# A Unified Framework for Grammar Error Correction

**Longkai Zhang Houfeng Wang**
Key Laboratory of Computational Linguistics (Peking University) Ministry of Education, China
zhlongk@qq.com, wanghf@pku.edu.cn

## Abstract

In this paper we describe the PKU system for the CoNLL-2014 grammar error correction shared task. We propose a unified framework for correcting all types of errors. We use unlabeled news texts instead of large amount of human annotated texts as training data. Based on these data, a tri-gram language model is used to correct the replacement errors while two extra classification models are trained to correct errors related to determiners and prepositions. Our system achieves 25.32% in $f_{0.5}$ on the original test data and 29.10% on the revised test data.

## 1 Introduction

The task of grammar error correction is difficult yet important. An automatic grammar error correction system can help second language(L2) learners improve the quality of their writing. Previous shared tasks for grammar error correction, such as the HOO shared task of 2012 (HOO-2012) and the CoNLL-2013 shared task(CoNLL-2013), focus on limited types of errors. For example, HOO-2012 only considers errors related to determiners and prepositions. CoNLL-2013 further considers errors that are related to noun number, verb form and subject-object agreement. In the CoNLL-2014 shared task, all systems should consider all the 28 kinds of errors, including errors such as spelling errors which cannot be corrected using a single classifier.

Most of the top-ranked systems in the CoNLL-2013 shared task(Ng et al., 2013) train individual classifiers or language models for each kind of errors independently. Although later systems such as Wu and Ng (2013); Rozovskaya and Roth (2013) use Integer Linear Programming (ILP) to decode a global optimized result, the input scores for ILP still come from the individual classification confidence of each kind of errors. It is hard to adapt these methods directly into the CoNLL-2014 shared task. It will be both time-consuming and impossible to train individual classifiers for all the 28 kinds of errors.

Besides the classifier and language model based methods, some systems(Dahlmeier and Ng, 2012a; Yoshimoto et al., 2013; Yuan and Felice, 2013) also use the machine translation approach. Because there are a limited amount of training data, this kind of approaches often need to use other corpora of L2 learners, such as the Cambridge Learner Corpus. Because these corpora use different annotation criteria, the correction systems should figure out ways to map the error types from one corpus to another. Even with these additions and transformations, there are still too few training data available to train a good translation model.

In contrast, we think the grammar error correction system should 1) correct most kinds of errors in a unified framework and 2) use as much unlabeled data as possible instead of using large amount of human annotated data. To be specific, our system do not need to train individual classifiers for each kind of errors, nor do we need to use manually corrected texts. Following the observation that a correction can either replace a wrong word or delete/insert a word, our system is divided into two parts. Firstly, we use a Language Model(LM) to correct errors with respect to the wrongly used words. The LM only uses the statistics from a large corpus. All errors related to wrongly used words can be examined in this unified model instead of designing individual systems for each kind of errors. Secondly, we train extra classifiers for determiner errors and preposition errors. We further consider these two kinds of errors because many of the deletion and insertion errors belongs to determiner or preposition errors. The



*Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 96–102,
Baltimore, Maryland, 26-27 July 2014. 

training data of the two classification models also come from a large unlabeled news corpus therefore no human annotation is needed.

Although we try to use a unified framework to get better performance in the grammar error correction task, there are still a small portion of errors we do not consider. The insertion and deletion of words are not considered if the word is neither a determiner nor a preposition. Our system is also incapable of replacing a word sequence into another word sequence. We do not consider these kinds of errors because we find some of them are hard to generate correction candidates without further understanding of the context, and are not easy to be corrected even by human beings.

The paper is structured as follows. Section 1 gives the introduction. In section 2 we describe the task. In section 3 we describe our algorithm. Experiments are described in section 4. We also give a detailed analysis of the results in section 4. In section 5 related works are introduced, and the paper is concluded in the last section.

## 2 Task Description

The CoNLL-2014 shared task focuses on correcting all errors that are commonly made by L2 learners of English. The training data released by the task organizers come from the NUCLE corpus(Dahlmeier et al., 2013). This corpus contains essays written by L2 learners of English. These essays are then corrected by English teachers. Details of the CoNLL-2014 shared task can be found in Ng et al. (2014).

## 3 System Overview

### 3.1 Overview

It is time-consuming to train individual models for each kind of errors. We believe a better way is to correct errors in a unified framework. We assume that each word in the sentence may be involved in some kinds of errors. We generate a list of correction candidates for each word. Then a Language Model (LM) is used to find the most probable word sequences based on the original sentence and the correction candidates for each word. An illustrative example is shown in figure 1.

