Francis, W. N. and H. Kucera (1979). *Brown Corpus Manual*. Tech. rep., Department of Linguistics, Brown University, Providence, Rhode Island, US.

Fraser, Ian S. and Lynda M. Hodson (1978). Twenty-one Kicks at the Grammar Horse. *English Journal* 67(9), 49–54.

Fujii, Atsushi and Tetsuya Ishikawa (2000). Utilizing the world wide web as an encyclopedia: extracting term descriptions from semi-structured texts. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, ACL '00, pp. 488–495.

Gale, William, Kenneth Church and David Yarowsky (1992). Work on statistical methods for word sense disambiguation. In *Proceedings of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*.

Gamon, Michael (2010). Using Mostly Native Data to Correct Errors in Learners' Writing: A Meta-Classifier Approach. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Gamon, Michael (2011). High-Order Sequence Modeling for Language Learner Detection High-Order Sequence Modeling for Language Learner Error Detection. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*.

Gamon, Michael, 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*.

Golding, Andrew R. and Dan Roth (1996). Applying WINNOW to Context-Sensitive Spelling Correction. In *Proceedings of the International Conference on Machine Learning*. Morgan Kaufmann, pp. 182–190.

Granger, Sylvaine (2004). Computer learner corpus research: current status and future prospects. In U. Connor and T. Upton (eds.), *Applied Corpus Linguistics: A Multidimensional Perspective*, Amsterdam & Atlanta: Rodopi, pp. 123–145.

Granger, Sylviane (2003a). Error-Tagged Learner Corpora and CALL: A Promising Synergy. *CALICO Journal* 20(3), 465–480.

Granger, Sylviane (2003b). The International Corpus of Learner English: A New Resource for Foreign Language Learning and Teaching and Second Language Acquisition Research. *TESOL Quarterly* 37(3), 538–546.

Granger, Sylviane, Estell Dagneaux, Fanny Meunier and Magali Paquot (2009). *International Corpus of Learner English V2*. Presses universitaires de Louvain.

Gravano, Agustin, Martin Jansche and Michiel Bacchiani (2009). Restoring punctuation and capitalization in transcribed speech. In *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing*.

Gui, S. and H. Yang (2003). Zhongguo Xuexizhe Yingyu Yuliaohu (Chinese Learner English Corpus). Shanghai Waiyu Jiaoyu Chubanshe. (In Chinese).

Guo, Yan and Gulbahar H. Beckett (2007). The hegemony of English as a global language: reclaiming local knowledge and culture in China. *Convergence* 40.

Hagen, Kirk (1995). Unification-based parsing applications for intelligent foreign language tutiring system. *CALICO Journal* 2(2), 2–8.

Hagen, Kirk (1999). Spanish for Business Professionals (SBP). Project Web Page.

Han, Chung-Hye, Na-Rare Han, Eon-Suk Ko and Martha Palmer (2002). Development and Evaluation of a Korean Treebank and its Application to NLP. In *Proceedings of LREC-02*.

Han, Chung-Hye and Martha Palmer (2004). A Morphological Tagger for Korean: Statistical Tagging Combined with Corpus-Based Morphological Rule Application. *Machine Translation* 18(4), 275–297.

Han, Na-Rae, Martin Chodorow and Claudia Leacock (2004). Detecting Errors in English Article Usage with a Maximum Entropy Classifier Trained on a Large, Diverse Corpus. In *Language Resources and Evaluation Conference*.

Han, Na-Rae, Martin Chodorow and Claudia Leacock (2006). Detecting Errors in English Article Usage by Non-Native Speakers. *Natural Language Engineering* 12(2).

Han, Na-Rae and Shijong Ryu (2005). *Guidelines for Penn Korean Treebank version 2.0.*. Tech. rep., IRCS, University of Pennsylvania.

Han, Na-Rae, Joel Tetreault, Soo-hwa Lee and Jin-young Ha (2010). Using an error-annotated learner corpus to develop and ESL/EFL error correction system. In *Language Resources and Evaluation Conference*.

Hana, Jirka, Alexandr Rosen, Svatava Škodová and Barbora Štindlová (2010). Error-Tagged Learner Corpus of Czech. In *Proceedings of the Fourth Linguistic Annotation Workshop*. Uppsala, Sweden, pp. 11–19.

Hanaoka, Hiroki, Hideki Mima and Jun'ichi Tsujii (2010). A Japanese Particle Corpus Built by Example-Based Annotation. In *Proceedings of LREC 2010*. Valletta, Malta.

