Overview of NTLK NGRAMS

BY: BRIDGETTE BRYANT & TERA PARISH

What are NGRAMS?

N-grams are simply a way to view/split up text by sliding windows of n words at a time. Different types of n-grams include unigrams (which look at one word at a time), bigrams (which look at 2 words at a time), and trigrams (which look at 3 words at a time). When looking at ore than 3 words at a time (4+) they are just referred to as n-grams. These simple text chunks can be used to build language models by calculating the probability of a specific ngram's count over the total number of the words by themselves. Ngrams require a body of text called a corpus to be made, the choice of the corpus is highly influential in the probabilities and accuracy of the ngrams.

Applications that use NGRAMS:

Applications that use Ngrams include many auto-completion sentences such as in Microsoft Word, Gmail, Outlook, google, etc. By generating the predicted text by it's known ngrams and their counts. Also, in auto spell-check which is used in nearly every text-based application. By predicting the text spelling and grammar by its known ngram probabilities. Ngrams can also be used to predict a language for translation/other uses which is what our two python programs ngrams_create_dictionaries.py and ngrams_probabilites.py does in our github here with a 97% accuracy: https://github.com/BridgetteBXP13/CS-4395.001---Human-Language-Technologies/tree/main/Assignment%205

Probabilities in Ngrams

Probabilities for Unigrams and Bigrams are very simple, they focus on the number of counts specific words have in the corpus. For Unigrams let's say our corpus is about sea life and our unigram is "Turtle" we would divide the number of time "Turtle" appears in our corpus by the total number of words in our corpus. For Bigrams it is very similar taking our same example our bigram will be "Sea Turtle" it will divide the number of time "Sea Turtle" appears in our corpus by the number of times "Turtle" and "Sea" appear separately in the corpus.

Language Model Source Text

Choosing a proper corpus is very important as it is our source file for creating your language model. With a skewed/misinterpretation source model compared to what you are predicting will heavily impact your predictions and cause a low accuracy. For example, if you want a language model to predict auto-completion sentences, but your corpus was from a Shakespeare book. It will probably be highly inaccurate and even irritating for users to use as most of them probably don't speak similarly to how Shakespeare writes. Instead, you should probably use something much more modern such as a text collection of emails, texts, etc. depending on your use-case.

Smoothing

Probabilities for ngrams can have combinations which don't exist for example if we have "Sea Turtles swim" we will attempt to compute the probability of "Sea swim", which will likely have a count of 0. Making our probability 0 from multiplication, causing a very skewed and incorrect result. Smoothing attempts to fix theses 0 counts by filling in zero values with a small probability of the overall probabilities. Causing the distribution to smooth, there are many smoothing techniques used, in our program we used Laplace smoothing, also known as add-one smoothing. It simply adds 1 to the zero count so the zero becomes a 1, which doesn't affect the multiplication in our probability.

Text Generation

As talked about above, language models can be used to generate auto-completion for sentences. However, these can often be misleading or incorrect. Because they are highly limited as the ngrams lack a sense of context. This is also seen in spell-check and auto-correction, for example, Microsoft Word doesn't think ngrams is a word, however, we know it definitely is despite the probability. You can attempt to generate full stories, emails, messages, etc. with ngrams although just because words are common in a pattern by count, doesn't mean they will make any sense together.

Evaluating Language Models

We can evaluate our prediction and accuracy values from our language model. The ngram's predictions are usually quite small and show how likely that specific ngram will be. The accuracy is simply a percentage value of correctness compared to the actual answer when testing your language model.

Google's N-GRAM Viewer

Google actually has an ngram viewer you can use to search the web of ngrams from different sources and plots their popularity. For example, here is an ngram of "Turtles", "Snakes", and "Lizards":