NTHU at the CoNLL-2014 Shared Task

Jian-Cheng Wu*, Tzu-Hsi Yen*, Jim Chang*, Guan-Cheng Huang*, Jimmy Chang*, Hsiang-Ling Hsu+, Yu-Wei Chang+, Jason S. Chang*

* Department of Computer Science
+ Institute of Information Systems and Applications

National Tsing Hua University
HsinChu, Taiwan, R.O.C. 30013
{wujc86, joseph.yen, cwebb.tw, a2961353, rocmewtwo, ilovecat6717, teer1990, jason.jschang}@gmail.com

Abstract

In this paper, we describe a system for correcting grammatical errors in texts written by non-native learners. In our approach, a given sentence with syntactic features are sent to a number of modules, each focuses on a specific error type. A main program integrates corrections from these modules and outputs the corrected sentence. We evaluated our system on the official test data of the CoNLL-2014 shared task and obtained 0.30 in F-measure.

1 Introduction

Millions of non-native learners are using English as their second language (ESL) or foreign language (EFL). These learners often make different kinds of grammatical errors and are not aware of it. With a grammatical error corrector applies rules or statistical learning methods, learners can use the system to improve the quality of writing, and become more aware of the common errors. It may also help learners improve their writing skills.

The CoNLL-2014 shared task is aimed at promoting research on correcting grammatical errors. Types of errors handled in the shared task are extended from the five types in the previous shared task to include all common errors present in an essay.

In this paper, we focus on the following errors made by ESL writers:

- Spelling and comma
- Article and determiner
- Preposition
- Preposition + verb (interactive)
- Noun number
- Word form
- Subject-verb-agreement

For each error type, we developed and tuned a module based on the official development data. A main program combines the correction hypotheses from these modules and produces the final correction. If multiple modules propose different corrections to the same word/phrase, the correction proposed by the module with the highest precision will be chosen.

2 Method

2.1 Spelling and Comma module

In this section, we correct comma errors and spelling errors, including missing/extraneous hyphens. For simplicity, we adopt Aspell[1] and GingerIt[2] to detect spelling errors and generate possible replacements, considered as confusable words, which might contain the word with correct spelling. Then, we replace the word being checked with confusable words to generate sentences. Language models trained on well-formed texts are used to measure the probability of these

[1] `http://aspell.net/`

[2] `https://pypi.python.org/pypi/gingerit`
We use GingerIt only for correcting missing/extraneous hyphens

candidates. Candidate with the highest probability is chosen as correction.

Omitted commas form a large proportion of punctuation errors. We apply the CRF model proposed by Israel, et. al. (2012) with some modification. We replace distance features with syntactic features. More specifically, we do not use features such as distances to the start of sentence or last comma. And we add two features, one indicates whether a word is at the end of a clause, and the other indicates whether the current clause starts with a prepositional phrase.

2.2 Subject-verb-agreement module

This module corrects subject-verb-agreement errors in a given sentence. Consider the sentence "*The boy in blue pants are my brother*". The correct sentence should be "*The boy in blue pants is my brother*". Since a verb could be far from it's subject, using ngram counts may fail to detect and correct such an error.

We use a rule-based method in this module. In the first stage, we identify the subject of each clause by using information from the parser. Both the dependency relation and syntactic structure are used in this stage. Dependency relations such as `nsubj` and `rcmod` indicate subjects of subject-verb relation. If there is a verb that has not been assigned a subject via dependency relations, head of noun phrase in the same clause will be used instead. And in the second stage, we check whether subject and verbs agree, for each clause in the sentence.

Here we explain our correction process in more detail. For each clause, the singular and plural forms of verbs in the clause must be consistent with the subject of the clause unless the subject is a quantifier. Consider the following sentences:

The number of cats *is* ten.
A number of cats *are* playing.

Since our judgment only depends on the subject *number*, it's hard to tell whether should we use a plural verb or not in this case. The quantifiers we do not handle are listed as follow: *number, lot, quantity, variety, type, amount, neither*.

2.3 Number module and Forms module

We correct noun number error in two stages. In the first stage, we generate a confusion set for each word. While constructing confusion set for noun number, both of the singular form and plural form are included in the set. While constructing confusion set for word forms, we use the word families in Academic Word List (AWL)[3] and British National Corpus (BNC4000) [4]. Given a content word, all the words in the same family except antonyms are entered into the confusion set. However, comparative form and superlative form of an adjective are eliminated from the confusion set, since replacing an adjective with these forms is a semantic rather than syntactic decision. The following examples illustrate what kinds of alternatives are eliminated:

antonyms: *misleading* for the word *lead*
semantics: *higher* for the word *high*

Additionally, in the forms module, a correction is ignored, if it is actually correcting a verb tense, subject-verb-agreement, or noun number error. We use part-of-speech (POS) tag to check this. More specifically, any corrections that changes a word with a *VBZ* tag to a word with a *VBP* or *VBD* tag is ignored, and vice versa. And any corrections that switches a noun between it's singular form and plural form is also ignored.

