**N-Grams**

**N-grams and How They Are Used to Build a Language Model**

An n-gram can be described as a sliding window over text n words at a time. Unigrams look at a single word at a time, bigrams look at two words at a time, so on and so forth. N-gram are used to build a probabilistic model of language. A corpus is used to generate the model and the corpus greatly influences the language model.

**Applications where N-grams Can Be Used**

Some common applications for n-grams include, spelling correction, machine translation, speech recognition, and auto suggestions for typing, messaging, and searching.

**How Probabilities Are Calculated for Unigrams and Bigrams**

For unigrams, the probability of word occurring in the corpus would simply be the number of unigrams that are that word divided by the total number of unigrams.

For bigrams, the probability of a sequence of two words occurring would be the number of unigrams that are the first word divided by the total number of unigrams multiplied by the count of the sequence of words (bigrams) divided by the count of the first word.

**The Importance of Source Text in Building a Language Model**

The source text essentially determines what sequences of words are more probable or not. A corpus using Shakespeare would have drastically different results than a science textbook.

**The Importance of Smoothing and a Simple Approach**

Smoothing is important because it takes care of the sparsity problem, our data can’t possibly contain every sequence of words. Without smoothing, a sequence of words that doesn’t exist would automatically be assigned a probability of zero. The idea behind smoothing is that zero values are filled in with a little bit of probability of the overall mass.

One simple approach is Laplace smoothing also known as add-one smoothing. The idea to add 1 to all counts so that none are zero. To offset this, the vocabulary count is added to the denominator of the probability.

**How Language Models Can Be Used for Text Generation and its Limitations**

A simple approach to text generation using n-gram models is to pick a start word and use bigrams to generate the next word. A function can be used to look through all the bigram probabilities and find the one with the highest probability in the first position, then the next word of the bigram is used as the new start word.

This type of generator is limited by corpus size and simple approach, typically higher order n-gram models work better along with a larger corpus.

**How Language Models Can Be Evaluated**

Language models can be evaluated by extrinsic evaluation where human annotators evaluate the results using a predefined metric, however this is time-consuming and expensive.

Another method is intrinsic evaluation using some metric to compare models, one such metric is perplexity. Perplexity measure how well and language model predicts text in test data.

**A Quick Introduction to Google’s N-gram Viewer**

Google’s n-gram viewer allows user to select a word or phrases showing how often those words have appeared in a corpus of scanned books available in Google books. They are represented as percentages of the corpuses over a period.

Below is an example of the words ‘uncanny’ and ‘awesome.’

A picture containing graphical user interface

Description automatically generated

Awesome and Uncanny have appeared more than they ever have before apparently.