Computational Linguistics and Chinese Language Processing
Vol. 18, No. 4, December 2013, pp. 31-44

# Correcting Serial Grammatical Errors based on N-grams and Syntax

**Jian-cheng Wu[*], Jim Chang[*], and Jason S. Chang[*]**

**Abstract**

In this paper, we present a new method based on machine translation for correcting serial grammatical errors in a given sentence in learners' writing. In our approach, translation models are generated to translate the input into a grammatical sentence. The method involves automatically learning two translation models that are based on Web-scale n-grams. The first model translates trigrams containing serial preposition-verb errors into correct ones. The second model is a back-off model, used in the case where the trigram is not found in the training data. At run-time, the phrases in the input are matched and translated, and ranking is performed on all possible translations to produce a corrected sentence as output. Evaluation on a set of sentences in a learner corpus shows that the method corrects serial errors reasonably well. Our methodology exploits the state-of-the art in machine translation, resulting in an effective system that can deal with many error types at the same time.

**Keywords:** Grammatical Error Correction, Serial Errors, Machine Translation, N-grams, Language Model

## 1. Introduction

Many people are learning English as a second or foreign language: it is estimated there are 375 million English as a Second Language (ESL) and 750 million English as a Foreign Language (EFL) learners around the world, according to Graddol (2006). Three times as many people speak English as a second language as there are native speakers of English. Nevertheless, non-native speakers tend to make many kinds of errors in their writing, due to the influence of their native languages (*e.g.*, Chinese or Japanese). Therefore, automatic grammar checkers are needed to help learners improve their writing. In the long run, automatic grammar checkers also can help non-native writers learn from the corrections and

[*] Department of Computer Science, National Tsing Hua University
E-mail: {wujc86; jim.chang.nthu; jason.jschang}@gmail.com

gradually gain better command of grammar and word choices.

The grammar checkers available in popular word processors have been developed with a focus on native speaker errors, such as subject-verb agreement and pronoun reference. Therefore, these word processors (*e.g.*, Microsoft Word) often offer little or no help with common errors causing problems for English learners (*e.g.*, missing, unnecessary, or wrong article, preposition, and verb form) as described in The Longman Dictionary of Common Errors, second edition (LDOCE) by Heaton and Turton (1996). The LDOCE is the result of analyzing errors encoded in the Longman Learners' Corpus.

The LDOCE shows that grammatical errors in learners' writing can either appear in isolation (*e.g.*, the wrong proposition in "*I want to improve my ability of* [*in*] *English*.") or consecutively (*e.g.*, the unnecessary preposition immediately followed by a wrong verb form in "*These machines are destroying our ability of thinking* [*to think*].''). We refer to two or more errors appearing consecutively as *serial errors*.

Previous works on grammar checkers either have focused on handling one common type of error exclusively or handling it independently in a sequence of errors. Nevertheless, when an error is not isolated, it is difficult to correct the error when another related error is in the immediate context. In other words, when serial errors occur in a sentence, a grammar checker needs to correct the first error in the presence of the second error (or *vice-versa*), making correction difficult to achieve. These errors could be corrected more effectively if the corrector recognized them as serial errors and attempted to correct the serial errors at once.

Consider an erroneous sentence, "*I have difficulty to understand English*." The correct sentence 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 correcting it to "*difficulty in understanding*," we can handle this kind of problem more effectively.

Input: *I have difficulty to understand English.* .

| Phrase table of translation model: | Back-off translation model: |
|---|---|
| difficulty of understanding \|\|\| difficulty in understanding \|\|\| 0.86 | difficulty of VERB+ing \|\|\| difficulty in VERB+ing \|\|\| 0.34 |
| difficulty to understand \|\|\| difficulty in understanding \|\|\| 0.86 | difficulty to VERB \|\|\| difficulty in VERB+ing \|\|\| 0.34 |
| difficulty with understanding \|\|\| difficulty in understanding \|\|\| 0.86 | difficulty with VERB+ing\|\|\| difficulty in VERB+ing \|\|\| 0.34 |
| difficulty in understand \|\|\| difficulty in understanding \|\|\| 0.86 | difficulty in VERB \|\|\| difficulty in VERB+ing \|\|\| 0.34 |
| difficulty for understanding \|\|\| difficulty in understanding \|\|\| 0.86 | difficulty for VERB+ing \|\|\| difficulty in VERB+ing \|\|\| 0.34 |
| difficulty about understand \|\|\| difficulty in understanding \|\|\| 0.86 | difficulty about VERB+ing \|\|\| difficulty in VERB+ing \|\|\| 0.34 |

