**CSCI 517: Assignment 2: You Write I Fix**

**Dataset used to create language model:**

It is a concatenation of public domain book excerpts from Project Gutenberg and lists of most frequent words from Wiktionary and the British National Corpus.

**Language model:**

Frequency of words is calculated using Counter from collections. Program uses unigram language model. Probability of a word, P (word) is calculated by counting the number of times each word appears in a dataset divided by total number of words in dataset. Unigram model is used because it is simple to implement as it does not consider history and works well for non-word errors.

**Smoothing Function:**

Smoothing function is not used in program.

**How I measured the cost of edits:**

A simple edit to a word is a deletion (remove one letter), a reverse (swap two adjacent letters), a substitution (change one letter to another) or an insertion (add a letter). The function *edits\_with\_cost\_1* returns a set of all the edited strings (whether words or not) that can be made with one simple edit. Corrections that require two simple edits are also calculated using *edits\_with\_cost\_2*. Both the sets can be big sets. So to get smaller sets, words are restricted to words that are known—that is, in the dataset.

**Error model of program:**

All known words of edit distance 1 are more probable than known words of edit distance 2, and less probable than a known word of edit distance 0. So function *candidate\_words* produce the first non-empty list of candidates in order of priority:

1. The original word, if it is known; otherwise
2. The list of known words at edit distance one away, if there are any; otherwise
3. The list of known words at edit distance two away, if there are any; otherwise
4. The original word, even though it is not known.

**What program covers?**

Program is based on unigram language model. It uses simple edit distance algorithm. Program corrects non-word errors. Program takes input file path from user and generates corrected output file with corrected words in brackets.

**Limitations of program:**

Program does not use large data set such as COCA n-gram or Google n-gram. Quality of those datasets is better than the dataset used in program. So it may happen that program fails to generate correct spelling for particular word because that word was not in the dataset used. Program does not use bigram language model. So it fails to correct real world errors. To correct real word errors we need previous word to predict current word which is not possible in unigram language model. Program does not guarantee that it will generate correct word for misspelled word. For example, she is good *acress*. Program generates across and not actress. This problem would have solved if bigram model was used.

**Improvement in program if given more time:**

I could have used COCA n-gram dataset. It does not have unigrams. So I would have also calculated unigram frequencies from given bigram model of COCA. I would have used Bigram model to solve real world errors. Further while calculating probabilities, I would have used K-smoothing. For calculating scores of candidate words, I would have used conditional probabilities and cost matrices.