CmpE 493
Spring 2020
Assignment 1: Spelling Error Corrector

Gurkan Demir
2015400177

March 13, 2020

Contents

1 Introduction

In this assignment, we are expected to implement an isolated word spelling error corrector based on the noisy channel model. We are given set of files which are;

1. ***corpus.txt :*** Used for finding word frequencies.
2. ***spell-errors.txt :*** Used for constructing confusion matrices according to error types.
3. ***test-words-misspelled.txt :*** Example set of misspelled words for testing.
4. ***test-words-correct.txt :*** Example set of corrected versions of misspelled words for testing.

While creating a dictionary, we are expected to tokenize the file and perform case-folding. During correction of misspelled words, we are expected to find candidates with edit distance 1 according to Damerau-Levenshtein.

Also, we are expected to implement two versions of this corrector. First one is calculating probability of candidates with the help of language and noisy channel model without smooting. Second one is using add-one smoothing (Laplace smooting with alpha = 1).

Our program is expected to take a file containing a list of misspelled words (one word per line) as input, and produce a file with the predicted correct spellings of these words (one word per line) as output. If our program can not produce predictions for any of the words in the input file, the corresponding lines in the output file should be printed as blank lines.

2 Implementation

1. Read corpus, tokenize and construct dictionary.
2. Read spelling errors, and construct confusion matrices.
3. Get misspelled word, and find all words in corpus with edit distance 1.
4. Using language and noisy channel model, calculate probability of each candidate.
5. Return most probable candidate.

After performing case-folding while reading corpus, tokenization is done. Tokenization is performed by replacing each non-alpha character with space.

3 How to Run?

In order to run error spelling corrector, execute the following from the command line:

python3 corrector.py –corpus [CORPUS_FILE] –spell_errors [SPELL_ERRORS_FILE] –misspelled [MISSPELLED_FILE] –correct [CORRECT_FILE] –smooth

where;

1. **CORPUS_FILE :** Path of corpus.txt (***required***).
2. **SPELL_ERRORS_FILE :** Path of spell-errors.txt (***required***).
3. **MISSPELLED_FILE :** Path of misspelled words file (***required***).
4. **CORRECT_FILE :** Path of correct words file (***not required***).
5. **smooth :** Whether use alpha smoothing or not (***not required***).

3.1 Notes

1. Algorithm prints corrected versions of misspelled words in a file named output.txt.
2. Output file is located in the same directory with the execution.
3. CORRECT_FILE is not required while execution.
4. It is required when calculating accuracy is needed.
5. smooth is not required.
6. If you execute algorithm with smooth, it implements alpha smoothing. Otherwise, it does not implement smoothing.

Details of starting execution is mentioned in ReadME file.

4 Screenshots of Running System

5 Assumptions

- All words in corpus are spelled correctly.
- Empty string is returned as output for misspelled words which have edit distance more than 1.
- Laplace smoothing is implemented using alpha = 1.

6 Outputs

6.1 Confusion Matrices

6.2 Accuracy of System

7 Results

7.1 Analysis

According results of test misspelled words, there are some wrong corrections that my system produces. Those wrong can be divided into 3 as:

1. Misspelled word's edit distance is more than 1.
2. Correct version of misspelled word does not exist in corpus.
3. Misspelled word exists in corpus.

Due to the fact that we are expected to find errors with edit distance 1, first class of errors can be neglected.

Because of the fact that, corpus contains all words that we know, corrected versions of misspelled one must exist in corpus. However, some of the corrected versions do not exist in corpus, so my system can not produce output, or can not produce true output.

Since we assume all words in corpus are spelled correctly, third class of errors can be ignored, hence they did not misspelled.

7.2 Improvements

There are various ways to tokenize a file. My system implements tokenization by only replacing non-alpha characters with blank. In this case, tokenization is not done perfectly. We need to care about the words that contains apostrophe etc like ***shouldn't***.

Moreover, our dictionary is created according to unigram language model. In unigram language model, it is hard to estimate correct versions of misspelled words, since we have no idea about the words that before or after misspelled ones.

Also, context plays a significant role in spelling error correction. If there exists a way to understand what text means, we can produce more related outputs.