Because the LM is designed for the replacement errors rather than insertion and deletion errors, we train two extra classifiers for determiners and prepositions. The determiner model and the preposition model can improve the performance in our experiment.

### 3.2 Correction Candidate Generation

The correction candidate generation phase aims to generate a list of correction candidates for each word in the original sentence. We generate correction candidates based on the following rules:

1. Words with the same stem
2. Similar words based on edit distance

The first rule includes the words with the same stem as candidates. These candidates can be used later to correct the errors related to word form. For example, candidates for the word ‘time’ in the original sentence ‘This is a timely rain indeed.’ may include ‘timed’,‘time’,‘timed’,‘times’,‘timings’,‘timely’, ‘timees’ and ‘timing’, which all have the stem ’time’. The correct candidate ‘timely’ is also included in the candidate list and can be detected through further processing.

The candidate generated by the second rule are mainly used for spelling correction. For example, a such candidate for ‘beleive’ may be ‘belive’ or ‘believe’. To generate meaningful candidates while guarantee accuracy, we require that the candidate and the original word should have the same initial character. By examining the training data we experimentally find that very few L2 learners make spelling errors on the initial characters. For example, they may spell “believe” as “belive”. However, very few of them may spell “believe” as “pelieve” or “delieve”.

In our system, we generate 10 candidates for each word. To keep the decoding of the best word sequence controllable, we do not generate candidates for every word in the original sentence. We only generate the edit distance based candidates for the following words:

1. Words that never appear in the English gigaword corpus[1]
2. Words that appear in the gigaword corpus but with frequency below a threshold (we use 10 in the experiment)

Besides, we do not generate candidates for the words whose POS tags are “NNP” or “NNPS”.

[1] http://catalog.ldc.upenn.edu/LDC2003T05

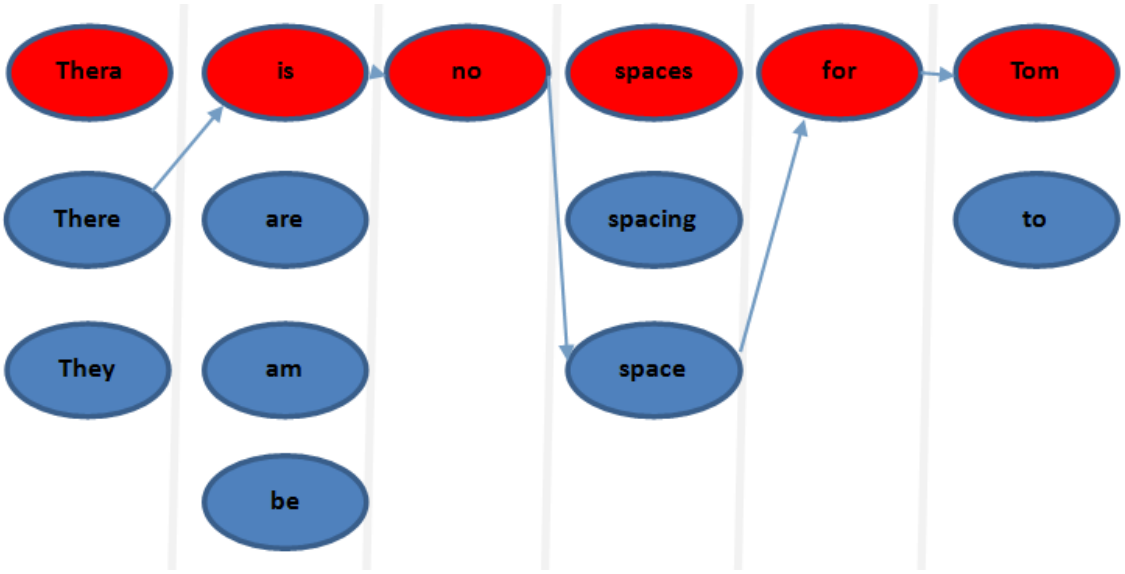


Figure 1: Correction of the original sentence "Thera is no spaces for Tom". We use red nodes to represent the original words in the sentence, and use blue nodes below each word to represent the candidate list of each word. We use arrows to show the final corrected word sequence with the highest probability.

These words are proper nouns. The correction of this kind of words should depend on more contextual information. For the stemming tools we use the snowball stemmer[2]. To generate candidates based on edit distance, we use the org.apache.lucene.search.spell.SpellChecker in Lucene[3]. Note that unlike other context based spell checkers such as the one in Microsoft Office, the SpellChecker class in Lucene is actually not a spell checker. For an input word $w$, it can only suggest words that are similar to $w$ given a predefined dictionary. We build the dictionary using all words collected from the English Gigaword corpus.