Output: *I have difficulty in understanding English.*

***Figure 1. Example session of correcting the sentence, "I have difficulty to understand English."***

We present a new system that automatically generates a statistical machine translation model based on a trigram containing a word followed by preposition and verb or by an infinitive in web-scale n-gram data. At run-time, the system generates multiple possible trigrams by changing a word’s lexical form and preposition in the original trigram. Example trigrams generated for “*difficulty to understand*” are shown in Figure 1. The system then ranks all of these generated sentences and use the highest ranking sentence as suggestion.

The rest of the paper is organized as follows. We review the related work in the next section. Then, we describe our method for automatically learning to translate a sentence that may contain preposition-verb serial errors into a grammatical sentence (Section 3). In our evaluation, we describe how to measure the precision and recall of producing grammatical sentences (Section 4) in an automatic evaluation (Section 5) over a set of marked sentences in a learner corpus.

## 2. Related Work

Grammatical Error Detection (GED) for language learners has been an area of active research. GED involves pinpointing some words in a given sentence as ungrammatical and offering correction if necessary. Common errors in learners’ writing include misuse of articles, prepositions, noun number, and verb form. Recently, the state-of-the-art research on GED has been surveyed by Leacock *et al.* (2010). In our work, we address serial errors in English learners’ writing which are simultaneously related to the preposition and verb form, an aspect that has not been dealt with in most GED research. We also consider the issues of broadening the training data for better coverage and coping with data sparseness when unseen events happen.

Although there are over a billion people estimated to be using or learning English as a second or foreign language, common English proofreading tools do not target specifically the most common errors made by second language learners. Many widely-used grammar checking tools are based on pattern matching and at least some linguistic analysis, based on hand-coded grammar rules (Leacock *et al.*, 2010). In the 1990s, data-driven, statistical methods began to emerge. Statistical systems have the advantage of being more intolerant of ill-form, interlanguage, and unknown words produced by the learners than the rule-based systems.

Knight and Chander (1994) proposed a method based on a decision tree classifier to correct article errors in the output of machine translation systems. Articles were selected based on contextual similarity to the same noun phrase in the training data. Atwell (1987) used a language model of a language to represent correct usage for that language. He used the language model to detect errors that tend to have a low language model score.

More recently, researchers have looked at grammatical errors related to the most common prepositions (9 to 34 prepositions, depending on the percentage of coverage). Eeg-Olofsson and Knuttson (2003) described a rule-based system to detect preposition errors for learners of Swedish. Based on part-of-speech tags assigned by a statistical trigram tagger, 31 rules were written for very specific preposition errors. Tetreault and Chodorow (2008), Gamon *et al.* (2008), and Gamon (2010) developed statistical classifiers for preposition error detection. De Felice and Pulman (2007) trained a voted perceptron classifier on features of grammatical relations and WordNet categories in an automatic parse of a sentence. Han *et al.* (2010) found that a preposition error detection model trained on correct and incorrect usage in a learner corpus works better than using well-formed text in a reference corpus.

In the research area of detecting verb form errors, Heidorn (2000) and Bender *et al.* (2004) proposed methods based on parse tree and error templates. Lee and Seneff (2008) focused on three cases of verb form errors: subject-verb agreement, auxiliary agreement, and verb complement. The first two types are isolated verb form errors, while the third type may involve serial errors related to preposition and verb. Izumi *et al.* (2003) proposed a maximum entropy model, using lexical and POS features, to recognize a variety of errors, including verb form errors. Lee and Seneff (2008) used a database of irregular parsing caused by verb form misuse to detect and correct verb form errors. In addition, they also used the Google n-gram corpus to filter out improbable detections. Both Izumi *et al.* (2003) and Lee and Seneff (2008) obtained a high error correction rate, but they did not report serial errors separately, making comparison with our approach is impossible.

