LING 570: Hw3
Due date: October 16, 2025 11pm
Total points: 100

IMPORTANT REMINDERS:

- Your code needs to work on patas. The submitted version of your code/output files should run on patas, using the python 3.10 environment in the class folder. When invoking python in your .sh file, **USE THE FULL PATH** to your class's python environment.

A few notes about this assignment:

- All the examples are under **/mnt/dropbox/25-26/570/hw3/examples/.** The files under that directory show the correct file formats; however, the numbers in those files can be based on different input files than the ones used in this hw, and thus these numbers might be different from your own output files.
- Also, see the slides for Hw3 which we went over in class and are posted to the tentative schedule.
- In Q1 and Q3, when you read in the training or test data, each line is a sentence. You should break the sentence into tokens by whitespace (and add BOS and EOS markers if needed). Other than that, do not process the sentence further; for instance, do not run a tokenizer or lowercase the tokens etc.
- Before you run your code on big input files, you might want to test it on some small files first, such as examples/training_data_ex and examples/test_data_ex.

Q1 (10 points): write a script, **ngram_count.sh**, that collects unigrams, bigrams, and trigrams.

- The command line format is: ngram_count.sh training_data ngram_count_file

- The format of the training data: w1 w2 … w_n; that is, one sentence per line (e.g., **examples/training_data_ex**)

- The format of ngram_count_file is: count word1 … word_k (e.g., **examples/ngram_count_ex**).

    - The delimiter for the line "count word1 … word_k" can be either a single whitespace or a tab, and our auto-grader code will treat them the same way.
    - In the file, list lines for unigrams first, bigrams next, and then trigrams.
    - For each n-gram "chunk", sort the lines by frequency in descending order, and then break ties by sorting alphabetically.
    - Caveat: The example file **examples/ngram_count_ex** is sorted by frequency only, not by ngrams to break the ties.

- When counting ngrams in a sentence, remember to "insert" ONE (not two) BOS marker "<s>" and ONE (not two) EOS marker "</s>" in each sentence. For instance, if the input sentence is
  John call Mary

  You should count the ngrams as if the sentence is written as
  <s> John call Mary </s>

Q2 (15 points): write a script, **build_lm.sh**, that builds an LM using raw ngram counts:

- The format is: build_lm.sh ngram_count_file lm_file
- ngram_count_file is an input file produced by Q1.
- lm_file is the output file, and it follows the modified ARPA format, as on the hw3 slides (e.g., **examples/lm_ex**).
- When printing out the prob and lgprob numbers on each line, as long as the numbers in your system output are close (say within 3%) to the ones in the gold standard, we will regard them as the same.
- **examples/lm_ex** shows correct formats

Q3 (35 points): Write a script, **ppl.sh**, that calculates the perplexity of test data given an LM. For smoothing, use interpolation (weighted combination of ngram probabilities, as in the Hw3 slides).

- The format is: ppl.sh lm_file l1 l2 l3 test_data output_file
- lm_file is an input file as created in Q2. If a token w does not appear in the lm_file, it means w is unknown (i.e., an OOV word) and you increase oov_num by one and skip the term P(w | w1 w2) when calculating the probability of the sentence.
- Use interpolation to calculate probability: l1, l2, and l3 are lambda_1, lambda_2, and lambda_3 in the interpolation formula, respectively. They are non-negative real numbers and the their sum should be equal to 1.
- test_data has the same format as the training data (e.g., **examples/test_data_ex**)

- The format of output_file is presented in the Hw3 slide deck and example file (e.g., **examples/ppl_ex**)
    - Like in Q2, you do NOT need exactly identical numbers in the output_file (e.g., the prob, logprob, ppl).
    - The number of OOVs is just the number of unknown words in the sentence. To determine whether a word is OOV or not, just check whether the word has a line in the unigram model in the lm_file.
    - The file **examples/ppl_ex** shows the correct format.

Q4 (15 points) Use examples/wsj_sec0_19.word as training data and calculate the perplexity of examples/wsj_sec22.word by running the following commands. Fill out the table below and submit all the output files (see submit-file-list):

./ngram_count.sh 570/hw3/examples/wsj_sec0_19.word wsj_sec0_19.ngram_count

./build_lm.sh wsj_sec0_19.ngram_count wsj_sec0_19.lm

./ppl.sh wsj_sec0_19.lm 0.05 0.15 0.8 570/hw3/examples/wsj_sec22.word ppl_0.05_0.15_0.8

./ppl.sh wsj_sec0_19.lm 0.1 0.1 0.8 570/hw3/examples/wsj_sec22.word ppl_0.1_0.1_0.8

…

./ppl.sh wsj_sec0_19.lm 1.0 0 0 570/hw3/examples/wsj_sec22.word ppl_1.0_0_0

lambda_1	lambda_2	lambda_3	Perplexity
0.05	0.15	0.8	
0.1	0.1	0.8	
0.2	0.3	0.5	
0.2	0.5	0.3	
0.2	0.7	0.1	
0.2	0.8	0	
1.0	0	0	

The submission should include:

- The readme.[txt | pdf] file that includes the table in Q4, as well as any observations about challenges encountered and how you resolved them (or not).

- hw.tar.gz includes all the files specified in submit-file-list, which are:

- ngram_count.sh
- build_lm.sh
- ppl.sh
- wsj_sec0_19.ngram_count
- wsj_sec0_19.lm
- ppl_0.05_0.15_0.8
- ppl_0.1_0.1_0.8
- ppl_0.2_0.3_0.5
- ppl_0.2_0.5_0.3
- ppl_0.2_0.7_0.1
- ppl_0.2_0.8_0
- ppl_1.0_0_0