Misspelling Correction with Pre-trained Contextual Language Model

Yifei Hu
Computer and Information Technology
Purdue University
hu381@purdue.edu

Xiaonan Jing
Computer and Information Technology
Purdue University
jing@purdue.edu

Youlim Ko
Computer and Information Technology
Purdue University
ko102@purdue.edu

Julia Taylor Rayz
Computer and Information Technology
Purdue University
jtaylor1@purdue.edu

***Abstract*—Spelling irregularities, known now as spelling mistakes, have been found for several centuries. As humans, we are able to understand most of the misspelled words based on their location in the sentence, perceived pronunciation, and context. Unlike humans, computer systems do not possess the convenient auto complete functionality of which human brains are capable. While many programs provide spelling correction functionality, many systems do not take context into account. Moreover, Artificial Intelligence systems function in the way they are trained on. With many current Natural Language Processing (NLP) systems trained on grammatically correct text data, many are vulnerable against adversarial examples, yet correctly spelled text processing is crucial for learning. In this paper, we investigate how spelling errors can be corrected in context, with a pre-trained language model BERT. We present two experiments, based on BERT and the edit distance algorithm, for ranking and selecting candidate corrections. The results of our experiments demonstrated that when combined properly, contextual word embeddings of BERT and edit distance are capable of effectively correcting spelling errors.**

***Index Terms*—Spelling Correction, BERT, Contextual Language Model, Natural Language Processing**

I. Introduction

Language is a unique gift for humankind. It conveys information using rules with a certain amount of informality. Human can quickly adapt to the informality and develop new rules to make sense from the informality. However, understanding the informality in human language has been a challenging task for computer algorithms. Nowadays, many state-of-the-art Natural Language Processing (NLP) tools and architectures are still vulnerable against all sorts of informality.

A simple spelling error can greatly impact the output of a sophisticated natural language processing tool for tasks that enjoy very small error rates. For example, Named Entity Recognition (NER) is one of the common NLP tasks, which automatically identifies named entities from a given text input. The following example is taken from AllenNLP [1] ELMO-based [2] online NER demo:

"*this shirt was bought* ***at*** *Grandpa Joe's in LA*."

In this sentence, two named entities, *Grandpa Joe's* (as an organization) and *LA* (as a city), are identified. However, if the input is slightly modified by introducing a misspelled word — "*at*" changed to "*ta*" (which is a common misspelling for "*at*") — an utterance"*Grandpa Joe's*", which was previously identified as an organization, will not no longer be identified as a named entity. Instead, "*Joe*", by itself, will be identified as a person.

Similarly, spelling correction affects another common NLP task, the semantic role labeling (SLR). Consider the following example:

"*the keys, which* ***were*** *needed to access the building, were locked in the car*"

SLR model based on BERT (Bidirectional Encoder Representations from Transformers) [3] identifies "the keys" as the only argument in this sentence. However, if the first word "*were*" is misspelled as "*weer*", the misspelled "*weer*" would be captured as a second argument.

For human, the spelling errors above do not impact how the sentences are interpreted. However, both the ELMO-based model and the BERT-based model changed their outputs as a result of misspellings present in the sentences. Similar to the above examples, many current NLP models are not robust enough when processing flawed input data. However, corpora collected through human generated texts often contain errors such as spelling errors, incorrect usage of words, and grammatical mistakes. These mistakes can easily introduce noise to the corpora and affect NLP parsers used for subsequent tasks. In many cases, minimizing noise in the corpora by correcting unintentional human generated mistakes serves an important procedure in performance enhancement for machine learning models.

This paper focuses on contextual correction of misspelled words — one of the most common type of errors in natural language texts. Spelling errors can be roughly divided into two types, non-word error and real-word error [4]. The former type results in a misspelled mistake which is not an actual English word (i.e. "weer" in the previous example) and the latter results

in a mistake that is accidentally an English word (i.e. misspell "access" as "assess"), but does not work well in context. While the detection of non-word error can be employed through a look-up in a large English vocabulary dictionary, the detection of real-word errors is more challenging. Real-word error detection often requires training of expensive neural network models with a large gold standard corpora in order to locate the desired error. Thus, training a robust error encoder which detects both types of errors can be difficult. On the other hand, both the correction of non-word and real-word errors can share a universal framework as both errors are a result of unintentional representation of the intended word. For instance, both "assess" (real-word error) and "acsese" (non-word error) are two edit distance away from the intended word "access" in the previous example. Fundamentally, the efforts spent in correcting both misspelled words to the intended word "access" are identical. With this intuition, we introduce a simple approach to correcting spelling errors in this paper which utilizes the edit distance mechanism and the recently popular neural network model BERT.