In a study more closely related to our work, Alla Rozovskaya and Dan Roth (2013) introduced a joint learning scheme to jointly resolve pairs of interacting errors related to subject-verb and article-noun agreements. They showed that the overall error correction rate is improved by learning a model that jointly learns each of these interacting errors.

## 3. Method

Correcting serial errors (*e.g.*, “*I have difficulty to understand English*.”) one error at a time in the traditional way may not work very well, but previous works typically have dealt with one type of error at a time. Unfortunately, it may be difficult to correct an error in the context of another error, because an error could only be corrected successfully within the correct context. Besides, such systems need to correct a sentence multiple times, which is time-consuming and more error-prone. To handle serial errors, a promising approach is to treat serial errors together as one single error.

## 3.1 Problem Statement

We focus on correcting serial errors in learners' writing using the context of trigrams in a sentence. We train a statistical machine translation model to correct learners' errors of the types of a content word followed by a preposition and a verb using web-scale n-grams.

*Problem Statement*: We are given a sentence $S = w_1, w_2, \ldots, w_n$, and web-scale n-gram, *webgram*. Our goal is to train two statistical machine translation model *TM* and back-off model $TM_{bo}$ to correct learners' writing. At run-time, trigrams ($w_i$ , $w_{i+1}$, $w_{i+2}$) in $S$ ($i$ =1, $n$-2) are matched and replaced using *TM* and the back-off model $TM_{bo}$ to translate $S$ into a correct sentence $T$.

In the rest of this section, we describe our solution to this problem. First, we describe the strategy to train *TM* (Section 3.2) and $TM_{bo}$ (Section 3.3) using *webgrams*. Finally, we show how our system corrects a sentence at run-time using *TM*, $TM_{bo}$, and a language model *LM* (Section 3.4).

## 3.2 Generating *TM*

We attempt to identify trigrams that fit the pattern of serial errors and correction we are dealing with in *webngram*, and we group the selected trigrams by their content words and verb lemmas. Our learning process is shown in Figure 2. We assume that, within each group, the low frequency trigrams are probably errors that should be replaced by the most frequent trigram: a *one construction per collocation* constraint. For example, when expressing "*difficulty*" and "*to understand*," any NPV constructs with low frequency (*e.g.*, "*difficulty for understanding*" and "*difficulty about understanding*") are erroneous forms of the most frequent trigram "*difficulty in understanding*". Therefore, we generate *TM* with such phrase to phrase translations accordingly.

(1) Select *trigrams* related to serial errors and corrections from *webngram* (Section 3.2.1)
(2) Group the selected trigrams by the first and last word in the *trigrams* (Section 3.2.2)
(3) Generate a phrase table for the statistical machine translation models for each group (Section 3.2.3)

***Figure 2. Outline of the process used to generate TM.***

### 3.2.1 Select and Annotate Trigrams

We select four types of trigrams ($t_1$, $t_2$, $t_3$) from *webngram*, including noun-prep-verb (NPV), verb-prep-verb (VPV), adj-prep-verb (APV), and adverb-prep-verb (RPV). We then annotate the trigrams with types and lemmas of content words $t_1$ and $t_3$ (*e.g.*, "*accused of being* 230633" becomes "*VPV*, *accuse be*, *accused of being* 230633). Figure 3 shows some sample annotated trigrams.

| | |
|---|---|
| *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 3. 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 4. 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 5. Sample phrase translations for a trigram group***

### 3.2.2 Group Trigrams

We then group the trigrams by types, the first words, and the verb lemmas. See Figure 4 for a sample VPV group of trigrams. This step should bring together the trigrams containing serial errors and their correction. Note that we assume certain serial errors will have a correction of the same length here, which is true in most cases.

