Assignment 3 Writeup

Murray Kang

- Problem 1.2

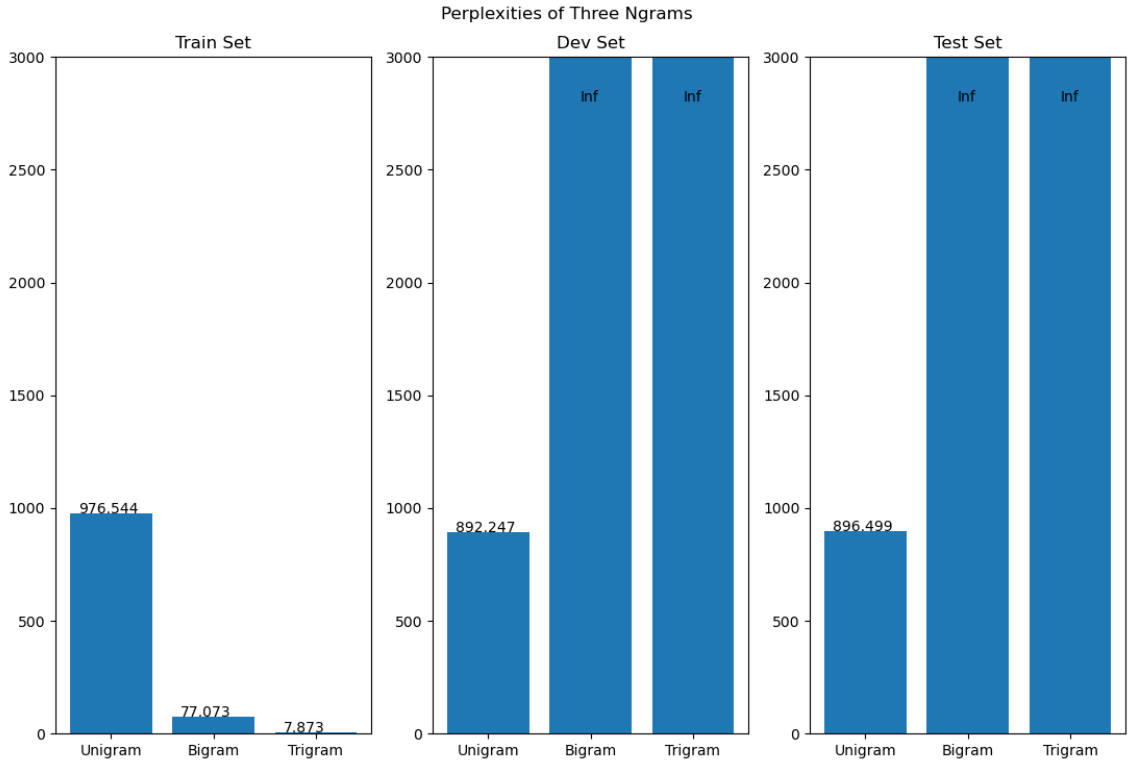
In the preprocess step, I firstly used a HashMap to store all words in the three documents, in which keys are words and values are the times of appearance. To UNK, I deleted the keys that have a value less than 3 and summed the values of all deleted words to the value of <UNK>.

For Unigram model, I also used a HashMap to store all unigram in the train set, in which the keys are unigrams and values are the times of appearance (nearly same as the dictionary I had in the preprocess step). Then, in the evaluation step, if there is a VOO, I would calculate its probability as the same of <UNK>. Then, cross entropy is the sum of all the probabilities of all unigrams in the file after log2 operations. To calculate the perplexity, I used 2**(- cross entropy / N~ (number of words in the file)).

For Bigram model, I also used a HashMap to store all bigrams in the train set, in which the keys are bigrams and values are the times of appearance. Then, in the evaluation step, if there is a VOO, I would calculate its probability as the same of <UNK>. But if there is a bigram that is not in the train set, I would directly return Inf, since if so, the probability would be zero and the perplexity would be infinite. If all bigrams are in the train set, then cross entropy is the sum of all the probabilities of all bigrams in the file after log2 operations. To calculate the perplexity, I used 2**(- cross entropy / N~ (number of words in the file)).

For Trigram model, I added two stop symbols at end of each sentence, and two start symbols at the beginning of each sentence. I also used a HashMap to store all trigrams in the train set, in which the keys are trigrams and values are the times of appearance. Then, in the evaluation step, if there is a VOO, I would calculate its probability as the same of <UNK>. But if there is a trigram that is not in the train set, I would directly return Inf, since if so, the probability would be zero and the perplexity would be infinite. If all trigrams are in the train set, then cross entropy is the sum of all the probabilities of all trigrams in the file after log2 operations. To calculate the perplexity, I used 2**(- cross entropy / N~ (number of words in the file)).

