N-gram report

N-grams are a contiguous block of text that span n words long. For example, an n-gram of length two (also known as a bigram) consists of two words, back to back. In the sentence “I had a lovely day at the park”, “had a”, “a lovely”, and “lovely day” are all bigrams. Likewise, 3-grams from the example sentence would be “had a lovely” “a lovely day” and “lovely day at”. Using n-grams, we can predict how likely a given sequence of words is to appear in a body of text, which helps us to build a language model.

N-grams can be used for autocompletion, generating text from a corpus, or for text summarization. Because N-grams are useful for predicting the probability of n string of words together, from unigrams up, they can be used to guess which word comes next in autocompletion, or try to guess what a summary or generative text would look like given a body of text to use. Now that we know what n-grams are and what they are used for, let’s have a look at how they are calculated.

Unigrams are calculated by taking the count of the unigram and dividing it by the total number of tokens in the text. Bigrams are calculated by taking the probability of the unigram of the first word in the bigram, and then multiplying that by the probability of seeing the second word in the bigram, given the first word in the bigram. In general ngram languages model probabilities are called using Maximum Likely Estimates (MLE’s). Okay, we have how n-grams are calculated. Now lets see other factors that affect the probabilities we see.

Source text is very important when working with n-grams. If you choose a source text that is written for elementary school kids, you may not get the same depth of vocabulary as a college level text. In a similar way, a science text and a body of poetry may yield different frequencies and usages of words, which would affect the n-grams you see. You may also have the issue where the probability of a unigram or bigram occurring in a test text ends up being zero. This would make the entire probability of the n-gram go to zero. In order to avoid this we can apply some smoothing.

Smoothing makes probabilities that would be zero, slightly non-zero, but smaller than other probabilities. One approach to smoothing is LaPlace smoothing, which adds 1 to the numerator so the value of P is never zero. In turn, the value V is added to the denominator. V is the size of the entire document, which makes any probabilities with 1 in the numerator very small.

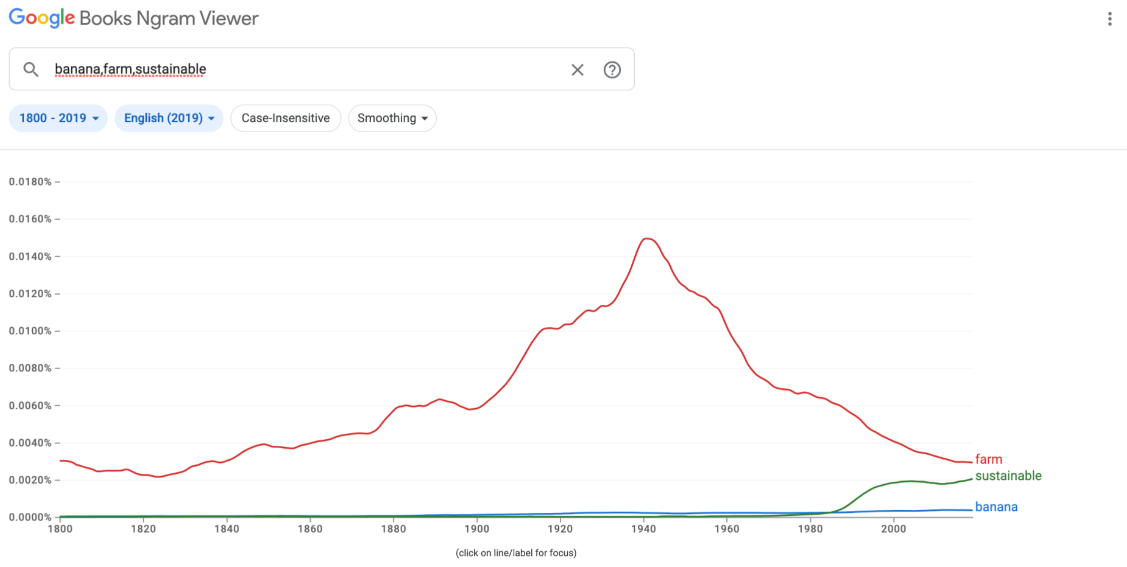
Like was said earlier in the report, one application of n-grams is generating text from a corpus. Language models can be used for text generation because they look at the probabilities of what words come next. We can use these probabilities, and pick the highest one to generate what word should come next. But how can we know if what we generated is any good?

Language models can be evaluated in two ways. The first is an extrinsic evaluation which is done through human annotators. The second way is with a

N-gram report

measurement called perplexity. This measure is the inverse probability of seeing words we observe normalized by the number of words to see.

Lastly, Google's n-gram viewer shows you different word frequencies for n-grams you search for in printed materials within the last five hundred years or so. Here is an example of an n-gram in Google's n-gram viewer:


Google Books Ngram Viewer
banana,farm,sustainable
English (2019)
Case-Insensitive
Smoothing
farm
sustainable
banana
(click on line/label for focus)