[COM6513] Lab 3: Statistical Language modelling

Instructor: Nikos Aletras

The goal of this lab (**not assessed**) is to implement three languge models to perform sentence completion, i.e. given a sentence with a missing word to choose the correct one from a list of candidate words. The way to use a language model for this problem is to consider a possible candidate word for the sentence at a time and then ask the language model which version of the sentence is the most probable one.

The sentences to be completed together with the candidate words are in this file: questions.txt. The word to be completed is denoted with '_____' while the pair of candidate words is at the end of the line (e.g. weather/whether). The character ':' between the sentence and the candidates is not part of the sentece. To apply a language model on a sentence for a given candidate word, you just need to replace '_____' with the candidate word.

The texts to train your language models are in this file: news-corpus-500k.txt (70MB), which is a small subset of the 1 Billion Word Benchmark. The text has already been tokenized and split into sentences (each line represents a sentence).

You will implement three language models for this task:

- unigram
- bigram
- bigram with add-1 smoothing, i.e. Laplace

Your code should be executable by running:

```
python lab3.py news-corpus-500k.txt questions.txt
```

and present for every question in 'questions.txt' the result of each of the three language models. You are advised to lowercase the text and remove punctuation (hint: Python's `string` module defines punctuation). You are not allowed to use any third-party libraries (e.g. NLTK, Spacy, etc.) to train your language models.

Tip: You can use a dictionary data structure to store the probabilities between ungigrams, and unigrams and bigrams, e.g. p['cat'] ($P(cat)$) or p['sat', ('cat')] ($P(sat|cat)$). Remember to append start ('<s>') and end ('<\s>') symbols before the first and last words of a sentence respectively.

- Compute the accuracy of each model on completing the sentences considering that: - in case a language model returns only 0 probabilities, its answer should be considered incorrect - in case of a tie with non-zero probabilities, its answer should be considered half-correct
- What is the best LM? Are the results expected?

Try to implement the language models using: (1) Numpy 2-d arrays and (2) Scipy sparse matrices instead of Python dictionaries. Are these implementations faster than using Python dictionaries? How big are in the memory using float64 or float32 precision?