Vicky Nguyen vtn180004, Meinhard Capucao mdc190005
CS 4395.001

N-Grams

N-gram is a text window that slides over n words at a time. Unigrams would take one word at a time e.g. “squirrel” while bigrams would take two words at a time e.g. “the squirrel”. If we were given a sentence such as “The squirrel ran across the street”, the first bigram would be “The squirrel” and the second bigram would become “squirrel ran”. There’s also trigrams that would take 3 words at a time and taking above 3 words would usually mean that they are n-grams with n specified as 4, 5, 6, etc. N-grams are used to build a probabilistic model of language, and for the program to learn that kind of model, they would require to have a body of text, corpus. The language model will be significantly impacted by the corpus selection.

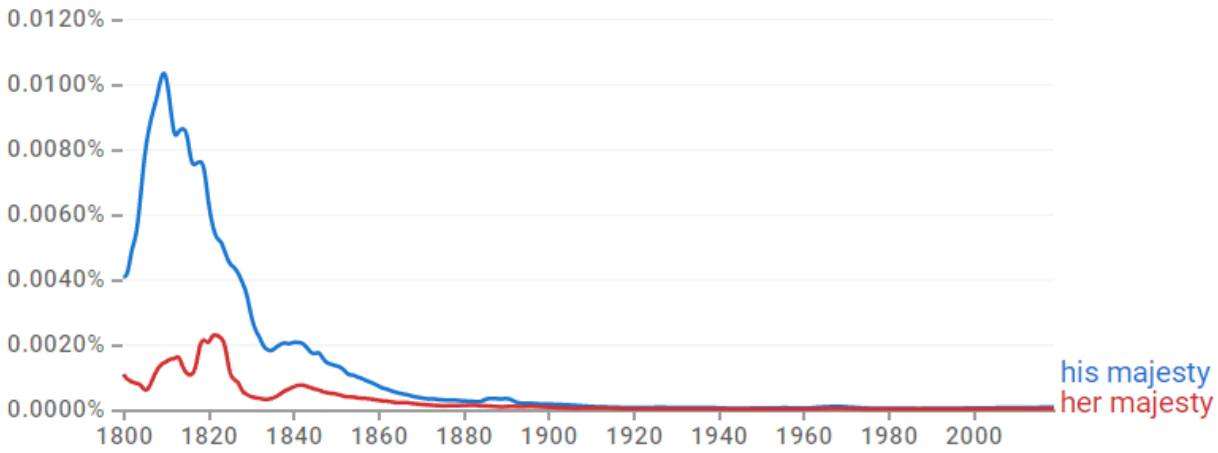
Many Natural Language Processing applications can be seen using n-grams. A few applications where n-grams could be used are spelling error detection and correction, dictionary look-up, language identification, etc. The probabilities for unigrams and bigrams are calculated by its n-gram model. This is found from the count of the first word divided by the number of tokens in the text. This is used to determine the probability for the next item in a sequence as well. Source text is a vital component in the accuracy of the language models because the probabilities of n-gram that are likely to follow another are generated from previous uses. The language model will most likely be more accurate if we had access to a large library of text from numerous sources rather than a few sentences as our source text.

There will inevitably be some n-grams missing from our dictionaries. In order to avoid assigning these scenarios the probability of zero, we must reassign the probability mass from more of the common events and give it to the events that we haven’t seen yet. This is referred to as smoothing and can be achieved by adding one to each count and dividing by an additional V word from the lexicon. Compared to how it would normally be calculated, this will most likely result in a more precise language model.

Language models can be used for text generation because they can accurately predict what might come after a word based on what has already been displayed. Bigrams can be utilized for this because it is not necessary to know the entire sentence to predict what word will come next, which can be done by looking at the previous word. Although, the main drawback to this method is that the calculation of the probability is based on previous text and will most likely be inaccurate if the model is based on a small quantity of data.

There are various techniques to evaluate language models. One of the techniques is called perplexity, which is a numerical value that is derived from a formula that compares words that are in a document. The score is determined by the probability distribution of the words used in the sentences, and the lower the perplexity score, the better the language model will be.

Google’s n-gram viewer allows the users to view the statistics/data of the word frequency that are in a corpus of English books over the past years (1800-2019). The user can also input multiple phrases at once to see a comparison on the graph. The image below shows an example of a comparison of the frequency between two bigrams, “his majesty” and “her majesty”


0.0120%
0.0100%
0.0080%
0.0060%
0.0040%
0.0020%
0.0000%
1800
1820
1840
1860
1880
1900
1920
1940
1960
1980
2000
his majesty
her majesty