國立清華大學

碩士論文

語言模型的文法改錯中之遺漏詞處理

Dealing with Missing Words in Grammatical Error Correction based on Language Model

系別: 資訊系統與應用研究所

指導教授: 張俊盛 博士 Dr. Jason S. Chang

學號姓名: 106065523 謝毅霖 Yi-Lin Xie

中華民國一〇九年六月

Abstract

We present a grammatical error correction system that automatically corrects a given sentence with potential grammatical errors without using a parallel data. In our approach, a language model approach is used, based on the idea that low probability sentences are more likely to contain grammatical errors than high probability sentences. The method involves generating and ranking corrective candidates using confusion sets, missing word recovery, and training an inference model to maximize the LM probability and correct grammatical errors. Preliminary evaluation shows that our approach achieves better performance compared to previous work, especially for insertion errors.

Keywords: Grammatical Error Correction, Language Model

摘要

本論文提出一個英文文法改錯的方法，可以在不依賴標注資料的情況下來改正輸入句子中潛在的文法錯誤。 我們採取應用語言模型（Language Model, LM）的研究路線——透過語言模型機率較高的句子較不容易含有文法錯誤的特性，來找到正確的改正結果。 此方法涉及建立勘誤表(Confusion Set)，遺漏詞復原，及訓練推論模型來生成、排序可能的改正結果，藉此將語言模型機率提升至最大，以改正文法錯誤。 實驗結果顯示，我們的方法較前人研究，獲得較佳的結果，且對於遺漏詞處理的方面上，有著顯著的改進。

關鍵字: 文法改錯、語言模型

致謝

僅以此篇論文獻給我的家人們，感謝父親和母親在我人生中求學階段給我的支持，讓我一路走來順利完成學業，取得碩士學位。

在清大資應求學的這段時間，特別感謝我的指導老師張俊盛教授在研究方向及實驗方法上給我的建議，尤其在學術英文寫作上不厭其煩的指導著實讓我受益良多，使我能如期完成碩士論文，在撰寫論文時對於老師不辭辛勞協助校閱不勝感激。

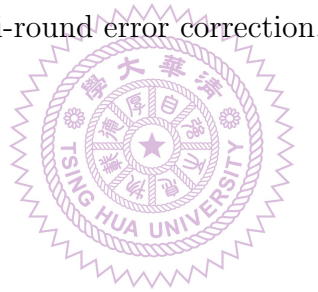
此外，要感謝實驗室的陳志杰學長提供我諸多寫程式上的技巧，且在我尋找海外實習機會時提供許多寶貴的意見。 也謝謝楊謦瑜學姊提供我許多關於語言學的知識，幫助我在自然語言的研究上能夠順利。 謝謝一起奮鬥的實驗室好夥伴文彬、仲庭、佳芳、名喬、鳳華、怡惠、沛臻和映竹，在新竹的這段日子令人難以忘懷，有你們的相伴使我的日常生活充實且不孤單。

謝毅霖

2020 於清大資應

Contents

List of Figures



List of Tables

Chapter 1

Introduction

Many sentences with potential errors (e.g., *"We look forward to see you."*) are submitted by learners of English as a Second or Other Language (ESOL) to writing services every day, and an increasing number of writing services specifically target Grammatical Error Correction (GEC). For example, *LanguageTool* [1] corrects grammar errors in submitted sentences using rules extracted from a corpus, while *Ginger* [2] and *Grammarly* [3] use proprietary algorithms to provide more powerful corrective feedback for language learners.

Grammar check services such as *Ginger* and *Grammarly* typically use Machine Translation (MT) based approaches, in order to provide correction for common grammatical errors in a given sentence. However, such MT-based approaches usually require a large amount of annotated training data, which are usually expensive, unreliable, or even unavailable. Even when reliable data is available, the size of the dataset may impose an upper bound on trained model. In other words, lack of high quality annotated training data is problematic for GEC. However, GEC can also be addressed by applying language model based approach, with the idea that a low-probability given sentence is more likely to be erroneous while high-probability correction sentences are likely to be correct. Using LM is advantageous because it only requires native training data, which is readily available in Web scale nowadays. Moreover, it is feasible to handle every error type, provided plausible edits can be generated. A GEC system could be more effective if we use

[1] `https://www.languagetool.org/`

[2] `https://www.gingersoftware.com/`

[3] `https://www.grammarly.com/`

a reasonable edit generating model and a large-scale language model.

Consider the sentence *"I am looking forway see you."*. A good correction might be *"I am looking forward to seeing you."*, a case of misspelling *"forward"*, missing *"to"*, and a verb-form misuse of *"seeing"*. Such errors are probably not easy to correct using a MT-based GEC system with limited and noisy training data. Intuitively, by using a language model and a confusion set with tense changes, preposition, and deletion, we can improve GEC.

In this thesis, we present a prototype system, *LessErrors* (`http://lesserrors.linggle.com`), that attempts to automatically correct grammatical errors without using a parallel data. An example of *LessErrors* handling the input sentence *"I am looking forway see you."* is shown in Figure 1.1. *LessErrors* has determined that *"I am looking forward to seeing you."* is the most probably correction. *LessErrors* corrects a given sentence by using a language model and incorporating a sophisticated corrective candidate generation and filtering strategy, which involves building confusion sets and training a missing word insertion model and an inference model. We will describe the process in more detail in Chapter 3.

At run-time, *LessErrors* starts with a sentence submitted by the user (e.g., *"I am looking forway see you."*), and generate corrective candidates with three kinds of edit. Then, *LessErrors* chooses the candidate that maximizes LM probability. This process repeats multiple times until the probability stops increasing. Finally, *LessErrors* returns the final result.

The rest of this paper is organized as follows. In the next chapter, we review related work. Then in Chapter 3, we describe the proposed GEC method. We describe the implementation, experiment settings, and the evaluation results in Chapter 4. Finally, we conclude and explore the future directions in Chapter 5.