### 3.3 Language Model for Candidate Selection

After given each word a list of candidates, we can now find the word sequence which is most likely to be the correct sentence. The model we use is the language model. The probability $P(s)$ of a sentence $s = w_0 w_1 ... w_{n-1}$ is calculated as:

$$P(s) = \prod_{i=0}^{n-1} P(w_i | w_0, ..., w_{i-1}) \quad (1)$$

The transition probability $P(w_i|w_0, ..., w_{i-1})$ is calculated based on language model. In our system we use a tri-gram language model trained on the gigaword corpus. Therefore, $P(w_i|w_0, ..., w_{i-1})$ is reduced to $P(w_i|w_{i-2}, w_{i-1})$. We do not use a fixed smoothing method. We just set the probability of an unseen string to be a positive decimal which is very close to zero.

The decoding of the word sequence that maximize $p(s)$ can be tackled through dynamic programming using Viterbi algorithm(Forney Jr, 1973). One useful trick is that to multiply $p(w_i|w_{i-2}, w_{i-1})$ with a coefficient (4 in our system) if $w_{i-2}$, $w_{i-1}$ and $w_i$ are all words in the original sentence. This is because most of the original word sequences are correct. If the system needs to make a correction, the corrected sequence should have a much higher score than the original one.

We do not generate candidates for determiners and prepositions. Firstly, they are all frequent words that are excluded by the rules we mentioned in this section. Secondly, the determiner and preposition errors are the main kinds of errors made by L2 learners. Some of the errors are related to the wrong deletions or insertions. Therefore we choose to take special care of determiners and prepositions to correct all their replacement, deletion and insertion errors instead of generating candidates for them in this stage.

### 3.4 Determiner Correction

After using LM, the spelling errors as well as ordinary word form errors such as noun numbers, verb

[2] http://snowball.tartarus.org/
[3] https://lucene.apache.org/

forms are supposed to be corrected. As we mentioned in the introduction, we should now handle the deletion and insertion errors. We choose to use special models for determiner and prepositions because many of the deletion and insertion errors are related to determiner errors or preposition errors. Also, these two kinds of errors have been considered in HOO-2012 and CoNLL2013. Therefore it's easier to make meaningful comparison with previous works. We use Maximum Entropy (ME) classifiers to correct the determiner and preposition errors. In this section we consider the determiner errors. The preposition errors will be considered in the next section. For both of the two parts, we use the open source tool MaxEnt[4] as the implementation of ME.

We consider the determiner correction task as a multi-class classification task. The input instances for classification are the space between words. We consider whether the space should keep empty, or insert 'a' or 'the'. Therefore, 3 labels are considered to indicate 'a', 'the' and 'NULL'. We use ''NULL' to denote that the correct space does not need an article. We leave the clarification between 'a' and 'an' as a post-process by manually designed rules. We do not consider other determiners such as 'this' or ''these' because further information such as the coreference resolution results is needed.

Instead of considering all spaces in a sentence, some previous works(AEHAN et al., 2006; Rozovskaya and Roth, 2010; Rozovskaya et al., 2013) only consider spaces at the beginning of noun phrases. Compared to these methods, our system do not need a POS tagger or a phrase chunker (which is sometimes not accurate enough) to filter the positions. All the operations are done on the word level. We list the features we use in table 1. Note that for 3-grams and 4-grams we do not use all combinations of characters because it will generate more sparse features while the performance is not improved.

Because there are limited amount of training data, we choose to use the English Gigaword corpus to generate training instances instead of using the training data of CoNLL-2014. Because the texts in the Gigaword corpus are all news texts, most of them are well written by native speakers and are proofread by the editors. Therefore they can serve as implicit gold annotations. We generate the training instances from the sentences in the Gigaword corpus with the following rules:

| | |
|---|---|
| 1-gram | $w_{-3}$, $w_{-2}$, $w_{-1}$, $w_1$, $w_2$, $w_3$ |
| 2-gram | all combinations of $w_i w_j$ where $i, j \in \{-3, -2, -1, 1, 2, 3\}$ |
| 3-gram | $w_{-3}w_{-2}w_{-1}$,$w_{-2}w_{-1}w_1$, $w_{-1}w_1w_2$, $w_1w_2w_3$ |
| 4-gram | $w_{-3}w_{-2}w_{-1}w_1$, $w_{-2}w_{-1}w_1w_2$,$w_{-1}w_1w_2w_3$ |

Table 1: The features used in our system. For a given blank(space), $w_i$ means the next $i$th word and $w_{-i}$ means the previous $i$th word. For the example of "I do not play balls .", if the current considered instance is the space between 'play' and 'balls', then $w_{-2}$ means 'not' and $w_1$ means 'balls'.

