A crucial question for automatically analyzing learner language is which grammatical information is relevant and useful for learner feedback. Based on knowledge about how learner language varies in its grammatical properties, we propose a framework for re-using analyses found in corpus annotation and illustrate its applicability to Korean postpositional particles. Simple transformations of the corpus annotation allow one to quickly use state-of-the-art parsing methods.

1 Introduction and Motivation

The goal of many intelligent computer-aided language learning (ICALL) systems is to provide intelligent feedback to learners on their language production (cf. Heift and Schulze, 2007), and the first step is generally to automatically assign a linguistic analysis to the sentence. This requires a grammatical description of appropriately or inappropriately used constructions. For situations such as describing subject-verb agreement, the grammatical model used is uncontroversial, as most grammars have subjects and verbs. This is naturally not the case with all linguistic constructions, and one must carefully consider the appropriate representation for a language in order to support the analysis of learner constructions.

Many approaches to “detecting ill-formed input” develop a grammar as a part of the process of developing a parser (e.g., Vandeventer Faltin, 2003; Schneider and McCoy, 1998; Menzel and Schröder, 1999). While potentially effective, this is a time-consuming process, limiting the reusability of such methods. Furthermore, there is often a gap between these methods and state-of-the-art parsing methods.
(e.g., Charniak and Johnson, 2005; Nivre et al., 2007b; Petrov et al., 2006). These modern statistical parsers are generally fast, accurate, robust, and reusable. The mechanisms for parsing are separate from the grammar, as the grammar is generally learned from a corpus containing syntactic annotation, i.e., a treebank (see, e.g., Abeillé, 2003). Given a treebank, one can quickly have a working parser. To the extent that the corpus annotation can be used for a grammar model, statistical parsing has the potential to speed up the process of analyzing learner data.

But is the grammatical annotation found in a corpus the most appropriate model to use for analyzing learner data? And, if not, how can we use the annotation in a fashion which supports analyzing learner language and providing intelligent feedback?

To answer this, we first need to make clear what we mean by “supporting” the analysis of learner language. We want to automatically acquire a model of correct usage, which can be adjusted for learner language. The goal of this paper is thus not to perform error detection or diagnosis of any construction, but to provide a framework for obtaining grammatical models from annotated corpora that identify the relevant features for learner language. This viewpoint draws much from the notion of annotation-based processing (Amaral and Meurers, 2007), where the analysis of learner data is a “process of enriching the learner input with annotations.” This slightly different use of the term annotation is based on the same idea of applying linguistic analysis to data. In corpus annotation, the data is annotated by hand, whereas in annotation-based processing, it is automatic. The crucial issue is to determine which properties should be annotated, and in this paper we investigate what can be done when the annotation provided in a corpus does not match the annotation desired for automatically analyzing learner language.

To make these issues concrete, we select a language construction in need of automatic analysis for learners, namely Korean postpositional particles. These function similarly to prepositions in English and have correlates in Japanese. Such particles have clear pedagogical needs and thus are the focus of ICALL systems for Korean and Japanese (Dickinson et al., to appear; Nagata, 1995). Crucially, particles make up a significant portion of learner errors (Ko et al., 2004; Lee et al., this volume), paralleling errors made by ESL learners for prepositions (Izumi et al., 2003; Tetreault and Chodorow, 2008). Thus, we need to determine how to automatically analyze Korean particles for learner language.
2 Background: Korean particles

In Korean, postpositional particles are used to indicate grammatical functions, thematic roles, and the locations of people and objects, as in neun (topic) and i (subject) in (1) (Dickinson et al., to appear).\textsuperscript{1} In some ways, then, they are similar to English prepositions, but, whereas prepositions are limited in their role as markers of grammatical function (e.g., the dative to), Korean postpositions are wider in scope (similar to other languages, e.g., Basque (de Ilarraza et al., 2008)), potentially marking the role of a word in a sentence, adding meaning to a word, or connecting a word to another word or to the whole discourse.

(1) Sumi-neun chaek-i pilyohae-yo
    Sumi-top book-sbj need-polite
    ‘Sumi needs a book.’

Since learners of Korean commonly omit a particle or substitute one particle for another (Lee et al., this volume), we might expect learners to make errors as in (2). The noun chaek (‘book’) must be marked with the subject particle i/kaka\textsuperscript{2} to indicate that the book is the subject that is needed, as in (1). However, English-speaking learners often use the object particle eul/reul with the noun as in (2), wrongly suggesting that the verb pilyohaeyo (‘need’) is a transitive verb.