Harrison, Richard (2013). Profiles in Social Networking Sites for Language Learning - Livemocha Revisited. In *Social Networking for Language Education*, Palgrave Macmillan.

Heidorn, G. E., K. Jensen, L. A. Miller, R. J. Byrd and M. S. Chodorow (1982). The EPISTLE Text-critiquing System. *IBM Syst. J.* 21(3), 305–326.

Heift, Trude (2010). Developing an Intelligent Language Tutor. *CALICO Journal* 27(3), 443–459.

Heift, Trude and Mathias Schulze (2007). *Errors and Intelligence in Computer-Assisted Language Learning: Parsers and Pedagogues*. Routledge.

Heinecke, Johannes, Jürgen Kunze, Wolfgang Menzel and Ingo Schröder (1998). Eliminative Parsing with Graded Constraints. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 1*. Stroudsburg, PA, USA: Association for Computational Linguistics, COLING '98, pp. 526–530.

Hirst, Graeme and Alexander Budanitsky (2005). Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering* 11(1), 87–111.

Hovermale, DJ and Dennis N Mehay (2009). Real-word Spelling Correction for CALL. Presentation given at the Midwest Computational Linguistics Colloquium (MCLC-6) workshop.

Ijaz, I. Helene (1986). LINGUISTIC AND COGNITIVE DETERMINANTS OF LEXICAL ACQUISITION IN A SECOND LANGUAGE*. *Language Learning* 36(4), 401–451.

Imamura, Kenji, 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*. Stroudsburg, PA, USA, ACL '12, pp. 388–392.

Islam, Aminul and Diana Inkpen (2009). Real-Word Spelling Correction using Google Web 1T 3-grams. In *Proceedings of the 2009 Conference on Empirical Methods* *in Natural Language Processing*. Singapore: Association for Computational Linguistics, pp. 1241–1249.

Israel, Ross, Joel Tetreault and Martin Chodorow (2012). Correcting Comma Errors in Learner Essays, and Restoring Commas in Newswire Text. In *Proceedings of NAACL-HLT 2012*.

Izumi, Emi, Kiyotaka Uchimoto and Hitoshi Isahara (2004). SST speech corpus of Japanese learners' English and automatic detection of learners' errors. *ICAME Journal* 28, 31–48. `http://icame.uib.no/ij28/Izumi.pdf`.

Izumi, Emi, Kiyotaka Uchimoto and Hitoshi Isahara (2005). Error Annotation for Corpus of Japanese Learner English. In *Linguistically Interpreted Corpora (LINC-2005)*.

Izumi, Emi, Kiyotaka Uchimoto, Toyomi Saiga, Thepchai Supnithi and Hitoshi Isahara (2003). Automatic Error Detection in the Japanese Learners' English Spoken Data. In *Proceedings of ACL-03*. Sapporo, Japan, pp. 145–148.

Jee, Min Jung and Min Jung Park (2009). Review of Livemocha as an online language-learning community. *CALICO Journal* 26(2).

Jensen, K., G. E. Heidorn, L. A. Miller and Y. Ravin (1983). Parse Fitting and Prose Fixing: Getting a Hold on Ill-formedness. *Computational Linguistics* 9(3-4), 147–160.

Jones, Rosie and Rayid Ghani (2000). Automatically Building a Corpus for a Minority Language from the Web. In *Proceedings of the Student Research Workshop at the 38th Annual Meeting of the Association for Computational Linguistics*. pp. 29–36.

Kang, Beom-mo and Hunggyu Kim (2004). Sejong Korean Corpora in the Making. In *Language Resources and Evaluation Conference*. European Language Resources Association.

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

Khan, Mohammad, Markus Dickinson and Sandra Kübler (2013). Does Size Matter? Text and Grammar Revision for Parsing Social Media Data. In *Proceedings of the Workshop on Language Analysis in Social Media*. Atlanta, GA USA.

Kilgariff, Adam and Gregory Grefenstette (2003). Introduction to the Special Issue on Web as Corpus. *Computational Linguistics* 29(3).

Kilgarriff, Adam, Pavel Rychly, Pavel Smrz and David Tugwell (2004). The Sketch Engine. In *Proceedings of EURALEX*.

Kim, Hansaem (2005). *Report of 'Construction of the primary data of the Korean language' project*. Tech. rep., The National Institute of the Korean Language, Seoul.