With the confusion sets, we proceed to the second stage. In this stage, we use words in the confusion set to attempt to replace potential errors. Language models trained on well-formed text are used to validate the replacement decisions. Given a word *w*, If there is an alternative *w'* that fits in the context better, *w* is flagged as an error and *w'* is returned as a correction.
Here is our formula for correcting errors

$$P(O) = \frac{P_{ngram}(O) + P_{rnn}(O)}{2}$$

$$P(R) = \frac{P_{ngram}(R) + P_{rnn}(R)}{2}$$

$$Promotion = \frac{P(R) - P(O)}{|O|}$$

While checking a content word *w*, we replace *w* in the original sentence *O* with alternatives and generate candidates *C*. We then generate the candidate *R* with the highest probability among all

[3] `http://www.victoria.ac.nz/lals/resources/academicwordlist/sublists`
[4] `http://simple.wiktionary.org/wiki/Wiktionary:BNC_spoken_freq`

candidates. We use an interpolated probability[5] of ngram language model P_{ngram} and recurrent neural network language model P_{rnn}. *Promotion* indicates the increase in probability per word after we replace sentence *O* with the candidate *R*. We use word number to normalize *Promotion* following Dahlmeier, et al. (2012). Corrections are made only if *Promotion* is higher than a empirically determined threshold.[6]

2.4 Article and Determiner module

In this subsection, we describe how we correct errors of omitting a determiner or adding a spurious determiner. The language models mentioned in the last subsection are also used in this module. We tune our thresholds for making corrections on development data[7], and found that deleting a determiner should have a lower threshold while inserting one should have a higher one, so we set different thresholds accordingly. [8]

To cope with the situation where a determiner/article is far ahead of the head of the noun phrase, we apply another constraint while making correction decision.

First, we calculate statistics on the head of noun phrases. We extract the most frequent 100,000 terms in Web-1T 5-gram corpus. These terms are used to search their definitions in Wikipedia (usually at the first paragraph). The characteristic of a definition is that it has no prior context and most of the noun phrases with a determiner are unique or known to the general public. Heads of these nouns phrases are likely to always appear with a determiner.

Heads that tend to appear with a determiner `the` help us to decide whether a determiner should be added to a noun phrase. We add a determiner using two constraints. We only insert a determiner or an article, if the statistics indicate that head of a noun phrase tends to have a determiner, or the promotion of log probability is much higher than the threshold. A similar constraint is also applied, for deleting a determiner or an article.

[5]the probabilities present in the formula are log probabilities

[6]the threshold for noun number module is 0.035 and 0.050 for word form module. These threshold were set empirically after testing on development data

[7]test data of the CoNLL-2013 shared task

[8]the threshold for insertion is 0.035 and 0.040 for deletion

2.5 Preposition module

For preposition errors, we focus on handling two types of errors: REPLACE and DELETE. A preposition, which should be deleted from the given sentence, is regarded as a DELETE error, whereas for a preposition, which should be replaced with a more appropriate alternative, is regarded as a REPLACE error. In this work, we correct the two types of errors based on the assumption that the usage of preposition often depends on the collocation relations with a verb or noun. Therefore, we use the dependency relations such as `dobj` and `pobj`, and `prep` to identify the related words, and then we validate the usage of prepositions, and correct the preposition errors.

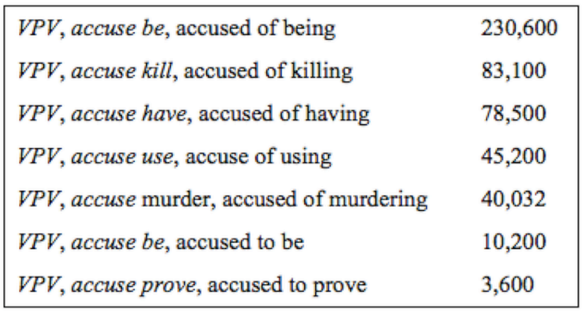
A dependency-based model is proposed in this paper to handle the preposition errors. The model consists of two stages: detecting the possible preposition errors and correcting the errors.

In the first stage, we use the Stanford dependency parser (Klein and Manning, 2003) to extract the dependency relations for each preposition. The relation tuples, which contain the preposition, verb or noun, and prepositional object. For example, the tuple of verb-prep-object (listen, to, music), or the tuple of noun-prep-object (point, of, view) are extracted for validation. We identify a preposition containing as an error, if the tuple containing the preposition does not occur in a reference list built using a reference corpus. In order to resolve the data sparseness and false alarm problems, we need a sufficiently large list of validated tuples. In this study, the reference tuple lists are generated from the Google Books Syntactic N-grams (Goldberg and Orwant, 2013)[9]. For example, we can find (come, to, end, 236864) and (lead, to, result, 57632) in the verb-preposition-object reference list. We have generated 21,773,752 different dependency tuples for our purpose.