(2) *Sumi-neun chaek-eul pilyohae-yo
    Sumi-top book-obj need-polite
    ‘Sumi needs a book.’

It is clear that particles are difficult to learn for non-native speakers (Ko et al., 2004). Korean locative particles mark distinctions that are not made in English, differentiating, for example, between the location of a static object versus the location of a dynamic activity. It is thus also no surprise that particles are also difficult to capture in linguistic theory (see, e.g., Lee, 2004; Yoon, 2005).

Lee et al. (this volume) and Ko et al. (2004) categorize particle errors by learners of Korean into six error types: omission, replacement, addition, malformation, paraphrasing, and spacing. With the exception of malformations (the wrong morphophonemic alternation) and spacing errors, these errors require contextual information to be detected. Of the remaining four types, paraphrasing errors are

\textsuperscript{1}For expository ease, we provide transliterations, using the Revised Romanization of Korean. Abbreviations used are: top=topic, sbj=subject, obj=object, dat=dative, decl=declarative

\textsuperscript{2}The distinction between ka and i is a simple morphotactic one; likewise for eul/reul.
beyond the scope of most ICALL work (see, however Bailey and Meurers, 2008),
and addition errors require a detailed analysis of complex particles (i.e., more
than one particle stacked together). Thus, for this study, we focus on delineating
the information needed to detect omission and replacement errors, which together
make up over 60% of particle errors made by beginning learners (Lee et al., this
volume).

As mentioned, particles often function as case markers, indicating nominative,
accusative, or dative case, as in example (3). For these particles, the relationship
between the verb and the noun needs to be known.

   Sumi-sbj Jisu-dat book-obj give-past-decl
   ‘Sumi gave Jisu a book.’

Another type of syntactic role that particles may indicate is that of modification.
This is where particles function most similarly to prepositions, indicating the
type of verbal activity, location of a noun, and so forth. As with prepositions (see,
e.g., Tetreault and Chodorow, 2008), this means that one needs to know specific
lexical, syntactic, and semantic information about the verb and the noun.

Other particles mark connectives, indicate information about a speaker’s in-
tention, or add meaning to a sentence, such as the topic marker (see (1)). Clearly,
discourse information is needed for this type of particle (see, e.g., Lee et al., 2005;
Hong, 2000). In this paper, we focus on what we refer to as syntactic postposi-
tional particles, those expressing syntactic relations among words, including both
argument and adjunct functions. In section 4.2, we fully outline the linguistic
properties needed to analyze the usage of these particles in learner language.

3 Using annotated corpora

To analyze learner language, we could use pre-built parsers for Korean (e.g.,
Chung, 2004; Seo, 1993), but these tools are designed for robust analysis and
not for learner language. Ungrammatical data has been shown to be a problem for
NLP technology used as a part of English learner language analysis (e.g., De Fe-
lie and Pulman, 2008; Lee and Knutsson, 2008), and we expect the same for
Korean. We want more flexibility to adapt the systems and thus look to training
technology from a grammatically-annotated corpus.

A potential pitfall in using grammatical corpus annotation to analyze learner
language is that the annotation may not be the most appropriate for the task at
hand. For example, English corpora often lack agreement features, obviously important for analyzing learner language. We address the general question of obtaining a grammar we want from the annotation we have in this section, and apply it to the specific case of Korean particles in section 4.

3.1 Add information

A general idea for parsing with treebanks is to extract extra information which is not explicitly encoded in the annotation. The first major way of doing this is to recover linguistic properties which are only implicitly annotated. The recovery of so-called latent annotation has been successfully employed to improve parsing by providing the parser with better information (cf. Klein and Manning, 2003; Pate and Meurers, 2007). For example, subject and object noun phrases (NPs) are not marked in the Penn Treebank (PTB, Marcus et al., 1993), but this distinction can be automatically recovered by including parent annotation in a label. In this case, subject NPs are re-annotated as NP’S, indicating that the parent is S, and object NPs are re-annotated as NP’VP.

Latent annotation is useful as information that goes into training the parser. In this case, it turns out that subject NPs (NP’S) are more likely to expand as pronouns, and thus including latent annotation in the corpus that the parser trains on improves accuracy. Whatever properties are important for a final analysis can benefit from being included in training, thereby providing more accurate statistics.