Kim, Hung-gyu, Beom-mo Kang and Jungha Hong (2007). 21st Century Sejong Corpora (to be) Completed. *Korean Language in America* 12(3).

Kim, Min-Joo (2002). Does Korean have adjectives? *MIT Working Papers in Linguistics* 43, 71–89.

Kim, Yong-Jin and Douglas Biber (1994). A corpus-based analysis of register variation in Korean. In *Sociolinguistic Perspectives on Register*, Oxford University Press.

King, Levi and Markus Dickinson (2013). Shallow Semantic Analysis of Interactive Learner Sentences. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*. Atlanta, GA USA.

Knight, Kevin and Ishwar Chander (1994). Automated postediting of documents. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI)*. Seattle, pp. 779–784.

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

Kukich, Karen (1992). Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys* 24(4), 377–439.

Lafferty, John D., Andrew McCallum and Fernando C. N. Pereira (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

Leacock, Claudia, Martin Chodorow, Michael Gamon and Joel R. Tetreault (2010). *Automated Grammatical Error Detection for Language Learners*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Lee, Iksop and S. Robert Ramsey (2000). *The Korean Language*. Albany, NY: State University of New York Press.

Lee, John and Ola Knutsson (2008). The Role of PP Attachment in Preposition Generation. In *Proceedings of CICLing 2008*. Haifa, Israel.

Lee, John and Stephanie Seneff (2006). Automatic Grammar Correction for Second-Language Learners. In *INTERSPEECH 2006*. Pittsburgh, pp. 1978–1981.

Lee, John and Stephanie Seneff (2008). Correcting Misuse of Verb Forms. In *Proceedings of ACL-08: HLT*. Columbus, Ohio, pp. 174–182.

Lee, Sun-Hee (2004). *Case Markers and Thematic Roles*. Seoul: Hankuk Publishing Co.

Lee, Sun-Hee, 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)*..

Lee, Sun-Hee, Seok Bae Jang and Sang-kyu Seo (2009a). Annotation of Korean Learner Corpora for Particle Error Detection. *CALICO Journal* 26(3).

Lee, Sun-Hee, Seok Bae Jang and Jae young Song (2009b). 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.

Lee, Sun-Hee and Jae-Young Song (2011). Particle Ellipsis in Korean Corpora. In *The 10th Conference for the American Association for Corpus Linguistics*. Atlanta, GA.

Leech, Geoffrey (1992). 100 Million Words of English: the British National Corpus. *Language Research* 28(1), 1–13.

Liaw, Meei-Ling (2011). Review of LIVEMOCHA. *Language Learning & Technology* 15(1).

Liu, Yang, Elizabeth Shriberg, Andreas Stolcke and Mary Harper (2005). Comparing HMM, maximum entropy, and conditional random fields for disfluency detection. In *In Proceeedings of the European Conference on Speech Communication and Technology*.

Lonsdale, Deryle and Diane Strong-Krause (2003). Automated rating of ESL essays. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing - Volume 2*. Stroudsburg, PA, USA: Association for Computational Linguistics, HLT-NAACL-EDUC '03, pp. 61–67.

Lüdeling, Anke, Maik Walter, Emil Kroymann and Peter Adolphs (2005). Multi-level error annotation in learner corpora. In *Proceedings of Corpus Linguistics 2005*. Birmingham.

Macdonald, N., L. Frase, P. Gingrich and S. Keenan (1982). The Writer's Workbench: Computer Aids for Text Analysis. *Communications, IEEE Transactions on* 30(1), 105–110.

Madnani, Nitin, Joel Tetreault and Martin Chodorow (2012). Exploring Grammatical Error Correction with Not-So-Crummy Machine Translation. In *Proceedings* *of the Seventh Workshop on Building Educational Applications Using NLP*. Montréal, Canada: Association for Computational Linguistics, pp. 44–53.

Malouf, Robert (2002). A comparison of algorithms for maximum entropy parameter estimation. In *proceedings of the 6th conference on Natural language learning - Volume 20*. Stroudsburg, PA, USA: Association for Computational Linguistics, COLING-02.

Matsumoto, Yuji, Akira Kitauchi, Tatsuo Yamashita, Yoshitaka Hirano, Hiroshi Matsuda and Asahara Masayuki (1999). *Japanese Morphological Anaylsis System ChaSen version 2.0 Manual 2nd edition*. Tech. rep., NAIST.