Below are the results of perplexities of three ngrams. From the graph, we can learn that although the trigram and bigram models have better performance in the train set, they are more easily to get infinite perplexity scores for the unseen data, which showed that a smoothing technique is needed for them.



- Problem 1.3

For this smoothed ngram model, I constructed three unigram count dictionary, two bigrams count dictionary, and one trigram count dictionary. For one of the unigram count dictionaries, I only added one stop symbol for each sentence. For the other one unigram and bigram dictionaries, I only added one start symbol and one stop symbol for each sentence. For the other dictionaries, I added two start symbols and two stop symbols for each sentence. Then, to calculate the probability of unigram model, I just used the first dictionary with no start symbol. To calculate the probability of bigram model, I used the two dictionaries with only one start and stop symbols. The probability of a given bigram (X, Y) would be the count (X, Y) / count(X). To calculate the probability of trigram model, I used the remaining three dictionaries with two start and stop symbols for each sentence. The probability of a given bigram (X, Y) would be the count (X, Y, Z) / count (X, Y). In the evaluation step, I firstly convert all OOV to <UNK>. Next, I calculated the log probability that is the weight sum of probabilities of three ngram models after log2. Finally, I used 2**(-log probability / N~ (# of words in the file) to get the smoothing perplexity score.

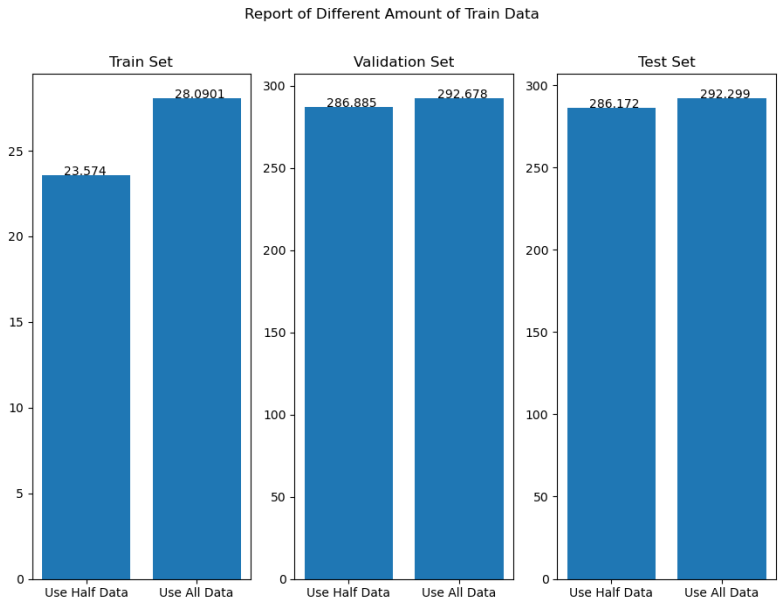
Below is the report of perplexities score of 4 different lambda choices. When lambda = (0.1, 0.3, 0.6), the perplexity score on the train set is 11.151, and that on the validation set is 352.234. Based on the validation perplexity score, I chose (0.1, 0.8, 0.1), and the test perplexity is **203.881**.

Perplexities of Different Lambdas

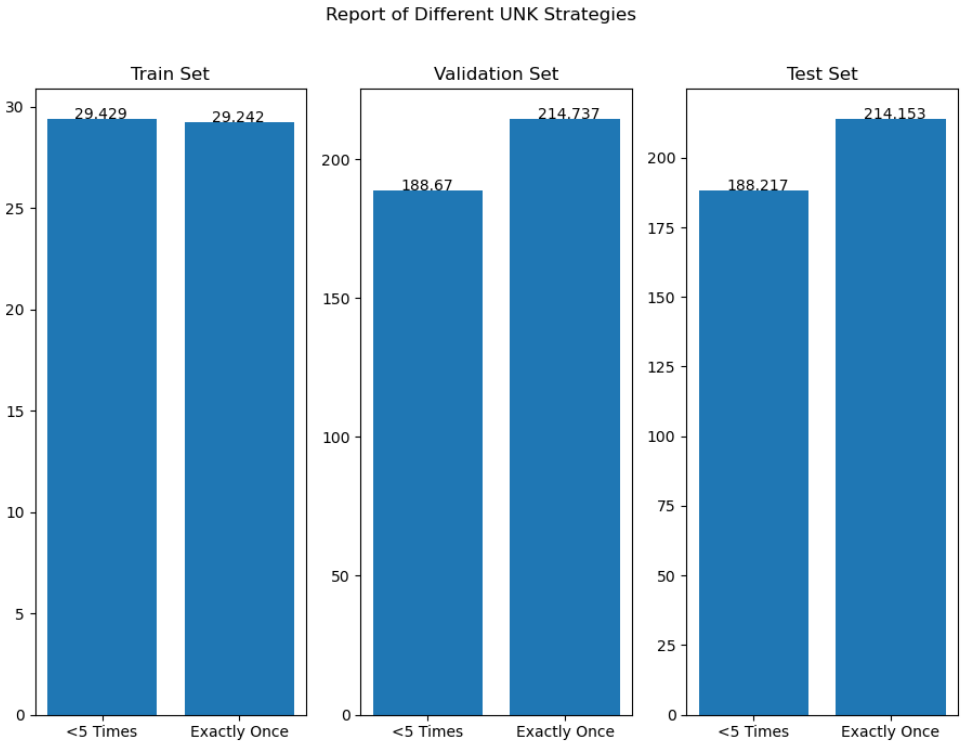
Lambda	Train Set	Validation Set
(0.1, 0.1, 0.8)	9.33	482.846
(0.1, 0.8, 0.1)	7.482	204.58
(0.8, 0.1, 0.1)	47.04	373.676
(0.1, 0.3, 0.6)	11.151	352.234