### 3.2.3 Generate Rules

For each group of annotated trigrams, we then generate phrase and translation pairs with probability as follows. Recall that we assume that the higher the count of the trigram, the more likely the trigram is to be correct. So, we generate "$l_1$, $l_2$, $l_3$ ||| $h_1$, $h_2$, $h_3$ ||| $p$ ," where $h_1$, $h_2$, $h_3$ is the trigram with the highest frequency count; $l_1$, $l_2$, $l_3$ is one of the trigrams with lower frequency count; and $p$ denotes the probability of $l_1$, $l_2$, $l_3$ translating into $h_1$, $h_2$, $h_3$. We define $p$=(highest frequency count)/(group frequency count).

## 3.3 Generating $TM_{bo}$

In addition to the surface-level translation model *TM*, we also build a back-off model as a way of coping with cases where the trigram ($t_1$, $t_2$, $t_3$) is unseen in *TM*. The idea is to assume the complement ($t_2$, $t_3$) of $t_1$ tends to be in a certain syntactic form regardless of the verb $t_3$, as dictionaries typically would describe the usage of "*accuse*" in terms of "*accuse somebody of doing something*." Our learning process for $TM_{bo}$ is shown in Figure 9.

| | |
|---|---|
| *VPV*, *accuse VERB*, accused of *VERB*-ing | 230,600 |
| *VPV*, *accuse VERB*, accused of *VERB*-ing | 83,100 |
| *VPV*, *accuse VERB*, accused of *VERB*-ing | 78,500 |
| *VPV*, *accuse VERB*, accuse of *VERB*-ing | 45,200 |
| *VPV*, *accuse VERB*, accused of *VERB*-ing | 40,032 |
| *VPV*, *accuse VERB*, accused to *VERB* | 10,200 |
| *VPV*, *accuse VERB*, accused to *VERB* | 3,600 |

***Figure 6. Sample annotated trigrams***

| | |
|---|---|
| *VPV*, *accuse VERB*, accused of *VERB-ing* | 870,600 |
| *VPV*, *accuse VERB*, accused to *VERB* | 50,200 |
| *VPV*, *accuse VERB*, accuse to *VERB* | 20,200 |

***Figure 7. Sample trigram group***

accused to VERB ||| accused of VERB-ing ||| 0.47

accused of VERB ||| accused of VERB-ing ||| 0.47

***Figure 8. Sample back-off translations***

| | |
|---|---|
| (1) | Select trigrams with specific forms from Web 1T n-gram |
| (2) | Reform trigrams W3 to W3's lexical |
| (3) | Group the selected trigrams using the first word |
| (4) | Group the selected trigrams using the first word |

***Figure 9. Outline of the process used to generate $TM_{bo}$***

### 3.3.1 Generalize Trigrams

First, we generalize the annotated trigrams (see Section 3.2.1) by replacing the verb form with its part of speech designator (*i.e.*, replace “accuse” with VERB, and replace “accusing” with VERB-ing).

### 3.3.2 Sum Counts

In this step, we group the identically transformed trigrams and sum up the frequency counts. See Figure 6 for sample results.

### 3.3.3 Group Trigrams of the Same Context

We then group the trigrams by type and by the first word (context). See Figure 7 for a sample “accuse P V” group of trigrams.

### 3.3.4 Generate Rules

For each group of generalized trigrams, we then generate the phrase and translation pair with the probability as described in Section 3.2.3. See Figure 8 for a sample of back-off translations.

## 3.4 Run-time Correction

If one loads *TM* and $TM_{bo}$ into memory before the decoding process (generating, ranking, and selecting translations), that would take up a lot of memory and slow the process of matching phrases to find translations. Therefore, we generate phrase translations on the fly for the given sentence before decoding. Our process of decoding to correct grammatical errors is shown in Figure 10.

(1) Tag the input sentence with part of speech information in order to find trigrams that fit the type of serial errors

(2) Search *TM* and generate translations for the input phrases

(3) Search $TM_{bo}$ and generate translations for the input phrases

(4) Run statistical machine translation

***Figure 10. Outline of the process used to correct the sentence at run-time***

### 3.4.1 Tag the Input Ssentence

We use a POS tagger to tag the input sentence, and we identify trigrams ($t_1$, $t_2$, $t_3$) consisting of a content word followed by a preposition and verb (belonging to the NPV, VPV, APV, or RPV types we described in Section 3.2.1).