Meurers, Detmar (2009). On the Automatic Analysis of Learner Language. Introduction to the Special Issue. *CALICO Journal* 26.

Meurers, Detmar (2013). Natural Language Processing and Language Learning. In Carol A. Chapelle (ed.), *Encyclopedia of Applied Linguistics*, Blackwell, pp. 1–13. to appear.

Mihalcea, Rada and Dan Moldovan (1999). A Method for Word Sense Disambiguation of Unrestricted Text. In *ACL '99: Proceedings of the 37th Annual Meeting on Association for Computational Linguistics*. pp. 152–158.

Mizumoto, Tomoya, 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 5th International Joint Conference on Natural Language Processing*. Chiang Mai, Thailand: Asian Federation of Natural Language Processing, pp. 147–155.

Montrul, Silvina (2010). Current Issues in Heritage Language Acquisition. *Annual Review of Applied Linguistics* 30, 3–23.

Nagata, Noriko (2009). Robo-Sensei's NLP-Based Error Detection and Feedback

Generation. *CALICO Journal* 26(3), 562–579.

Nam, Ki-Shim (1993). *The Usage of Korean Particles*. Seoul: Seokwang Academic Publisher.

Nam, Ki-shim (2005). *Modern Korean Syntax*. Seoul, Korea: Thaehaksa. trans. by Sun-Hee Lee and Allison Blodgett.

Nam, Ki-shim and Yong-kun Ko (2005). *Korean Grammar (phyocwun kwuke mwunpeplon)*. Seoul: Top Publisher.

National Virtual Translation Center (2007). Languages of the world: Critical languages. Retrieved May 5, 2008, from `http://www.nvtc.gov/lotw/months/november/criticalLanguages.html`.

Ng, Hwee Tou, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto and Joel Tetreault (2013). The CoNLL-2013 Shared Task on Grammatical Error Correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 1–12.

Nicholls, Diane (2003). The Cambridge Learner Corpus - error coding and analysis for lexicography and ELT. In *Corpus Linguistics*.

Oyama, Hiromi (2010). Automatic Error Detection Method for Japanese Particles. *Polyglossia* 18.

Park, Seok-Joon, Kil-Im Nam and Sang-Kyu Seo (2003). A Study on Distribution and Usage Patterns of Particles and Endings in a Spoken Text of College Students. *Text Linguistics* 14, 139–167.

Parton, Kristen, Joel Tetreault, Nitin Madnani and Martin Chodorow (2011). E-rating Machine Translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland: Association for Computational Linguistics, pp. 108–115.

Peng, Jing and Kenji Araki (2005a). Correction of Article Errors in Machine Translation Using Web-Based Model. In *IEEE NLP-KE*. Wuhan, China, pp. 393–397.

Peng, Jing and Kenji Araki (2005b). Correction of Article Errors in Machine Translation Using Web-Based Model. In *IEEE NLP-KE*. Wuhan, China, pp. 393–397.

Polinsky, Maria and Olga Kagan (2007). Heritage Languages: In the 'Wild' and in the Classroom. *Language and Linguistics Compass* 1(5), 368–395.

Pollock, Joseph J. and Antonio Zamora (1984). Automatic spelling correction in scientific and scholarly text. *Commun. ACM* 27(4), 358–368.

Ratnaparkhi, Adwait (1996). A Maximum Entropy Model for Part-Of-Speech Tagging. In Eric Brill and Kenneth Church (eds.), *Proceedings of the Empirical Methods in Natural Language Processing*.

Ratnaparkhi, Adwait (1998). Maximum entropy models for natural language ambiguity resolution. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA.

Rimrott, Anne and Trude Heift (2005). Language Learners and Generic Spell Checkers in CALL - CALICO Journal. *CALICO* 23(1).

Rosen, Alexandr, Jirka Hana, Barbora Štindlová and Anna Feldman (2013). Evaluating and automating the annotation of a learner corpus. *Language Resources and Evaluation* pp. 1–28.

Rosenfeld, Ronald (1994). Adaptive Statistical Language Modeling: A Maximum Entropy Approach. Ph.D. thesis, Carnegie Mellon University.

Rozovskaya, A. and D. Roth (2010a). Generating Confusion Sets for Context-Sensitive Error Correction. In *EMNLP*.

Rozovskaya, A. and D. Roth (2011). Algorithm Selection and Model Adaptation for ESL Correction Tasks. In *Proceedings of ACL-2011*.