Latent properties can also be recovered after parsing, i.e., from the resulting tree. This is appropriate when introducing the distinction does not help, or actually hurts, accuracy. Such degradations arise because having more distinctions means we have less data about each individual property (see section 3.2). Thus, latent annotation should be introduced only as needed into training.

The second major way to incorporate additional information into corpus annotation is to use one’s intuition, by encoding hand-crafted linguistic generalizations (cf., e.g., Dickinson, 2006). For example, if a treebank lacks agreement information, we can use knowledge about pronouns to add some of it. In (4a), for instance, the PTB tagset does not distinguish what type of personal pronouns (PRP) are used, but we can change the annotation to (4b), as He is always third person singular. This method is best when the amount of distinctions to be introduced is small, and their inclusion is highly reliable. As another example, for determiner error detection, Nagata et al. (2006) write rules to add mass/count noun distinctions to a corpus.
Finally, one can use an external source to make the desired distinctions. For example, if the PTB tagset (Marcus et al., 1993) lacks a needed distinction between subordinating conjunction and preposition, one can also tag the data using the Brown corpus tagset (Kucera and Francis, 1967), which makes this distinction, and merge the results. For general methods across corpora and languages, this is less desirable, as an additional resource may not always be available.

### 3.2 Remove information

As adding information to the annotation can help, so too can removing information, as there is a problem of data sparsity. The more information in the annotation, the more data is needed to obtain accurate statistics about these patterns. Removing information, often in the form of collapsing distinctions, can result in more effective technology. Additionally, many linguistic properties are not predictive of others. For example, Hana et al. (2004) demonstrate that verb tense in Russian does not predict noun case. Thus, for POS tagging, they train subtaggers where each only contains partial information, and then merge the results for full tagging. Feature-based models are similar in that they examine only predictive information.

Relatedly, a model with less information can provide different patterns, filling in what another model may not have captured. For example, Metcalf and Boyd (2006) train two parsers: the first contains lexical information to capture individual verb subcategorization properties, and the other, less informative model highlights more general verb subcategorization trends by not including lexical information. By comparing the output of these two models, they are able to identify verb subcategorization errors in the text.

### 4 A case study of modifying corpus annotation

#### 4.1 The data

The data we use for our case study is the Penn Korean Treebank (KTB), version 2.0 (Han et al., 2002), a syntactically-annotated corpus of 5,010 sentences (132,040 words) consisting of constituency (i.e., phrasal) annotation. In addition
to basic constituents, the annotation also consists of function labels (e.g., subject (SBJ)). We use this popular corpus because have easy access to it, and it is similar to other Penn corpora (for English, Chinese, and Arabic). One could also explore the Sejong Corpus (Kim, 2006), but the official version was not available when we started this work.

There are two points worth noting about the annotation of the KTB. First, due to Korean’s complex morphology (as an agglutinative language), internal morphemes of a so-called eajeol (‘word-phrase’) are represented not as separate tokens, but as bound morphemes, distinguished by using a plus sign between morphemes within a word. Even though each morphological unit is annotated, only a full word is available for the annotation of the tree. As learners will generally be assumed to write full words, we will follow the convention of using full words as syntactic units. Secondly, the treebank contains null elements, including traces and empty pronominals. Clearly, people do not write with such empty elements; thus, they have to be removed, as we describe in section 4.3.1.

4.2 Modeling correct particle usage

Turning to the question of how to tell whether a particle is being used appropriately, we have identified some main questions that need to be addressed by an analysis. The issue is whether these questions can be addressed directly by the annotation and, if not, how they can be derived, which we discuss in section 4.3.

4.2.1 Analysis vs. Annotation

The first question the annotation needs to have an answer for is straightforward: What is the verb, and what are the surrounding NPs? This is directly available in the annotation, as we can see in an example like (5). The verb is given the label VV, and the surrounding noun phrases are annotated as NPs.3

(5) (NP-SBJ toduk/NNX+i/PCA) (VP (NP-OBJ munseo/NNC+reul/PCA)
   a burglar+sblj a document+obj
   (VV humchi/VV+eo/ECS ka/VV+ass/EPF+ta/EFN)))
   steal+ADV go+PAST+DECL
   ‘A burglar stole a document’

3The remaining examples are from the KTB, represented as bracketed structures; + marks a morpheme boundary; / marks a POS label. For clarity, some annotation is left out.
Given that we have a verb and NPs, the next question is whether the annotation indicates which NPs depend upon which verb. This is partially available from the treebank: as can be seen in example (5), the subject (SBJ) NP and the object (OBJ) NP are clearly within the projection of the verb. However, in this and in cases like (6), which verb is not clear: both *humchi* (‘steal’) and *ka* (‘go’) are annotated as VV. The annotation needs to mark the head, so that the head verb is clear.

