N-grams are a tool that can be used to build a model of a language and can be thought of as a sliding view over a piece of text. By sliding window, we mean that it is a list of n tokens in a text that moves along 1 token at a time. For example, a unigram is an n-gram with a size of 1, and can be seen as equivalent to tokenized text. In contrast a bigram's window is of size 2, so each bigram consists of 2 tokens. It is important to note that bigrams do not simply consist of a list containing paired tokens, there is overlap that is created as the window moves. If there were 3 tokens, the 2 bigrams would include the first and second tokens, and the second and third. Ngrams are applicable in several situations, from providing a convenient way to organize and split up text, to being able to create a statistical language model that can be used for natural language processing and generation.

As stated previously, ngrams are used to build a statistical model of a language. In order to do this, we make use of the Markov assumption, which states that only the current state (in our case, the most recent word) should be accounted for. With this in mind, we are able to take the probability of a word X appearing in the corpus, and multiply that with the probability of another word Y given word X. By doing this, we are able to statistically determine what are the most likely bigrams and with it combinations of words in a text.

However, not every possible ngram will be present in the training text. In order to prevent this from messing with our model, we will need to implement smoothing, which gives every single possibility with some probability mass. One very simple way to provide some smoothing to a language model is to give the possibilities with a probability of 0 with the probability of words that only happen once. This is known as Modified Good-Turing Smoothing.

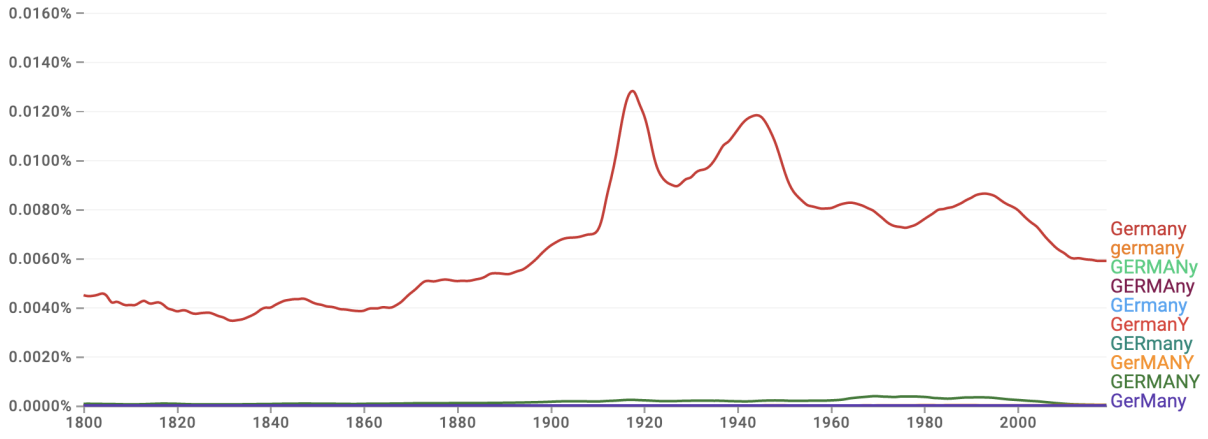
When working with ngrams, the source text the language model is using is very important. Since this is the only frame of reference that the model will be able to use in order to statistically determine what is most probable, one must make sure it is of high enough quality. For example, the source text must be large enough that the model is able to have a good reference of the properties of natural speech. Another important factor is that the source text should tonally and subject wise fit with the topic of the text it is meant to analyze or generate. A language model that was created using transcripts from court cases will have difficulty analyzing love letters, for example.

Ngrams can also be used to generate text, not just analyze it. A start word is chosen or randomly selected, and the model simply determines what is the most likely word to follow. While this is simple, it has its very evident limitations. The largest being that while the generated text may start off seeming reasonable, it rapidly becomes more incoherent as more text is generated.

Language models such as those created by n-grams can be judged and evaluated in various ways. The first way is, of course, simply having the model's

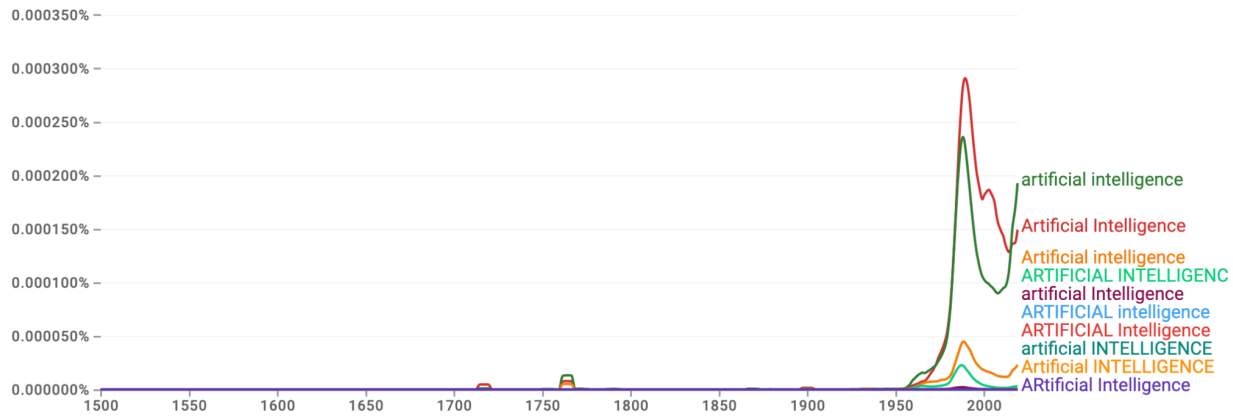
performance be evaluated by a human. Another way it could be judged is by determining if the perplexity is low.

The Google Books Ngram Viewer provides users with an easy tool to analyze the popularity of ngrams over a certain period of time using the text from Google Books. Users can choose which phrase to search for, the language to search it in, the date range, and even the smoothing.



(click on line/label for focus, right click to expand/contract wildcards)

(click on line/label for focus, right click to expand/contract wildcards)



(click on line/label for focus, right click to expand/contract wildcards)