Language Technology - EDAN20
Lab Session 0 - Fall 2020

Spell Checker - Python

Hicham Mohamad (hi8826mo-s)

September 20, 2020

1 introduction

Spell-checking is the task of predicting which words in a document are misspelled. Correction is the task of substituting the well-spelled hypotheses for misspellings. In this assignment, we need to run and comment the spell checker implemented by Peter Norvig in [2].

2 probabilistic approach

When there is no way to know for sure the corrected word, it is usual to use probability theory and in particular **Bayes theorem**. The problem is to find the correction c out of all the possible candidate corrections that maximizes the probability that c is the intended correction, given the original word w. Since $P(w)$ is the same for every possible candidate c, we can factor it out in Bays equation, giving:

$$\arg max_{c \in candidates} P(c/w) = \arg max_{c \in candidates} P(c)P(w/c) \tag{1}$$

In this equation we can distinguish four parts:

1. Selection Mechanism: $argmax$
2. Candidate Model: $c \in candidates$
3. Language Model: $P(c)$
4. Error Model: $P(w/c)$

3 Programming approach

The same four parts in the probabilistic approach can be mapped in the Python program:

1. Selection Mechanism: in Python `max` with a `key` argument to make *argmax*
2. Candidate Model: a **simple edit** to a word is a deletion (remove one letter), a transposition (swap two adjacent letters), a replacement (change one letter to another) or an insertion (add a letter), as shown in the function `edits1()`.
3. Language Model: the probability of a word, `P(word)`, can be estimated by counting the number of times each word appears in a text file of about a million words, `big.txt`, as represented in the function `words()` and the variable `WORDS`.
4. Error Model: it is defined a trivial, *flawed error model* that says: all known words of edit distance 1 are infinitely more probable than known words of edit distance 2, and infinitely less probable than a known word of edit distance 0, as shown in the function *candidates()*.

4 Instructions about the implementation

The program for running the spell corrector is implemented in Python. In the following skeleton, we describe the methods used in the program to solve the problem of spell-checking and auto-correction.

1. `correction()` tries to choose the most likely spelling correction.
2. `edits1()` returns a set of all the edited strings (whether words or not) that can be made with one simple edit.
3. `edits2()` consider corrections that require two **simple edits**. This generates a much bigger set of possibilities, but usually only a few of them are known words. We say that the results of edits2(w) have an edit distance of 2 from w.
4. `words()` breaks text into words.
5. `P()` estimates the probability of each word.
6. `candidates()` produce the first non-empty list of candidates in order of priority.
7. `word()` restrict ourselves to words that are known, that is, in the dictionary, then the set is much smaller.

5 Appendix

References

[1] Pierre Nugues, *Language processing with Perl and Prolog*, 2nd edition, 2014, Springer.

[2] Peter Norvig, *How to Write a Spelling Corrector*, `http://norvig.com/spell-correct.html`

[3] P. Nugues, Language Technology - EDAN20, Lectures notes: `https://cs.lth.se/edan20/`

[4] VanderPlas, A Whirlwind Tour of Python. O'Reilly 2016. Online reading: `https://jakevdp.github.io/WhirlwindTourOfPython/`

[5] Tf-Idf measure: `https://en.wikipedia.org/wiki/Tf-idf`.

[6] Geron, Jupyter notebook on linear algebra from Machine Learning and Deep Learning in Python using Scikit-Learn, Keras and TensorFlow: `https://github.com/ageron/handson-ml2`