### 3.4.2 Search TM and Generate Translation Rules

We then search for the group of trigrams (indexed by POS type and $t_1$, $t_3$) in *TM* containing the trigrams ($t_1$, $t_2$, $t_3$), found in Step 3.4.1. We find the trigram ($h_1$, $h_2$, $h_3$) with the highest count in that group. With that, we can dynamically add the translation, "$t_1$, $t_2$, $t_3$ ||| $h_1$, $h_2$, $h_3$ ||| 1.0" to the cache of *TM* in memory (*e.g.*, "difficulty to understand ||| difficulty in understanding ||| 1.0") to speed up the subsequent decoding process.

### 3.4.3 Search $TM_{bo}$ and Generate Translation Rules

Just like in 3.4.2, we use $t_1$ and its part of speech $p_1$ to search $TM_{bo}$ for the generalized trigram group that matches ($t_1$, $t_2$, $t_3$). We then find the most frequent generalized trigram ($h_1$, $h_2$, $h_3$) in that group. After that, we need to specialize ($h_1$, $h_2$, $h_3$) for $t_3$ by replacing $h_3$ with the verb form of $t_3$ for the designator $h_3$, resulting in ($h_1$, $h_2$, $h'_3$). Consider the generalized trigram "accused of VERB-ing" and $t_3$ = "murder," the specialized trigram would be "accused of murdering." Finally, we add "$t_1$, $t_2$, $t_3$ ||| $h_1$, $h_2$, $h'_3$ ||| 1.0" (*e.g.*, "accused to murder ||| accused of murdering ||| 1.0") to the cache of *TM* in memory for the same purpose of speeding up decoding.

### 3.4.4 Decode the Input Sentence without Reordering

Finally, we run a monotone decoder with the cache *TM* and a language model *LM*. By default, any word not in *TM* will be translated into itself.

## 4. Experimental Setting

Our system *DeeD* (Don'ts-to-Do's English-English Decoder) was designed to correct preposition-verb serial errors in a given sentence written by language learners. Nevertheless, since large-scale learner corpora annotated with errors are not widely available, we have resorted to Web scale n-grams to train our system, while using a small annotated learner corpus to evaluate its performance. In this section, we first present the details of training *DeeD* for the evaluation (Section 4.1). Then, Section 4.2 lists the grammar checking systems that we used in our evaluation and comparison. Section 4.3 introduces the evaluation metrics for the performance of the systems, and details of the sentences evaluated and performance judgments are reported in Section 4.4.

## 4.1 Training DeeD

We used the Web 1T 5-grams (Brants & Franz, 2006) to train our systems. Web 1T 5-grams is a collection that contains 1 to 5 grams calculated from a 1 trillion words of public Web pages provided by Google through the Linguistic Data Consortium (LDC). There are some ten million unigrams, 3 hundred million bigrams, and around 1 billion trigrams to fivegrams. We obtained 104,537,560 trigrams, containing only words in the General Service List (West, 1954) and Academic Word List (Coxhead, 1999). These trigrams were further reduced to 4,486,615 entries that fit the patterns of four types of serial errors and corrections: an adjective, noun, verb, or adverb followed by a preposition (or infinitive *to*) and a verb.

To determine the part of speech of words in the n-gram, we used the most frequent tag of a given word in BNC to tag words in the trigram.

## 4.2 Grammar Checking Systems Compared

Once we have trained *DeeD* as described in Section 3, we evaluated its performance using two datasets. The first dataset contained sentences written by an ESL or EFL learner with the serial errors with corrections. The second dataset contained mostly correct sentences in British National Corpus (BNC) with mostly published works written by native, expert speakers.

