Homework 2: Ngram Language Model

Objectives:

- Gain experience creating ngrams from text
- Build a language model from ngrams
- Reflect on the utility of ngram language models

Turn in:

- The two programs described below should be written in Jupyter-style notebooks
- Print each run notebook to pdf and upload the 2 pdfs

Overview:

- In this homework you will create bigram and unigram dictionaries for English, French, and Italian using the provided training data where the key is the unigram or bigram text and the value is the count of that unigram or bigram in the data. Then for the test data, calculate probabilities for each language and compare against the true labels.

Instructions:

1. Notebook 1: Build separate language models for 3 languages as follows.
    a. create a <u>function</u> with a filename as argument; the function will:
        - read in the file text and remove newlines
        - tokenize the text
        - use nltk to create a bigrams list
        - use nltk to create a unigrams list
        - use the bigram list to create a bigram dictionary of bigrams and counts,
            - 'token1 token2' -> count
            - note that ('token1', 'token2') also works as a key
        - use the unigram list to create a unigram dictionary of unigrams and counts,
            - 'token' -> count
        - return both the unigram dictionary and bigram dictionary from the function
    b. call the function 3 times, once for each training file (you can hard code the file names), pickle the 6 dictionaries, and save to files with appropriate names. The reason we are pickling them in one notebook and unpickling them in another is that NLTK ngrams can be slow and if you put this all in one program, you may waste a lot of time waiting for ngrams() to finish

2. Notebook 2.
    a. read in your pickled dictionaries
    b. for each line in the test file, calculate a probability for each language (see note below) and write the language with the highest probability to a text file
    c. compute and output your accuracy as the percentage of correctly classified instances in the test set. The file LangId.sol holds the correct classifications.
    d. output your accuracy, as well as the line numbers of the incorrectly classified items

Upload your 2 print-to-pdf notebooks to eLearning, making sure that the print pdf shows the output of the code blocks

Hints for the programs:

Creating the dictionaries in Notebook 1:
You can use the NLTK ngrams() function to create a bigrams and a unigrams generator object. Then you can iterate over each to create the dictionary using Python's .count() string method to extract counts from the text you read in.

Calculating probabilities in Notebook 2:
The probabilities will be large enough so that you don't need to use logs, we will simply multiply the probabilities together. Each bigram's probability with simplified Laplace smoothing is: (b + 1) / (u + v) where b is the bigram count, u is the unigram count of the first word in the bigram, and v is the total vocabulary size (add the lengths of the 3 unigram dictionaries). The probabilities will be very near zero, that is expected. However small, one will be larger than the others.

Grading Rubric for the code:

Element	Points
Notebook 1	50
Notebook 2	50
Total	100