Nickolas Wang, CSCI 366, Professor Alla Rozovskya Fall 2024

To start, NLP is a field in artificial intelligence that focuses on the interaction between humans and computers through
natural language; and the importance of this is crucial in our society which is why chat bots and AI like ChatGPT are around.

This report details the implementation of unigram and bigram language models, including the smoothed version using Add-One Smoothing.
This program also represents the calculated perplexities for a given sentence and discusses the differences between the models based on output results.

Overview:

Counting Words:
The count_words function reads the training file and creates a dictionarythat maps each word to its count.

Words are converted to lowercase to ensure uniformity.

Handling Unknown Words:

The unk_sentences function processes the test and training files, replacing
singleton words (words that appear only once in the training data) and
banned words (words present in the test data but absent in the training data) with <unk>.

Creating the Language Models:

The unigramModel function builds a probability distribution for each word based on its frequency.
The createBigram function constructs bigrams (pairs of consecutive words) and counts their occurrences.
The bigramModel function calculates probabilities for each bigram based on the frequencies of the first word in the pair.

Add-One Smoothing:

The addOneSmoothingModel function modifies the bigram probabilities to account for unseen bigrams by adding one to each count.

Log Probability Calculations:

Functions like calculateUnigramLog, calculateBigramLog, and calculateAddOneSmoothingLog compute the
log probabilities of sentences based on the respective models.

Perplexity Calculation:

The calculatePerplexity function calculates the perplexity based on the log probabilities, giving insight into how well the model predicts the test data.

The Results:

So, from the results of the output of perplexity_7.py, we see three unique numbers representing the perplexity:

Unigram: 1.0112

Bigram: 1.0056
Add-One Smoothing: 1.0087

From this, we can infer that:

1. Unigram Model (1.0112): The perplexity indicates that the unigram model is fairly consistent in predicting the words in the sentence, with a slight tendency toward higher uncertainty.

2. Bigram Model (1.0056): This model shows lower perplexity, indicating improved prediction accuracy due to the consideration of word pairs, which in my opinion better captures context compared to the unigram model.

3. Add-One Smoothed Bigram Model (1.0087): The smoothed model's perplexity is slightly higher than that of the regular bigram model, suggesting that while smoothing helps in avoiding zero probabilities, it introduces a small amount of uncertainty.

Discussing the differences:

The bigram model outperforms the unigram model as it incorporates the context of preceding words. The addition of smoothing in the Add-One model addresses the issue of unseen bigrams, or "zero" probabilities but it can be uncertain as the perplexity is a decimal, adding to the small amount of uncertainty.

The use of the <unk> token is crucial in the test models since the treatment of unknown or rare words affects the overall perplexity of the corpus. This in turn, demonstrates the importance of handling these cases when training language models.

Conclusion:

The implementation of unigram and bigram models along with Add-One smoothing has provided a comprehensive understanding of language modeling. The analysis of perplexity values illustrates the importance of context in language predictions. Future work may explore additional smoothing techniques and the impact of varying training data sizes on model performance.