Cady Baltz
CS 4395.001 Human Language Technologies
Portfolio Chapter 8: Ngrams
3/3/2023

a) What are n-grams and how are they used to build a language model

An n-gram allows you to process text ‘n’ words at a time, rather than looking at single words individually. Unigrams take one word at a time, bigrams take two words, and trigrams take three. However, you can specify any value of n if you want to process even larger phrases.

N-grams are specifically used to build probabilistic language models. These models are useful because languages have certain sequence patterns that appear more commonly than others. Humans themselves learn a lot of language through pattern recognition. For example, if you hear the two words “Buckle your”, then you can predict intuitively that the next word will likely be “seatbelt.” N-grams allow us to incorporate this same probabilistic reasoning into NLP applications.

b) Applications where n-grams could be used

One application is text auto-completion. For example, in Google Search, when you start to type a search query, certain results are suggested to you automatically. If you type “how to make”, then Google suggests “how to make money” and “how to make a paper airplane”. To accomplish this behavior with an n-gram, you could treat “how to make” as a trigram, and then use a probabilistic language model to predict the next words the user may type in their query.

Another application is language translation. When translating one natural language to another, if you translated each word individually, you would produce unnatural results. By using n-grams, you can produce more natural language by using phrases that are more commonly spoken by native speakers.

A third application is speech recognition. If a machine is unsure exactly what word was spoken, it can help choose the word based on what the previous words were using n-grams.

c) How probabilities are calculated for unigrams and bigrams

You can calculate the probability of a unigram by simply dividing the count of that unigram in the text by the total number of unigrams in the text. This can be written as:

P(word) = unigram_count(word) / total_number_of_unigrams

To calculate the probability of a bigram, such as “how to”, you would need to use the following formula: *P(how, to) = P(how) P(to | how)*. In this formula, you calculate the probability of seeing the unigram “how” as described above.

Then, you calculate the probability of seeing “to” after “how” by dividing the count of the bigram “how to” by the unigram count of “how”. This can be written as:

P(to | how) = bigram_count(how to) / unigram_count(how)

You then multiply these two values to get the probability of the bigram “how to”.

d) Importance of the source text in building a language model

In order to create a probabilistic model, you need to calculate word occurrences from some existing source text (also known as the corpus). The choice of corpus will greatly impact the model itself, as the frequency of certain phrases will vary widely in different texts. For example, a bigram like “cell phone” would have a high probability if your source text was modern news articles, but an incredibly low probability if your source text was books written in the 1800s.

Additionally, the size of the corpus will also impact the model. If your source text is too small, a lot of n-grams will appear to have zero probability simply because they never happened to appear in the text. Generally, the larger the corpus, the more powerful your probabilistic language model can be.

e) Importance of smoothing

Because a corpus is not infinite, it is not possible that it will contain every possible sequence of words. Therefore, when calculating the probability of a unigram, we may get zero. This “sparsity problem” would cause calculation issues because, as described in part (c), to calculate the probability of a bigram we must divide it by the unigram count of the first word.

There are several ways to deal with the sparsity problem, and one of them is called “smoothing”. Smoothing allows us to replace zero values with very small, but non-zero, probabilities. There are several ways to implement smoothing. One simple smoothing method is called Laplace smoothing, which adds one to all counts, and adds the total vocabulary count to the denominator of calculations to balance this change out. This simple method allows us to remove denominators containing zero from our calculations, while preserving the relative probabilities of different phrases.

f) How language models can be used for text generation, and the limitations

Language models can be used for text generation by first calculating a probability for every possible n-gram. Then, to generate a phrase, the model can continually select the n-gram with the highest probability based on the last word from the previous phrase. Generally, this will be more accurate with higher values of n, as this allows for longer coherent units of text to be grouped together when generating text.

Of course, there are many limitations to this approach. While it may produce isolated phrases that make sense, it most likely will not produce large bodies of logical text. This is because the probabilistic model itself cannot account for the context or the actual semantics of the text being generated.

g) How language models can be evaluated

One way of evaluating language models is through intrinsic evaluation, which uses some metric to compare models. For instance, models can be compared based on their perplexity. To calculate perplexity, a small amount of the test data is set aside when creating the model. Then we use this data to calculate the inverse probability of seeing these words, normalized by the total number of words. Models with a lower perplexity also have a lower entropy, meaning we have less choices for the next word given a current word. Perplexity will vary based on how diverse the corpus is, as more specialized text will naturally have lower entropy.

Another way of evaluating language models is through extrinsic evaluation. In this type of evaluation, human annotators evaluate the results produced by the model using a predefined metric. Due to human involvement, this method tends to be more expensive and time-consuming, so it is used sparingly.

h) Introduction to Google's ngram viewer

One way to explore n-grams using a large corpus is through the Google Books Ngram Viewer. Through this interactive web app, you can select a corpus of books, such as "English Fiction", and compare trends in ngrams over any time period. Due to the size and large time period of the corpus, this is a powerful tool to observe trends in human language.

For an example of observing trends in this tool, I entered the bigram "electronic mail", the unigram "postcard", and the unigram "mailman". One trend I found interesting is how "electronic mail" rapidly grew in usage through the 1980s and 1990s, only to become even less common than the word "mailman" today. Another trend is the word "postcard" spiking in the 1950s, and it makes me curious as to whether postcards and leisurely travel were trendy during that time. Overall, I enjoyed using this tool to explore how new words and phrases have entered the English language over the years.

Google Books Ngram Viewer

electronic mail,postcard,mailman

1860 - 2019 | English (2019) | Case-Insensitive | Smoothing