The first testset is a subset of the Cambridge Learner Corpus, the CLC First Certificate Exam Dataset (CLC/FCE). This dataset contains 1,244 exam essays written by students who took the Cambridge ESOL First Certificate in English (FCE) examination in 2000 and 2001. For each exam script, the CLC/FCE Dataset includes the original text annotated with error, type, and correction. From the 34,893 sentences in the 1,244 exam essays, we extracted 118 sentences that contained the serial errors in question. Other types of errors were replaced with corrections in these sentences.

The second testset is a random sample of 1000 sentences containing trigrams that fit the error patterns also used to evaluate our system. The four system and testset combinations evaluated are:

—Learner corpus without back-off model (**LRN**): The proposed system using only the surface-level translation model was tested on the first testset obtained from a learner corpus.

—Learner corpus with back-off model (**LRN-BO**): The proposed system with the additional back-off model was tested on the first testset obtained from a learner corpus.

—BNC without back-off model (**BNC**): The proposed system using only the surface-level translation model was tested on the first testset obtained from the British National Corpus.

—BNC with back-off model (**BNC-BO**): The proposed system without the back-off model was tested on the first testset obtained from the British National Corpus.

## 4.3 Evaluation Metrics

English correction systems usually are compared based on the quality and completeness of correction suggestions. We measured the quality using the metrics of precision, recall, and error rate. For the first testset, we measured precision and recall rates while, for the second testset, we measured the error rate (false alarms). We define precision and recall as:

Precision = C/S (1)

Recall = C/N (2)

where N is the number of serial errors, S is the number of corrections our system found, and C is the number of corrections where our system was correct. We also computed the corresponding F-score. Error rate was used in the second dataset described above, and we define the error rate as follows:

Error Rate = E/T (3)

where E is the number of corrections our system found (which are all wrong, since we were testing sentences with no errors) and T is the number of sentences tested.

## 5. Evaluation Results

In this section, we report the results of the evaluation using the dataset and environment mentioned in the previous section. During this evaluation, 118 sentences with serial errors were used to evaluate the two systems: **LRN** and **LRN-BO**. Table 1 shows the average precision, recall, and F-score of **LRN** and **LRN-BO**. As we can see, **LRN** performs better in precision, which is reasonable since the back-off model corrects errors without the information of the verb involved. **LRN-BO** performs better in recall because the back-off model applies when the original model does not cover the case. Overall, **LRN-BO** performs better in F-score.

***Table 1. Average precision, recall, and F-score of LRN and LRN-BO***

| | F-Score | Precision | Recall |
|---|---|---|---|
| LRN | 0.43 | 0.71 | 0.31 |
| LRN-BO | 0.45 | 0.68 | 0.33 |

***Table 2. Average error rate of BNC and BNC-BO***

| | Error Rate |
|---|---|
| BNC | 0.10 |
| BNC-BO | 0.13 |

During this evaluation, 1000 sentences in BNC that fit the pattern of serial errors but in fact do not contain errors, were used to evaluate the same two systems: **BNC** and **BNC-BO**. Table 2 shows the average error rate of BNC and **BNC-BO**. It is not surprising that **BNC** performs better than **BNC-BO**, since **BNC** always makes fewer corrections than **BNC-BO**. Nevertheless, **BNC-BO** is only slightly worse than **BNC**.

## 6. Conclusions

Many avenues exist for future research and improvement of our system. For example, spell checking can be done before correcting grammatical errors. Context used to “translate” the serial errors can be enlarged from one word to two or more words (immediately or closely) preceding the errors. We can also add one more level of backing off for the context word preceding the serial errors: from surface word to lemma or from a proper name to named entity type (PERSON, PLACE, ORGANIZATION). We also can improve the accuracy of part of speech tagging used in applying the back-off model.

Additionally, an interesting direction to explore is extending this approach to handle other types of isolated and serial errors commonly found in learners’ writing. Yet another direction of research would be to consider corrections resulting in more or fewer words (*e.g.*, one less word as in **spend time for work* vs. *spend time working*). Or, we could also combine n-gram statistics from different types of corpora: a Web-scale corpus, a reference corpus, and a learner corpus. For example, the translation probability can be determined via statistical classifier training on the learner corpus with features extracted from n-grams of multiple corpora.

