Programming Assignment 2 (due: 22 February 2022)

Design and implement a Python program called ngram.py that will learn an N-gram language model from an arbitrary number of plain text files. Your program should generate a given number of sentences based on that N-gram model.

Your program should work for any value of n, and should output m sentences. Before learning the N-gram model, convert all text to lower case, and make sure to include punctuation in the n-gram models. Separate punctuation from words before learning the N-gram model. Your program should learn a single n-gram model from any number of input files.

As a benchmark for performance, your program should be able to generate results for a trigram model (n=3) based on 1,000,000 words (tokens) of text in under five minutes.

Your program should run as follows:

```
ngram.py n m input-file/s
```

n and m should be integer values, and input-file/s should be a list of one or more file names that contain the text you are building your ngram model from. For example you could run your program like this :

```
ngram.py 3 10 pg2554.txt pg2600.txt pg1399.txt
```

This command should result in 10 randomly generated sentences based on a tri-gram model learned from these three files (which happen to be Crime & Punishment, War & Peace, and Anna Karenina from Project Gutenberg. You can find plain text versions of many literary works at Project Gutenberg (http://www.gutenberg.org) You may find it interesting to develop your program using works from an author you are familiar with and enjoy reading. You may use whichever files you wish from Project Gutenberg, but make certain the total number of tokens in all your files is more than 1,000,000.

Your program should be designed so that any number of input files (1 or more) can be processed by your program without modification. You should *not* hard code file names or the number of files your program is able to process.

Make sure that you separate punctuation marks from text and treat them as tokens. Also treat numeric data as tokens.

So, in a sentence like :

my, oh my, i wish i had 100 dollars.

You should have 12 tokens :

my , oh my , i wish i had 100 dollars .

Your program will need to identify sentence boundaries, and your ngrams should *not* cross these boundaries. For example, you could have input like this:

He went down the stairs

and then out the side door.

My mother and brother

followed him.

You should treat this as two sentences, as in:

He went down the stairs and then out the side door .

My mother and brother followed him .

To identify sentence boundaries, you may assume that any period, question mark, or exclamation point represents the end of a sentence. (In general this assumption is not correct in all cases, but is perfectly adequate for our purposes here. It is ok if your sentence boundary detection isn't completely accurate.) When generating a sentence, keep going until you find a terminating punctuation mark. Once you observe that then the sentence is complete.

If the length of a sentence in the input text file is less than n, then you may simply discard that sentence and not use it when computing n-gram probabilities.

Your program should output an informative message as a first line, stating what this program is and who is the author. Then it should output the value of the command line options, followed by the m sentences generated by your program.

For example....

```
%python ngram.py 2 10 pg2554.txt pg2600.txt pg1399.txt

This program generates random sentences based on an Ngram model.

Command line settings : ngram.py 2 10

[followed by 10 random sentences]
```

Please submit a copy of your source code (ngram.py) on the Canvas.

Note that you can download Project Gutenberg files via your web browser (http://www.gutenberg.org) or the wget command. You will want to save the .txt versions of files since the others (html etc) contain a lot of mark-up (html, etc) that would affect your models. You may use code as found in pythondoc, Learning Python, or Programming Python as a part of your assignments, however, this must be attributed in your source code. You may also use modules from CPAN if they are not NLP specific.

Make sure to review the programming assignment grading rubric to see how points will be distributed on each assignment. Your program should be commented such that I can understand the overall design and detailed workings of your program.