CSCI 6350 (Barbosa)
Project 2: N-Grams

Due: See course calendar for due date. May not be turned in late.

Assignment ID: **proj2**
File(s) to be submitted: **proj2.py, myout.log**

Objective(s): 1) Use n-grams for language modeling; 2) predict words using n-grams.

Project description:
In this assignment you will have the opportunity to recall what you've previously learned about n-grams and their use in modeling language. Write a small Python program that constructs bigrams and trigrams language models from a training corpus, and use the resulting models to predict words in a test corpus.

Requirements:

1. (**60%**) Your program must provide the functionality listed below, in a single file named ***proj2.py***:

- *Input* – The training data input file (available on the course calendar) should be read into appropriate variables. The test data will be read from a separate file and processed as indicated below. These files are gathered from over 1,700 CNN news stories. Each set contains a single sentence per line. The punctuation, possessives ('s) and negations (n't) have been separated from adjacent text to enable tokenization by using the Python string ***split*** method. Each line in the test file has an integer postfix. This integer is the index (zero based) of the token that is to be predicted.
- *Processing* – The processing requirements include:
  - Each line in the file is considered a sentence. N-grams should be extracted at the line/sentence level, and should not crossover line boundaries. Note: in the unlikely event that some lines contain more than one sentence, they should be handled as a single sentence in this assignment.
  - The data structure used to hold the tokens/words in each sentence should be augmented with the special start sentence ***<s>*** and end sentence ***</s>*** symbols, as required for each n-gram.
  - The extracted unigrams, bigrams, and trigrams should be kept in separate data structures (dictionaries indexed by tuples work well for this).
  - Once the n-grams have been extracted, the test file will be read for testing. The last element of each line (an integer) is the index of the token to be predicted. The context is taken from the sentence:

    Example sentence: ***My car is painted dark <u>blue</u> . 5*** - Since the (zero based) index for the target word is 5, the word being predicted is *blue* (shown underlined). The highest probabilities (frequency of occurrence) bigrams where the first word is *dark*, and trigrams where the first two words are *painted dark* are assessed.

- *Output* – The program should display the following output:
  - The number of unique unigrams, bigrams, and trigrams extracted
  - The number of correct predictions found within the top 1, 3, 5, and 10 bigrams and/or trigrams with the highest probabilities. The prediction is considered correct when either the bigram or trigram having the correct conditional context is followed by the word being predicted. See example output below:

```
Unique unigrams extracted: 25905
Unique bigrams extracted: 173178
Unique trigrams extracted: 306551
# of times correct word found in top 1 highest probability n-grams 719 of 4332 predictions.
# of times correct word found in top 3 highest probability n-grams 1128 of 4332 predictions.
# of times correct word found in top 5 highest probability n-grams 1323 of 4332 predictions.
# of times correct word found in top 10 highest probability n-grams 1599 of 4332 predictions.
```

2. (**15%**) Test your program – Re-arrange the data to experiment with different splits for training and test.

3. (**5%**) Code comments - Add the following comments to your code:

- A section at the top of the source file(s) with the following identifying information:

  Your Name
  CSCI 6350-001
  Project #X
  Due: mm/dd/yy

- Below your name add comments in each file that gives an overview of the program or class.
- Place a one or two line comment above each function that summarizes the workings of the function.

4. (**20 %**) Analysis – Provide a brief answer (as commented code) to the following questions:

   1. Why you believe the results are as output by your code.
   2. The test sentences are extracted randomly from the news stories used in the training data. A sentence is in either the training data or the test data, but not both. Discuss the pros and cons of this approach.
   3. What, in your opinion, is a better way to split the data into training and test components? Give reasons for your answer.

Project submission:

Log into the ranger desktop with your class student account. Once logged into ranger, go to the directory containing your Python program. At the Linux $ prompt, type in ls (for list) to make sure the Python script file is listed in this directory. Issue the following commands at the prompt to create a script file to be submitted with the Python code.

- *script myout.log*
- *pwd*
- *ls -l*
- *cat –n proj2.py*
- *python3 proj2.py*
- *exit*

The handin command allows assignments to be submitted electronically. The command handin must be followed by assignment ID and the files required. The files must be named exactly as listed here (although they may appear in any order). Submit your assignment from the directory where your source and log files are, with the following command (you may resubmit your work – it simply overwrites the previous submission. Your last submission will be graded).

handin proj2 proj2.py myout.log

Alternately, go to the course syllabus web page, and select the Gus icon near the bottom of the page's contents. Login with your class account username and password. Select the appropriate project from the dropdown menu, check the Submit button, and click the Perform Action button. Use the Choose File button to navigate to the location of the files. Please note that the filename is case-sensitive, and must be entered exactly as shown on the page. Click the Upload button to submit your assignment. You may submit a newer version of the assignment, all the way up to the deadline. It overwrites your previous submission (which is not recoverable). All project files must be included in each and every submission.