Homework 6

Reading: Jurafsky and Martin, SLP 3 - 3.5

Ex1 and Ex2 Due Wed, Dec. 7, 17:00
Ex3 Due Wed, Dec. 14, 17:00

Submission Instructions

Submit Ex1 and Ex2 as usual. Due Wed, Dec 7, 17:00.

For Ex3, upload your implementation of bigram_model.py (don't forget to fill in the honor code), and your discussion of the perplexity value. Due Wed, Dec 14, 17:00.

Exercise 1 (3 pts)

(a) Using Figure 3.2 and the useful probabilities below it, calculate the probability of the sentence: *i want chinese food*

(b) Now suppose that the unigram count for the word *chinese* is 110 instead of 158. Assuming that the bigram counts in Figure 3.1 remain unchanged:

(i) What is P(chinese|food)?
(ii) What is P(food|chinese)?

Exercise 2 (4 pts)

Consider the following corpus:	And the test sentence:
I am Sam Sam I am I like Sam I do not like eggs	*I like eggs*

(a) (i) Calculate the unigram probabilities of the corpus
(ii) Find the perplexity of the test sentence using the unigram model

(b) (i) Calculate the bigram probabilities of the corpus
(ii) Find the perplexity of the test sentence using the bigram model

Exercise 3 (25 pts)

(a) (22 points) Implement the code to train a bigram model using the starter code (bigram_model.py) and tests (test_bigram_model.py) provided. Train a bigram model on sentences taken from Jane Austen's novel *Emma* in Data/ja-emma-train.txt, and calculate the perplexity of test sentences, different sentences taken from the same novel, in Data/ja-emma-test.txt.

(b) (3 points) Provide a brief discussion (max 10 sentences) of your results. What does the perplexity value tell you about your model? What could you do to be able to interpret it more meaningfully?

Implement the following functions in the starter code. You will find descriptions of the functions, their parameters, and return values in the documentation of each function.

- preprocess_sentence()
- load_data()

- get_unigram_counts()
- get_vocab_index_mappings()
- get_bigram_counts()
- counts_to_probs()
- to_log10()
- generate_sentence()
- get_sentence_logprob()
- get_perplexity()
- main()

Create a virtual environment with Python version 3.6 or above, and install **numpy**. Please do not import any additional packages.

The following variable/constants are defined near the top of bigram_model.py, which can be used anywhere in the code:

- rng (a random number generator)
- BOS_MARKER (beginning of sentence marker)
- EOS_MARKER (end of sentence marker)
- UNKNOWN (special token for OOV words)

Unit tests are provided for all of the functions except main(). The tests are not exhaustive, but if all tests pass it's a good indication that the code works as intended. Unit tests not only make sure your code is working, but they also demonstrate how a function should be called and what values should be returned. Have a look at them if you are unsure about the arguments or return values of a function.

For full credit, all tests must pass and your main() method must train the model and correctly calculate the perplexity of the test data. You must also provide a short discussion of the perplexity value you calculated.