Programming Assignment

Due: Friday, March 8, 11:30 p.m.

Your task for this assignment is to build 3 N-gram language models using Python 3. Your language modeling program should accept two input files: (1) a training corpus file and (2) a test sentences file. Your program should accept these files as command-line arguments in the following order: yourname\_PA2.py <training\_file> <test\_file>

**Input File Format**

The training file will consist of sentences, with one sentence per line. For example, a training file might look like this:

I love natural language processing .

Boulder is a fun place to live !

You should divide each sentence into unigrams based solely on white space. Note that this can produce isolated punctuation marks (when white space separates a punctuation mark from adjacent words) as well as words with punctuation symbols that are still attached (when white space does not separate a punctuation mark from an adjacent word). For example, consider the following sentence:

“ N-grams are my favorite topic ” , she said .

This sentence should be divided into exactly eleven unigrams: (1) “ (2) N-grams (3) are (4) my (5) favorite (6) topic (7) “ (8) , (9) she (10) said (11) . The test file will have exactly the same format as the training file, and it should be divided into unigrams exactly the same way.

**Building the N-gram Language Models**

To create the N-gram language models, you will need to generate frequency counts from the training corpus for unigrams (1-grams) and bigrams (2-grams). An N-gram should not cross sentence boundaries. All of your N-gram tables should be case-insensitive (i.e., “the”, “The”, and “THE” should be treated as the same word).

1 You should create three different types of language models:

(a) A unigram language model with no smoothing.

(b) A bigram language model with no smoothing.

(c) A bigram language model with add-k smoothing. Have k = .0001

You can assume that the set of unigrams found in the training corpus is the entire universe of unigrams. I will not give you test sentences that contain unseen unigrams. So the vocabulary V for this assignment is the set of unique unigrams that occur in the training corpus. However, I will give you test sentences that contain bigrams that did not appear in the training corpus. For the bigram model without smoothing, the unseen bigrams should be assigned a probability of zero. For the last language model (c), you should use add-k smoothing to compute the probabilities for all of the bigrams. Use k = .0001.

Since the probabilities will get very small, you must do the probability computations in log space (as discussed in class). Please do these calculations using log base 10. You can use the log10 function in Python’s math module: **math.log10(x)**.

**Computing Sentence Probabilities**

For each of the language models, you should create a function that computes the probability of a sentence P(w1...wn) using that language model.

**Organizing your Code**

Please create the following functions so we can easily see how you computed the counts and probabilities:

GetUnigramCounts()

GetBigramCounts()

Or GetUnigramAndBigramCounts()

GetUnigramProbabilities()

GetBigramProbabilities()

GetSmoothedProbabilities()

Or GetBigramAndSmoothedProb()

GetUnigramSentenceProbability()

GetBigramSentenceProbability()

GetSmoothedSentenceProbability()

**Output Specifications**

Your program should print the following information for each test sentence. When printing the logprob numbers, please only print 4 digits after the decimal point. For example, print -8.9753864210 as -8.9754. If P(S) = 0, then the logarithm is not defined, so print logprob(S) = undefined.

Please print the following information, formatted like this:

S = <sentence>

Unigrams: logprob(S) = <number>

Bigrams: logprob(S) =<number>

Smoothed Bigrams: logprob(S) = <number>

For example, your output might look like this (the examples below are not real, they are just for illustration):

S = Elvis has left the building .

Unigrams: logprob(S) = -9.9712

Bigrams: logprob(S) = -12.2933

Smoothed Bigrams: logprob(S) = -8.4819

**Report**

I will post specific questions about your output that you should answer in a brief report.