Daniyah Arshad

CS 4395

03 March 2023

# Ngrams

An n gram is a set of sections of size n over a group of text. For example, the unigrams in a sentence would simply be every word in that sentence. The bigrams in a sentence would be a set of every pair of words in that sentence. This continues on and on, with ngram sizes varying based on the purpose of the user. Ngrams are typically used to build language models, by calculating counts of ngrams in a set of training text and using that to calculate the probability of certain words occurring before or after each other. With these probabilities, a language model can then predict what word will most likely occur after an existing word, and more complex language models can also take into account the probabilities of the entire string of existing words to predict what will come next. Additionally, in a similar fashion, they can also be used to generate sentences and text using this same predictive ability based on probabilities.

Ngrams can be used in predictive text applications, chatbot applications, and grammar checking applications. All of these applications  rely on predicting and generating text, potentially based on observed probabilities.

The probability of a unigram is calculated by the count of how many times that unigram appears in the text, and dividing that count by the total count of unigrams in the text. The probability of a bigram is calculated by multiplying the probability of the first word in the

bigram by the second word in the bigram, and these single word probabilities are derived from the unigram probability calculation method.

The source text for building a language model greatly influences the range and depth of a language model, as language is diverse and all-encompassing, it is important to have a source text that can encompass as many aspects of language as possible. Or, if a language model intends to model a specific subset of language, such as medical language, internet colloquialisms, or engineering jargon, it is important to choose source text(s) that represent a big range of that specific subset. It is also important to simply have a large source text in general, as the more words there are in a source text, the better and more accurate probabilities can be calculated.

In a source text, there will likely be many words that are only used once or twice, and thus, will have a very low probability. When considering bigrams, there will be many bigrams with a probability of 0. This can skew calculations in an unintentional way, and as such, smoothing is used to bump up the probabilities of very rare words and bigrams, so as to allow calculations to proceed without a hitch.

As described earlier, language models using n grams can be used to generate text through the probability calculation method. By choosing a starting word, a language model can look through probabilities of what word is most likely to follow that starting word, and it can then keep choosing words based on their probabilities until it generates a full text. This can be very limiting, as language is not simply defined by purely mathematically sequenced words, it is defined by many complex rules defined by culture, tone, setting, and more, and as such, a probabilistic language model will not be able to account for all the nuances in language.

Language models can be evaluated extrinsically and intrinsically, extrinsically by comparing human generated text vs language model generated text, intrinsically by using specific tools that mathematically analyze the accuracy of the text a language model generates with itself.

Google's n-gram viewer allows you to choose from a variety of corpuses, then type in your n grams, and then visually displays those n grams as they appear through that corpus across time. This is a fun analytical tool that can be used to observe the frequency of words and phrases through time.