1. for each space between words, we treat it as an instance with label 'NULL', which means no article is needed. We use the 3 words before the space as $w_{-3}$, $w_{-2}$, $w_{-1}$ and the 3 words after the space as $w_1$, $w_2$, $w_3$ to generate features. We name this kind of instances 'Space Instance' to indicate we operate on a space. This kind of training instances can convey the information that in this context no article is needed.

2. for each word that is an article, we assume it as an instance, with the label 'a' or 'the' depending on itself. We use the 3 words before it as $w_{-3}$, $w_{-2}$, $w_{-1}$ and the 3 words after is as $w_1$, $w_2$, $w_3$. In this case we do not use the article itself as the context. We name this kind of instances 'article Instance' to indicate we operate on an article. This kind of training instances can convey the information that in this context a particular article should be added.

The testing instance are also generated following the previously mentioned rules. The decoding process is as follows. If an instance is a 'space instance' and is predicted as 'a' or 'the', we then add 'a' or 'the' in this space. If an instance is an 'article instance', the situation is a bit complex. If it is predicted as another article, we replace it with the predicted one. If it is predicted as 'NULL', we should delete the article to make it a space.

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

To guarantee a certain level of precision, we require the decoding should only be based on confident predictions. We use the probability calculated by the classifier as the confidence score and require the probability of the considered predictions should exceed a threshold.

### 3.5 Preposition Correction

The preposition model is similar to the article model. We use the same set of features as in table 1. The training and testing instance generation is similar except now we consider prepositions instead of articles. The decoding phase is also identical to the determiner model.

### 3.6 Post Processing

The post processing in our system is listed as follows:

1. Distinguish between "a" and "an". We use rule based method for this issue.

2. Splitting words. If a word is not in the dictionary but one of its splitting results has a high frequency, we will split the word into two words. For example, "dailylife" is an out of vocabulary word and the splitting result "daily life" is common in English. Then we split "dailylife" into "daily life".

3. We capitalize the first character of each sentence.

## 4 Experiment and Analysis

We experiment on the CoNLL-2014 test data. We evaluate our system based on the M2 scorer which is provided by the organizers. Details of the M2 scorer can be found in Dahlmeier and Ng (2012b). We tune the additional parameters like all the thresholds on the CoNLL-2014 official training data. We use all the text in the Gigaword corpus to train the language model. We use 2.5 million sentences in the Gigaword corpus to train the extra two classifier.

Results of our system are shown in table 2. LM refers to using language model alone. LM+det refers to using a determiner classifier after using a language model. LM+prep refers to using a preposition classifier after using a language model. LM+det+preposition refers to using a preposition classifier after LM+det, which is the method used in our final system.

| Model | P | R | F0.5 |
|---|---|---|---|
| LM | 29.89% | 10.04% | 21.42% |
| LM+det | 32.23% | 13.64% | 25.33% |
| LM+prep | 29.73% | 10.04% | 21.35% |
| LM+det+prep(all) | 32.21% | 13.65% | 25.32% |

Table 2: The experimental results of our system in the CoNLL-2014 shared task. The threshold for determiner model and preposition model is 0.99 and 0.99. Parameters are tuned on the CoNLL-2014 training data.

| Model | P | R | F0.5 |
|---|---|---|---|
| LM+det+prep(all) | 36.64% | 15.96% | 29.10% |

Table 3: The experimental results of our system in the CoNLL-2014 shared task on the revised annotations. The threshold for determiner model and preposition model is 0.99 and 0.99. Parameters are tuned on the CoNLL-2014 training data.

From the results we can see that the main contribution comes from the LM model and determiner model. The preposition model can correct part of the errors while introduce new errors. The preposition model may harm the overall performance. But considering the fact that the grammar error correction systems are always used for recommending errors, we still keep the preposition model in real applications and suggest the errors predicted by the preposition model.

One limitation of our system is that we only use a tri-gram based language model as well as up to 4-gram features for limited instances. Previous works(Rozovskaya et al., 2013; Kao et al., 2013) have shown that other resources like the Google 5-gram statistics can help improve performance. For the determiner and preposition models, we experiment on different size of training data, from near zero to the upper bound of our server's memory limit (about 72GB). We find that under this limitation, the performance is still improving when adding more training instances. We believe the performance can be further improved.

Scores based on the revised annotations is shown in table 3.

