N-grams

What are n-grams? How are they used to build a language model?

N-grams are sliding windows of length n over a given corpus (collection of text), where n is the number of adjacent words or characters from a sequence of one or more words or characters. Unigrams are n-grams of length 1 and contain a single word, where its next word in the sequence is the next unigram. Bigrams are n-grams of length 2 and contain two words at a time, where the window slides one word over each time to get the next bigram so that the second word of its previous bigram is its first word. Trigrams are n-grams of length 3, contain three words at a time, and slide the window one word over for each trigram. N-grams larger than trigrams are simply referred to as n-grams for each n greater than 3.

A probabilistic model of language can be built from n-grams created from a corpus. The corpus and dictionaries of n-grams from the corpus, including the count of each n-gram, can serve as training data. N-grams allow the language model to generate probabilities over the n-grams of words to determine how often the words occur in a natural language. When building n-gram language models, start and stop symbols may be added to the start and end of each sentence of the text.

Applications of N-grams

N-grams can be used in building probabilistic models of language to determine the probability of occurrence of words of a certain length from a body of text, as described above. They are useful in other natural language processing applications such as language generation or identification, speech or character recognition, word comparison, machine translation, spelling check, autocorrect suggestions, and glossary search.

Probabilities for Unigrams and Bigrams

Probabilities can be calculated for unigrams and bigrams of a corpus by first using NLTK and its `ngrams()` function to obtain separate lists of unigrams (tokens) and bigrams for the corpus and then creating dictionaries of counts for each list. Generally, the probability of a unigram u in a given text is the count of unigrams that are u in the text divided by the total number of unigrams, or tokens, in the text. The probability of a bigram b in a text is the count of that bigram divided by the count of the first word in the bigram. It is given by the probability of the first word in the bigram being multiplied by the probability of the count of bigrams that are b divided by the count of the first word in the bigram.

Smoothing can be used in code to compute the probability for a bigram and may use logs to account for underflow due to very low probabilities. Laplace smoothing multiplies probabilities of the bigram's unigrams together and uses (n + 1) / (d + v) for the test data, where n is the count of bigrams in the text, d is the count of unigrams that are the first word in the bigram, and v is the total vocabulary size of unique unigrams in the text being used as training data.

Importance of Source Text in Building Language Models

The source text is the corpus for building a language model. It is important because it can be used to create n-grams and provide the training data required to learn the language model. Each language model depends on what specific type of text it learns from, so the source text determines the probabilities over its n-grams of words.

Importance of Smoothing

Smoothing is important when computing probabilities of n-grams because it handles the sparsity problem that may result from zero counts of an n-gram in a given text. As dictionaries of data won't always contain every sequence of possible words from the source text, smoothing can fill in the zero values with enough mass for an overall smoother probability distribution.

A simple approach to smoothing is Laplace, or add-one, smoothing. Adding 1 to all counts of n-grams smooths the 0 count of n-grams because it ensures that any possible counts of 0 are not directly used in the probability calculation. The total vocabulary count of unique tokens is added to the denominator to balance the offset. Although Laplace smoothing is simple, it affects the accuracy of all the probabilities. Good-Turing smoothing results in a more accurate probability because it replaces any possible zero n-gram counts with words occurring only once.

Language Models in Text Generation

Language models can be used for text generation by converting n-grams to probabilities, using the text corpus to create probability dictionaries of n-grams. A possible approach to language generation is using a start word to locate the next word with the highest probability from the n-grams' probabilities and then adding that word to the phrase being generated. Text continues generating in this way until the last token signals the end of a sentence.

There exist limitations of this approach. The generator may be limited by the small size of the source text. As larger n-grams yield a better result, a large collection of text is needed. Creating the probability dictionaries of n-grams involves creating dictionaries of counts over a large corpus, which can also be a lengthy process.

Evaluating Language Models

Language models can be evaluated intrinsically, where an internal metric like perplexity is used to compare models, or extrinsically, a more costly method where human annotators use a predefined metric to evaluate the models. The size of text can help determine the quality of a language model. A small amount of test data can be used to calculate perplexity, which measures how well the language model predicts the text in the test data based on the number of available choices for the next possible word given a current word. Perplexity (PP) is the inverse probability of seeing words observed by humans, normalized by the number of words. A lower perplexity serves as a better evaluator.

Google Ngram Viewer

The Google Books Ngram Viewer is an online search engine that displays the frequencies of a set of given words, using counts of n-grams from Google's large corpus of books published in multiple languages between the years 1500 and 2019. The tool accepts input words as n-grams, compares them to matching words in the text corpus, and charts all words appearing in a minimum number of books in the corpus as a graph that shows how often the words were used over the years.

The following example shows the search results of the set of phrases "Hitler", "World War", "Great Depression", "recession", "evolution", and "Cold War":