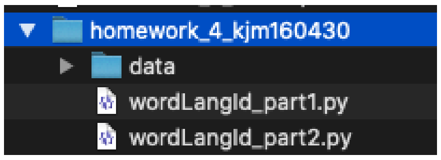
HLT – Fall 2021

Homework 4: Language Models

Objective: Use n-gram models for text analysis.
Turn in: your Python programs, zipped (just your 2 programs)

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.

Note: put the data files in a folder called 'data' in the same folder as your python files, like this:



homework_4_kjm160430
data
wordLangId_part1.py
wordLangId_part2.py

then use relative paths, like this:
'data/LangId.train.English'

Instructions:

1. Program 1: Build separate language models for 3 languages as follows.
    a. create a function with a filename as argument
    b. read in the text and remove newlines
    c. tokenize the text
    d. use nltk to create a bigrams list
    e. use nltk to create a unigrams list
    f. use the bigram list to create a bigram dictionary of bigrams and counts, ['token1 token2'] -> count
    g. use the unigram list to create a unigram dictionary of unigrams and counts, ['token'] -> count
    h. return the unigram dictionary and bigram dictionary from the function
    i. in the main body of code, call the function 3 times for each training file, pickle the 6 dictionaries and save to files with appropriate names. The reason we are pickling them in one program and unpickling them in another is that NLTK ngrams is slow and if you put this all in one program you will waste a lot of time waiting for ngrams() to finish.
2. Program 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 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

See Hints next page

Creating the dictionaries in Program 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 Program 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 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).