In summary, we have introduced a new method for correcting serial errors in a given sentence in learners’ writing. In our approach, a statistical machine translation model is generated to attempt to translate the given sentence into a grammatical sentence. The method involves automatically learning two translation models based on Web-scale n-grams. The first model translates trigrams containing serial preposition-verb errors into correct ones. The second model is a back-off model for the first model, used in the case where the trigram is not found in the training data. At run-time, the phrases in the input are matched using the translation model and are translated before ranking is performed on all possible translation sentences generated. Evaluation on a set of sentences in a learner corpus shows that the method corrects serial errors reasonably well. Our methodology exploits the state of the art in machine translation, resulting in an effective system that can deal with serial errors at the same time.

## References

Atwell, E. S. (1987). How to detect grammatical errors in a text without parsing it. In *Proceedings of the Third Conference of the European Association for Computational Linguistics (EACL)*, 38-45, Copenhagen.

Bender, E. M., Flickinger, D., Oepen, S., & Baldwin, T. (2004). Arboretum: Using a precision grammar for grammar checking in CALL. In *Proceedings of the Integrating Speech Tech- nology in Learning/Intelligent Computer Assisted Language Learning* *(inSTIL/ICALL) Symposium: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice.

Brants, T., & Franz, A. (2006). *The Google Web 1T 5-gram corpus version 1.1*. LDC2006T13.

Coxhead, A. (2000). A new academic word list. *TESOL quarterly*, *34*(2), 213-238.

De Felice, R., & Pulman, S. G. (2009). Automatic detection of preposition errors in learner writing. *CALICO Journal*, *26*(3), 512-528.

Eeg-Olofsson, E., & Knuttson, O. (2003). Automatic grammar checking for second language learners - the use of prepositions. In *Proceedings of the 14th Nordic Conference in Computational Linguistics* (NoDaLiDa).

Gamon, M. (2010). Using mostly native data to correct errors in learners' writing. In *Proceedings of the Eleventh Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Los Angeles.

Gamon, M., Gao, J., Brockett, C., Klementiev, A., Dolan, W. B., Be-lenko, D., & Vanderwende, L. (2008). Using contextual speller techniques and language modeling for ESL error correction. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, 449-456, Hyderabad, India.

Graddol, D. (2006). *English next: Why global English may mean the end of 'English as a Foreign Language*.' UK: British Council.

Han, N.-R., Tetreault, J., Lee, S.-H., & Ha, J.-Y. (2010). Using error-annotated ESL data to develop an ESL error correction system. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC)*, Malta.

Heidorn, G. E. (2000). Intelligent writing assistance. In R. Dale, H. Moisl, and H. Somers, editors, *Handbook of Natural Language Processing*, 181-207. Marcel Dekker, New York.

Izumi, E., Uchimoto, K., Saiga, T., Supnithi, T., & Isahara, H. (2003). Automatic error detection in the Japanese learners' English spoken data. In *Companion Volume to the Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, 145-148.

Knight, K., & Chander, I. (1994). Automated postediting of documents. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI)*, 779-784, Seattle.

Leacock, C. *et al*. 2010. Automated grammatical error detection for language learners. *Synthesis Lectures on Human Language Technologies*, *3*(1), 1-134.

Lee, J., & Seneff, S. (2006). Automatic grammar correction for second-language learners. In *Proceedings of the Ninth International Conference on Spoken Language Processing (Interspeech)*, 1978-1981.

Lee, J., Tetreault, J., & Chodorow, M. (2009b). Human evaluation of article and noun number usage: Influences of context and construction variability. In P*roceedings of the Third Linguistic Annotation Workshop (LAW)*, 60-63, Suntec, Singapore.

Rozovskaya, A., & Roth, D. (2013). Joint Learning and Inference for Grammatical Error Correction, In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 791-802.

West, M. (1953). *A General Service List of English Words*. London: Longman, 1953.

Yannakoudakis, H., Briscoe, T., & Medlock, B. (2011). A New Dataset and Method for Automatically Grading ESOL Texts, In *Proceedings of the Annual Meeting of the Association for Computational Linguistic*.