Additionally, the object NP in (6) is handled via the use of the empty element *T*. We need to discard such null elements, in order to obtain only a surface string. Ideally, the the verb will also be connected to the string acting as the object, namely *komunseo* (‘old document’), which one obtains by following the linking of traces, as shown by the underlined indices.

(6) (NP (S (NP-ADV jinan 1866 nyeon byeonginyangyo dangsi) (S (NP-SBJ peurangseugun+i) (VP (NP-OBJ *T*-1) (CV (VV yaktal/VV+ha+eo ka/VV+eun))) (NP-1 oegyujangkak komunseo))) (NP-1 oegyujangkak komunseo)))

‘Documents that the French army stole at an event in 1866.’

Although it is useful to know simply which words are related, it is even more helpful for particle usage to know what type of relationship a verb and its dependent NPs share. Again, the information is only partially provided, as the annotation scheme maintains somewhat coarse function labels. In example (7), for instance, we have the relations SBJ and COMP, but COMP is a very general grammatical term for “NP complements that occur with adverbial postposition[s]” (Han et al., 2001, p. 4) and is realizable by several kinds of particles. Thus, we have to find a way to insert more fine-grained information into the function labels.

(7) (S (VP (S (NP-SBJ yancheuk+i) (VP (NP-COMP WTO+e) (VV jesो+ha+ki+euro))) (NP-1 oegyujangkak komunseo)))

‘Both companies decided to sue each other before the WTO.’

Relatedly, is the particle annotation fine-grained enough to distinguish different uses that learners have to distinguish? Particles are attached to nouns as sub-word units, and their annotation is restricted to five different tags: PCA (case),
PAD (adverbial), PAN (adnominal), PCJ (conjunctive), and PAU (auxiliary [including topics]). Given the discussion in section 2, we restrict our attention only to the particles which focus on syntactic roles, namely PCA, PAD, and PAN. These labels make important distinctions, such as between arguments and adjuncts (PCA vs. PAN/PAD) and as such should be included, but they are again not rich enough to provide feedback on usage. Consider (8) (where (8c) is a hypothetical example). Whether used correctly or not, all three locative markers (-e, -buteo, and -eso) are labeled PAD and form part of an NP-ADV. The label inventory does not distinguish these cases the way it distinguishes SBJ from OBJ use, largely because these are lexical and semantic differences, an issue we return to in section 4.3.1.

(8) a. (NP-ADV naenyeon-e/PAD) boneos-reul batneunta
   next year+at bonus-OBJ receive
   ‘(They) receive a bonus next year.’

   b. (NP-ADV naenyeon-buteo/PAD) boneos-reul batneunta
   next year+from bonus-OBJ receive

   c. * (NP-ADV naenyeon-eso/PAD) boneos-reul batneunta
   next year+from bonus-OBJ receive

4.2.2 Dependency structures

The annotation we have been describing as desirable is essentially dependency annotation, a common form of annotation to identify grammatical relations between words. An example is in (9), with arrows drawn from heads to dependents. Not surprisingly, dependencies have been argued to be appropriate for Korean and Japanese (e.g., Chung, 2004; Seo, 1993; Kudo and Matsumoto, 2000).

(9) toduk+i munseo+reul humchi+eo ka+ass+ta
   a burglar a document steal go

In fact, for detecting preposition errors in English, grammatical functions are among the most important features (De Felice and Pulman, 2007; Lee and

\[\text{4Note that whatever method of error detection is used, it must allow for more than one correct particle in the same context, similar to English prepositions (cf. Tetreault and Chodorow, 2008).}\]
Knutsson, 2008). Chodorow et al. (2007) use information from the surrounding heads of noun and verb phrases, and they mention the need to distinguish argument and adjunct uses, all of which is captured in a dependency analysis. The advantage of using a full dependency structure, instead of simple context-based features (e.g., Tetreault and Chodorow, 2008) is that a parser has a better chance of accounting for word order variation. This is relevant, in that Korean allows for relatively free word order, or scrambling (see, e.g., Chung, 2004). Additionally, a dependency analysis should provide the relevant grammatical relations for feedback: once an error has been detected, the dependency relations can be consulted to see what type of function the particle has.

