Bigram Language Modeling

Goals  The main goal of this module is for you to implement and play around with a bigram language model, to get experience with these types of techniques and understand what this looks like.

Code Setup

The code you have contains the following files:

wiki.train.tokens
wiki.valid.tokens
BigramLanguageModel.java
Main.java

You can either run the project on the command line or using repl.it. In either case, you want to run Main.java.

Your Task

Question 1  Comment out everything below checkNormalization in main. Add the following lines of code to query the LM with the following arguments:

query_lm(lm, "I like to")
query_lm(lm, "I want to")

This will print out a list of the top words that can follow these contexts along with their probabilities. What do you notice about what the LM returns in these cases?

Question 2  Implement the getBestWord method. This method takes two arguments: first, the language model, and second, a context word to start with. It should return the highest probability next word. You can then use getBestSentence, which calls getBestWord, to generate a whole sentence from the language model.

You'll want to use getVocabulary, which returns an iterable set of words from the language model. You can then loop over all words in the vocabulary, compute the probability of each, and return the highest probability word. Hint: use two variables, one to track the best word seen so far, and the other to track its probability, so you can see when a new word has higher probability.

a)  Implement the code and test your implementation by commenting in the GETBESTSENTENCE call. You can print the probability of the word being picked at each step; try to verify that this is actually the highest-probability word each time.

b)  Try at least five different prefixes. What do you observe about the sentence completions?

Question 3  Implement sampleWord. This function takes two arguments: first, the language model, and second, a context word to start with. You can then use sampleSentence, which calls sampleWord to sample whole sentences.
A sampling algorithm is described in the videos. A Random r is declared in Main. You can use r.nextDouble() to get a random number between 0 and 1, and then iterate through the vocabulary and sum as you go to find the word corresponding to this random sample.

a) What happens if you samples from the same prefix repeatedly? How good are these samples?

b) What happens if you samples from the same prefix repeatedly? How good are these samples?

c) Try at least five different prefixes and see what you observe.

d) How do these sentences compare to the sentences from getBestWord?