Rozovskaya, Alla and Dan Roth (2010b). Annotating ESL Errors: Challenges and Rewards. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*.

Rozovskaya, Alla and Dan Roth (2010c). Training Paradigms for Correcting Errors in Grammar and Usage. In *Proceedings of HLT-NAACL-10*.

Ryu, Su-Rin (2007). Synchrone Kasusalternation und Grammatikalisierung. *German Linguistics* 16, 71–90.

Sakaguchi, Keisuke, Tomoya Mizumoto, Mamoru Komachi and Yuji Matsumoto (2012). Joint English Spelling Error Correction and POS Tagging for Language Learners Writing. In *Proceedings of COLING 2012*. Mumbai, India: The COLING 2012 Organizing Committee, pp. 2357–2374.

Schmidt, Thomas (2005). EXMARaLDA und die Datenbank Mehrsprachigkeit - Konzepte und praktische Erfahrungen. *Heterogeneity in Focus: Creating and Using Linguistic Databases* .

Schmidt, Thomas (2010). Linguistic Tool Development between Community Practices and Technology Standards. In *Proceedings of the LREC Workshop Language Resource and Language Technology Standards - state of the art, emerging needs, and future developments*. Valletta, Malta: European Language Resources Association (ELRA).

Schneider, David A. and Kathleen F. McCoy (1998). Recognizing Syntactic Errors in the Writing of Second Language Learners. In *Proceedings of ACL-98*. Montreal, pp. 1198–1204.

Sharoff, Serge (2006). Creating General-Purpose Corpora Using Automated Search Engine Queries. In *WaCky! Working papers on the Web as Corpus. Gedit*.

Shieber, Stuart M. and Xiaopeng Tao (2003). Comma Restoration Using Constituency Information. In *Proceedings of the 2003 Human Language Technology Conference and Conference of the North American Chapter of the Association for Computational Linguistics*.

Sinclair, John (ed.) (1987). *Looking Up: An Account of the COBUILD Project in Lexical Computing*. Collins.

Sohn, Ho-Mihn (2001). *The Korean Language*. Cambridge University Press.

Suppes, Patrick, Dan Flickinger, Elizabeth Macken, Jeanette Cook and Tie Liang (2012). Description of the EPGY Stanford University Online Courses for Mathematics and the Language Arts. In *Proceedings of the International Society for Technology in Education*.

Suzuki, Hisami and Kristina Toutanova (2006). Learning to Predict Case Markers in Japanese. In *Proceedings of COLING-ACL-06*. Sydney, pp. 1049–1056.

Tajiri, Toshikazu, 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*. Stroudsburg, PA, USA, ACL '12, pp. 198–202.

Taylor, Insup (1980). The Korean writing system: An alphabet? A syllabary? a logography? In PaulA. Kolers, MeraldE. Wrolstad and Herman Bouma (eds.), *Processing of Visible Language*, Springer US, vol. 13 of *Nato Conference Series*, pp. 67–82.

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

Tetreault, Joel and Martin Chodorow (2009). Examining the Use of Region Web Counts for ESL Error Detection. In *Web as Corpus Workshop (WAC-5)*. San Sebastian, Spain.

Tetreault, Joel, Jennifer Foster and Martin Chodorow (2010). Using Parse Features for Preposition Selection and Error Detection. In *Proceedings of the ACL 2010 Conference Short Papers*.

Ueyama, Motoko (2006). Evaluation of Japanese Web-based Reference Corpora: Effects of Seed Selection and Time Interval. In *WaCky! Working papers on the Web as Corpus. Gedit*.

Ueyama, Motoko and Marco Baroni (2005). Automated construction and evaluation of a Japanese web-based reference corpus. In *Proceedings of Corpus Linguistics 2005*.

Wagner, Joachim, Jennifer Foster and Josef van Genabith (2007). A Comparative Evaluation of Deep and Shallow Approaches to the Automatic Detection of Common Grammatical Errors. In *Proceedings of EMNLP-CoNLL 2007*. pp. 112–121.

Williamson, David M., Xiaoming Xi and F. Jay Breyer (2012). A Framework for Evaluation and Use of Automated Scoring. *Educational Measurement: Issues and Practice* 31(1), 2–13.

Wunsch, Holger, Sandra Kübler and Rachael Cantrell (2009). Instance Sampling Methods for Pronoun Resolution. In *Proceedings of RANLP 2009*. Borovets, Bulgaria.

