LING 82100 Writing Systems
Final Project
Yuying Ren

English Spelling Error Detection and Correction with Contexts

Abstract

This project has built a misspelling detection and correction system for English from the scratch. Levenshtein Distance was used to calculate the orthographic similarities between the error and candidate words. Furthermore, a trigram model was built in order to make the prediction of the correction with inspecting the contexts.

Index Terms— *spelling error correction, non-word error, Levenshtein Distance, n-gram language model.*

1. Introduction

Spell checking and correction are the classic tasks in the field of natural language processing, they have a long history in NLP (Flor et al., 2019). With the advancement of technology, there are many text processing applications such as Microsoft Word, Ispell, Aspell etc, have complete these tasks well.
However, with the growth of online communication, online text data has enormously increased as well. The issue of misspellings becomes more ubiquitous and accordingly, spell checking and correction is considered as the fundamental step of text preprocessing in NLP. For example, misspelled query terms can significantly affect the performance of web searching results, Cucerzan & Brill (2004) has found that 10% to 15% queries are misspelled in their investigation of search engine's log data. Moreover, the issue of misspellings can have serious impact in some specific domains such as medical records (Nazir et al., 2018).

Flor & Futagi (2012) has suggested that misspelling can happened to both native English speakers and English Language Leaners, there are 6.5% spelling errors of all errors found in a US national sample of 3000 college composition essays. In addition, spell errors are usually divided in to two types: non-word errors and real word errors. For instance, in a sentence of '*Sally mode a pancake',* the word *mode* is a real word error because it is a real word in English but doesn't form any semantic meanings with its contexts in the sentence. Instead, if the word is not an English word like *mgde* or *madr,* then it is a non-word error. With the knowledge of spelling error above, this project intended to build a naïve system for non-word spelling error detection and correction. The purpose for this project is to establish a better understanding of English writing system and the computational method of dealing with English textual data.

2. Related Work

As Samanta & Chaudhuri (2013) has concluded, spelling errors is broadly classified in two categories: non-word errors and real word errors. This section will discuss two previous studies dealing with the detection and correction of both real word errors and non-word errors.
Samanta & Chaudhuri (2013)'s work dealt with the real word error detection and correction. They have generated a set of confusion words for each word in the test texts. Confusion words are selected from an English dictionary by calculating the

Levenshtein Distance between each word in the dictionary and the target word. Furthermore, they built a language model which retrieve the bigram and trigrams of each word in the text data. The bigram and trigram contexts are then used to rank the confusion words by referring the frequency of their cooccurrence with each of the confusion word. Therefore, the scores of each word in the ranking are calculated like as:

$$Score(W_j^i) = \lambda_1 P_1(W_j^i|W^{i-1}) + \lambda_2 P_2(W_j^i|W^{i+1}) + \lambda_3 P_3(W_j^i|W^{i-1}, W^{i+1})$$

(Samanta & Chaudhuri, 2013)

While Samanta & Chaudhuri (2013)'s language model is concentrating on the detection of real word errors, this project will adapt their method for the purpose of correcting the non-word errors. Instead of inspecting all the words in the texts, the model will only generate candidate words for the error words and predict the corrections by comparing their trigram contexts frequencies.

Flor et al. (2019) has released a benchmark corpus of English misspellings based on the TOEFL essays written by English Language Learnners. In addition, their context-aware approach with machine learning for spell correction has reached an accuracy of 88.12% and successfully applied to data in other domains such as medical records.
Flor et al. (2019)'s study has provided theoratical supports for this project. Their experiment has investigated 7 features (separated in two categories: contextual and non-contextual features) with Logistic Regression and Averaged Perceptron. Based on their finding that the orthographic similarity of non-contextual features and the n-grams (they used window size 4) of the contextual features are the two most informative features, this project will build an edit model to calculate the orthographic similarities and a language model to identify the trigrams contexts in order to find the most possible corrections.