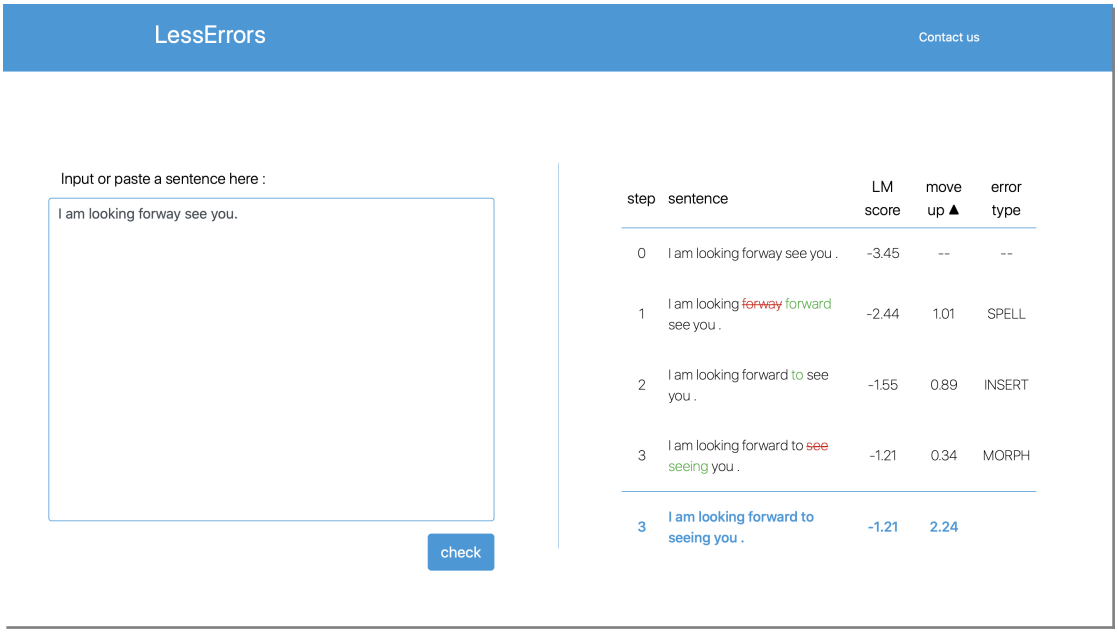


Figure 1.1: A screenshot of the system handle the sentence “I am looking forway see you.”.

Chapter 2

Related Work

Grammatical Error Correction (GEC) has been an area of active research. Recently, researchers have begun applying neural machine translation (NMT) models to solve GEC problems, and gained significant improvement (e.g. Yuan and Briscoe (2016); Xie et al. (2016)). In our work, we focus on using a language model based approach to correct grammatical errors with a sophisticated candidate generation strategy. While recent GEC systems are pursuing NMT-based models with huge amounts of annotated training data, our approach only requires limited annotated data for tuning parameters.

Early work on GEC task typically uses rule-based and statistical approaches with rules learned from an annotated corpus (Leacock et al., 2010). Rule-based approaches usually use phrasal and grammatical rules formulated by experts to generate possible corrections (McCoy et al., 1996; Park et al., 1997; Schneider and McCoy, 1998; Michaud et al., 2000; Dodigovic, 2007). Unlike rule-based approaches which require expertise and extensive labor, statistical approaches extract knowledge automatically by building learning-based classifiers (Han et al., 2006; Chodorow et al., 2007; Lee and Seneff, 2008; De Felice and Pulman, 2008; Tetreault et al., 2010).

In recent years, Statistical Machine Translation (SMT) has been applied to correct grammatical errors. Junczys-Dowmunt and Grundkiewicz (2016) presented a system using phrase-based SMT approach that incorporates features such as edit operation counts, a word class language model, the Operation Sequence Model (OSM) (Durrani et al., 2015), and sparse edit operations. Yannakoudakis et al.

(2017) explored the strength of both error detection model and SMT-based approaches, and integrated both to build a hybrid GEC system. They used an additionally learned sequence labeling model that calculates the probability of each token being correct or incorrect to re-rank SMT model's N-best candidates, with the goal of finding the best correction.

More recently, Neural Machine Translation (NMT) has been adopted in grammatical error correction and has achieved state-of-the-art performance. Yuan and Briscoe (2016) presented the very first NMT model for grammatical error correction of English sentences and proposed a two-step approach to handle the out-of-vocabulary problem in NMT. Xie et al. (2016) presented a character-based NMT model for language correction, with the goal of avoiding the out-of-vocabulary problem. Chollampatt and Ng (2018) proposed a multilayer convolutional encoder-decoder neural network to correct grammatical, orthographic, and collocation errors. The NMT models usually require a large amount of annotated training data, but the largest set of publicly available parallel data (Lang-8 Learner Corpora) in GEC has only two million sentence pairs (Mizumoto et al., 2011). To cope with data sparseness, researchers have proposed methods for augmenting data by using native-speaker data (Xie et al., 2018; Ge et al., 2018; Lichtarge et al., 2019; Zhao et al., 2019; Kiyono et al., 2019).

In a study closer to our work, Kao et al. (2013) presented a language model (LM) based approach for grammatical error correction. They used frequency and dependency relation to build several modules, aimed at correcting determiner, noun number, verb form, and preposition errors independently. Bryant and Briscoe (2018) presented a similar approach which only requires a small annotated dataset. They used spellchecker, human-defined rules, and language model to iteratively correct replacement and deletion errors. However, their method is limited by the ability to deal with insertion errors, which amount to 20% of all errors. Our method, which we will describe in the next chapter, extends the method of Bryant and Briscoe (2018) by incorporating a novel candidate generation strategy to address the insertion errors.

In the research area of missing word recovery, The Billion Word Imputation

Challenge [1] presented a corpus of sentences each with one word missing, based on One Billion Word Benchmark dataset (Chelba et al., 2013). In this task, an incomplete sentence with one word erased was provided as an input to the models, and the goal is to predict the erased words correctly. Mani (2015) proposed a method that uses a recurrent neural network language model (RNNLM) to predict the likelihood of possible candidates with a word inserted in the given sentences. However, this approach is computationally expensive in that corrective candidates are generated by inserting all the word from a pruned vocabulary into every position between words, leading to a $O(VN)$ computational cost of processing a sentence (where N is the number of words in the sentence and V is the size of a pruned vocabulary) and making it impractical for a real-time system. In contrast to researches with the direct focus of missing word recovery, Yuan et al. (2016) presented a long short-term memory (LSTM) model for Word Sense Disambiguation (WSD). Their approach relies on predicting the held-out word in a sentence given the surrounding context. In our work, we use a similar method with a different goal of correcting missing errors.

In the research area of Natural Language Inference (NLI), Bowman et al. (2015) presented a large, high quality dataset and created a benchmark for evaluation. In this task, a text classification model would be given a *premise* and a *hypothesis*, and the goal is to predict whether the *hypothesis* is an entailment, contradiction, or being neutral with respect to the *premise*. For example, the sentence *"A man inspects the uniform of a figure in some East Asian country."* are contradictory to the sentence *"The man is sleeping."* In relation to GEC, a possible application of NLI is to filter inappropriate corrective candidates by detecting semantic shift from the given sentence. Our approach, which we will describe in the next chapter, uses a similar approach to reduce false positive when attempting to correct insertion errors.

