IDS 703 HW Assignment II
Submitted by Song Young Oh & Jiahui (Heather) Qiu
September 20, 2022

Assignment Objectives: Build a spelling corrector based on the Levenshtein distance. Document some specific cases in which the Levenshtein-distance spelling corrector works well and in which it works poorly. Why is it working poorly and how could it be improved?

Our Python script for spelling correction includes four parts. In the first section, we wrote the function of Levenshtein distance. We started by measuring the length of two words (w1 and w2), noted as w1_len and w2_len. Then we created an edit distance matrix with the size of (w1_len +1) by (w2_len + 1) and populated the matrix with the minimum edit distance. In the second section, we imported the linked txt file, which includes all words and their respective frequency of use. In the third section, we defined the spelling corrector function by asking it to sort the corpus based on their edit distance to a particular word. In the last section, we passed a list of test words to the spelling corrector function.

The spelling corrector worked well on some test words such as “honsty”, “pamily”, “fredom”, “amfibian”, and “countrary”. It accurately corrected them into “honesty”, “family”, “freedom”, “amphibian”, and “contrary”, respectively. The behavior aligns with our expectations because the correct words have the minimum edit distance to the test words among all the other words in the vocabulary.

However, the spelling corrector worked poorly on other test scenarios. For instance, it failed to correct misspelled words such as “noticable”, “mashine”, “wonderfull”, “beutiful”, “devlopment”, and “sence”. Our code merely returned the test words as-is because they are also part of the corpus, resulting in a Levenshtein distance of 0. Therefore, when the spelling corrector calculated the minimum edit distance, it returned the same misspelled word. The spelling correction at times also returned non-word characters when provided input. A case in point is “jyv” returned “jv”. Neither combination of letters represents a valid English word.

Additionally, our spelling corrector only evaluates a word by calculating its minimum edit distance to the vocabulary data set. It, however, cannot consider the context. For instance, a test word like “griss” returned as “gross” based on Levenshtein distance, but we argue that “greece” might be a better correction given its pronunciation. Another example is when we input “johnedw” the spelling corrector returned “joined” due to the minimum edit distance of 2 based on the Levenshtein algorithm. However, one might reason that “johnedward” is a better match in the context of a name, even though it has a Levenshtein distance of 3 from “johnedw”.

There are several ways to improve the results of our spelling corrector. First, we can start by referencing the corpus to an actual English dictionary and removing combinations of letters that are not words. This step will resolve the issue of our spelling corrector returning the misspelled word or even returning something that is not a word. Second, we can set an arbitrary frequency threshold to remove non-word characters in the vocabulary. However, the downside of doing this is that we might remove valid words in the process. Last, we can incorporate advanced algorithms to Levenshtein distance by taking context into consideration.