CS4395.001 Human Language Technologies Ngram Narrative

a. N-grams are a chunk of n words from a sentence that overlap each other if the n-gram is longer than a single word. N-grams give a good idea of what words often accompany each other in sentences by being able to calculate the probabilities that a word will be followed by a specific word. N-grams can be used to build language models by first having a corpus that can be broken down into n-grams. Then n-gram dictionaries are created to determine the count of specific n-grams in the corpus. Using this data, you can look at other bodies of text to determine how similar your language model is to different text examples.

b. A few applications where n-grams could be used is in a language recognition program like google translate, which can give a good estimate of your inputted language based on how often those sets of words are seen in different languages. Another application of n-grams is in a chatbot application, in which, given a large corpus of text and probabilities of n-grams corresponding to the corpus, can determine the next most likely word in the sequence in order to construct a sentence that makes sense to the user.

c. To find the probability of a unigram occurring, you first need to determine how many times that unigram showed up in your training data set. This is often done in a dictionary of unigrams followed by their counts. You then divide the count of that specific unigram by the count of all the unigrams in your data set to determine its probability. For bigrams, you need the count of the bigram divided by the count of the first word in that bigram. The count of the bigram is oftentimes stored in a dictionary of bigrams followed by their counts.

d. To build a good language model, you need to start with a good source text, or corpus. Your corpus needs to be representative of the body of text that you will be analyzing using the source text, or else you may get an inaccurate idea of any text that you analyze. For instance, if you train your language model on a very technical text like research papers, but then you use that language model to analyze normal everyday conversations, then your language model may miss very common words that are used more casually or any slang that a person uses. It may also think that more technical words should be found more often than usual, so they may put more weight on those words than others. Your corpus must also be very expansive. For instance you try to generate text using your language model, but the corpus you used was very small, it may repeatedly use the same sorts of phrases more often than a human would. Therefore, its important that a language model is built off of a large corpus and also a corpus whose source material will be similar to that of the text you will be analyzing.

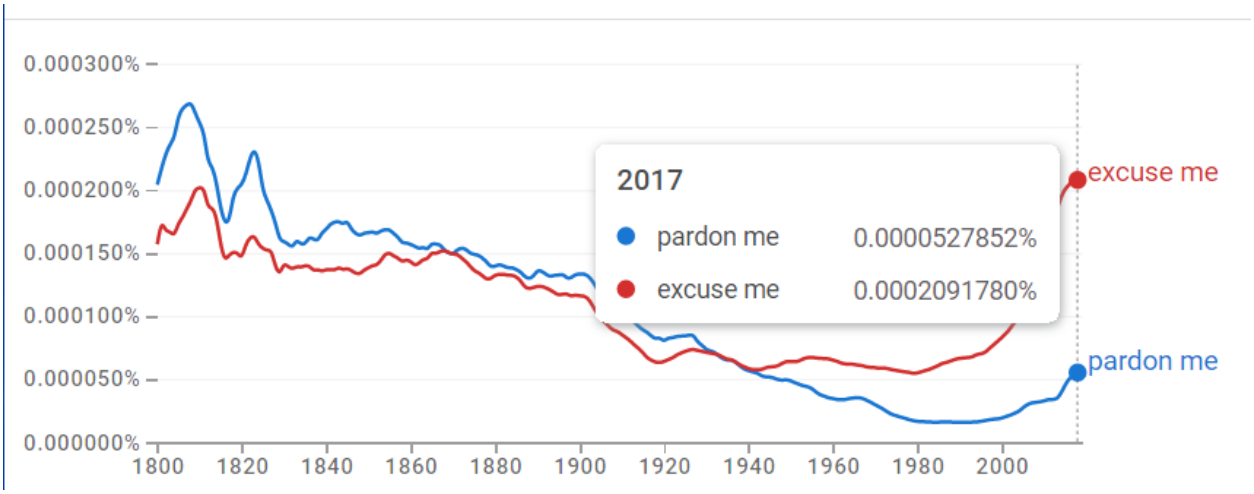
e. Smoothing is important since it is very likely that your corpus is not going to contain every single word that you will encounter when using your language model on text. Smoothing makes it so that any word that is not in the corpus still receives some value so that the probability of the word appearing will be a nonzero number. A simple approach to smoothing is Laplace smoothing, which adds one to the count of every word so that words that have a count of zero

can be accounted for. To compensate for this, the vocabulary count is added to the denominator so that the probabilities will be balanced out.

f. Language models can be used for text generation. When a sentence is generated, a program could look at the first word in that sentence, look through its n-gram dictionaries it created to find the word inside of the n-gram, and then look at all the matches and pick the n-gram whose count was the largest. You can continue to do this until you get to an n-gram with an ending punctuation mark. While this is just one way to do this, n-grams can be used in other approaches to determine which words are the next likely candidates when constructing a sentence. However, this approach is limited by how good your corpus is. If your corpus is too small or too limited, then your constructed sentences may sound too repetitive. To get a good result, you need to build a good corpus, which could take a lot of time to find just the right dataset to use.

g. Language models can be evaluated either through extrinsic evaluation or intrinsic. Extrinsic evaluation involves having human annotators analyze the result of your language model. This can be costly, so this evaluation is often used sparingly. Intrinsic evaluation involves using a metric to compare models. One type of metric used for this is perplexity, which is performed on a small subset of the data and measures the accuracy of your language model in predicting the text in the test data.

h. Google's n-gram viewer takes as its corpus a set of books published over decades that were written in English. Using that data, you can input a few n-grams to see how often that n-gram occurs in the data set and at what points in time were they used most or least often. The following is an example of using the n-gram viewer:



According to the viewer, the phrase "pardon me" was more common in the 1800s than "excuse me." However, in the 2010s it is much more common to say excuse me rather than pardon me, even though their meanings are relatively the same. This shows that, even with similar words, their usage largely depends on what timeframe you are looking at.