In contrast to the previous work in GEC, we extend LM-based approach to GEC with performance rivaling the latest NMT approaches that rely on a large amount of annotated training data. The method involves generating and ranking corrective candidates using confusion sets, missing word recovery, and Natural

[1] `https://www.kaggle.com/c/billion-word-imputation`

Language Inference, with the goal of maximizing the LM probability and correcting grammatical errors. The results show that our approach achieves better performance compared to previous work, especially for insertion errors.


TSING HUA UNIVERSITY

Chapter 3

Methodology

Submitting a sentence with potential grammatical errors (e.g., *"I am looking for-way see you."*) to MT-based GEC systems with limited training data often does not work very well. MT-based GEC systems typically require a huge amount of annotated training data. Unfortunately, the largest set of publicly available parallel data (Lang-8 Learner Corpora) in GEC has only two million sentence pairs, and has a high false negative rate. When encountering new and unseen errors and contexts, these systems might not be able to correct such errors. To provide a more effective GEC system, a promising way is to build confusion set, use a missing word insertion model and an inference model, and rely on language model that leverages the high availability of the monolingual text.

3.1 Problem Statement

We focus on correcting grammatical errors and offering suggestions by considering using an alternative word, or deleting/inserting a word and ranking alternative edits. These top-ranking edits are then returned as the output of the system. The returned edited sentences can be viewed by the users directly for self-editing, or passed on to other applications such as automatic essay rater and assisted writing systems. Thus, it is important that the grammatical errors in a given sentence be corrected as many as possible. At the same time, the system should avoid making false corrections. Therefore, our purpose is to return edited sentences with the most grammatical errors corrected, while keeping false positive reasonably low.

We now formally state the problem that we are addressing.

Problem Statement: We are given a sentence with potential spelling or grammatical errors. The sentence is a sequence of word tokens $S = (w_1, w_2, ..., w_n)$, where n is the length of the sentence. Our goal is to return the edited sentence Y with m word tokens $(y_1, y_2, ..., y_m)$. For this, we repeatedly generate and rank edits for the given sentence, aiming at maximizing the LM probability of Y.

In the rest of the chapter, we describe the solution to this problem. First, we describe how to train all our necessary models in Section 3.2. Then, we define a strategy for correcting grammatically erroneous sentences in Section 3.3. This strategy relies on generating candidates with varieties of edit operations, including replacement, deletion, and insertion. In this section, we also describe our method for automatically filtering inappropriate candidates and predicting the most likely correction.

3.2 Training Phase

3.2.1 Training a Language Model

In the first stage of the training process, we train a language model (LM) to assess the grammatiaclity of a given sentence. The input to this stage is a monolingual corpus (e.g. Wikipedia). The output of this stage is a full-fledged language model.

The most common way to build a language model is to compute the probabilities of short windows of text (n-grams) from a large corpus of monolingual text. In the general case, the probability of any window of N words (from word w_{n-N+1} to word w_n) can be expressed as equation 3.1 and the chain rule can be applied to calculate probabilities of sequences longer than N, as shown in equation 3.2.

$$Pr(w_k|w_{k-N+1}, ..., w_{k-1}) = \frac{Count(w_{k-N+1}, ..., w_k)}{Count(w_{k-N+1}, ..., w_{k-1})} \tag{3.1}$$

$$Pr(w_1, w_2, ..., w_n) = \prod_{k=N}^{n} Pr(w_k|w_{k-N+1}, ..., w_{k-1}) \tag{3.2}$$

In this work, we employ the modified Kneser-Ney smoothing LM proposed by Chen and Goodman (1996), which is perhaps the best smoothing method widely

used today. We train our LM with an existing open-source toolkit, KenLM [1], which is fast, memory-efficient and adaptable to various programming language (e.g., Python, C, Java) (Heafield, 2011). The training details of our LM will be described in Section 4.3.

3.2.2 Training a Missing Word Insertion Model

In the second stage of the training process, we train a missing word insertion model to recover the missing words in a given sentence. The input to this stage is a monolingual corpus (e.g. Wikipedia). The output of this stage is a missing word insertion model.

To train the missing word insertion model, we first mask some tokens of the input sentences at random by substituting a special token `<mask>`. For example, the sentence *"I am looking forward to seeing you"* is converted to *"I* `<mask>` *looking forward* `<mask>` *seeing you"*. After that, we use these masked sentences to train our model, with the goal of predicting the masked tokens.

In our approach, we use a sequence tagging model for missing word recovery. The model predicts a missing word given the surrounding context. For the word sequence $(w_1, ..., w_n)$ with a `<mask>` token at position k, the missing word insertion model are trained to predict the masked token w_k given the surrounding context $W_{-k} = (w_1, ..., w_{k-1}, w_{k+1}, ..., w_n)$, which can be written as the following equation:

$$w_k = \arg\max_{w} Pr(w|W_{-k}) \tag{3.3}$$

In this work, we utilize RoBERTa (Robustly optimized BERT approach) (Liu et al., 2019), a variant of Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018), as the missing word insertion model. RoBERTa is a deep bidirectional language model trained using a dynamic masking strategy, which changes the masking pattern every time a sequence is fed into the model. The implementation details of the missing word insertion model will be described in Section 4.3.

[1] `https://github.com/kpu/kenlm`

Premise	Hypothesis	Label
A soccer game with multiple males playing.	Some men are playing a sport.	entailment
A black race car starts up in front of a crowd of people.	A man is driving down a lonely road.	contradiction
A smiling costumed woman is holding an umbrella.	A happy woman in a fairy costume holds an umbrella.	neutral

Table 3.1: Example of annotated data for natural language inference

3.2.3 Training a Natural Language Inference Model

In the third stage of the training process, we train a natural language inference (NLI) model to detect semantic shift between a given sentence and its corrective candidates. The input to this stage is an annotated dataset for natural language inference, each consisting of a premise, a hypothesis, and their logic relationship (i.e., entailment, contradiction, or neutral). The output of this stage is a NLI model.

In our approach, we train a multi-label classifier on The Stanford Natural Language Inference Corpus (SNLI) (Bowman et al., 2015) to detect the semantic discrepancy. The model trained on SNLI predicts the logic relationship y between a premise $a = (a_1, a_2, ..., a_n)$ and a hypothesis $b = (b_1, b_2, ..., b_m)$, which can be written as the following equation:

$$y_i = \arg\max_{y} Pr(y|a; b), \tag{3.4}$$

where $y \in \{entailment, contradiction, neutral\}$ and n, m is the sequence length of a, b, respectively.

