N-grams

By: Aditi Chaudhari

An n-gram is a sliding window over text that slides n words at a time. For instance, a unigram is an n-gram with a sliding window of size 1. A bigram is an n-gram with a sliding window of size 2. A trigram in an n-gram with a sliding window of size 3. Anything over a sliding window of size 3 is referred to as a n-gram, with the n specifying the size of the window. N-grams can be used to build a probabilistic model of language. The example below provides a clearer image of n-grams.

Example:
Let's say that our text is: "the quick brown fox jumps over the lazy dog."

A unigram of the text would be:
["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog" "."]

A bigram of the text would be:
["the quick", "quick brown", "brown fox", "fox jumps", "jumps over" "over the", "the lazy", "lazy dog", "dog."]

A trigram of the text would be:
["the quick brown", "quick brown fox", "brown fox jumps", "fox jumps over", "jumps over the", "over the lazy", "the lazy dog" "lazy dog."]

N-grams have a wide range of applications. Within the context of NLP, they can be used for text prediction. For instance, a person may guess that the sentence "Add in a tablespoon of _____" ends with the word "salt." Another person may guess that the sentence ends with the word "sugar." A language generation model built with n-grams may predict either "salt" or "sugar" based on what data it was trained on. Different source texts create different probabilistic models of language. Because of this, a model trained with cooking recipes may predict the word "salt" but a model trained with baking recipes may predict the word "sugar." N-grams can also be used for language generation. A naïve approach to language generation involves beginning with a start word, looking though the bigram probabilities to find the bigram with the start word in the first position that has the highest probability, concatenating that word, and then repeating this process with the next word over and over again until it finds a stop token, such as a period. Using n-grams for language generation not ideal since it is limited by a small size and a naïve approach. The bigger the size of the n-gram, though, the better the language generation will be. N-grams also have important applications outside of NLP. For instance, they can be used in computational biology for DNA sequencing.

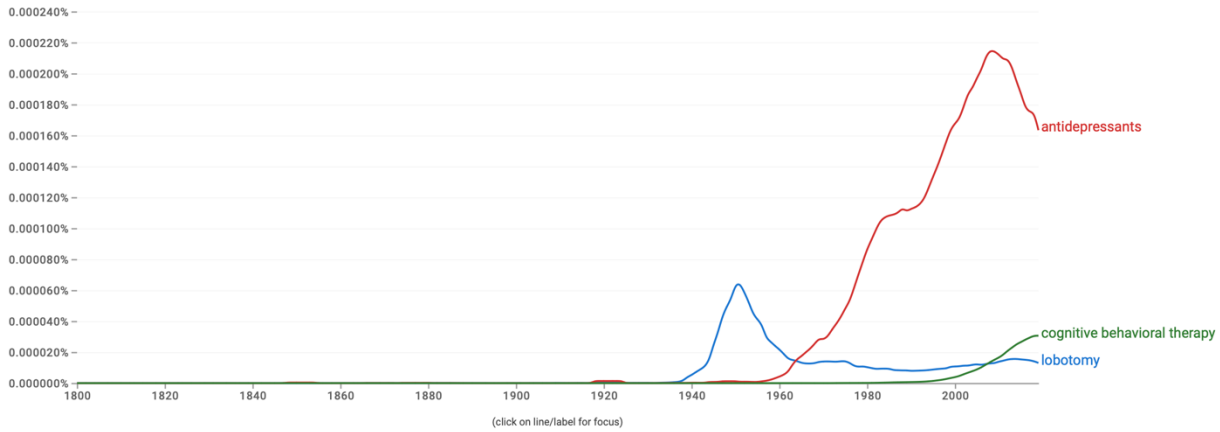
After creating an n-gram out of a text, a probabilistic model of language can be created. To assist in calculating probabilities in text, dictionaries that hold a count of each n-gram are created. Using the example above, for instance, the unigram dictionary entry for 'the' is: 'the':

2. The bigram entry for ‘brown fox’ is: ‘brown fox’: 1. If we wanted to find the probability of ‘brown fox’ occurring in this corpus, we would need to find P(‘brown’) • P(‘fox’ | ‘brown’), or more generally P(w1, w2) = P(w1) • P(w2 | w1). In this example, the probability of P(‘brown’) is the number of unigrams that are ‘brown’ divided by the total number of unigrams. The probability of P(‘fox’ | ‘brown’) is the count of ‘brown fox’ bigrams divided by the count of ‘brown’. Using counting, we can use n-grams to create a probabilistic model of language. What if we wanted to calculate the probability of ‘lazy cat’, though? If we try to calculate the P(‘lazy’) • P(‘cat’ | ‘lazy’), we will run into an issue due to the fact that the word ‘cat’ has a count of 0. Our corpus cannot possibly contain all possible sequences of words, and this is known as the sparsity problem. To tackle the sparsity problem, we can use smoothing, which involves filling in 0 values with a little bit of probability of the overall mass. In this assignment, I used Laplace smoothing by adding 1 to all the 0 counts so that the overall probability isn’t 0. In fact, 1 is added to all counts since we cannot tell which counts are zero ahead of time. To balance out adding 1 to each count, the total vocabulary count is added to the denominator.

To evaluate language models, an extrinsic evaluation using human annotators evaluating the result against a predefined metric can be used. However, these tend to be quick time-consuming and expensive, so intrinsic evaluations using some metric to compare models are used more frequently. A common metric for evaluating language models is perplexity, which measures how well the language model predicts text in test data. In order to calculate perplexity, a small amount of test data must be set aside prior to training the model.

Google has an amazing resource known as the “Google Books Ngram Viewer,” which allows the user to enter in phrases and then displays a graph of how the phrases have occurred in a corpus of books published over the years.

Personally, I am interested in the intersection between computer science, psychology, and neuroscience. As a result, one of the courses I am taking this semester is Abnormal Psychology. Throughout the semester, we have looked at how the definition of “abnormal” has changed throughout the decades and how the treatment for those with psychological disorders has changed, as well. The n-grams that I fed into “Google Books Ngram Viewer” were “lobotomy”, “antidepressants”, and “cognitive behavioral therapy.” I wanted to see if the popularity of each treatment for depression could be mapped to a certain time based on when these treatments were being mentioned in literature. Based on the graph, lobotomies were popular in the 1950s, but the use of lobotomies rapidly declined after that. Antidepressants started gaining popularity in the 1960s and are still popular today. A quick Google search revealed that cognitive behavioral therapy was invented in the 1960s, but the “Google Books Ngram Viewer” reveals that it has only recently started gaining popularity in literature. It is fascinating to see how the popularity of treatments have changed over time.


0.000240%
0.000220%
0.000200%
0.000180%
0.000160%
0.000140%
0.000120%
0.000100%
0.000080%
0.000060%
0.000040%
0.000020%
0.000000%
1800
1820
1840
1860
1880
1900
1920
1940
1960
1980
2000
antidepressants
cognitive behavioral therapy
lobotomy
(click on line/label for focus)