4.3 Recovering information from annotation

4.3.1 Acquiring dependencies

After analyzing Korean particles, we have determined that a dependency representation would be appropriate for learner language, but the treebank contains only constituencies. The problem of converting constituency structures to (unlabeled) dependencies is not a new one, however, and such a conversion can be done once one knows what the heads of phrases are (cf., e.g., Collins, 1999; Nilsson and Hall, 2005). A list of so-called head rules indicate how to determine the head category of a phrase, for a particular annotation scheme. To derive a list of such rules requires only a small amount of knowledge of the annotation scheme. The full list of head rules we use is given in appendix A.

In addition, even though we have dependencies, we still need to remove empty elements, in order to obtain only surface strings. We do this after extracting dependency relations, and this allows us to obtain dependencies between all and only actual words in the sentence. This process is straightforward, except for 68 sentences, where the trace is the head; we remove these sentences from the data.

4.3.2 Acquiring grammatical relations

In order to be able to provide relevant feedback to learners, we not only need to know which words are dependent upon which other words, but the specific relationship they have. In other words, we need dependency labels. The most obvious starting point for this is simply to use the function labels that are included
in the treebank on phrases, namely SBJ, OBJ, COMP, and ADV. While these can be easily extracted, the set of syntactic function labels is too coarse to be able to say whether one particle is being used correctly (cf. (8)). We expand the set of function labels by augmenting each relation with more specific information about the type of particle being used.

But what kind of particle information can be included? On the one hand, we could add the particle’s POS tag, to indicate more properties of each particle. But this information is not really any more fine-grained than the current labels; for example, the difference between PCA and PAD is close to the difference between OBJ and ADV. On the other hand, we could include each particle name in the relation. This would put information about the type of relation into the label, e.g., ADV-ege. Aside from its redundancy (though, see section 4.4), with 59 particle types in the KTB (32 for PCA, PAD, and PAN), using individual particle names means we might not have enough data to obtain accurate statistics.

To approach these issues, we use two strategies, normalizing and thresholding. The intuition behind normalization is that some particles function in the same manner, and their selection relies on non-syntactic factors, such as morphophonemic alternations or pragmatic choice. Thus, we group particles into classes, using linguistic intuition, as shown in table 1, and treat the class as a label. All other relations receive a generic label. We follow the conventions in the KTB, even though, for example, europuteo could be considered a stacked particle and ko could be considered a complementizer.

For PCA, we use the function labels SBJ and OBJ because most PCA particles can be replaced by -ka/i (SBJ) or -reul/eul (OBJ). Adverbial particles (PAD) are not easy to group, however, and thus we use a frequency restriction, or threshold, to focus on particles which appear over 50 times in the corpus, giving us 16 particles. This is similar to work on detecting errors in English prepositions, in which a subset of them is selected to analyze (De Felice and Pulman, 2008; Tetreault and Chodorow, 2008; Gamon et al., 2008). Extending the method to rarer particles would likely require more data.

4.4 Removing information from annotation

With these divisions into new dependency labels, our parser can learn the general distribution of the types of particles that learners are attempting to use. But there

---

5We do not use the two other function tags, VOC (vocative) and LV (light verb): VOC only appears once, and LV can be collapsed into OBJ for our purposes.
is massive redundancy in the labeling; EGE, for example, will be used whenever ege is encountered.

We can remove this redundancy in two ways: the label can return to being coarse-grained, or the word token itself can be changed, such that it no longer contains the particle. This latter option is what we want, as particles are exactly what we expect learners to misuse. The particle they use (if any) may not match what was intended. Whether or not we actually include particles in the corpus, the labels we have should still predict the presence of a particular type of particle.

Thus, we create a second corpus to train from, namely one which is identical to the dependency-annotated corpus, but does not contain the particles of interest. Training a parser on this corpus allows us to capture what might have been meant by a learner, as it is less influenced by the actually-realized particles. In other words, this model captures the general relations between words, irrespective of which particle is actually used; this is akin to feature-based models which predict the correct preposition based on the surrounding context, without using information about the preposition itself (cf. Tetreault and Chodorow, 2008; De Felice and Pulman, 2008).

