“You Write I Fix”

Torumoy Ghoshal

**Language Model**

**Choices:** Google N-grams, COCA n-gram, text files compiled by Peter Norvig (big.txt, count\_2w.txt), names.txt

**What I used**

I started with Google N-grams. However, this corpus is huge, and my program became too slow. I decided not to use Google n-grams even though it would help the model become more accurate. I finally used big.txt compiled by Peter Norvig, and generated counts on my own. I also used count\_2w.txt and names.txt files for different parts of the program.

**Smoothing Function**

I have not implemented a smoothing function directly. If an unknown word is encountered, all possible variations are generated, and invalid variations are pruned. In this way, it is very unlikely that at least one candidate word will not be found given the size of the corpus. The exceptions are names and all capital words (e.g. DC). If there exists a word with the first/all letters capital in the middle of a sentence, it is most likely to be a name( e.g. DC, Paul). These words are not corrected. The digits are also kept unchanged.

**Methodology**

The whole methodology can be broken down into two parts.

1. Non-word correction: Candidate words are generated by finding valid words from all possible variations of a word. The words are then sorted based on their edit distance. In most cases, a word within 1 edit-distance will be found. After sorting, the first word will be chosen.

I used a file ‘names.txt’ collected from the internet. If a word exists in that file, that word is excluded from correction.

1. Real-word correction: This part is much harder than non-word correction. For example, in the phrase, “Until there days ago”, all words are spelled correctly. However, intuitively we understand that the writer of the sentence didn’t mean to write “there.” This can be corrected using a bigram model. Using my model, the two likely outcomes are:
   1. Until three days ago
   2. Until these days ago

Both outcomes make more sense subjectively than “there days.” However, in this context, “three days” is the correct choice. Given the random choice, I have found “three” to appear two times out of three times the program is run.

For the bigram model, I used the file count\_2w.txt compiled by Peter Norvig. It contains 286358 bigrams, and their counts. If the count of the original bigram (created from original words in the input file) is less than 10, and if there is a bigram created from the variations from the original words that has a count > 100000, then the newer bigram is chosen.

Some very popular words like a, the, too etc. are excluded from bigram calculation.

**Cost measuring:**

I used the cost matrices as stated in the paper <http://www.aclweb.org/anthology/C90-2036>. Using the cost matrices for insertion, deletion, substitution, I implemented a dynamic programming algorithm that finds the Levenshtein Distance between two words. In most cases, a word within one distance is found. The cost matrices ensure that the distance is not their alphabetical distance (e.g. a,b,c etc), but real word spelling mistakes (e.g. insertion cost between a and b is not 1 anymore).

**Limitations of the program:**

* As mentioned before, the program is likely to choose between “three” and “these” instead of “there.” If randomness was not introduced, the program would always choose “these.” This is because “these days” has much higher bigram count than “three days.” I cannot think of an easier way to solve this problem without using trigram or n-gram with higher numbers. I also do not think unigram probability would solve this problem.
* Another likely scenario is that the word “campagn” might be corrected with either “camplaign” or “campan”. Although this happens rarely, this is also caused by the randomness of the program.
* My program cannot correct “too (two) federal charges” and “as well.” I think this problem would also be solved by using a trigram model.

**What I would improve if more time was given:**

* The program runs a bit slow. It may take 5~10 mins to complete for the given input text. I would like to improve the speed if more time was given.
* I would like to implement the trigram model.
* I would use the Google n-gram model and find a fast way to implement it.