3. Data

Two datasets are used for the spelling check and correction tasks in this project. An English word dictionary for detecting the non-word errors, which would be the out of vocabulary words of this dictionary. In addition, this dictionary is also used for providing candidates of the correction. Since it is difficult to find a free high-quality n-grams corpus, this project also built a trigram corpus using a text data, again, this text data is also used for error simulation and evaluation.
The word dictionary data used in this project is a free open source of English vocabulary on Github called List Of English Words which contains 370k single words. The text data used for building trigram corpus and evaluation is The International Magazine Vol.III, No.2, which contains 41 articles that are written by various authors. This text data has 140k words in total, generated a corpus contains 170k trigrams and is divided into 1,456 small text files for the evaluation.

4. Method

4.1. Error Detection

This project focuses on the detection and correction of non-word errors of English. Although The task of detecting non word errors is simpler than detecting real word errors, it still can be problematic depending on the quality of the text data. For error detection and correction, 20 text files are randomly selected, which contains 4.5k words in total and yet some words are not English nor in the standard form.
The non-English words will be identified by the error detection model and simply ignored by the error correction system in this project. However, the ill formed words in the texts are vary from each other. Some words contain numbers, or non-white

space separators like underscores, hyphen etc. Some of the words are proper names contain uppercased letters. These problems can be solved with some text normalization methods such as regular expression. Some words are lack of separators between each other, which cannot be solved by regular expressions would be ignored and pass to the error detection, error simulation and error correction models.

4.2. **Candidate Selection**

There are two steps of the error correction part of this Project. First, an edit model compares the orthographic similarities between words in lexicon and each of the errors. For the sake of convenience, this model will choose words are only 1 edit distance from the error word and form a candidate set for each error. The edit distance is determined by the algorithm of Levenshtein Distance (minimum edit distance). Levenshtein Distance (1966) is a method that compares the similarity between two strings by calculating how many edit operations (including insertion, deletion and substitution) are needed to convert one string to the other. One example of the error *aave*'s candidate set would be {*ave, dave, have ... tave, wave*}, and words like *drave, made* will not be included.
The construction of edit model followed Jurafsky & Martin (2020)'s algorithm of Levenshtein Distance.

4.3. **Error Correction**

The numbers of candidates for error correction are significantly reduced by the one edit Levenshtein Distance model, which is a preparation for extracting the context information. Researchers have the flexibility of experiment with different n-grams of the contexts, Flor et al. (2019) inspected the 4-grams of the error word's contexts, Samanta & Chaudhuri (2013) combined the probabilities of bigram and trigrams. This project will examine the trigram context, which are the words on the left and right side of the error.
Each of the candidates would be combined with contexts and their frequencies would be recorded from the trigram corpus and ranked for the final decision of the correction. Candidate with the highest frequency would be the most possible correction to the error. This idea comes from the intuition that words with higher frequency in general are more likely to be misspelled, more importantly, they will have more unique trigram pairs and higher frequency of each pair.
An example of the procedure is as below:

Input sentence:
*We **aave** the portrait of George W. Kendall.*
↓
Extract the trigram context of: (*we, aave, the*)
↓
Rank the candidates by their frequency with the contexts in the trigram corpus:

Candidates with contexts	Frequency
(*we, have, the*)	1.8e-5
(*we, gave, the*)	6e-6
(*we, dave, the*)	0
…	0

In this case of the error *aave*, only two candidates have the same contexts with it, the rest candidates-contexts pairs that are not found in the trigram corpus will be assigned to a value 0. Since *have* the is highest frequent word in the trigram data, it will be the final decision of the correction.

5. Evaluation

5.1. **Error Simulation**

A proper corpus for evaluation of spell checking and correction is not easy to find. Some studies have adopted the writing samples from English Language Learner, like essays from TOEFL test in

Flor et al.(2019). Other researchers have created their own data by simulating errors with selected words in the texts. This project has adopted the second method.

The simulation model includes two steps. First, an error generation function will select words from the original texts to create the error. Similar to the edit model, there are three edit operation in this step: insertion, deletion and substitution, each operation will insert, delete or substitute one character in the word with another random character from the alphabet. With an iteration, one of the three operations would be randomly applied to words until they become a non-word error. Second, an error insertion function will put the errors created back to the original texts.

