Kyle Chan kxc180021, Ryan Banafshay rcb170002

N-grams

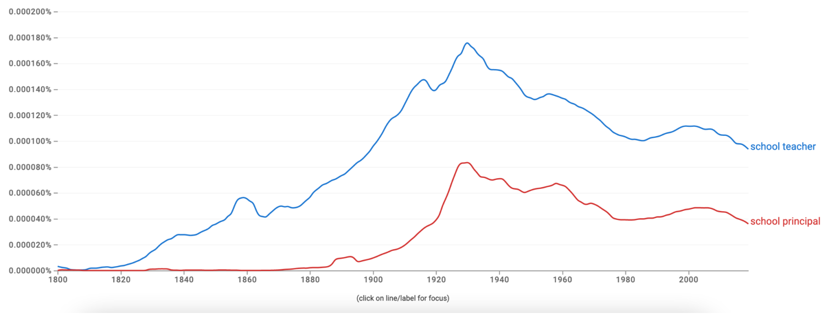
An n-gram is simply just a sequence of words with a variable length n. For example, “cat” would be an n-gram of one (unigram) while “the cat” is an n-gram of two (bigram). If a sentence is “The cat jumps on the table”, the first bigram would be “The cat” and the second would be “cat jumps”. N-grams iterate word by word as they are mostly used to predict the sequencing or ordering of different phrases. This gives an accurate representation of what words tend to follow other words.

The application of n-grams can be seen in a variety of Natural Language Processing programs. Auto completion of sentences in emails, spell checks in messaging apps, and even checking for grammar in word applications are all functions that can be tackled using n-grams. This is done using a probabilistic method. We can assign the odds of an n-gram appearing again based on the times it has been used already in the text. This value can then be used to estimate what n-gram is likely to follow another. Due to the fact that these probabilities are calculated from previous uses, the source text is a crucial element in the accuracy of these predictive models. If we have a large library of text over multiple different sources, the predictive model will likely be much more accurate than if the source text was just a few sentences long.

It is almost inevitable that certain n-grams will be left out of our dictionaries. To avoid giving a 0 probability to these scenarios, we have to reassign the probability mass from some more common events and give it to the events we’ve never seen. This is known as smoothing. We can achieve smoothing by adding one to every count and divide it by an extra V words in the vocabulary. This will likely give a more accurate predictive language model than what would be calculated normally. There are a variety of different ways to evaluate these language models. One example of these methods is a technique known as perplexity. Perplexity is a numerical value that is derived from a formula which compares words in a document. The probability distribution of the words within sentences gives out the score. The lower the perplexity score, the better the language model.

Language models built with n-grams can be used for text generation as they have a good ability to predict what could come next after a word based on what has been shown previously. Bigrams can be used as you do not need a whole sentence to predict what is to come next. This very well can be done by just looking at the previous word. The biggest limitation to this method is that the probabilities are determined based on previous text. If you are only basing this model on a small amount of data, it will almost certainly lack accuracy.

Google’s n-gram viewer allows you to display the frequency of words in a corpus of English books over the years. This can be done with multiple phrases at once on a single graph. Here is an example of a comparison of the frequency between the two bigrams “school teacher” and “school principal”.


school teacher
school principal
(click on line/label for focus)