Abdullah Hasani
AHH190004

Assignment 4: N-grams

N-grams are a technique in natural language processing (NLP) that can be used to build language models. An n-gram is a sliding window of size n over a segment of text. To build a language model, a model can be trained over a large dataset and the number of times an n-gram appears can be found out, and using this information, a prediction can be made on what words might come next in a sentence.

N-grams can be used in a large number of NLP applications, including chatbots and speech recognition. As an extreme example of a chatbot, ChatGPT by OpenAI uses n-grams as a part of its architecture to provide coherent and lengthy answers in response to a question. N-grams are also used in speech recognition software, where a spoken word may sound like several different words to a machine. To remedy this, there is an added layer of using a statistical model to estimate the probability of the next word given the sequence of characters prior, and in doing so can mitigate errors in speech recognition software.

To calculate the probability for a unigram or bigram, there is a technique used called “maximum likelihood estimation”, or MLE, which finds the parameter values that maximize the likelihood of the data. The method for calculating MLE for unigrams versus for bigrams differs slightly, in that the probability of a word is the frequency of the term divided by the number of words in the text. However, for bigrams, the likelihood is calculated as the number of times the bigram appears in the corpus divided by the probability of the first word appearing in the text. This allows the user to see if the two words occurring together in the bigram occur together often or if it is just a coincidence and the words do not occur together often.

The source text, or training data, is critical in all these cases. This is because the n-grams model is heavily reliant on the source text as its “source of truth”, where if multiple words are strung together often then the results would be skewed to favor those words appearing in the same or a similar order as they were in the training data. For instance, if a model is trained on a corpus describing coding basics, then “finding a bug” would have a very different meaning than if the model was trained on a corpus about pest control.

Smoothing in the context of NLP is an important part of text processing, especially in the case of n-grams. Smoothing is used to account for zero probabilities and for unseen or unknown n-grams. Zero probabilities are problematic as they can result in predictions of zero for a segment of text, even though it may be grammatically correct. Unknown or unseen n-grams can occur when the training data does not contain all possible n-grams (which is almost always), and not smoothing out the data would result in a lack of robustness in the model.

Language models can be used for text generation as they string words together using the probability that a word is found next to another word. However, this is very much dependent

on the training data that is fed into the model as the probabilities are computed given that training data. The larger the training data set, the better the text generation will work. However, there are still limitations in generating text using this approach. For instance, the training data could be biased or incorrect, and the generated outputs would also reflect the imperfections of the training model. There is also a lack of originality in the texts that are generated, as the generated text is derivative from other texts and not innovative in any way.

A language model, despite its flaws, can still be evaluated and tuned to provide better results. One way of checking the accuracy of a model is to have a human manually check the results of a model, and rate its quality. There can then be some penalties associated with low-quality outputs, which will then reduce the number of unwanted outputs. There are also automated ways of calculating how good a language model is, and one such example is by using the perplexity of the model and minimizing it as much as possible to get an ideally low perplexity number.

Google has an n-gram viewer that allows for analyzing and searching through corpora. An example of Google's Books Ngram Viewer in action is when the words "horse" and "car" are inputted. Between 1800 and 1902, "horse" showed up in more books than "car". But 1902 was the turning point, and ever since then "car" has been the more popular word.