For the convenience of future meaningful comparison, we report the result of our system on the CoNLL-2013 data set in table 4. We tune the additional parameters like all the thresholds on the CoNLL-2013 official training data. Note that in CoNLL-2013 the scorer considers F1 score in-

| Model | P | R | F1 |
|---|---|---|---|
| CoNLL13 1st | 23.49% | 46.45% | 31.20 % |
| CoNLL13 2nd | 26.35% | 23.80% | 25.01 % |
| LM | 18.92% | 14.55% | 16.45% |
| LM+det | 23.76% | 36.15% | 28.67% |
| LM+prep | 18.89% | 14.55% | 16.44% |
| **LM+det+prep** | **23.74**% | **36.15**% | **28.66**% |

Table 4: The experimental results of our system on the CoNLL-2013 shared task data. The threshold for determiner model and preposition model is 0.75 and 0.99. Parameters are tuned on the CoNLL-2013 training data. CoNLL13 1st is Rozovskaya et al. (2013) and the 2nd is Kao et al. (2013)

stead of F0.5. Therefore some of the thresholds are different with the ones in the CoNLL-2014 system. Because the CoNLL-2013 shared task only considers 5 types of errors, it will be much easier to design components specially for each kind of errors. Therefore our system is a bit less accurate than the best system. In this system, we restrict the candidates to be either noun or verb, and omit the spell checking model. We also omit some post-processings like deciding whether a word should be split into two words, because these kinds of errors are not included.

## 5 Conclusion

In this paper we describe the PKU system for the CoNLL-2014 grammar error correction shared task. We propose a unified framework for correcting all types of errors. A tri-gram language model is used to correct the replacement errors while two extra classification models are trained to correct errors related to determiners and prepositions. Our system achieves 25.32% in $f_{0.5}$ on the original test data and 29.10% on the revised test data.

## Acknowledgments

This research was partly supported by National Natural Science Foundation of China (No.61370117, No.61333018), National High Technology Research and Development Program of China (863 Program) (No.2012AA011101) and Major National Social Science Fund of China(No.12&ZD227).

## References

AEHAN, N., Chodorow, M., and LEACOCK, C. L. (2006). Detecting errors in english article usage by non-native speakers.

Dahlmeier, D. and Ng, H. T. (2012a). A beam-search decoder for grammatical error correction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 568–578. Association for Computational Linguistics.

Dahlmeier, D. and Ng, H. T. (2012b). Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572. Association for Computational Linguistics.

Dahlmeier, D., Ng, H. T., and Wu, S. M. (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*, pages 22–31.

Forney Jr, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Kao, T.-h., Chang, Y.-w., Chiu, H.-w., Yen, T.-H., Boisson, J., Wu, J.-c., and Chang, J. S. (2013). Conll-2013 shared task: Grammatical error correction nthu system description. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 20–25, Sofia, Bulgaria. Association for Computational Linguistics.

Ng, H. T., Wu, S. M., Briscoe, T., Hadiwinoto, C., Susanto, R. H., and Bryant, C. (2014). The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2014 Shared Task)*, pages 1–12, Baltimore, Maryland, USA. Association for Computational Linguistics.

Ng, H. T., Wu, S. M., Wu, Y., Hadiwinoto, C., and Tetreault, J. (2013). The conll-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared*

*Task*, pages 1–12, Sofia, Bulgaria. Association for Computational Linguistics.

Rozovskaya, A., Chang, K.-W., Sammons, M., and Roth, D. (2013). The university of illinois system in the conll-2013 shared task. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 13–19, Sofia, Bulgaria. Association for Computational Linguistics.

Rozovskaya, A. and Roth, D. (2010). Training paradigms for correcting errors in grammar and usage. In *Human language technologies: The 2010 annual conference of the north american chapter of the association for computational linguistics*, pages 154–162. Association for Computational Linguistics.

Rozovskaya, A. and Roth, D. (2013). Joint learning and inference for grammatical error correction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 791–802, Seattle, Washington, USA. Association for Computational Linguistics.

Wu, Y. and Ng, H. T. (2013). Grammatical error correction using integer linear programming. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1456–1465, Sofia, Bulgaria. Association for Computational Linguistics.

Yoshimoto, I., Kose, T., Mitsuzawa, K., Sakaguchi, K., Mizumoto, T., Hayashibe, Y., Komachi, M., and Matsumoto, Y. (2013). Naist at 2013 conll grammatical error correction shared task. *CoNLL-2013*, 26.

Yuan, Z. and Felice, M. (2013). Constrained grammatical error correction using statistical machine translation. *CoNLL-2013*, page 52.