Jesse Truong
Anthony Chen

Portfolio Chapter 8: Ngrams

N-grams are basically sliding window problems which get analysis to create a probabilistic model. N-gram is a field of computational linguistics which is used to make predictions on the next words after a word. N-gram assigns probability to word sequences, for example if i had the word race track, if the model saw only the word race, there would be a probability that track would be the next word. Having a magnitude of these probabilities would create a N-gram model or a language model. For the word in English we can have a model that would help us with autocorrect, NLP noise-assumption (speech recognition), text mining, text generation, etc.

N-grams are calculated through bigrams and unigrams, unigrams are just a dictionary of words with the number of occurrences of the word as the value, while a bigrams are also a dictionary that would instead take every 2 words, in the style of a sliding window. With both the dictionary of unigram and bigrams we can now calculate the probabilities, by taking the number of occurrences of a word and dividing it by the total number of words in the unigrams. To calculate the bigrams, you would take the 2 word sequence occurrences and divide it by the number of occurrences of the first word, giving us the probabilities of the 2 word combination for the first word of the combination.

Having a robust source text is extremely important for these language models. They must have a large dataset that has a large coverage of the language to prepare itself for every possible combination of words. As well as representing the overall english language for an extremely accurate language model, with the overall occurrence of every word accurate to modern english. This is so the correct probabilities can most predict the incoming text used for whatever the application.

Smoothing is the process of “flattening” probabilities that are in the language model. This is so that a reasonable word sequence can occur with probabilities rather than words having no probabilities or word combinations dominating probabilities. Smoothing starts with the process of expanding the distribution by moving dominating probabilities to regions with low or zero probabilities. Note that there are many different ways to smooth, one of which is simple smoothing, which makes the language think that every bigram occurs 1 more than it does on the source text.

Text generation with a n gram language model is possible. Generation of a text with the sequences of words, through the probabilistic of the occurrences of the text. Note that the larger the n in n-gram the better results. So a tri-gram would do better than bi-gram as more context is given to the model, allowing for better generation. But ultimately a

large corpus would be the best. A small corpus would fail to generate a good text. But this approach isn't the best as the model only has the context of n and given a large n, the time complexity would increase for building and generating.

Language models can be evaluated through metrics. Through the process of evaluation of extrinsics which are human annotators or intrinsic which are internal metric like perplexity.

The google n-gram viewer shows the occurrences of the text imputed throughout time. The sources of google are from literature all the way back from 1800 to 2019. We use the example of pokemon and pokemon. This is useful as we can see the explosion of pokemon go caused a increase of pokemon text on the internet