Yannakoudakis, Helen, Ted Briscoe and Ben Medlock (2011). A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, OR, pp. 180–189.

Yarowsky, David (1994). Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French. In *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 88–95.

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

Yoon, Juntae and Kisun Choi (1999a). *Study on KAIST corpus*. Tech. rep., Department of Computer Science, KAIST. Written in Korean.

Yoon, Juntae and Kisun Choi (1999b). *Study on POS tagged corpus for Korean*. Tech. rep., Department of Computer Science, KAIST. Written in Korean.

Appendix A

Non-heritage Intermediate-based Seed Terms

Travel		Learning Korean	
kihayngmwun	'essay on travel'	*haksup*	'lesson or learning'
yehayngci	'travel location'	*swuep*	'class'
kwanchalhata	'to observe'	*kanguyhata*	'to lecture'
kyengpi	'travel expense'	*sihem*	'exam'
kitayhata	'to expect'	*phyengka*	'evaluation'
cito	'map'	*kyosa*	'teacher'
kwankwangkayk	'tourist'	*thongyek*	'interpretation'
pangmwunhata	'to visit'	*swucwun*	'level'
kilokhata	'to keep a record'	*yuchanghata*	'to be fluent'
philo	'fatigue'	*mokphyo*	'goal'

Korea

hanpato	'Korean Peninsula'
pwungsup	'traditional customs'
kacita	'tradition'
wentayhata	'to change'
kecwuhata	'to reside'
inkwu	'population'
inceng	'generosity'
thongil	'unification'
kyengcey	'economy'
pwuntan	'separation(North/South)'

Happiness

kaceng	'home'
salang	'love'
canye	'children'
whamok	'harmony'
kachikwan	'personal value'
salm	'life'
anceng	'stability'
cengsincek	'metal, psychological'
pwungyolopta	'to be abundant or to be nourishing'
chwukwuhata	'to pursue'

Personality

thukcing	'characteristics'
yenghyang	'effect or influence'
naysengcek	'introvert'
oyhyangcek	'outgoing'
inseng	'personality'
yoin	'factor'
paltalhata	'to develop'
kyelcenghata	'to decide or determine'
yucen	'inheritance'
hyengsenghata	'to form'

Exercise

kyuchikcek	'regular'
sinchey	'human body'
taieth	'diet'
swumyeng	'lifespan'
hohup	'breathing'
cilpyeng	'disease'
yepanghata	'to prevent (disease)'
piman	'obesity'
cheycwung	'weight'
wumcikita	'to move'

Friend		Gathering	
wuceng	'friendship'	*tongchanghwoy*	'reunion'
kyocey	'interaction or dating'	*hwoysik*	'dinner with members'
inkankwankyey	'human relationship'	*yenmal*	'end of a year'
iseng	'different sex'	*chamsekhata*	'to attend'
sakwita	'to hang out or to have a relationship'	*tomohata*	'to promote'
kaltung	'conflict'	*hwoypi*	'membership fee'
hwahay	'reconciliation'	*tongbanhata*	'to accompany'
paylye	'consideration'	*chotay*	'invitation'
towum	'help'	*chinmok*	'friendship'
cinsim	'sincerity'	*cwunpihata*	'to prepare'

Appendix B

Heritage Intermediate-based Seed Terms

Travel	
yehayngsa	‘travel agency’
yehayngci	‘travel location’
yeceng	‘itinery’
konghang	‘airport’
kyothongpyen	‘transportation methods’
kwankwang	‘tour’
swukpak	‘accomodation’
pangmwunhata	‘to visit’
kyenghem	‘experience’
chwuek	‘memory’

Learning Korean	
haksup	‘lesson or learning’
swuep	‘class’
kanguyhata	‘to lecture’
sihem	‘exam’
phyengka	‘evaluation’
kyosa	‘teacher’
thongyek	‘interpretation’
swucwun	‘level’
yuchanghata	‘to be fluent’
mokphyo	‘goal’

This Semester's Plan	
kwamok	'course'
swuep	'class'
swukanghata	'to take (a course)'
kwyosu	'professor'
mokphyo	'goal'
sillyek	'capacity'
paywuta	'to learn'
alupaith	'part time work'
hakcem	'GPA'
kimal	'sihem final exam'

Sense of value	
insayng	'life'
milay	'future'
seywuta	'set up'
hayngpok	'happiness'
kwakliphata	'to establish'
kayin	'individual'
chengsonyen	'youth'
phantan	'judgement'
pikyohata	'to compare'
sachwunki	'adolescence'

