Aleena Sayed
Norah Khan

Ngrams Part Three: Narrative

An ngram defines a sequence of n words or tokens. There can be a variety of uses for ngrams and different ways to classify them. An ngram with one word or token is called a unigram. Another commonly used ngram involving two sequences of words/tokens is called a bigram.

A common use of ngrams is autocomplete, a feature that is on most word processing applications. If an ngram probably is computed it can predict which sequences of words are most likely to show up together. Additionally, many applications use ngrams for voice recognition. Another use would be the actual processing of phrases for smart assistants. If an exact sentence isn't given every time, using ngrams the machine can still interpret the sentence being said, this logic can be applied to both smart assistants and voice recognition. Overall, ngrams are a useful tool for text processing and to assist in optimizing machinery. They can essentially allow a text document that has no structure to be converted into data that can be interpreted by a machine or program, this is how a language model is developed.

Probability of ngrams can be determined by dividing the count of the given ngram by the total number of tokens. The source text is vital to having an accurate language model. For example, if a text is very small it will be difficult for a model to grasp the probability of words. If a text is longer but does not follow correct language rules or sentence structure in some places it may skew probabilities and lead to results that may not be realistic or comprehensible. Regardless of the corpus size and variety, you may also consider pre-processing a corpus by doing POS tagging or omitting words that may be difficult for the model to process.

Even after creating a language model and generating the probabilities for all possible ngrams, there may still be potential ngrams that haven't been generated by the model. This is where smoothing is applied. Smoothing gives a probability, some weight, to ngrams that do not exist in the dictionaries. One approach to smoothing is simply applying the same weight to zero probability ngrams that would be given to a single word.

Text generators are a common use of language models and they work particularly well with ngrams that have a higher degree. The first step is calculating the probability of ngrams and then saving that information in a dictionary. This data can then be used to generate text, such as in autocomplete. The only way this process can actually accurately perform is if the corpus is very large, so that it can be trained efficiently.

After creating a language model the issue of evaluation of accuracy and correctness arises. There are two approaches, straightforwardly having human annotators or a perplexity

metric. A low perplexity shows that there is not a large difference between the probability of the words that are appearing in the model and the original corpus.

Google's ngram viewer makes charts of the most commonly used ngrams in a variety of corpora. The search engine measures the frequencies of strings. It typically searches through a document or other corpus and then once provided with a string it will determine the frequencies of the ngrams. The interface of the ngram viewer allows the user to select a corpus, a period of time, and an ngram. After entering the information for these fields the ngram viewer will return a chart showing the frequency of the words.