Both models provide a different picture of the data, in some sense aiming to model both the learner’s intentions (no particles) and their production (particles). This additional model, while using less information, can supplement the original

<table>
<thead>
<tr>
<th>POS</th>
<th>Class</th>
<th>Particles</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>SBJ</td>
<td>-kkeseo, -seo, -ka/-i, -eseo</td>
</tr>
<tr>
<td></td>
<td>OBJ</td>
<td>-eul/-reul</td>
</tr>
<tr>
<td>PAN</td>
<td>UI</td>
<td>-ui</td>
</tr>
<tr>
<td>PAD</td>
<td>EUROSEOE</td>
<td>-eurosseo</td>
</tr>
<tr>
<td></td>
<td>EUROPUTEOE</td>
<td>-europeutoe</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>-e</td>
</tr>
<tr>
<td></td>
<td>EGE</td>
<td>-ege</td>
</tr>
<tr>
<td></td>
<td>KO</td>
<td>-ko</td>
</tr>
<tr>
<td></td>
<td>ESEO</td>
<td>-eseo</td>
</tr>
<tr>
<td></td>
<td>EURO</td>
<td>-euro</td>
</tr>
<tr>
<td></td>
<td>IRAGO</td>
<td>-irago</td>
</tr>
<tr>
<td></td>
<td>PODA</td>
<td>-poda</td>
</tr>
<tr>
<td></td>
<td>KWA</td>
<td>-kwa</td>
</tr>
</tbody>
</table>

Table 1: The mapping of particles into classes
one in error detection and diagnosis, whether by examining mismatches or as features for a machine learner.

5 Parsing experiments

Sections 4.3 and 4.4 discussed how the treebank alterations are appropriate for learner language. We now want to show that they are able to be accurately applied in an automatic way.

5.1 Dependency parsing

We extract dependency relations from a sentence before training our parser, for a number of reasons. The first is simply that, since they are only determining word-word relations, dependency parsers are very efficient (cf. Nivre, 2003; McDonald and Pereira, 2006) and can run in a real-time ICALL setting. Secondly, with methods devoted to multi-lingual dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007a), using dependency parsing will better ensure a greater degree of applicability to new languages. A final point about training a parser specifically on the properties we intend to annotate is that learning is optimized for those distinctions. For example, if dependencies are desired for a language like Korean, then the parser can learn that word order is not as important a feature for determining the subject as much as a case marker or specific lexical items.

5.2 Evaluation

5.2.1 Experiment details

To evaluate the parser on the KTB, we use tenfold cross-validation: we run the parser ten times, each time training on nine-tenths of the corpus and testing on the other tenth. For our experiments, we use the gold standard POS tags found in the treebank; future work should incorporate POS tagging (e.g., Han and Palmer, 2004).

We run two different sets of experiments, to gauge accuracy. The first is to evaluate the parser as a straight dependency parser: are we achieving reasonable accuracy on regular, in-domain language? As a subpart of this evaluation, we also see whether either of our two parsing models is able to correctly assign a head and a relation label for each word. This will tell us whether the two models are
providing complementary information, and thus what the potential is for getting the dependency relation correct.

Our second evaluation is to create a small evaluation corpus from the treebank, consisting of 100 sentences, which includes randomly inserted errors (cf. De Felice and Pulman, 2007). These 100 sentences were removed from the training data; in other words, the data sets are disjoint. With these 100 sentences, we created two sets of data, one with randomly-selected substitutions and one with randomly-selected omissions. This allows us to see how the parser works irrespective of other learner errors, such as misspellings. With such a small data set, we must be careful in drawing too many conclusions, but it can at least demonstrate the potential for the methodology.

For all experiments, we use MaltParser (Nivre et al., 2007b), a freely available, state-of-the-art dependency parser; and we report unlabeled and labeled attachment scores (UAS, LAS), that is, the percentage of words which are correctly attached to the correct head. Given that one may select a different parser, or even perform the annotation modifications after parsing, the results we present are only indicative of general effectiveness.

5.2.2 Parsing results

We have two models, one with particles (Model 1) and one without (Model 2), using the same set of relation labels. As we can see from the first row of table 2, Model 1 serves as an effective dependency parser, comparable to results for other languages (Nivre et al., 2007b).