Friend	
wuceng	'friendship'
kyocey	'interaction or dating'
inkankwankyey	'human relationship'
iseng	'different sex'
sakwita	'to hang out or to have a relationship'
kaltung	'conflict'
hwahay	'reconciliation'
paylye	'consideration'
towum	'help'
cinsim	'sincerity'

Invasion of privacy	
kansep	'interference'
kwansim	'interest'
hokisim	'curiosity'
nochwul	'exposure'
cengpo	'information'
poho	'protection'
nonlan	'controversy'
chimpemhata	'to intrude'
pwulpep	'illegality'
kayin	'individual'

Stress elimination	
philo	'fatigue'
hwoypokhata	'to recover'
wuntong	'exercise'
kwalo	'overwork'
cengsin	'spirit'
kenkang	'health'
ssahita	'to stack up'
phwulta	'to get rid of'
yeka	'leisure activities'
chwimi	'hobby'

College entrance exam	
swunungsihem	'learning-aptitude exam'
kwawoy	'private tutoring'
cwunpihata	'to prepare for (an exam)'
chiluta	'to take (an exam)'
kyocay	'reference materials'
kyokwase	'text books'
nanita	'difficulty rate'
cemswu	'score'
wense	'application document'
hapkyekca	'people who passed an exam.'

Health management	
pyengwen	‘hospital’
kenkang	‘kemcin health checkup’
kyuchikcek	‘regular’
wuntong	‘exercise’
taieth	‘diet’
saynghwal	‘supkwan living habits’
piman	‘obesity’
swumyen	‘sleep’
cwuuyhata	‘to be cautious’
siksa	‘meal’

Generation gap	
nai	‘aga’
celmum	‘youth’
noin	‘old people’
pwumonim	‘parents’
kukpokhata	‘to get over’
tayliphata	‘to confront’
ihay	‘understanding’
nukkita	‘to feel’
sakopangsik	‘ways of thinking’
kaltung	‘conflict’

Life dreams	
canglay	'future'
huymang	'hope'
kacita	'have'
wentayhata	'to be big/broad'
silhyenhata	'to accomlish'
kanungseng	'possibility'
tocenhata	'to challenge'
nolyekhata	'to make an effort'
iluta	'to accomlish'
phokihata	'to give up'

Happiness	
kaceng	'home'
salang	'love'
canye	'children'
whamok	'harmony'
kachikwan	'personal value'
salm	'life'
anceng	'stability'
cengsincek	'metal, psychological'
pwungyolopta	'to be abundant or to be nourishing'
chwukwuhata	'to pursue'

Gathering	
tongchanghwoy	'reunion'
hwoysik	'dinner with members'
yenmal	'end of a year'
chamsekhata	'to attend'
tomohata	'to promote'
hwoypi	'membership fee'
tongbanhata	'to accompany'
chotay	'invitation'
chinmok	'friendship'
cwunpihata	'to prepare'

Curriculum Vitae

Ross Israel

raisrael@indiana.edu

Education

- Ph.D. Computational Linguistics, Indiana University, December 2014
  - Dissertaion Title: *Building a Korean particle error detection system from the ground up*
  - Dissertation Committee: Markus Dickinson, Sandra Kübler, Lawrence Moss, Joel Tetreault
- BA English, West Virginia University, May 2007

Employment

Research Programmer October 2012 – Current

USC Information Sciences Institute Los Angeles, CA

Developed code for a natural language understanding system based on abductive reasoning. Worked under the supervision of Jerry Hobbs and Ekaterina Ovchinnikova. Main duties included building a pipeline for processing Spanish text from raw data into logical form for use with an abductive reasoner, building a knowledge base of linguistic axioms for processing of multiple languages.

Natural Language Processing Intern May 2011 – August 2011

Educational Testing Service Princeton,NJ

Developed comma error detection system for scoring learner essays. Worked under mentors Joel Tetreault and Martin Chodorow. Main duties included developing annotation guidelines, overseeing corpus annotation, and developing a machine learning-based approach to automatically detecting comma errors in learner English.

Research Analyst May 2010 – December 2010

Rivera Consulting Group Sellersburg, IN

Worked as a senior member on a team of researchers developing deception detection software. Main duties included developing annotation for a corpus of data collected from the web, managing a group of annotators, and designing a pipeline and software for deception detection.

