Ngrams Narrative
Liam Leece & Kyle Zarzana

This project gave us insight into the world of ngrams and how they are used in computational linguistics. An n-gram is a sequence of n items from a corpus. The most common n-grams are unigrams and bigrams, which contain one word and two, respectively. We worked with these n-grams in this assignment to make a language model from three different corpsuses. To do this, the corpus is divided up into bigrams and unigrams, with each having a count of how many times the word or phrase appears. This is done with all three corpuses in the three different languages. After this knowledge base is built with the n-grams, we can compare words and sentences of unknown origin with each dictionary of n-grams to find which language or corpus it most likely originates from. Language identification is just one of the many uses of n-grams, others being speech recognition and spellcheck. There are a few different ways to calculate probability using unigrams and bigrams. In our application, we used the laplace method which, for each bigram, took the bigram count plus one and divided it by the unigram count plus the total count of all corpuses. This very small number is the likelihood that the given phrase would appear in the dictionary. When compared to dictionaries from other languages, the correct language was chosen around 98% of the time.

Language modeling relies on strong source text to accurately determine the rules of a language. The source text needs to be long enough and cover a large number of grammar rules. Of course, the source text will not contain every possible n-gram available in the language. In order to work around the zeroes, the process of smoothing adjusts the probabilities to more accurately reflect the words used in the language. Good Turing smoothing methods estimate the probabilities of previously unseen words. A simple approach to smoothing is the replacing the probability of words that appear zero times in the training data with the probability of other words that occur only once.

Language models can be used for text generation however, there are limitations. A language model can generate text based on the context of the words around it. It uses the text it was trained on to calculate the most likely next word it should generate. The problem with this approach is that there are many sentences in the English language that don't seem to make sense on their own. If the language model attempts to recreate one of those sentences in a different context, it could ruin the meaning of the generated text. Language models can be evaluated based on several different aspects including accuracy, simplicity, and space and time complexity. Our program was evaluated based on accuracy by comparing the results to a list of manmade accurate results.

Google has an n-gram viewer that works similar to the one we created in this project. You simply type in a word or phrase and it gives a graphic representation of how often that phrase occurs in the chosen language over time. The viewer even has different corpuses for different accents of the same base language. Below is an example of the n-gram viewer, notice how the

word Python has started to rise in usage in correspondence with the popularity of the programming language.