In this work, we utilize RoBERTa mentioned in Section 3.2.2 as our NLI model. The implementation details of our NLI model is described in Section 4.3

3.3 Correcting Errors based on Language Model

Once we obtain the language model, missing word insertion model, and inference model, we apply these models at run-time. We attempt to correct grammatical errors by generating corrective candidates with varieties of edit operations and scoring corrective candidates with language model. Our correction process is shown in Figure 3.1.

```
procedure CorrectGrammaticalErrors(Sentence)
        copy Sentence to originalSentence
        hasError = True
        While hasError
            candidates = ∅
(1)         candidates += GenerateReplaceDeleteCandidates(Sentence)
(2)         candidates += GenerateInsertCandidates(Sentence)
(3a)        sentScore = Score(Sentence)
            bestScore = -inf
            bestSentence = ∅
            For each <threshold, candidate> in candidates
(3b)            candScore = Score(candidate)
(4a)            If candScore > sentScore * threshold and candScore > bestScore
(4b)                If SemanticCheck(originalSentence, candidate) is entailment
                        bestScore = candScore
                        bestSentence = candidate
            If bestSentence is not ∅
(5)             Sentence = bestSentence
            else
                hasError = False
(6)     return Sentence
```

Figure 3.1: Correcting Sentences at run time

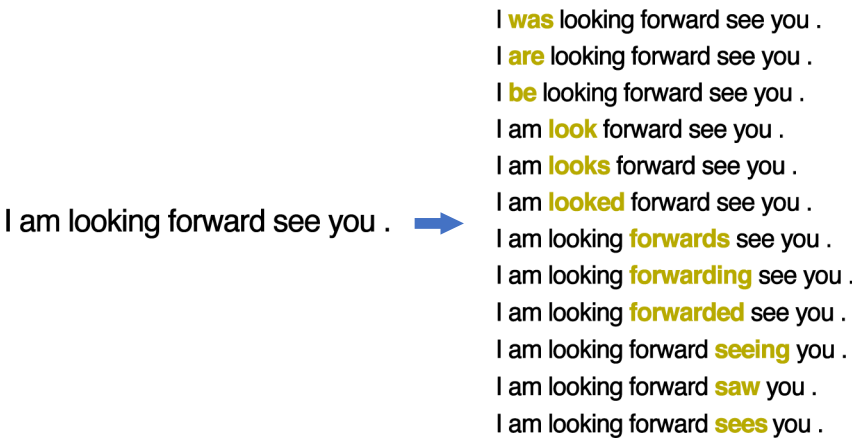


Figure 3.2: Generating corrective candidates using confusion sets.

3.3.1 Generating Replacement and Deletion Candidates

In the first stage of the correction process (Step (1) in Figure 3.1), we generate corrective candidates with replacement or deletion edit operation by creating confusion sets. More specifically, we use a confusion set based on all potential alternative words for each token in the input sentence, where the alternative words are variously generated by a spellchecker, morphological inflection database, and other human-defined rules. After creating confusion sets, we then generate corrective candidates by replacing each token in the input sentence with the alternative words from confusion sets. Figure 3.2 shows an example of generating corrective candidates for the sentence *"I am looking forward see you."* using confusion set. The input to this stage is the input sentence, and the output of this stage is a group of corrective candidates.

In this work, we follow Bryant and Briscoe (2018) and use confusion sets to generate candidate corrections related to non-words, morphology, articles, and prepositions, which constitute a large proportion of errors in learner text.

- **Non-words:** We generate a set of new candidates by replacing a misspelled word with each correction proposed by a spellchecker. Non-words include genuine misspellings, such as [*freind* → *friend*], and inflectional errors such as [*advices* → *advice*]. We use CyHunspell [2] as our spellchecker.

- **Morphology:** Morphological errors include noun number (e.g. [*cat* →

[2] `https://pypi.org/project/CyHunspell/`

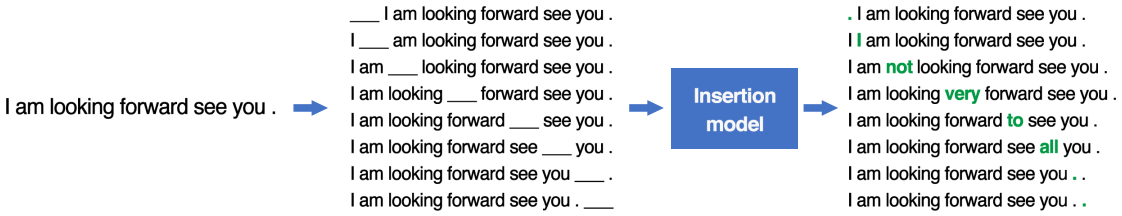


Figure 3.3: Generating insertion candidates using the missing word insertion model.

cats]), verb tense (e.g. [*eat* → *ate*]), and adjective form (e.g. [*big* → *bigger*]). In this work, we use an Automatically Generated Inflection Database (AGID) [3], which contains all the morphological forms of most common English words, to generate corrective candidates for morphological errors.

- **Articles and Prepositions:** We defined confusion sets for these errors by rules. The article confusion set comprises {ϵ, a, an, the}, while the preposition comprises {ϵ, about, at, by, for, from, in, of, on, to, with}, which is the set of the top 10 most frequent prepositions. A null character ϵ is contained in both sets to represents a deletion.

3.3.2 Generating Insertion Candidates

In the second stage of the correction process (Step (2) in Figure 3.1), we generate corrective candidates with insertion edit operations for correcting missing word errors. Such correction can not be done by creating confusion sets because it is difficult to know what and where to insert a missing word. Therefore, we use a different generation strategy for generating insertion candidates.

First, we insert slots into the input sentence. For a sequence of length n, we generate $n + 1$ candidates by inserting a word at every position between words, before the first word, and after the last word. Then, we apply the missing word insertion model to generate the corrective candidates by filling the slots. An example of generating insertion candidates for the sentence *"I am looking forward see you."* is shown in Figure 3.3, where the input to this stage is the input sentence, and the output of this stage is a set of insertion candidates.

[3] `http://wordlist.aspell.net/other/`

3.3.3 Assessing Grammaticality with LM

In the third stage of the correction process (Step (3a) and (3b) in Figure 3.1), we use the language model to assess the grammaticality of the input sentence and each generated candidate. Here we evaluate sentence grammaticality in terms of normalized log probabilities at the sentence level. Normalization by sentence length is necessary to help overcome the tendency for shorter sequences to have higher probabilities than longer sequences. The normalized log probability can be written as:

$$score_{LM}(S) = \frac{1}{|S|} \log Pr(S), \tag{3.5}$$

where S is a sentence and $|S|$ is the sentence length in tokens.

