Korea University System in the HOO 2012 Shared Task

Jieun Lee[†] and **Jung-Tae Lee**[‡] and **Hae-Chang Rim**[†]
[†]Dept. of Computer & Radio Communications Engineering, Korea University, Seoul, Korea
[‡]Research Institute of Computer Information & Communication, Korea University, Seoul, Korea
{jelee,jtlee,rim}@nlp.korea.ac.kr

Abstract

In this paper, we describe the Korea University system that participated in the HOO 2012 Shared Task on the correction of preposition and determiner errors in non-native speaker texts. We focus our work on training the system on a large collection of error-tagged texts provided by the HOO 2012 Shared Task organizers and incrementally applying several methods to achieve better performance.

1 Introduction

In the literature, there have been efforts aimed at developing grammar correction systems designed especially for non-native English speakers. A typical approach is to train statistical models on well-formed texts written by native English speakers and apply the learned models to non-native speaker texts to correct textual errors based on given context. This approach, however, fails to model the types of errors that non-native speakers usually make. Recent studies demonstrate that it is possible to improve the performance of error correction systems by training the models on error-annotated non-native speaker texts (Han et al., 2010; Dahlmeier and Ng, 2011; Gamon, 2010). Most recently, a large collection of training data consisting of preposition and determiner errors made by non-native English speakers has been released in the HOO (Helping Our Own) 2012 Shared Task, which aims at promoting the research and development of automated tools for assisting authors in writing (Dale et al., 2012).

In this paper, we introduce our error correction system that participated in the HOO 2012 Shared Task, where the goal is to correct errors in the use of prepositions and determiners by non-native speakers of English. We mainly focus our efforts on training the system using the non-native speaker texts provided in the HOO 2012 Shared Task. We also share our experience in handling some issues that emerged while exclusively using the non-native speaker texts for training our system. In the following sections, we will describe the system in detail.

2 System Architecture

The goal of our system is to detect and correct preposition and determiner errors in a given text. Our system consists of two types of classifiers, namely *edit* and *insertion* classifiers. Inputs for the two types of classifiers are noun phrases (NP), verb phrases (VP), and prepositional phrases (PP); we initially pre-process the text given for training/testing by using the Illinois Chunker[1] and the Stanford Part-of-Speech Tagger (Toutanova et al., 2003). For learning the classifiers, we use maximum entropy models, which have been successfully applied to many tasks in natural language processing. We particularly use Le Zhang's Maximum Entropy Modeling Toolkit[2] for implementation.

2.1 Edit Classifiers

The role of an edit classifier is to check the source preposition/determiner word originally chosen by the author in a given text. If the source word is incorrect, the classifier replaces it with a better choice. For every preposition/determiner word,

[1]Available at http://cogcomp.cs.illinois.edu
[2]Available at http://homepages.inf.ed.ac.uk/lzhang10/

The 7th Workshop on the Innovative Use of NLP for Building Educational Applications, pages 251–256,
Montréal, Canada, June 3-8, 2012.

we train a classifier using examples that are observed in training data. The choice for prepositions is limited to eleven prepositions (*about, at, as, by, for, from, in, of, on , to, with*) that most frequently occur in the training data, and the candidates for determiner choice are *the* and *a/an*. In summary, we train a total of thirteen edit classifiers, one for each source preposition or determiner. For each edit classifier, the set of candidate outputs consists of the source preposition/determiner word itself, other confusable preposition/determiner words, and *no preposition/determiner* in case the source word should be deleted. Note that the number of confusable words for each source preposition is decided flexibly, depending on examples observed in the training data; a similar approach has been proposed earlier by Rozovskaya and Roth (2010a). For a particular source preposition/determiner word in the test data, the system decides whether to correct it or not based on the output of the classifier for that source word.

2.2 Insertion Classifier

Although the edit classifiers described above are capable of deciding whether a source preposition/determiner word that appears in the test data should be replaced or removed, a large proportion of common mistakes for non-native English writers consists of missing prepositions/determiners (i.e., leaving them out by mistake). To deal with those types of errors, we train a special classifier for insertions. A training or testing event for this particular classifier is any whitespace before or after a word in a noun or verb phrase that is a potential location for a preposition or determiner. Table 1 shows the five simple heuristic patterns based on part-of-speech tags that the system uses in order to locate potential sites for prepositions/determiners. Note that *s* is a whitespace to be examined, an asterisk (*) means wildcard, and *NN* includes the tags that start with NN, such as NNS, NNP, and NNPS. *VB* is also treated in the same manner as *NN*. The set of candidate outputs consists of the eleven prepositions, the two determiners, and *no preposition/determiner* class. Once a candidate position for insertion is detected in the test data, the system decides whether to make an insertion or not based on the output of the insertion classifier.

Pattern	Example
s+NN	I'll give you all **information**
s+*+NN	I need **few days**
s+VB	It may seem relaxing at **beginning**
s+*+VB	Buy **new colored** clothes
VB+s	I'm looking **forward** your reply

Table 1: Patterns of candidates for insertion

2.3 Features

Both edit and insertion classifiers can be trained using three types of features described below.