Below is repot of comparison experiment of using half of training data and all data.

I randomly selected half of lines in the training dataset and used the same hyperparameter to calculate the unseen data. As we can see, using half of training data would lead to **lower** perplexity score on the unseen dataset. The reason the <UNK> token represents an equivalence class of tokens. As less training data is used, this equivalence class increased. Because the meaning of the <UNK> token is changing, model perplexities are not directly comparable. Especially when the amount of training is not large enough, reducing the amount of data will rapidly increase the model probability of <UNK>, causing the entropy and perplexity of the model distribution to **decrease**.



Below the report of using different UNK strategies. Based on the result, UNKing the words that appearance less than five times would lead to less perplexity score on **validation dataset**.



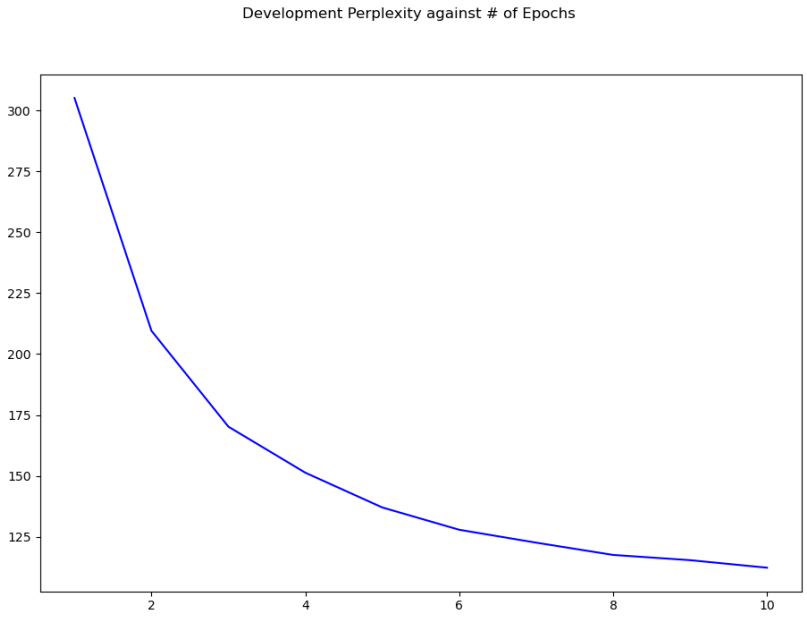
- Problem 2

I used a LSTM model. In my model, I used a Dropout layer and a LSTM layer, and an encoder for embedding and a linear decoder. After tuning hyperparameters, my chosen hyperparameters are embedding size = 650, number of hidden units per layer = 650, number of layers = 2, batch size = 20, learning rate = 5, maximum epochs = 50, sequence length = 35, dropout rate = 0.5, and gradient clipping = 0.5