3.3.4 Filtering out Inappropriate Candidates

In the fourth stage of the correction process (Step (4a) and (4b) in Figure 3.1), we filter out inappropriate corrective candidates generated in Section 3.3.1 and Section 3.3.2. Since it is conceivable that there might be some cases where corrections are false positives. For example, even though the correction [*"I am a teacher."*→*"I was a teacher."*] increases the probability from -1.537 to -1.534, the edit [*am* → *was*] is not necessarily an error. To deal with such an issue, we use two candidate filtering strategies to minimize false positive.

Setting Threshold

The above-mentioned issue usually occurs in a correction with a tiny improvement. A simple yet efficient way is to set a threshold α for a candidate correction. We only accept corrections that improve the probability by at least α% than before:

$$score_{cand} > \alpha \cdot score_{input}, \alpha \in (0, 1] \tag{3.6}$$

where $score_{cand}$ denotes the LM score of a corrective candidate and $score_{input}$ denotes the LM score of the input sentence [4].

[4] As we will describe in Section 3.3.5, we use a multi-round correction strategy to correct the grammatical errors. For clarification, the input sentence here is not the user's original input sentence, but each round's input sentence.

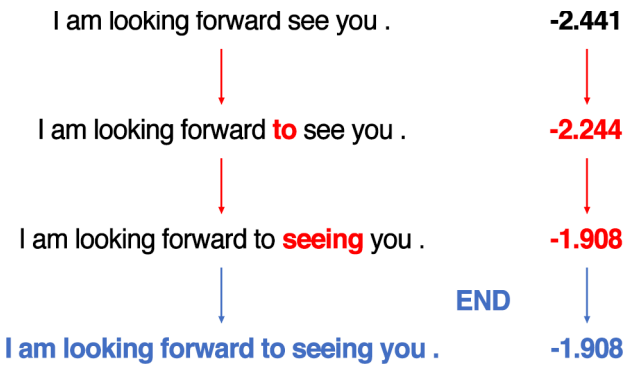


Figure 3.4: An overview of multi-round error correction.

In our work, we use two different thresholds for filtering inappropriate candidates generated in Section 3.3.1 and Section 3.3.2, respectively. We describe how to choose these thresholds in Section 4.2.

Ensuring semantic consistency with NLI model

For insertion edit operation, adding a single word sometimes would result in semantic shift. For example, the correction [*"I am surprised."*→*"I am not surprised."*] increases the LM probability from -1.772 to -1.322. However, the correction of inserting *"not"* in this case is not only a false positive, but also changes the original semantic meaning. This suggests that setting threshold is insufficient for getting a good trade-off between precision and recall. To avoid this issue, we apply the previously trained NLI model to prevent semantic discrepancy. In our approach, we take the user's original input sentence as the premise and each candidate to be evaluated as the hypothesis. We then discard candidates whose prediction results are not *entailment* (i.e., *neutral* or *contradiction*), in order to cope with errors caused by semantic inconsistency.

3.3.5 Iterative Correction with Generated Candidates

In the fifth and final stage of the correction process (Step (5) and (6) in Figure 3.1), we correct the grammatically erroneous sentence using the filtered corrective candidates generated in Section 3.3.4. In this stage, a multi-round correction strategy is adopted, where only one correction is considered at a time. At each round, we correct the sentence by selecting the correction that leads to highest LM probabil-

ity, and the correction process continues as long as the LM probability is increased. For a sentence with multiple errors, usually the most severe errors will be corrected first. After that, the corrected parts will make the context more informative for another error to be corrected. Figure 3.4 shows an example of the proposed multi-round correction process.


TSING HUA UNIVERSITY

Chapter 4

Experiment and Evaluation

LessErrors was designed to correct grammatical errors written by ESOL learners. As such, the system will be developed and evaluated on a collection of ungrammatical sentences written by non-native learners. In this chapter, we first give a brief description of the datasets and toolkits used in the experiments in Section 4.1, and describe the tuning process for various thresholds in Section 4.2. Then, Section 4.3 describes the hyper-parameters setting of the language model, missing word insertion model, and NLI model. Section 4.4 describes several systems with different experimental settings for comparing performance. We introduce the evaluation metrics for evaluating the performance in Section 4.5, and finally discuss the results and analyze system errors in Section 4.6.

4.1 Datasets and Tools

We used the following datasets for development and evaluation: CoNLL-2013 and CoNLL-2014 test sets, and the JFLEG corpus. Specifically, we used CoNLL-2013 and CoNLL-2014 test sets to evaluate how our systems improve grammaticality, and used JFLEG corpus to evaluate how our systems improve fluency. Additionally, we also used two toolkits in our experiment: a text processing toolkit (spaCy) and a grammatical error annotator (ERRANT).

CoNLL-2013 and CoNLL-2014: These two datasets are the official test sets of CoNLL-2013 and CoNLL-2014 Shared Task (Ng et al., 2013, 2014), which contain 1,381 and 1,312 sentences, respectively. To evaluate grammaticality, we first

used CoNLL-2013 test set as our development set for threshold tuning. We then used CoNLL-2014 test set with original annotations for evaluation. In addition, we also used another annotations released by CoNLL-2014, which combines gold-standard answers with alternative answers proposed by participants, to analyze error types in terms of edit operation.

JFLEG: The JHU FLuency-Extended GUG corpus (JFLEG) (Napoles et al., 2017) is a parallel corpus for developing and evaluating GEC systems with respect to fluency. It consists of 1,501 source sentences (with 754 in development set and 747 in test set) written by students with different native languages and proficiency levels. To evaluate the performance of fluency, we first used the development set for threshold tuning. Then, we used the test set for evaluating the system performance.

spaCy: spaCy (Honnibal and Johnson, 2015) [1] is a free open-source library for text processing that supports Python. It supports tokenization, which keeps track of the character offsets of the tokens it separates and makes it a lot easier to map characters to tokens. We used spaCy to tokenize the input sentences for preprocessing.

ERRANT: ERRor ANnotation Toolkit (ERRANT) (Bryant et al., 2017) [2] is an automatic annotation tool for error type analysis, which provides a detailed view of the system performance in terms of edit operation. We used ERRANT for threshold tuning and error type analysis.

4.2 Threshold Tuning

Although annotated training data is not required in our LM based approach, a small amount of annotated development data is helpful to reduce false positive by exploring different values for thresholds, as described in Section 3.3.4. In our work, we set two different thresholds for candidates generated using confusion sets (Section 3.3.1) or the missing word insertion model (Section 3.3.2): t_c and t_i.

In our work, we used ERRANT $F_{0.5}$ for tuning a threshold. Since it is impractical to exhaustively test all the combinations of t_c and t_i, we only optimize t_c and

[1] `https://spacy.io/`

[2] `https://github.com/chrisjbryant/errant`

t_i in the range of 0.85-1.0 on the given development set.

