EDAN20 - Assignment 0

Oliver Cosic

September 2021

1 Introduction

There are basically four parts to the program by Norvig that I will comment on. These are: selection mechanism , candidate model , language model and error model. Together these parts makes the Spelling corrector work properly.

2 Selection mechanism

When correcting a word , the selection mechanism chooses the candidate with the highest combined probability. This is done by taking all the candidates and using max('argmax') in Python.

3 Candidate model

The candidate model basically tells us which candidates to consider as corrections for the word in question.

We use the function *edits1* to get a set of all strings that can be created with **one** edit. That is with either deletion(remove one letter), transposition(swap two adjacent letter), replacement(change one letter to another) or insertion (add a letter). Note that this returns string wheter they are words or not so one can imagine that the set will be huge.

But since we only want to correct words with words that actually exists we can restrict ourselves to words that are in the dictionary and then the set becomes much smaller. This is what the function *known* is used for.

The program also considers corrections that require two edits. As one can imagine this gives a even bigger set but usually only a few are known words.

Figure 1: Showcases how the number of words in the sets described above

4 Language model

The program estimates the probability of a word with the function P. This function takes in the number of times a word is used in a text and divides it by the total number of words in that same text. So the function *words* breaks text into words, then the variable *WORDS* holds a Counter of how often each word appears. And then with P it estimates the probability of each word. For example the word *the* returns a 7% probability.

5 Error Model

The program uses an error model that says that all known words of edit distance 1 are infinitely more probable than known words of edit distance 2, and infinitely less probable than known words of edit distance 0. The function *candidates* uses this and produces a list of candidates in order of priority:

- The original word if its known
- The list of known words at edit distance 1
- The list of known words at edit distance 2
- The original word even if it is not a word

Each candidate at the chosen priority will have the same probability according to this model.