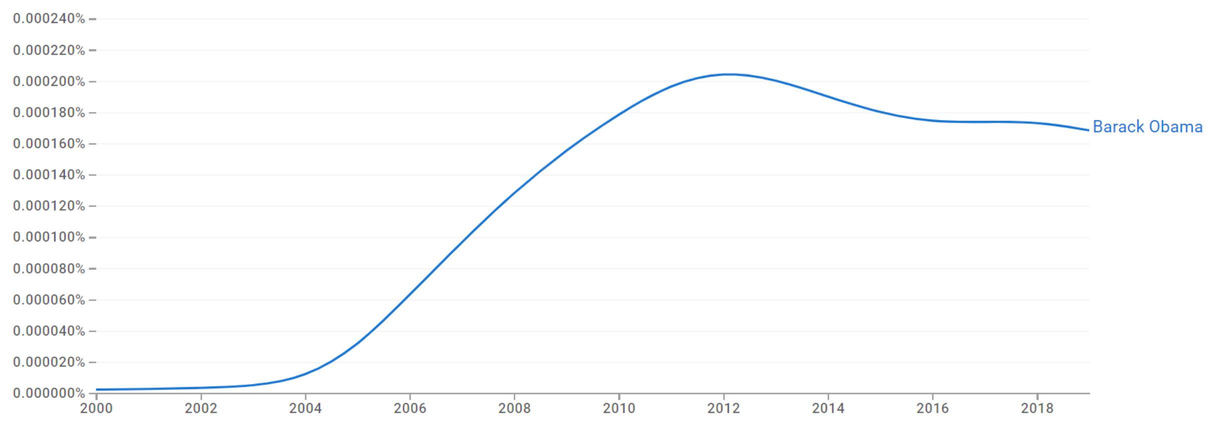
Aditya Guin (asg180005)
Varin Sikand (vss180000)
CS 4395.001
10/02/2022

N-Grams

An n-gram refers to a sliding window of size n over some text. Ngrams allow us to calculate probabilities of a sequence of words, which can help create language models for different languages. These language models can also be used for predicting the next word given a sequence of words. There are various practical applications of Ngrams; protein and DNA sequencing make use of the Markov models created from ngrams. In addition, they are used in statistical natural language processing for speech recognition.

Unigrams are the individual tokens of a text (using nltk's work_tokenize). The probability of a certain unigram is that unigram divided by total unigrams (or total tokens). Bigrams are consecutive words from a text. If (w1, w2) represents a bigram, the probability of this bigram is the unigram probability of w1 multiplied by the probability that w2 appears right after w1. This idea can be extended to n-grams as well. Source text for these language models is important since the model should be trained well to perform its task; for example, if the model is used for classifying tech journals, it shouldn't be trained using medical journals. This is because the words would be focused on medical terms, which wouldn't help for technology-based classification. A common issue with N-gram modeling is inherent data sparseness. For a large token set size, there can be minimal occurrences of an n-gram. Therefore instead of having near-0 probabilities, smoothing is used to flatten the probability distribution so that all word sequences have some probability of occurring. More precisely, filling in all zero probabilities with a little bit of the probability with the overall mass. A simple approach to smoothing would be the LaPlace smoothing. In LaPlace smoothing, one is added to the count of the word, while dividing by the vocabulary and token size. However, studies have shown that LaPlace smoothing isn't the best smoothing technique; the Good-Turing smoothing technique replaces all zero occurrences of a word with counts of words that occur only once. Language models can be used to generate text. Given a set of bigrams and a starting word, we can use the model to "construct" sentences by appending the word associated with the highest bigram probability. We continue this process until we reach a punctuation mark or a certain word count. This approach with bigrams isn't as accurate, because we're only looking at two words at a time. A more accurate model would include n-grams with a higher degree. Language models can be evaluated by extrinsic or intrinsic evaluations. Extrinsic evaluation refers to human annotators manually reviewing the results based on some predefined metric. This process can be time-consuming for huge output. The intrinsic evaluation uses a specific metric to compare models. An example would be perplexity, a metric used to test how well a language model predicts the text in test data. This is calculated by taking the inverse probability of seeing the words we observe, normalized by the number of words. Google's n-gram viewer is an online search engine that gives the frequency of an n-gram. The user inputs a phrase, and google shows the frequency that phrase has appeared in printed sources from 1950-2019. In the image below, an example is shown for the phrase "Barack Obama".

Aditya Guin (asg180005)
Varin Sikand (vss180000)
CS 4395.001
10/02/2022



The upward trend around the years 2008-2013 makes sense, given that's when he was running for president. One interesting feature of the graph is the scale of the y-axis, specifically the percentages being extremely small. This also is logical, given that there are millions of printed sources per year.

In conclusion, N-grams are effective in generating language models and using those models for text prediction. The various applications, including machine translation and text prediction have proven N-grams to be significant importance in the realm of natural language processing.