4.3 Model Implementation

4.3.1 Language Model

For grammaticality assessment in GEC, we trained a language model with KenLM to calculate the sequence probabilities, as mentioned in Section 3.2.1. The n-gram degree of LM is set to 5. We used a language model trained on the One Billion Word Benchmark dataset (Chelba et al., 2013), which consists of nearly a billion words of English taken from news articles on the Web.

4.3.2 Missing Word Insertion model

For correcting missing words, we used HuggingFace's Transformers library (Wolf et al., 2019) [3] for the RoBERTa model to generate insertion candidates, as mentioned in Section 3.2.2. Here, we utilized the released pretrained RoBERTa$_{LARGE}$ model, which employs the same architecture as BERT$_{LARGE}$ ($L = 24$, $H = 1024$, $A = 16$, 355M parameters). RoBERTa$_{LARGE}$ was trained on over 160 GB of raw text from different English-language corpora, including BookCorpus (Zhu et al., 2015) [4], Wikipedia [5], a portion of the CC-NEWS dataset [6], OpenWebText corpus of Web content extracted from URLs shared on Reddit (Gokaslan and Cohen) [7], and a subset of CommonCrawl (Trinh and Le, 2018) [8].

4.3.3 Natural Language Inference Model

To ensure semantic consistency, we implemented an NLI model using AllenNLP library (Gardner et al., 2018) [9]. In this study, we used the publicly available NLI

[3] https://github.com/huggingface/transformers
[4] https://github.com/soskek/bookcorpus
[5] https://github.com/google-research/bert
[6] https://github.com/fhamborg/news-please
[7] https://github.com/jcpeterson/openwebtext
[8] https://github.com/tensorflow/models/tree/master/research/lm_commonsense
[9] https://allennlp.org/

model, which is based on RoBERTa$_{LARGE}$ and was additionally finetuned on the SNLI dataset, to support our system.

4.4 Systems compared

In evaluation, we compared three systems with different experimental settings to correct grammatical errors:

- **LMGEC**$_{NLI}$ uses candidates generated by confusion sets; then correction is applied only if NLI model's prediction result is *entailment*.
- **LMGEC**$_{M}$ uses candidates generated by both confusion sets and missing word insertion model.
- **LMGEC**$_{M+NLI}$ uses candidates generated by both confusion sets and missing word insertion model; then a correction is applied only if NLI model's prediction result is *entailment*.

We use the results of Bryant and Briscoe (2018) as our baseline system.

4.5 Evaluation Metrics

GEC systems are usually compared based on the quality of the correction. This quality is traditionally qualified using two metrics: MaxMatch (Dahlmeier and Ng, 2012) and GLEU (Napoles et al., 2015).

4.5.1 MaxMatch

The MaxMatch (M^2) scorer, proposed by Dahlmeier and Ng (2012), is used to evaluate system performance by how well its proposed corrections or edits matching the gold-standard edits. It computes the sequence of phrase-level edits between a source sentence and a system's candidate that achieves the highest overlap with the gold-standard annotation. Evaluation is performed by computing precision (P), recall (R) and $F_{0.5}$:

$$P = \frac{\sum_{i=1}^{n} |e_i \cap g_i|}{\sum_{i=1}^{n} |e_i|} \tag{4.1}$$

$$R = \frac{\sum_{i=1}^{n} |e_i \cap g_i|}{\sum_{i=1}^{n} |g_i|} \tag{4.2}$$

$$F_{0.5} = (1 + 0.5^2) \times \frac{P \times R}{(0.5^2 \times P) + R} \tag{4.3}$$

where $e_i = (e_1, e_2, ..., e_n)$ is the system's candidate edit set and $g_i = (g_1, g_2, ..., g_n)$ is the gold-standard edit set. The intersection between e_i and g_i is defined as:

$$e_i \cap g_i = \{e \in e_i \mid \exists\ g \in g_i : match(e, g)\} \tag{4.4}$$

where $match(e, g)$ determines whether the system edit e matches the reference edit g and depends on the chosen task.

The M^2 metric was the official evaluation metric for the CoNLL-2014 shared task on GEC. We followed the previous research and used M^2 precision, recall and $F_{0.5}$ to evaluate the CoNLL-2014 test set's grammaticality.

4.5.2 GLEU

Generalized Language Evaluation Understanding (GLEU), proposed by Napoles et al. (2015), is a simple variant of BLEU (Papineni et al., 2002) for GEC which takes the original source into account. It focuses on fluency rather than grammaticality. GLEU modifies the n-gram precision in BLEU to assign extra weight to n-grams present in the candidate that overlap with the reference but not the source ($R \backslash S$), and penalize those in the candidate that are in the source but not the reference ($S \backslash R$). For a correction candidate C with a corresponding source S and reference R, the modified n-gram precision ($p_n{}'$) for $GLEU(C, R, S)$ is defined as:

$$p_n{}' = \frac{\sum_{ngram \in C} count_{R \backslash S}(ngram) - \lambda(count_{S \backslash R}(ngram)) + count_R(ngram)}{\sum_{ngram \in C} count_S(ngram) + \sum_{ngram \in R \backslash S} count_{R \backslash S}(ngram)} \tag{4.5}$$

where the weight λ is determined by how much incorrectly changed n-grams are penalized. Given a bag of n-grams B, the counts in Equation 4.5 are collected as:

$$count_B(ngram) = \sum_{ngram' \in B} d(ngram, ngram') \tag{4.6}$$

$$d(ngram, ngram') = \begin{cases} 1, & \text{if } ngram = ngram' \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

By applying the n-gram precision in Equation 4.5, GLEU is defined as:

$$GLEU(C, R, S) = BP \cdot \exp \sum_{n=1}^{N} w_n \log p'_n \tag{4.8}$$

$$BP = \begin{cases} 1, & \text{if } c > r \\ e^{(1-c/r)}, & \text{if } c \leq r \end{cases} \tag{4.9}$$

In our evaluation, we use $N = 4$ and uniform weights $w_n = 1/N$, which are the same parameters as in BLEU. In our evaluation, we use $N = 4$ and uniform weights $w_n = 1/N$, which are the same parameters as in BLEU.

Since JFLEG was originally released alongside the GLEU metric, most systems evaluating on JFLEG use GLEU rather than M^2. Thus, we used GLEU score to evaluate the JFLEG test set's fluency.

4.6 Evaluation Results

In this section, we report the results of the experimental evaluation using the metrics described in the previous section. Specifically, we first evaluate our systems in terms of grammaticality on the CoNLL-2014 test set with M^2 metric. Then, we measure the fluency of the outputs on the JFLEG test set with GLEU score. We also report the error type analysis with respect to edit operation on CoNLL-2014 test set using ERRANT. The reported results are based on the best performing thresholds on each dataset.

