Natural Language Processing – Assignment #1

Assignment description

In this assignment you will perform two tasks. Both make use of a large English dataset -- download a subset of 2018 Wikipedia dump (large well-formed English corpus) via this link -- https://drive.google.com/file/d/15H-pg40Epx2u6GE14vquCdF_-VlOgOG7/view?usp=sharing. These are 10M rows from the full dataset that should suffice for the purpose of this assignment.

Task #1: testing the Heaps' law in natural language

1. The python file `plot_heaps_zipf_laws.py` contains a function plotting log-log scale Zipf's distribution. Run it and make sure you see the expected output for `plot_zipf_law()`, and understand the code producing it.
2. Implement a new function that plots the Heaps' law for the dataset you downloaded. Does the plot fit your intuition? Copy the plot into a document to be submitted for this assignment.

Task #2: statistical language models – solving a cloze

This task deals with solving a cloze: a short text where certain words were removed, and you're required to fill in the missing items from a given list. Cloze is a popular task in language proficiency level assessment, and requires understanding of words and their context.

Input:

(1) a large well-formed English corpus

(2) a file with cloze: a short text where removed words were replaced with "__________"

(3) a file with words removed from the text (in a random order) – use these words to solve the cloze

(4) a file with top-50K most frequent English words in the given Wikipedia sample – if willing to build a single LM that will serve you for multiple clozes, you will need to use the lexicon

Output: print to the console a list of words in the order they are assigned to placeholders in (2)

For example, this input paragraph:

```
The wooden bridge, dating from the Middle Ages, across the Aare was destroyed by floods three __________ in thirty years, and was replaced with a steel suspension __________ in 1851. This was replaced by a __________ bridge in 1952. The city was linked up to the Swiss Central Railway in 1856.
```

And the list of words: `bridge, concrete, times`

The output should be (a list with words):

```
['times', 'bridge', 'concrete']
```

You are given two files: `config.json` and `main.py`, where `config.json` contains corpus, input cloze, candidates and lexicon files location. You can change these locations in `config.json`, but not the code parsing it in main. Your task is to implement the function `solve_cloze()`:

```
def solve_cloze(input, candidates, lexicon, corpus):
      # todo: implement this function
      ...
```

The accuracy of the solution will be estimated as the ratio of correctly assigned words out of total.

Question: Do you have any estimation of the accuracy of a random word selection?

One way to estimate that is via experiment: generate 100 random solutions for the given cloze and compute their mean accuracy: you are supposed to get a very low number, meaning the **chance accuracy** is very low. Write down to the document the chance accuracy you got for the candidate list in the assignment. Compare the chance accuracy to the accuracy you got for your solution – although not perfect, it should be much better than chance.

Comments:

- Although generally desirable, you are not required to perform tokenization and lemmatization in this assignment due to runtime constraints. Working with large enough data, we're hoping that the quantity makes up for the quality in this case.
- Assume that the words used to solve a cloze appear in the large corpus you are using.
- Note that you cannot expect a perfect result: the task is hard, we are limited to the set of tools we learned so far, and runtime constraints. However, you will get accuracy much higher than chance.
- It's recommended to try out your code on additional input files, not only the attached one. Create your own examples to test various cases! Keep the number of missing words small, e.g., 10.
- You can (and encouraged) to break down your code into additional functions, for good design and readability. Document your code with comments where needed.
- Make sure your code runtime does not exceed 20 minutes.

Submission

Submit a single zip file – assignment1_ xxxxxxxxx_xxxxxxxxx.zip , where “xxxxxxxxx” stands for a student id. Please specify two student ids (your and your partner’s). It should include two files:

(1) A document with Heaps’ law plot for task #1, and the chance accuracy for random solution you got for task #2. No need to submit you code for task #1.

(2) Your implementation for task #2 (including computation of chance accuracy) -- `main.py`.

Grading criteria include: correctness (the major part), code design and readability.

Good Luck!