In the second stage, we attempt to correct all potential preposition errors. At first, a list of candidate tuples is generated by substituting the original preposition in the error tuple with alternative prepositions. For example, the generated candidate tuples for the error tuple (join, at, party) will include (join, in, party), (join, on, party), etc. On the other hand, the tuple, (join, party), which

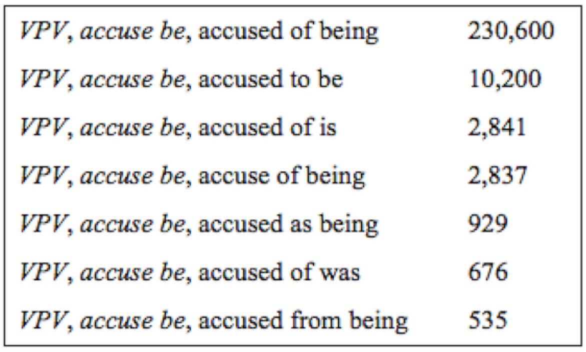
[9]Data sets available from `http://commondatastorage.googleapis.com/books/syntactic-ngrams/index.html`

deletes the preposition, is also taken into consideration. All candidates are ranked according to the frequency provided by the reference lists. The preposition in the tuple with the highest frequency is returned as the correction suggestion.



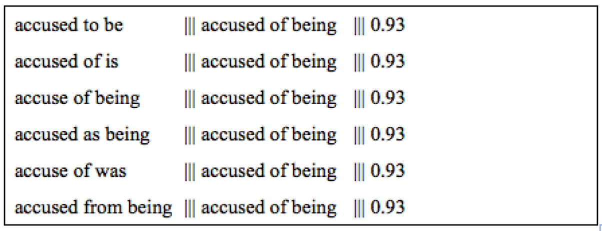
VPV, *accuse be*, accused of being	230,600
VPV, *accuse kill*, accused of killing	83,100
VPV, *accuse have*, accused of having	78,500
VPV, *accuse use*, accuse of using	45,200
VPV, *accuse* murder, accused of murdering	40,032
VPV, *accuse be*, accused to be	10,200
VPV, *accuse prove*, accused to prove	3,600

Figure 1: Sample annotated trigrams



VPV, *accuse be*, accused of being	230,600
VPV, *accuse be*, accused to be	10,200
VPV, *accuse be*, accused of is	2,841
VPV, *accuse be*, accuse of being	2,837
VPV, *accuse be*, accused as being	929
VPV, *accuse be*, accused of was	676
VPV, *accuse be*, accused from being	535

Figure 2: Sample trigram group



accused to be	||| accused of being	||| 0.93
accused of is	||| accused of being	||| 0.93
accuse of being	||| accused of being	||| 0.93
accused as being	||| accused of being	||| 0.93
accuse of was	||| accused of being	||| 0.93
accused from being	||| accused of being	||| 0.93

Figure 3: Sample phrase translation for a trigram group

2.6 Interactive errors module

This module uses a new method for correcting serial grammatical errors in a given sentence in learners writing. A statistical translation model is generated to attempt to translate the input with serial and interactive errors into a grammatical sentence. The method involves automatically learning translation models based on Web-scale n-gram. The model corrects trigrams containing serial preposition-verb errors via translation. Evaluation on a set of sentences in a learner corpus shows that the method corrects serial errors reasonably well.

Consider an error sentence *"I have difficulty to understand English."* The correct sentence for this should be *"I have difficulty in understanding English."* It is hard to correct these two errors one by one, since the errors are dependent on each other. Intuitively, by identifying *difficulty to understand* as containing serial errors and correct it to *difficulty in understanding*, we can handle this kind of problem more effectively.

First, we generate translation phrase table as follows. We begin by selecting trigrams related to serial errors and correction from Web 1T 5-gram. Figure 1 shows some sample annotation trigrams. Then, we group the selected trigrams by the first and last word in the trigrams. See Figure 2 for a sample VPV group of trigrams. Finally, we generate translation phrase table for each group. Figure 3 shows a sample translation phrase table.