Table 4.1 shows the evaluation results on the CoNLL-2014 test set. As we can see, all our systems outperform previous work on $F_{0.5}$. Among all systems, LMGEC$_{M+NLI}$ achieves the best performance with 40.97 precision and 35.61 $F_{0.5}$. This indicates that incorporating missing word recovery to correct missing errors and NLI model to filter inappropriate candidates improves the overall GEC performance.

Table 4.2 shows the evaluation results on the JFLEG test set. As we can see, all our systems outperform previous work on GLEU score and LMGEC$_M$ achieves the best performance among all systems (49.80 GLEU). This means that our systems successfully correct sentences to improve fluency. It is interesting

System	P	R	$F_{0.5}$
Bryant and Briscoe (2018) (t_c: 0.98)	40.56	20.81	34.09
LMGEC$_{NLI}$ (t_c: 0.98)	40.75	20.84	34.21
LMGEC$_{M}$ (t_c: 0.98, t_i: 0.92)	39.59	23.79	34.95
LMGEC$_{M+NLI}$ (t_c: 0.98, t_i: 0.92)	**40.97**	23.37	**35.61**

Table 4.1: Evaluation on CoNLL-14 shared task using MaxMatch(M^2)

System	GLEU
Bryant and Briscoe (2018) (t_c: 0.95)	48.75
LMGEC$_{NLI}$ (t_c: 0.94)	49.01
LMGEC$_{M}$ (t_c: 0.94, t_i: 0.88)	**49.80**
LMGEC$_{M+NLI}$ (t_c: 0.94, t_i: 0.89)	49.61

Table 4.2: Evaluation on JFLEG test set using GLEU

to note that incorporating NLI model to reduce semantic discrepancy improves grammaticality but harms fluency.

Table 4.3 shows the error type analysis in terms of three edit operations: missing, replacement, and unnecessary, depending on whether tokens are inserted, substituted or deleted respectively. It is observed that our proposed method perform reasonably well in correcting grammatical errors, especially for missing words. Moreover, by leveraging NLI model to reduce semantic discrepancy, the false positive of missing errors decreased from 176 to 132, resulting in significant improvement on precision and $F_{0.5}$. This indicates that the meaning preserving strategy plays an important role in successfully correcting missing word errors.

In this work, we used LMGEC$_{M+NLI}$ developed with CoNLL-2013 test set as *LessErrors*'s configuration setting.

LessErrors provides reasonably effective corrective feedback of grammatical errors for language learners. However, we note that for some error types the systems are less successful. Here, we show some unsuccessful correction and analyze the reasons behind these correction failures.

- **Spelling Error:** We observed that sentences containing misspelling words would sometimes lead to semantic noise. This might result in the corrective

System	Type	TP	FP	FN	P	R	$F_{0.5}$
	Missing	0	0	321	–	–	–
$LMGEC_{NLI}$	Replacement	464	727	1082	38.96	30.01	36.77
(t_c: 0.98)	Unnecessary	29	18	330	61.70	8.08	26.51
	Total	493	745	1733	39.82	22.15	34.34
	Missing	76	176	286	30.16	20.99	27.74
$LMGEC_{M}$	Replacement	467	744	1081	38.56	30.17	36.53
(t_c: 0.98, t_i: 0.92)	Unnecessary	32	19	335	62.75	8.72	28.02
	Total	575	939	1702	37.98	25.25	34.50
	Missing	71	**132**	287	**34.98**	19.83	**30.34**
$LMGEC_{M+NLI}$	Replacement	459	718	1083	39.00	29.77	36.72
(t_c: 0.98, t_i: 0.92)	Unnecessary	30	19	336	61.22	8.20	26.69
	Total	560	869	1706	39.19	24.71	**35.08**

Table 4.3: Error type analysis on CoNLL-2014 dataset with alternative annotations

candidate of the right answer being filtered out. For example, Table 4.4 shows that the system corrects the non-word *"pratical"* to *"piratical"* rather than *"practical"*, which is because the corrective candidate containing *"practical"* was filtered out by the NLI model. To address this issue, we could exempt spelling candidates from semantic preservation.

- **Replace Verb Error:** Using the wrong verb in a given sentence is the most common type of grammatical errors. However, it is challenging to solve such an error without using a collocation database. For example, Table 4.4 shows that the system fails to detect the verb misuse with repect to the noun *"education"*, where *"accept"* is not appropriate and *"receive"* would be a better choice. To correct this type of error, we could add common verb errors into confusion sets using datasets such as WikEd Error Corpus (Grundkiewicz and Junczys-Dowmunt, 2014).

- **Word Order Error:** Syntactic errors are difficult to correct using the correction-at-a-time strategy, especially for sentences containing interrogative words. For example, Table 4.4 shows that no error is detected in the

Error	Data
Spelling Error	Source: The teacher gave us many pratical suggestions. System: The teacher gave us so many piratical suggestions. Reference: The teacher gave us many practical suggestions.
Replace Verb Error	Source: Most people in Japan accept higher education. System: Most people in Japan accept higher education. Reference: Most people in Japan receive higher education.
Word Order Error	Source: Do you know what time is it? System: Do you know what time is it? Reference: Do you know what time it is?
Insertion Overcorrection	Source: I am writing a book yesterday. System: I was writing a book about yesterday. Reference: I was writing a book yesterday.

Table 4.4: The commonly seen correction failures of *LessErrors*

sentence *"Do you know what time is it?"*, where *"is it"* in this sentence should be corrected to *"it is"*. To correct such an error, a promising way is to leverage part-of-speech tagging and grammar pattern to generate corrective candidates.

- **Insertion Overcorrection:** The proposed filtering strategies significantly reduce the false positive rate. However, it is still easy to over-correct occasionally. Take the sentence in Table 4.4 for example. Despite the misuse of verb tense *"am"* being successfully corrected to *"was"*, the system additionally inserts the preposition *"about"* before *"yesterday"* and changes the meaning. To address this issue, we could learn a missing word insertion model that is able to determine not to add any word if the input sentence is already good enough.


TSING HUA UNIVERSITY

Chapter 5

Conclusion and Future Work

We have presented a GEC system that applies a language model based correction strategy to provide suggestions for correcting grammatical errors in a given sentence. The method involves generating and ranking corrective candidates using confusion sets, missing word recovery, and training an inference model to maximize the LM probability and correct grammatical errors. The result shows that the proposed method performs reasonably well in correcting missing words, and outperforms previous work in CoNLL-2014 with the JFLEG test set. While recent GEC systems are based on NMT-based models with a huge amount of parallel data, we propose a method that only requires limited annotated data for tuning and is also applicable to GEC systems for other languages with limited annotated data (e.g. Japanese, Korean).

