Spelling Correction Homework

Authors: Erika Fox and Raza Lamb

Our spelling corrector is too inefficient to run on the entire text, so to examine how it performs, we ran it on the first 131 lines of the corrupted version of Jane Austen's *Sense and Sensibility*. From this output we were able to identify 4 different types of outcomes that our code produced, in comparison to the desired outcome. Below we document each of these cases, give a specific example, and if relevant, propose a potential solution.

Properly Corrected Errors

In this case, our code performed exactly as intended: first it identifies an error in the corrupted text, and identifies a potential correction. We can then compare this correction to the original Sense and Sensibility to confirm whether or not the correction was appropriate. Below are several cases in which our code performed as designed, with the corrupted word on the left and the corrected word on the right.

- "estete" → "estate"
- "thfs" → "this"
- "attacsment" → attachment
- "hveart" → "heart"

Missed Errors

Here, our code failed to identify corrupted words, and left them unchanged. Some examples are included below, with the corrupted word on the left, and the correct word on the right.

- "i" → "in"
- "Norlad" → "Norland"
- "nd" → "and"
- "o" → "of"
- "Daswood" → "Dashwood"
- "hi" → "him"

Within this case, there are two subcases: first, cases in which the corrupted word is in our dictionary of English words, so the code does not identify it as a mistake. In this case, in order to correct this behavior, we would need to add a grammatical component to our code, checking to see if the word grammatically fits in the flow of the sentence. Or, we could implement a word predictor that could help us choose the most likely word if there are multiple possibilities of replacement words like there are in this case.

The second case demonstrated above is proper nouns. Our code is designed to ignore any capitalized words, because our dictionary is unlikely to contain these words, and we would end up doing more harm than good. However, in order to fix this, we would need to include proper nouns relevant to the book to our dictionary. This could include places, common names, etc.

Improperly Corrected Errors

In this case, our code properly identified corrupted words, but did not implement the correct solution. We can identify these cases similarly to how we identified properly corrected errors. Below are several examples, with the incorrect word on the left, the word our code replaces it with on the right, and the correct word in parentheses.

- "inheritkr" → "inherited" ("inheritor")
- "sorid" → "solid" ("sordid")
- "wifl's" → "will's ("wife's")

While this error case is less frequent than the others, it still occurs. It mainly appears to occur due to the dictionary that we are using. The dictionary we are using is modern, and organized by usage in modern language. Our code would be more accurate to use a dictionary representing the english language in the time in which Sense and Sensibility was written. This type of error would also be helped through the addition of a grammatical component to the model, as mentioned earlier.

Error Introduction

This is the most frequent error our code produced. Our code incorrectly identifies an uncorrupted word, and replaces it with a new word. A few examples are included below, with the uncorrupted word on the left and the word our code replaced it with on the right.

- "respectable" → "respective"
- "acquaintance" → "assistance"
- "nephew" → "new"
- "economically" → "economic"

These cases are almost entirely caused by the dictionary we are using. Again, this dictionary is modern, while the text is not. Usually a more appropriate dictionary in this setting would resolve most of these errors.

Corner Cases

Having covered the main four categories of changes, we want to cover just one special case in which our code does not work as desired. The first example is words which are at the beginning of sentences. Our code does not attempt to correct any word that is capitalized, which includes words at the beginning of sentences, in addition to proper nouns. One potential way to correct this would be to build code that can differentiate between proper nouns and words that begin a sentence, such as by building a regular expression that does well at generally recognizing names, or inserting code that recognizes when a word is starting a sentence. However, this proved to be complicated in this instance, as if we use periods as identifiers, our code will correct words that follow "Mrs.", for example. Additionally, proper nouns can begin a sentence, which adds further complexity.