BERT uses transformer [6] architecture to predict masked word given the context. The attention mechanism in the transformers allows the model to compute an attention score for each context word and subsequently use the attention score to predict the masked word. We are interested in examining BERT's ability to predict correct words given the masked spelling errors.

II. Related Works

A. Traditional Spelling Error Correction

Traditional spelling error correction approaches include noisy channel models and n-gram models. Church and Gale [5] presented a context-based system with a noisy channel model, employing a simple word bi-gram model and Good-Turing frequency estimation. A similar model from IBM Watson Research using tri-gram was proposed in [7]. Carlson and Fette [8] illustrated the effectiveness of n-gram models in spelling error corrections. They compared the n-gram models with the GNU Aspell [9] model on Google n-gram dataset. On non-word error corrections, n-gram model showed a 36%-49% improvements on top correct word ranking, with highest accuracy achieved at 92.4% for detecting insertion error with 5-gram. A recent study on real-time spelling error correction proposed a Symmetric Delete Algorithm (SDA) with n-grams approach [10]. A weighted sum of unigrams, bigrams, and trigrams was used as the metric for ranking SDA generated candidate words. The model was tested on 24 languages with spelling errors generated by manipulating edit distance and bi-gram probability, with a 97.1% accuracy on top English language errors. However, the model showed only 68% top correction accuracy when tested on public dataset such as Wikipedia, although it outperforms popular industrial models Aspell and Hunspell [11].

B. Neural Network Based Spelling Error Correction

Sequence-based neural network models have also been applied in spelling error correction tasks. Li et al. trained a nested RNN (Reccurent Neural Network) model with a large-scale pseudo dataset generated from phonetic similarity [12]. The model outperforms other state-of-the-art sequential models, scRNN [13] and LSTM-Char-CNN [14] – which combined long short-term memory (LSTM) and character-level convolutional neural networks (Char-CNN) – by 5% on precision and 2% on recall. Ge et al. trained a neural sequence-to-sequence model in grammatical error correction (GEC) [15]. Through fluency boost mechanism, which allows multi-round sequence-to-sequence corrections, multiple grammatical errors in one sentence can be corrected in one setting. However, such training often requires large amount of data to enhance the performance of the error encoder. Oftentimes, spelling error dataset with gold standard labels are small on size and generated errors can be drastically different from how human writers make mistakes. In a recent attempt, Shaptala and Didenko [16] proposed an alternative GEC approach using pre-trained BERT with fully connected network to detect and correct errors. However, the error type resolution given by the model does not consider the dependency between different errors and error types, and the decoder tends to mistakenly remove end of sentence tokens. The result of [16] again demonstrated the difficulties in detecting errors without type independence and the limitation of neural network decoders.

III. Dataset

The dataset used in the paper is a subset from the CLC (Cambridge Learner Corpus) FCE (First Certificate in English) Dataset [17], which consists of 5,124 sentences written by English as second language learners, from speakers of 16 native languages. CLC FCE dataset contains 75 types of annotated errors and corresponding corrections in the original essays. We extracted a subset which contains only spelling errors, annotated as "S", "SA", "SX" to represent "spelling", "American spelling", and "spelling confusion" errors respectively. A total of 2,075 sentences was extracted from the original dataset. Each extracted sentence was limited to contain only one misspelled word in order to provide BERT with the correct contextual information. TABLE I shows the distribution of the dataset. We used the Brown corpus (49815 vocabulary) from NLTK (Natural Language Toolkit) [18] to determine whether the error was a real-word.

TABLE I
DATASET STATISTICS

sentence statistics	average sentence length	20.68
	std of sentence length	11.00
	maximum length	1
	minimum length	99
error types	number of real-word errors	225
	number of non-word errors	1850

IV. Proposed Method

We propose two methods which combine the per-trained BERT language model and the edit distance algorithm. The pre-trained BERT model is capable of predicting a masked word and provide a list of candidate words that make sense in a given context. The edit distance algorithm can be used to find the words that are similar to the misspelled word. The detail of this mechanism will be explained in the following section.