For future work, we could further explore a different candidate generation strategy with the goal of correcting other error types, such as replace verb errors (e.g. [*accept education* → *receive education*]). For this, we could add common verb errors into confusion sets using datasets such as WikEd Error Corpus. Additionally, we could use a more powerful neural language model such as GPT-2 (Radford et al., 2019) to improve grammaticality assessment, thereby exploring better correction results.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. *ArXiv*, abs/1508.05326, 2015.

Christopher Bryant and Ted Briscoe. Language model based grammatical error correction without annotated training data. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 247–253, 2018.

Christopher Bryant, Mariano Felice, and Ted Briscoe. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1074. URL `https://www.aclweb.org/anthology/P17-1074`.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, California, USA, June 1996. Association for Computational Linguistics. doi: 10.3115/981863.981904. URL `https://www.aclweb.org/anthology/P96-1041`.

Martin Chodorow, Joel Tetreault, and Na-Rae Han. Detection of grammatical errors involving prepositions. In *Proceedings of the Fourth ACL-SIGSEM*

Workshop on Prepositions, pages 25–30, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W07-1604.

Shamil Chollampatt and Hwee Tou Ng. A multilayer convolutional encoder-decoder neural network for grammatical error correction. *ArXiv*, abs/1801.08831, 2018.

Daniel Dahlmeier and Hwee Tou Ng. 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, Montréal, Canada, June 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N12-1067.

Rachele De Felice and Stephen G. Pulman. A classifier-based approach to preposition and determiner error correction in L2 English. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 169–176, Manchester, UK, August 2008. Coling 2008 Organizing Committee. URL https://www.aclweb.org/anthology/C08-1022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Marina Dodigovic. Artificial intelligence and second language learning: An efficient approach to error remediation. 2007.

Nadir Durrani, Helmut Schmid, Alexander Fraser, Philipp Koehn, and Hinrich Schütze. The operation sequence Model—Combining n-gram-based and phrase-based statistical machine translation. *Computational Linguistics*, 41(2):157–186, June 2015. doi: 10.1162/COLI_a_00218. URL https://www.aclweb.org/anthology/J15-2001.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP:

A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-2501. URL https://www.aclweb.org/anthology/W18-2501.

Tao Ge, Furu Wei, and Ming Zhou. Reaching human-level performance in automatic grammatical error correction: An empirical study. *CoRR*, abs/1807.01270, 2018.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. The wiked error corpus: A corpus of corrective wikipedia edits and its application to grammatical error correction. In *International Conference on Natural Language Processing*, pages 478–490. Springer, 2014.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. Detecting errors in english article usage by non-native speakers. *Nat. Lang. Eng.*, 12:115–129, 2006.

Kenneth Heafield. Kenlm: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pages 187–197. Association for Computational Linguistics, 2011.

Matthew Honnibal and Mark Johnson. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1162. URL https://www.aclweb.org/anthology/D15-1162.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1161. URL https://www.aclweb.org/anthology/D16-1161.

Ting-Hui Kao, Yu-Wei Chang, Hsun-Wen Chiu, Tzu-Hsi Yen, Joanne Boisson, Jian-Cheng Wu, and Jason S. Chang. 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, August 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W13-3603.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. An empirical study of incorporating pseudo data into grammatical error correction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1236–1242, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1119. URL https://www.aclweb.org/anthology/D19-1119.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. Automated grammatical error detection for language learners. *Synthesis lectures on human language technologies*, 3(1):1–134, 2010.

John Lee and Stephanie Seneff. Correcting misuse of verb forms. In *Proceedings of ACL-08: HLT*, pages 174–182, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P08-1021.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. Corpora generation for grammatical error correction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3291–3301, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1333. URL https://www.aclweb.org/anthology/N19-1333.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

A. Mani. Solving text imputation using recurrent neural networks. 2015.

Kathleen F. McCoy, Christopher A. Pennington, and Linda Z. Suri. English error correction : A syntactic user model based on principled " mal-rule " scoring. 1996.

Lisa N. Michaud, Kathleen F. McCoy, and Christopher A. Pennington. An intelligent tutoring system for deaf learners of written english. In *Assets '00*, 2000.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. Mining revision log of language learning SNS for automated Japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155, Chiang Mai, Thailand, November 2011. Asian Federation of Natural Language Processing. URL `https://www.aclweb.org/anthology/I11-1017`.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-2097. URL `https://www.aclweb.org/anthology/P15-2097`.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. Jfleg: A fluency corpus and benchmark for grammatical error correction. *arXiv preprint arXiv:1702.04066*, 2017.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 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, August 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W13-3601`.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-1701. URL https://www.aclweb.org/anthology/W14-1701.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://www.aclweb.org/anthology/P02-1040.

Jong C. Park, Martha Palmer, and Clay Washburn. An English grammar checker as a writing aid for students of English as a second language. In *Fifth Conference on Applied Natural Language Processing: Descriptions of System Demonstrations and Videos*, pages 24–24, Washington, DC, USA, March 1997. Association for Computational Linguistics. doi: 10.3115/974281.974296. URL https://www.aclweb.org/anthology/A97-2014.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

David Schneider and Kathleen F. McCoy. Recognizing syntactic errors in the writing of second language learners. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 1198–1204, Montreal, Quebec, Canada, August 1998. Association for Computational Linguistics. doi: 10.3115/980691.980765. URL https://www.aclweb.org/anthology/P98-2196.

Joel Tetreault, Jennifer Foster, and Martin Chodorow. Using parse features for preposition selection and error detection. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 353–358, Uppsala, Sweden, July 2010. Association

for Computational Linguistics. URL https://www.aclweb.org/anthology/P10-2065.

Trieu H. Trinh and Quoc V. Le. A simple method for commonsense reasoning. *ArXiv*, abs/1806.02847, 2018.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y. Ng. Neural language correction with character-based attention. *ArXiv*, abs/1603.09727, 2016.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Ng, and Dan Jurafsky. Noising and denoising natural language: Diverse backtranslation for grammar correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 619–628, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1057. URL https://www.aclweb.org/anthology/N18-1057.

Helen Yannakoudakis, Marek Rei, Øistein E. Andersen, and Zheng Yuan. Neural sequence-labelling models for grammatical error correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2795–2806, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1297. URL https://www.aclweb.org/anthology/D17-1297.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. Semi-supervised word sense disambiguation with neural models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1374–1385, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL https://www.aclweb.org/anthology/C16-1130.

Zheng Yuan and Ted Briscoe. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1042. URL https://www.aclweb.org/anthology/N16-1042.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 156–165, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1014. URL https://www.aclweb.org/anthology/N19-1014.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, 2015.