<table>
<thead>
<tr>
<th>Model</th>
<th>LAS</th>
<th>UAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>81.77%</td>
<td>85.27%</td>
</tr>
<tr>
<td>Model 2</td>
<td>67.15%</td>
<td>74.41%</td>
</tr>
</tbody>
</table>

Table 2: Dependency parsing results (tenfold cross-validation)

Removing particles, perhaps unsurprisingly, results in a degradation in accuracy, down to 67.15% in labeled attachment. Crucially, though, the model is providing complementary information. If we compare the models by examining every word to see whether either of the two models correctly predicts the head

---

6 An initial experiment for Model 1 with a reduced set of labels showed only marginal improvement.
and the dependency relation, we find that one model is correct 85.33% of the time (cf. 67.15% and 81.77% LAS). This tells us the potential accuracy of using both models and confirms that they are capturing different, useful information.

In fact, examining the differences can provide an initial gauge on how the models can inform error detection. From the first experiments, we have ten sets of training data, and so we can compare each of the ten trained Model 1 models against their Model 2 counterparts on our 100-sentence evaluation data. When we compare the differences between the models on the evaluation set with substitution errors, we find the two models on average agree on 2102 relations and disagree in 423 cases. Crucially, most agreements (99%) are for correct usage, while 18.41% of disagreements are particle errors, identifying on average 78 of the 100 errors. This figure of 78% is what is most important to focus on now: even with this rather crude way of mismatching models, most of the errors are identified by discrepancies between the models. The results for the omission data set show similar trends, with even more errors detected. The models agree on 2087 relations and disagree on 438, with 99.36% of agreements being correct usage and 19.81% of disagreements being incorrect, and we find 87% of the errors. We have thus outlined a way to identify where to suspect misusage in particle selection.

6 Summary & Outlook

We have shown how the annotation found in a corpus can be adapted for situations requiring analysis of learner language. We examined the specific case of providing a parsing model to provide accurate information about Korean postpositional particles, but the methods of using more or less information are quite general. The ability to use models with differing information allows us to highlight cases which are more likely to be erroneous.

Given that the treebank we used contains newspaper data, a next step is to make the parser more aware of learner data. For statistical parsers, one can perform so-called domain adaptation by self-training (McClosky et al., 2006). We can retrain the parser by first running it on a diverse corpus of Korean (see Han et al., 2006, p. 5) and then retraining the parser on these trees. Interestingly, learning from even a small set of corrected learner sentences can improve performance (cf. Nagata et al., 2006). Alongside this, we must test the methods on a real learner corpus, such as the one described by Lee et al. (this volume).

Following that, more thorough error detection and diagnosis needs to be done, such as predicting which particle should have been used or extending the method-
ology to addition errors arising from stacked particles. The methodology of adapt-
ing corpus annotation described here could be used to provide features for ma-
chine learning methods (De Felice and Pulman, 2008; Tetreault and Chodorow,
2008; Gamon et al., 2008), rule-based methods (de Ilarraza et al., 2008; Eeg-
Olofsson and Knutsson, 2003), or other error detection methods.

Acknowledgements

We would like to thank Sun-Hee Lee, Seok Bae Jang, Soojeong Eom, and San-
dra Kühler for their comments at various stages of this work and the anonymous
reviewer for his/her helpful suggestions.

References


Amaral, Luiz and Detmar Meurers (2007). Putting activity models in the driver’s
seat: Towards a demand-driven NLP architecture for ICALL. Talk given at
EUROCALL. University of Ulster, Coleraine.

Bailey, Stacey and Detmar Meurers (2008). Diagnosing Meaning Errors in Short
Answers to Reading Comprehension Questions. In The 3rd Workshop on
Innovative Use of NLP for Building Educational Applications (ACL08-NLP-

Buchholz, Sabine and Erwin Marsi (2006). CoNLL-X Shared Task on Multiling-
gual Dependency Parsing. In Proceedings of the Tenth Conference on Compu-
tational Natural Language Learning (CoNLL-X). New York City, pp. 149–164.

Charniak, Eugene and Mark Johnson (2005). Coarse-to-fine n-best parsing and
MaxEnt discriminative reranking. In Proceedings of ACL-05. Ann Arbor, MI,
USA, pp. 173–180.

Chodorow, Martin, Joel Tetreault and Na-Rae Han (2007). Detection of Gram-
matical Errors Involving Prepositions. In Proceedings of the 4th ACL-SIGSEM

the Surface Contextual Information. Ph. d. thesis, Korea University, Seoul.

