Six Wires
CS 4395
Dr. Karen Mazidi
October 1, 2022

Ngram Analysis

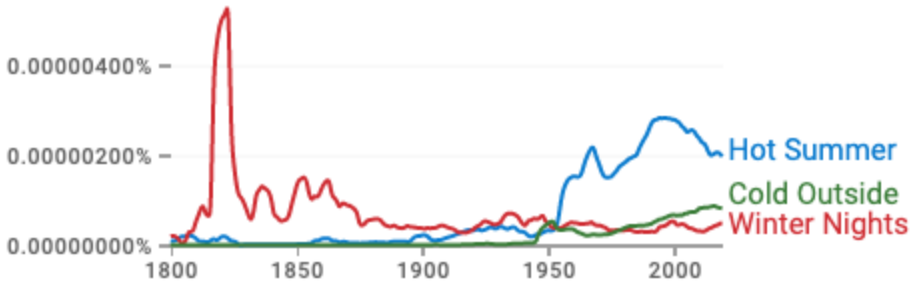
In Natural Language Processing, we often want to find a way to predict the most possible next occurring word in a sentence, document, media, or other. N-grams allow us to break up sentences into pieces that we can use with statistics to produce a model that allows us to predict these next occurring words. “Simply put, n-gram language models codify that intuition” that we have to *fill in the blank* (“Understanding N-Gram Language Models”). As mentioned, this can be used in text prediction, to allow text suggestions, email suggestions, and more.

N-grams can be broken up into associations of multiple length. Unigrams are single words, and bigrams are two words occurring next to each other in a sentence (in order). By coding our text into bigrams and unigrams, we can calculate the probability of a following gram by using laplace smoothing to calculate the probability based on the overall text fed into the test unigram and bigram dictionaries. Because of this, source text is very important. We want our source text to contain many examples of the possible text topics that we are going to want to predict with the model. The more examples of specific bigrams and their associations, the more accurately we’ll be able to predict future bigrams.

Because we’ll be using our n-grams for predictions, we want our data to be consistent throughout the set. Regardless, we won’t get every possible combination of n-gram from our source text, which is where smoothing comes in handy. This allows us to fill out n-grams that don’t occur naturally within the text with some probability mass. This allows us to have more consistent data throughout the set. To achieve this goal, we can use techniques like laplace smoothing, which takes the probability of each n-gram given the entire vocabulary.

With the model that we’ve created with our n-grams, we can create a language ruleset to be able to generate language. To do this, we need to calculate the probability of each unigram and bigram. With this model, we can output automatically generated text based on the rules created from the source text and n-grams. N-grams with a higher n value help in creating the rules. The limitations of this are that the text is only generated based on the limited amount of text fed in, and will only generate text based around the pre-existing word associations fed in. This dissalows for any creativity or learning from the code. We can evaluate these models based on how accurate our target values are to the actual predicted values.

Google has a cool n-gram viewer that shows us how often a specific n-gram has been used over time via a graph. In the image below, I show the usage of the phrases “hot summer”, “winter nights”, and “cold outside” over time.



Work Cited

"Understanding N-Gram Language Models." *Rev*, https://www.rev.com/blog/resources/understanding-n-gram-language-models. Accessed 2 October 2022.