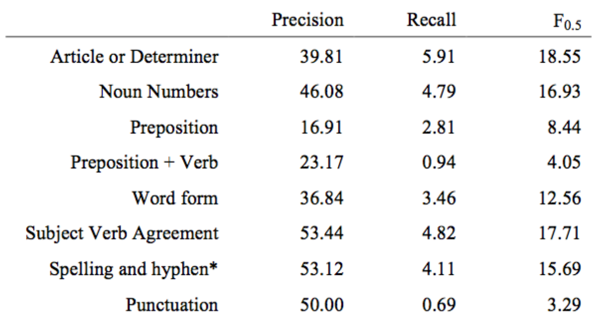
At run time, we tag the input sentence with part of speech information in order to find trigrams that fit the type of serial errors. Then, we search phrase table and generate translations for the input phrases in a machine translation decoder to produce a corrected sentence.

3 Experiment

Two types of trigram language models, ngram model and recurrent neural network (RNN) model, are used in correcting spelling, noun number, word form, and determiner errors. We trained the ngram language model on English Gigaword and BNC corpus, using the SRILM tool (Stolcke, 2002). We train the RNN model with RNNLM toolkit (Mikolov et al., 2011). Complexity of training the RNN language model is much higher, so we train it on a smaller corpus, the British National Corpus (BNC).

We used the Stanford Parser (Klein and Christopher D. Manning, 2003) to obtain dependency relations in the preposition module, and to obtain POS tags for the word form module. The subject-verb-agreement module also uses dependency relations contained in test data. Dependency relations in Google Books Syntactic N-grams (Goldberg and Orwant, 2013) were also used to develop our dendepency-based model in the preposition module.

To assess the effectiveness of the proposed method, we used the official training, development, and test data of the CoNLL-2014 shared task. On the test data, our system obtained the precision, recall and $F_{0.5}$ score of 0.351, 0.189, and 0.299. The following table shows the performance breakdown by module.



	Precision	Recall	$F_{0.5}$
Article or Determiner	39.81	5.91	18.55
Noun Numbers	46.08	4.79	16.93
Preposition	16.91	2.81	8.44
Preposition + Verb	23.17	0.94	4.05
Word form	36.84	3.46	12.56
Subject Verb Agreement	53.44	4.82	17.71
Spelling and hyphen*	53.12	4.11	15.69
Punctuation	50.00	0.69	3.29

Figure 4: The performance breakdown by module. (Displayed in %)

In the spelling and hyphen module, candidates from *Aspell* include words that only differ from the original word in one character, s. Language models are then used to choose the candidate with highest probability as our correction. The module therefore gives some corrections about noun numbers or subject-verb-agreement. As a result, some corrections made by this module overlap with corrections made by the noun numbers module and the subject-verb-agreement module, which makes the recall of correcting spelling and hyphen errors, 4.11%, overestimated.

4 Conclusion

In this work, we have built several modules for error detection and correction. For different types of errors, we developed modules independently using different features and thresholds. If multiple modules propose different corrections to a word/phrase, the one proposed by the module with higher precision will be chosen. Many avenues for future work present themselves. We plan to integrate modules in a more flexible way. When faced with different corrections made by different modules, the decision would better be based on the confidence of each correction with a uniform standard, but not on the confidence of modules.

Acknowledgments

We would like to acknowledge the funding supports from Delta Electronic Corp and National Science Council (contract no: NSC 100-2511-S-007-005-MY3), Taiwan. We are also thankful to anonymous reviewers and the organizers of the shared task.

References

Daniel Dahlmeier, Hwee Tou Ng, and Eric Jun Feng Ng 2012. NUS at the HOO 2012 Shared Task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, Association for Computational Linguistics, June 7.

Daniel Dahlmeier and Hwee Tou Ng, 2012. Better Evaluation for Grammatical Error Correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2012).*,568-672

Daniel Dahlmeier, Hwee Tou Ng, Siew Mei Wu. 2013. Building a Large Annotated Corpus of Learner English: The NUS Corpus of Learner English. In *Proceedings of the 8th Workshop on Innovative Use of NLP for Building Educational Applications(BEA 2013)*

Yoav Goldberg and Jon Orwant 2013. A dataset of syntactic-ngrams over time from a very large corpus of English books. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, Atlanta, GA, 2013.

Ross Israel, Joel Tetreault, and Martin Chodorow 2012. Correcting Comma Errors in Learner Essays, and Restoring Commas in Newswire Text. In *Proceeding of the 2012 Conference of the North America Chapter of the Association for Computational Linguistics: Human Language Technologies*,284-294, Montreal Canada, June. Association for Computational Linguistics

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, 423-430.

Tomas Mikolov, Anoop Deoras, Dan Povey, Lukar Burget, and Jan Honza Cernocky 2011. Strategies for Training Large Scale Neural Network Language Models *Proceedings of ASRU 2011*

Andreas Stolcke 2002. SRILM-An Extensible Language Modeling Toolkit In *Proceedings of the International Conference on Spoken Language Processing*, vol 2, 901-904