Spelling Correction

Yiheng Shu*

*Software College, Northeastern University, China

July 2019

1 Introduction

This report is for *Homework 1: Spelling Correction* of *DATA130006: Introduction to Natural Language Processing*.

The project attempted to correct the spelling of a piece of English news text. The types of spelling errors are mainly divided into **non-word errors** and **real word errors**.

For non-word errors, the words that need to be checked are compared to all words in the vocabulary, and words that are not in the vocabulary must be non-word errors. The rest of the words appear in the vocabulary, which may be real word errors or correct words.

The algorithms involved in this project are mainly the noisy-channel model and the n-gram language model.

2 Algorithm

From the perspective of code implementation, the flow of program execution is as follows.

1) Preprocess. The program downloads nltk Reuters corpus and count the corpus for n-gram. In the program, preprocessing involves loading the confusion matrix and vocabulary from external files.

2) Error statistics and positioning. The number and location of non-word errors are determined for each sentence based on the number of sentence errors and the vocabulary provided by the test data. If the number of incorrect words in a sentence is equal to the number of non-word errors judged by the program, then we can assume that all other words except these non-word errors are correct. Otherwise, if the number of incorrect words in the sentence is greater than the number of non-word errors determined by the program, the program also needs to handle the real word error.

3) Noisy channel model. In this model, the goal is to find the intended word given a word where the letters have been scrambled in some manner. Given an

alphabet Σ, let Σ^* be the set of all finite strings over Σ. Let the dictionary D of valid words be some subset of Σ^*, i.e., $D \subseteq \Sigma^*$. The noisy channel is the matrix $\Gamma_{ws} = P(s|w)$,

where $\omega \in D$ is the intended word and $s \in \Sigma^*$ is the scrambled word that was actually received.

4) N-gram language model. Considering both the model effect and the computational performance, the program uses the Bigram model, which is a form of n = 2 in the n-gram model. Bigrams help provide the conditional probability of a token given the preceding token, when the relation of the conditional probability is applied:

$$P(W_n|W_{n-1}) = \frac{P(W_{n-1}, W_n)}{P(W_{n-1})}$$

3 Implementation Details

In the process of algorithm implementation, there are some details that need to be considered. These problems may not be considered in the design of the algorithm, but they occur during program debugging.

3.1 The Integrity of Vocabulary

The first is the integrity of the thesaurus given by the title. For checking for non-word errors, the larger the lexicon, the better. The lexicon given by the title has a total of more than 48,000 lines. Excluding the repetition of uppercase and lowercase words, it actually contains more than 20,000 common words.

In natural language, vocabulary does not always appear as a root. Nouns may be plural, and verbs may be in the form of third-person singular forms, past participles, and present participles. Each word may be followed by punctuation marks such as commas, periods, and question marks, as well as suffixes required for grammars such as tenses and possessives.

The given vocabulary includes nouns with capitalized initials, nouns in plural, and third-person singular forms of verbs. When the program judges a non-word error, it first removes the suffix of the word and converts it into the form contained in the vocabulary. After finding the word with the highest probability in the vocabulary, the separated suffix is added to the end of this corrected word.

Some common words are not included in the vocabulary, such as *won't*, *don't* and *can't*, so they are manually added to the vocabulary.

3.2 Confusion Matrix

A confusion matrix is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one. This program directly uses the confusion matrix provided by [1]. The four confusion matrices are (1) del[x,y], the number of times that the characters xy (in the correct word)

were typed as x in the training set; (2) add[x,y], the number of times that x was typed as xy; (3) sub[x,y], the number of times that y was typed as x; (4) rev[x,y], the number of times that xy was typed as yx.

3.3 Edit Distance Implementation

In this task, the Damerau-Levenshtein distance is used to calculate the minimum number of operations for a word to be modified to another word. The types of operations include insertion, deletion, single-character substitution, and transposition of two adjacent characters. After discovering that the edit distance calculation function directly implemented in Python (without speeding up tools such as numpy) is inefficient, the program uses *pyxDamerauLevenshtein* package of PyPI to calculate the edit distance. It implements algorithm for Python in Cython for high performance, and it runs in $O(N * M)$ time using $O(M)$ space.

4 Experiments

4.1 Experiment Setup

In summary, the test data used in this experiment contains a thousand sentences selected from the news text, mainly containing non-word errors and containing a small number of real word errors.

The corpus used in this experiment is the Reuters news corpus provided by *nltk* package. In addition to the Reuters corpus, the Gutenberg and the Brown corpus was also tried, but was found to be ineffective in improving the model effectiveness.

4.2 Experiment Result

In 1000 sentences, if the error type is not distinguished, the total correction accuracy is about 70.3000%. Detailed results can be found in the output file attached to the source code repository. The program execution takes only a few minutes. For most sentences, the correction is instantaneous, and the longer sentence correction will not exceed several seconds.

References

[1] Mark D Kernighan, Kenneth W Church, and William A Gale. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th conference on Computational linguistics-Volume 2*, pages 205–210. Association for Computational Linguistics, 1990.