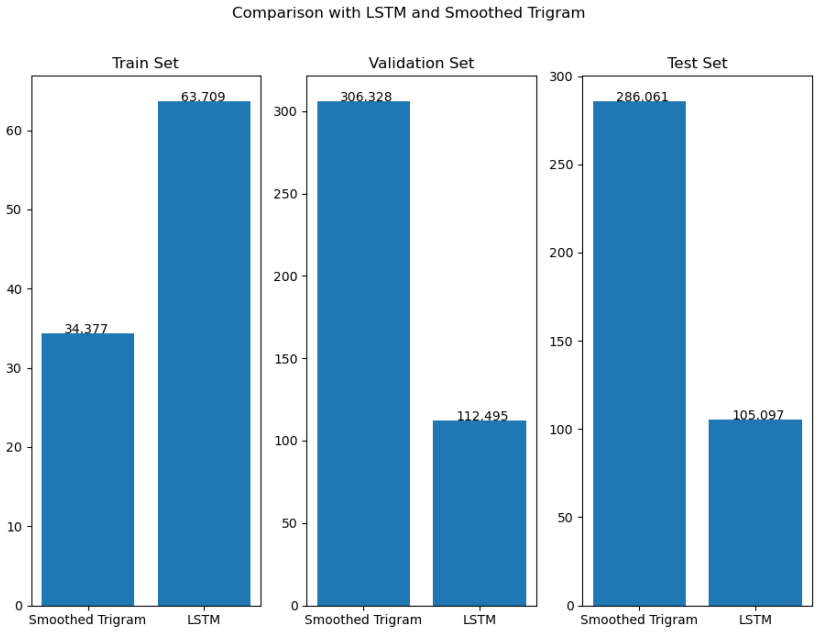
Firstly, I built up one word-to-index dictionary and one index-to-word dictionary. Next, I tokenized the train, valid, and test datasets, and store them in the two dictionaries. Next, I batchified the train, development, and test sets by evenly dividing the data across the batch-sized batches. In the training step, in each epoch, I called train method once. In each time of train method, starting each batch, I detached the hidden state from how it was previously produced. If we didn't, the model would try backpropagating all the way to start of the dataset. I used clip_grad_norm to prevent the exploding gradient problem in LSTM. Then, I used Adam optimizer to optimize the parameters. In the evaluation step, the perplexity is the total loss / number of the tokens in the file. The total loss is the sum of all (cross entropy loss * sequence length used). For each batch, I wrapped hidden states in new Tensors, and detached them from their history. Also, I store the best validation loss ever in a variable to compare the validation loss after every epoch, and I stop training if the validation loss started to increase.

For the training time, in each epoch of LSTM model, it took about 77-80 seconds, and there are 10 valid epochs, so the total training time is 13.2 minutes. For the ngram model, it only took 3.4 seconds to train the model. Thus, LSTM took more times than the ngram models.

Below is the report of development perplexity against # of epochs. The test perplexity score is 105.097.



Below is the report of the perplexities of the LSTM and Ngram model. LSTM have better performance than the trigram model on this dataset.



- Problem 3

After reading the ideas of Bender and Koller as outlined within their study, I am more convinced by the argument presented in the paper. After thoroughly evaluating the paper, I concluded that a valid meaning could not be generated from only a single form relative to NLP. That implies the large frameworks of languages such as BERT (Bidirectional Encoder Representations Transformers). The frameworks do not learn to have a system of learning languages; however, they have a system on which they reflect the linguistic arrangements that are considered vital in applications. During the scope of the study, BERT has been outlined as a way of enhancing the overall performance relative to the meaning associated activities. As a result, the framework must have acquired significant knowledge relative to the meanings. Such learning relative to meaning is logically compared to how the syntax and semantic similarity can trap some data relative to meaning. The paper presents a way in which a specific language learning is encouraged when evaluating achievement of the present frameworks and incorporation of humility when assessing the natural language.

The existence of intelligence in the world of technology has been triggered by capturing the aspects of meaning and understanding. The communication of machines such as Computers is supported when people execute the machine learning process. Machine learning is a system instituted to ensure that artificial intelligence is promoted to the next level. Machine learning gives computers the power to think and respond similarly to human beings. That reveals the amazing nature of NLP activities. However, the accomplishments are on other occasions considered as hypes on which various frameworks obtain the portrayal of “understanding” of acquiring the meaning.

Relative to the topic under study, the research presented improved my understanding of the intention of communication and the implication of messages delivered during communication. The study outlined a fundamental objection on why communication is executed among human beings. One of the vital

goals of communication is to stimulate people to do something or pass a piece of given information. The argument enhanced my understanding that the purpose of communication falls outside the scope of a language. As a result, the intent of communicating is enclosed in the real-life situation. A wide range of reasoning tends to separate the conventional meaning of communication from the linguistic communicative intent. The conventional implication of a given expression during communication remains unchanged during different contexts where the phrase or the word can be utilized.

Before reading the study outlined under the paper, I believed there were various issues relative to NLP that I am no longer thinking about them. Earlier, I believed that it was easier to incorporate the generation of natural language, speech recognition, and the understanding of natural language in natural language processing. I believed that there was no challenge involved in integrating artificial intelligence with human language. However, reading the article has proved that the process is extensive and may spark various problems that will translate to adopting a systematic procedure and framework that will stimulate a clearer understanding of the speech and text processing frameworks.