5.2. **Measurements**

While the evaluation of error correction is straight forward: compute the percentage of correctly corrected words of the total real errors detected, the evaluation of error detection can be implicit due to the complexity of the text data. As described in the Data section, the non-English words and ill formed words in the text data can affect the performance of error detection and correction.

In order to solve this issue, the statistical measurement, F-score is adopted to evaluate the error detection function, where the F-score is calculated as below:

$$F\text{-}score = 2 \times \frac{precision \times recall}{precision + recall}$$

The precision and recall are calculated as:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ negatives}$$

In this case, the TP would be the number of errors correctly detected (errors are simulated and detected), TP + FP is the number of errors detected in total and TP + FN is the number of errors actually generated by the error simulation model.

6. Result

The text data contains 140k words in total and was randomly divided into 1,456 small files. For the evaluation, 20 of them (4.5k words in total) are randomly selected. Furthermore, 246 errors in total are generated by the error simulation model and 240 of them are correctly detected, in addition, the number of words that are over detected is 64. Therefore, the precision and recall of the error detection are calculated as below:

$$precision = \frac{number\ of\ correct\ detections}{number\ of\ total\ detections} = 78.75\%$$

$$recall = \frac{number\ of\ correct\ detections}{number\ of\ errors\ simulated} = 97.56\%$$

Moreover, the F-measure of the error detection is:

$$F\text{-}score = 2 \times \frac{precision \times recall}{precision + recall} = 86.28\%$$

The error correction model has ignored the over detected errors and reach to an accuracy of 95.41%. The reasons for this relatively accuracy could be that only a small portion of texts are used for the test. More importantly, the test data are extracted from the same text data set of building the trigram corpus, which is a relatively small data set for an error detection and correction task.

Conclusion

This is an experimental project of building and evaluating a system of error detection and correction. The experiment focused on the dealing with the non-word errors and adopted methods of Levenshtein Distance and n-gram model to select the candidates and predict the correction of the spelling errors in the texts.

The evaluation of the system is based on a small text data that contains simulated non-word spelling errors. 246 errors are generated in total, 240 of them are correctly detected and there are 64 over detected words. Therefore, the F-score of the error detection is 86.28%. In addition, there are 11 miss corrections of all the 240 corrections has made by the system and the accuracy of correction is 95.41%.

This project has limited the edit model to obtain the candidate words from the lexicon that are only one edit distance from the error word. Although most of the misspellings are only one edit distance from their correct form (Flor et al., 2019), a more convenient and efficient method is needed for the edit model to select the proper candidates. The current trigram model was built on a relatively small trigram corpus that created from the same dataset for the test, most of the candidates couldn't find their trigram pairs in the corpus, therefore, a larger trigram corpus should improve the performance of the error correction model.

References

Samanta P. & Chaudhuri B. A Simple Real-word Error Detection and Correction Using Local Word Bigram and Trigram. Proceedings of the 25th Conference on Computational Linguistics and Speech Processing (ROCLING 2013), pages 211-220, 2013

Flor M., Fried M. & Rozovslaya A. A Benchmark Corpus of English Misspellings and A Minimally-supervised Model for Spelling Correction. Proceedings of the 14th Workshop on Innovative Use of NLP for Building Education Application, pages 76-86, 2019

Cucerzan S. & Brill E. Spelling correction as an iterative process that exploits the collective knowledge of web users. Proceedings of EMNLP'04, pages 293300, 2004.

Lai H., Topaz M., Goss R. & Zhou L. Automated misspelling detection and correction in clinical free-text records. Journal of Biomedical Informatics, page 188-195, 2015

Flor M., Futagi Y., Lopez M., & Mulholland M. Patterns of misspellings in L2 and L1 English: a view from the ETS Spelling Corpus. In Learner Corpus Research: LCR2013 Conference Proceedings, volume 6 of Bergen Language and Linguistic Studies (BeLLS), pages 107–132. 2015

Levenshtein VI. Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady, No. 10,707-10, 1966.