Outreach Assistant September 2007 – August 2008

East Asian Studies Center at Indiana University Bloomington, IN

Assisted in maintaining the NCTA outreach program. Main duties included authoring and designing recruiting materials, liaising between seminar participants, program leaders, and university staff.

University Appointments

- Indiana University Department of Linguistics, Bloomington, IN

Graduate Assistant January 2011 – May 2012

Main duties included maintaining departmental servers, editing the computational linguistics website, and maintaining the departmental computer lab.

Research Assistant January 2010 – August 2010

Project – Evidence-based Fusion of Hard and Soft Information for the Assessment of Reliability of Soft Information. Worked under supervisors Sandra Kübler and Matthias Scheutz. Main duties included developing code for part of speech tag domain adaptation for search tasks.

Research Assistant January 2009 – May 2009

Project – Analyzing Korean Learner Particles for Use in Computer Assisted Language Learning. Worked under supervisor Markus Dickinson. Main Duties included developing a machine learning based approach to Korean grammatical error correction, including learner corpus annotation, web scraping, and developing code for a Korean NLP pipeline.

Research Assistant May 2008 – December 2008

Project – Infraware Dictation Recognition Engine Project. Worked under supervisors Markus Dickinson and Sandra Kübler. Main duties included developing code for extracting phonetic information from medical dictation, and developing a pipeline for a machine learning-based approach to automatic sentence boundary detection and punctuation insertion.

Publications

- Ross Israel, Markus Dickinson, and Sun-Hee Lee (IJCNLP 2013). Detecting and Correcting Learner Korean Particle Omission Errors. *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Nagoya, Japan.
- Sun-Hee Lee, Markus Dickinson, and Ross Israel (LCR 2013). Challenges in annotating Korean particle errors. In Sylviane Granger, Gaëtanelle Gilquin, Fanny Meunier (eds.) *Twenty Years of Learner Corpus Research. Looking Back, Moving Ahead Proceedings of the First Learner Corpus Research Conference.* Louvain-la-Neuve, Belgium: Presses universitaires de Louvain.
- Martin Chodorow, Markus Dickinson, Ross Israel, and Joel Tetreault (COLING 2012). Problems in Evaluating Grammatical Error Detection Systems. *Proceedings of the 24th Conference on Computational Linguistics*. Mumbai, India.

- Sun-Hee Lee, Markus Dickinson, and Ross Israel (LAW 2012). Developing Learner Corpus Annotation for Korean Learner Particle Errors. *Proceedings of the 6th Linguistic Annotation Workshop*. Jeju, Republic of Korea.

- Ross Israel, Joel Tetreault, and Martin Chodorow (NAACL 2012). Correcting Comma Errors in Learner Essays, and Restoring Commas in Newswire Text. *Proceedings of the 2012 Meeting of the North American Association for Computational Linguistics: Human Language Technologies*. Montreal, Canada.

- Ning Yu, Sandra Kübler, Joshua Herring, Yu-Yin Hsu, Ross Israel, and Charese Smiley (LASSA 2012). LASSA: Emotion Detection via Information Fusion. *Biomedical Informatics Insights*. Volume 5 (Supplement 1).

- Markus Dickinson, Ross Israel, and Sun-Hee Lee (2011). Developing Methodology for Korean Particle Error Detection. *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications*. Portland, OR.

- Sandra Kübler, Matthias Scheutz, Eric Baucom, and Ross Israel (TLT 2010). Adding Context Information to Part of Speech Tagging for Dialogs. *Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories*. Tartu, Estonia.

- Markus Dickinson, Ross Israel, and Sun-Hee Lee (WAC 2010). Building a Korean Web Corpus for Analyzing Learner Language. *Proceedings of the 6th Web as Corpus Workshop*. Los Angeles, CA.

Teaching

- Guest Lecturer:
  - Linguistics 245, Language and Computers. Language Tutoring Systems. February 13 & 15, 2012.
  - Linguistics 555, Programming for Computational Linguistics. Unix. September 6 & 8, 2011.
  - Linguistics 515, The Computer and Natural Language. Special Topic: NLP at Educational Testing Service. September 14, 2011.
  - Linguistcs 545, Computation and Linguistic Analysis. Error-Driven Part-of-Speech Tagging. February 15, 2011.
  - Linguistics 515, The Computer and Natural Language. Special Topic: Grammatical Error Detection. October 15, 2010.