- **LEX/POS/HEAD** This feature set refers to the contextual features from a window of n tokens to the right and left that are practically used in error correction studies (Rozovskaya and Roth, 2010b; Han et al., 2010; Gamon, 2010). Such features include lexical features, part-of-speech tags, and head words of the preceding and the following chunks of the source word. In this work, we set n to be 3.
- **HAN** This represents the set of features specifically used in the work of Han et al. (2010); they demonstrate that a model trained on non-native speaker texts can outperform one trained solely on well-formed texts.
- **L1**[3] L1 refers to the first language of the author. There have been some efforts to leverage L1 information for improving error correction performance. For example, Rozovskaya and Roth (2011) propose an algorithm for adapting a learned model to the L1 of the author. There have been many studies leveraging writers' L1. In this work, we propose to directly utilize L1 information of the authors as features. We also leverage additional features by combining L1 and individual head words that govern or are governed by VP or NP.

3 Additional Methods for Improvement

The training data provided in the HOO 2012 Shared Task consists of exam scripts drawn from the publicly available FCE dataset (Yannakoudakis et al.,

[3]L1 information was provided in the training data but not in the test data. Therefore, the benefits of using L1 remain inconclusive in this paper.

a/an	the	*NULL*
6028	114	203

Table 2: Training data distribution for *a/an* classifier

about	as	at	by	for	from
0	3	2510	1	2	3
in	**of**	**on**	**to**	**with**	***NULL***
75	7	20	30	3	41

Table 3: Training data distribution for *at* classifier

2011) with textual errors annotated in HOO data format. From this data, we extract examples for training our classifiers. For example, let w be a source word that we specifically want our classifier to learn. Every use of w that appears in the training data may be an example that the classifier can learn from. However, it is revealed that for all w, there are always many more examples where w is used correctly than examples where w is replaced or removed. Table 2 and Table 3 respectively show the class distributions of all examples for source words *a/an* and *at* that are observable from the whole training data for training *a/an*- and *at*-specific classifiers. We can see that various classes among the training data are unevenly represented. When training data is highly skewed as shown in the two tables, constructing a useful classifier becomes a challenging task. We observed from our preliminary experiments that classifiers learned on highly unbalanced data hardly tend to correct the incorrect choices made by non-native speakers. Therefore, we investigate two simple ways to alleviate this problem.

3.1 Filtering Examples Less Likely to be Incorrect

As mentioned above, there are many more examples where the source preposition/determiner is used without any error. One straightforward way to adjust the training data distribution is to reduce the number of examples where the source word is less likely to be replaced or removed by using language model probabilities. If a language model learned on a very large collection of well-formed texts returns a very high language model probability for a source word surrounded by its context, it may be reasonable to assume that the source word is used correctly. Therefore we build a language model trained on the English Gigaword corpus by utilizing trigrams. Before providing examples to the classifiers for training or testing, we filter out those that have very high language model probabilities above a pre-defined threshold value.

Class	Initial Distribution	After Filtering	After Adding
about	0	0	528
as	3	3	275
at	2510	2367	2367
by	1	1	207
for	2	2	1159
from	3	3	550
in	75	75	1521
of	7	7	1454
on	20	20	541
to	30	30	2309
with	3	3	727
NULL	41	41	41

Table 4: Refined data distribution for *at* classifier

3.2 Adding Artificial Errors

Our second approach is to introduce more artificial examples to the training data, so that the class distribution of all training examples becomes more balanced. For example, if we aim at adding more training examples for *a/an* classifier, we would extract correct phrases such as "*the* different actor" from the training data and artificially convert it into "*a* different actor" so that an example of *a/an* being corrected to *the* is also provided to *a/an* classifier for training. When adding aritificial examples into the training data, we avoid the number of examples belonging to each class exceeding the number of cases where the source word is not replaced or removed. Table 4 demonstrates the results of both the filtering and adding approaches for training the *a/an* classifier.

4 Experiments

4.1 Runs

This section describes individual runs that we submitted to the HOO 2012 Shared Task organizers. Table 5 represents the setting of each run.

Runs	Models	Features	Filtering Threshold	Adding
Run0	LM	n/a	n/a	
Run1	ME	LEX/POS/HEAD	X	X
Run2	ME	HAN	X	X
Run3	ME	LEX/POS/HEAD	-2	X
Run4	ME	LEX/POS/HEAD	-2	O
Run5	ME	LEX/POS/HEAD, L1	-2	O
Run6	ME	LEX/POS/HEAD, L1, age	-2	O
Run7	ME	Insertion: POS/HEAD Other: LEX/POS/HEAD	X	X
Run8	ME	LEX/POS/HEAD	-3	X

Table 5: The explanation of each runs

- **Run0** This is a baseline run that represents the language model approach proposed by Gamon (2010). We train our language model on Gigaword corpus, utilizing trigrams with interpolation and Kneser-Ney discount smoothing.

- **Run1, 2** Run1 and 2 represent our system using the LEX/POS/HEAD feature sets and HAN feature sets respectively. Neither additional method described in Section 3 is applied.