A. Treating Misspelled Words As Masked Words

BERT provides a functionality of predicting a word in a sentence, marked as needed to be predicted, based on the context of a sentence. Due to this mechanism, known as masked word prediction, BERT only takes a sentence with one masked token (word) as input at a time and output a list of candidate words with probabilities accordingly from high to low. Here is an example of a typical masked word prediction process:

1) Raw sentence: "*How are you today?*"
2) Replace a word with a MASK: "*How [MASK] you today?*"
3) Predictions (with probability) of the MASK: "*are (0.88)*", "*do (0.04)*", "*about (0.03)*", "*have (0.003)*"

Based on this mechanism, it is possible to encode the misspelled words as masked words in each sentence and utilize the one to one prediction feature in candidate correction rankings. The number of candidate words is adjusted by a parameter N which can be as large as the vocabulary size. However, one limitation for the BERT model is that if the expected output is not in BERT's vocabulary, BERT will not be able to give the correct predictions that would match a gold standard. In reality, BERT is trained on various corpora with diverse vocabulary, which makes it very unlikely for BERT to encounter an out-of-vocabulary word. In this paper, we uses a pre-trained BERT-Large-Cased model [19] in candidate correct words generation.

It should be noted that BERT and similar architectures are not the only mechanisms of providing an unknown word based on context. An ontology-based approach was described by Taylor and Raskin [20] by using fuzzy membership functions of words based on the context of a sentence. The differences between human and computer processing of information in context was also touched on by Jing et al. [21].

B. Damerau–Levenshtein Distance

Edit distance [22] [23] measures the similarity between two words syntactically. In this paper, edit distance refers to the Damerau–Levenshtein Distance. There are 4 different operations: insertion, deletion, substitution, and transposition. Below are the examples for each type of operation:

- Insertion: "*wat*" $\rightarrow$ "*what*"
- Deletion: "*whaat*" $\rightarrow$ "*what*"
- Substitution: "*wgat*" $\rightarrow$ "*what*"
- Transposition: "*waht*" $\rightarrow$ "*what*"

Edit distance is calculated as the count of a minimum number of operations above needed to covert one word to another.

In this paper, edit distance is used in two different ways, comparison and generation. First, edit distance is used to compare the candidate corrections from the BERT prediction to the misspelled word. Ideally, the corrections should be very similar to the misspelled words in terms of edit distance. The candidate correction with the lowest edit distance is selected as the correct prediction of the misspelled word. It is possible that the lower edit distance is selected for a word that is also slightly lower on the BERT's list in terms of predictions than a word with a higher edit distance. However, since all BERT's predictions are contextual, and people tend to recover similar words faster, we choose to optimize edit distance rather than contextual ranking. Second, edit distance is used to find all similar words of a target token. Given a corpus, we can extract existing lexical units that are within K edit distance away from the given word and then test it with BERT. For instance, in the Brown corpus, all the words close to "annoying" with an edit distance under 2: enjoying, annoying. These two approaches can be seen as bidirectional: one takes BERT's interpretation of misspelled words and then applies edit distance to it; the other one applies edit distance first, and then checks whether it agrees with BERT's interpretation of what fits well in a sentence.

V. Experiments and Results

A. Experiment 1: Applying BERT before edit distance

In this experiment, we first apply the pre-trained BERT-Large-Cased [19] model to predict the masked misspelled words, then employ the edit distance to rank the candidate corrections. The BERT model outputs a list of N candidate words, where we chose N to be from 10 to 500 for this experiment. The edit distance between each candidate word and the misspelled token can be then calculated and ranked in ascending order. The word with the lowest edit distance is chosen as the final prediction. When multiple candidate words have the same lowest edit distance, the word with the highest prediction probability from BERT is chosen as the final prediction.

The result for experiment 1 is shown in Table II. Three metrics are applied in evaluating the accuracy of the corrections:

- **accuracy top@1**: measure how often the label matches with top 1 prediction.
- **accuracy top@N**: measures how often the label appears in the top N output of BERT.
- **P(top@1|top@N)**: measures the conditional probability for the edit distance to select top 1 correct prediction given the label appears in top N output of BERT.

With N = 500, 73.25% of the misspellings were corrected. According to Fig. 1, by increasing N, the final accuracy increased accordingly; however, the trend was slowing down. At the same time, as we increased N, the edit distance algorithm started to miss more correct answers from the candidate list.