Mini project 3 report

Caleb Doiron and Jade Gonzalez

May 2020

1 Introduction

The objective was to create an efficient spell check and predictive text system. We wanted to create a system that can suggest viable options for the words you are using as you write. Our system has 2 distinct modes for this: read mode and write mode. Read mode will read in a file you provide and automatically make any spelling corrections that it can make and then write it out to a file. Write mode will allow you to type sentence-by-sentence and the system will suggest possible next words as you type or spelling corrections and write each finished sentence to file. You can use an exit command to exit the current mode.

2 Algorithm

In general, the algorithm will assume that a letter starting the sentence is capitalized. If a word is in app capital letters, the algorithm assumes it is an acronym and leaves it unchecked. Words that start with a number are ignored. If there are apostrophes or any such symbol after the word, it is ignored and then reapplied to the corrected word.

2.1 Correcting One Word

The algorithm checks if a word is real. If it is not real it is likely incorrect. The algorithm will generate words one error off and test which ones are actual words. It will also do so for words two edits off as well if needed. If the word is real and is not the first word in the sentence, the algorithm checks for context using the previous words to make it make logical sense. If it does not, then the algorithm will act as if it were incorrect and make suggestions. Otherwise, the algorithm assumes it really is correct and leaves it alone.

2.2 Finding the Intended Word

Probability of an incorrect word is predicted with following considered:
Amount of edits - finds words that are one edit away but will also look for those two edits away if there are no correct or low probability results for one edit.

Word pair frequency - word pair frequency of the surrounding words.
Word frequency - The frequency of the individual word.
Neighboring keys - if the key inserted or replaced is a neighbor of the intended key.
Viability of the original is also considered with the frequency of each trigram and bigram averaged off.
In choosing the best suggestions, first the algorithm considers the words that are one letter key off on the keyboard. If they exist, these should be the most likely errors.
Next, the algorithm ranks each word from 0 to 10000 on frequency and word pair frequency on the surrounding words. If suggestion has a transpose edit or a deletion/replacement of a letter with close keyboard approximation to the intended or surrounding letter, the score is multiplied by a factor of 10. This is under the assumption that small mistakes from 'fat fingering' or incorrect orders of pressing keys are the most likely mistakes. It chooses the word with the largest score sum as this would be the most likely candidate for the intended word. If candidate words have a low score around zero, the viability is checked if its under the a under a predetermined score, the user is queried on what word was intended. The user can keep the original, choose a new one or cancel that word. The ideal viability score was concluded by testing words that would not appear in the archives.

2.3 Predicting the Next Word

The algorithm checks for context and will suggest the next word based on word pair frequency. The word it predicts will be picked from the most possible word that could come next in the current sentence being written. If a user enters a line of text without ending the last sentence with punctuation, the program assumes the user wants to choose from a selection of likely choices. The choices are considered and ranked by searching a large text file with multiple books and novels. The algorithm begins by searching for the all the words in the current sentence in that specific order. When there is a match it records the word next word. If two words have the suffix "'s" they are considered a match. If if there are 15 words or more found, the words with the highest count are chosen to be printed. If multiple words have the same count, the word with the highest frequency in the English language is chosen. If there are less then 15 words found the program does the same process but narrows the scope by removing the first word in the sentence until 15 are found or there is no words left. If this process was unable to find 15 likely words the rest are filled with the most common words in the English language like "the" and "it". If the program is in writer mode, each word is written to a file specified by the user and the current sentence with corrections are printed to the screen. If the program is in reader mode, the user will not be prompted for unfinished sentences.

3 Conclusion

The final program works very well except for some biases and wait times for predicting of large sentences. Some recommended outputs seem to be focused more around wars and emperors which we had improved by including free romance novels from the gutenburg project in the archive text file. Some improvements that would be helpful would be a better system for ranking candidate words and classification of words to improve the prediction process. If we were able to filter out specific names from the novels, the results would be less biased. It would also be helpful to incorporate the words frequent words surrounding an individual words that are not just the immediate words before and after a target word.