- **Run3, 8** These runs represent our system using LEX/POS/HEAD features (Run1), where examples that are less likely to be incorrect are filtered out by consulting our language model. The threshold value is set to -2 and -3 for Runs 3 and 8 respectively.

- **Run4** This particular run is one where we introduce additional errors in order to make the class distribution of the training data for the classifiers more balanced. This step is incrementally applied in the setting of Run3.

- **Run5, 6** Run5 and 6 are when we consider L1 information and age respectively as additional features for training the classifiers. The basic setup is same as Run4.

- **Run7** This run represents our system with its insertion classifier trained using POS and HEAD features only. No LEX features are used.

Runs	Precision	Recall	F-score
Run0	1.45	15.45	2.65
Run1	1.35	10.82	2.39
Run2	1.23	11.48	2.22
Run3	1.33	10.6	2.36
Run4	1.19	11.26	2.15
Run5	1.02	10.38	1.87
Run6	0.99	9.93	1.79
Run7	1.16	11.26	2.1
Run8	1.46	11.04	2.58

Table 6: Correction before test data revisions

5 Results

Table 6 shows the correction scores of the individual runs that we originally submitted. Unfortunately, we should confess that we made a vital mistake while generating the runs from 1-8; the modules implemented for learning the insertion classifier had some bugs that we could not notice during the submission time. Because of this, our system was unable to handle MD and MT type errors properly. This is the reason why the performance figures of our runs are very low. For reference, we include Tables 7-10 that illustrate the performance of our individual runs that we calculated by ourselves using the test data and the evaluation tool provided by the organizers.

We can observe that Run3 outperforms Run1 and Run4 performs better than Run3, which demonstrates that our attempts to improve the system performance by adjusting training data for classifiers

Runs	Precision	Recall	F-score
Run1	42.67	7.06	12.12
Run2	49.28	7.51	13.03
Run3	47.62	6.62	11.63
Run4	45.45	7.73	13.21
Run5	33.82	10.15	15.62
Run6	8.68	18.54	11.82
Run7	33.33	10.82	16.33
Run8	50.0	7.28	12.72

Table 7: Recognition before test data revisions (system revised)

Runs	Precision	Recall	F-score
Run1	32.0	5.3	9.09
Run2	42.03	6.4	11.11
Run3	34.92	4.86	8.53
Run4	37.66	6.4	10.94
Run5	26.47	7.94	12.22
Run6	5.68	12.14	7.74
Run7	24.49	7.95	12.0
Run8	42.42	7.28	10.79

Table 8: Correction before test data revisions (system revised)

Runs	Precision	Recall	F-score
Run1	49.33	7.82	13.50
Run2	52.17	7.61	13.28
Run3	52.38	6.98	12.31
Run4	51.95	8.46	14.55
Run5	37.5	10.78	16.75
Run6	9.29	19.03	12.5
Run7	36.73	11.42	17.42
Run8	51.52	7.18	12.62

Table 9: Recognition after test data revisions (system revised)

Runs	Precision	Recall	F-score
Run1	34.67	5.5	9.49
Run2	42.02	6.13	10.7
Run3	36.51	4.86	8.58
Run4	38.96	6.34	10.91
Run5	29.42	8.45	13.13
Run6	6.40	13.11	8.61
Run7	25.85	8.03	12.26
Run8	42.42	5.92	10.39

Table 10: Correction after test data revisions (system revised)

help. Moreover, we can also see that L1 information helps when directly used for training features.

6 Conclusion

This was our first attempt to participate in a shared task that involves the automatic correction of grammatical errors made by non-native speakers of English. In this work, we tried to focus on investigating simple ways to improve the error correction system learned on non-native speaker texts. While we had made some critical mistakes on the submitted runs, we were able to observe that our method can potentially improve error correction systems.

Acknowledgments

We would like to thank Hyoung-Gyu Lee for his technical assistance.

References

Daniel Dahlmeier 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*, ACL-HLT '11, pages 915–923, Portland, Oregon.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. Hoo 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the 7th Workshop on Innovative Use of NLP for Building Educational Applications*, HOO '12, Montreal, Canada.

Michael Gamon. 2010. Using mostly native data to correct errors in learners' writing: a meta-classifier approach. In *Human Language Technologies: Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL-HLT '10, pages 163–171, Los Angeles, California.

Na-Rae Han, Joel Tetreault, Soo-Hwa Lee, and Jin-Young Ha. 2010. Using an error-annotated learner corpus to develop an esl/efl error correction system. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, LREC '10, pages 763–770, Malta.

Alla Rozovskaya and Dan Roth. 2010a. Generating confusion sets for context-sensitive error correction.

In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 961–970, Cambridge, Massachusetts.

Alla Rozovskaya and Dan Roth. 2010b. Training paradigms for correcting errors in grammar and usage. In *Human Language Technologies: Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL-HLT '10, pages 154–162, Los Angeles, California.

Alla Rozovskaya and Dan Roth. 2011. Algorithm selection and model adaptation for esl correction tasks. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ACL-HLT '11, pages 924–933, Portland, Oregon.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Edmonton, Canada.

Helen Yannakoudakis, 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 - Volume 1*, ACL-HLT '11, pages 180–189, Portland, Oregon.