Collins, Michael (1999). Head-Driven Statistical Models for Natural Language


Han, Chung-hye, Na-Rae 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. Las Palmas, Canary Islands, Spain.


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


Heift, Trude and Mathias Schulze (2007). Errors and Intelligence in Computer-
Izumi, Emi, Kiyotaka Uchimoto, Toyomi Saiga, Thepchai Supnithi and Hitoshi
In Proceedings of ACL-03. Sapporo, Japan.
Ko, S., M. Kim, J. Kim, S. Seo, H. Chung and S. Han (2004). An analysis of
Kucera, Henry and W. Nelson Francis (1967). Computational Analysis of Present-
Kudo, Taku and Yuji Matsumoto (2000). Japanese Dependency Analysis Based on
Support Vector Machines. In Proceedings of the Joint SIGDAT Conference on
Empirical Methods in Natural Language Processing and Very Large Corpora.
Hong Kong. pp. 18–25.
Lee, John and Ola Knutsson (2008). The Role of PP Attachment in Preposition
Generation. In Proceedings of CICLing 2008, 9th International Conference on
Intelligent Text Processing and Computational Linguistics. Haifa, Israel.
Lee, Sun-Hee, Seok Bae Jang and Sang kyu Seo (this volume). Annotation of
Korean Learner Corpora for Particle Error Detection. CALICO Journal.
McClosky, David, Eugene Charniak and Mark Johnson (2006). Reranking and
Self-Training for Parser Adaptation. In Proceedings of COLING-ACL-06. Sydney,
Australia, pp. 337–344.
McDonald, Ryan and Fernando Pereira (2006). Online learning of approximate
dependency parsing algorithms. In Proceedings of the 11th Conference of the
European Chapter of the Association for Computational Linguistics (EACL).
Menzel, Wolfgang and Ingo Schröder (1999). Error Diagnosis for Language


Vandeventer Faltin, Anne (2003). Syntactic error diagnosis in the context of computer assisted language learning. Thèse de doctorat, Université de Genève,
A Head rules

The head rules in table 3 work as follows: within a rule, we find the leftmost or rightmost element which is a possible head. For $S \rightarrow NP VP$, for example, $VP$ is the rightmost possible head and thus is the head.

<table>
<thead>
<tr>
<th>Mother</th>
<th>Direction</th>
<th>Possible heads</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADCP</td>
<td>Right</td>
<td>ADC</td>
</tr>
<tr>
<td>ADJP</td>
<td>Right</td>
<td>PAD ECS ADJP LV NNC VJ</td>
</tr>
<tr>
<td>ADV</td>
<td>Right</td>
<td>NNC VJ</td>
</tr>
<tr>
<td>ADVP</td>
<td>Right</td>
<td>NP ADV ADVP S</td>
</tr>
<tr>
<td>CV</td>
<td>Left</td>
<td>VV</td>
</tr>
<tr>
<td>DANP</td>
<td>Right</td>
<td>DAN DANP</td>
</tr>
<tr>
<td>LV</td>
<td>Left</td>
<td>VV VJ</td>
</tr>
<tr>
<td>S</td>
<td>Right</td>
<td>EAN PAN PAU PAD ECS S NP VP NNC ADJP TRACE</td>
</tr>
<tr>
<td>VJ</td>
<td>Right</td>
<td>NNC VJ TRACE</td>
</tr>
<tr>
<td>VP</td>
<td>Right</td>
<td>PAD ECS EAN VP VX VV CV LV NP TRACE</td>
</tr>
<tr>
<td>VV</td>
<td>Right</td>
<td>EAN EFN NNC XSV VJ VV TRACE</td>
</tr>
<tr>
<td>VX</td>
<td>Right</td>
<td>NNX VJ VX TRACE</td>
</tr>
<tr>
<td>WHNP</td>
<td>Right</td>
<td>TRACE</td>
</tr>
<tr>
<td>NP</td>
<td>Right</td>
<td>PRN NNC NNU ADV NP NNX NPR NFW NPN S TRACE</td>
</tr>
<tr>
<td>PRN</td>
<td>Right</td>
<td>NP ADJP ADVP NNC NNX NNU NFW NPR NP S VP</td>
</tr>
</tbody>
</table>

Table 3: Head rules for the Korean Treebank, v. 2.0