Name: Bushra Rahman
Class: CS 4395.001
Assignment: N-grams

How N-Grams Are Used

N-grams suggest a number of words for a given text, and the word suggestion depends on the text. The number of words n-grams goes over depends on the mode, which can be: unigrams, bigrams, trigrams, or n-grams. If the word is unigram, the model would go over only one word at a time, and make suggestions for that word. Hence, n-grams use a probabilistic approach to suggest solutions to the text or corpus. Thus, the language model depends on the probabilistic approach that n-grams use.

Applications Using N-Grams

Because n-grams are used mostly in text, most applications that use n-grams are related to text predictions (such as auto suggestion), translation, correction, and speech recognition.

Calculating Probabilities of Unigrams and Bigrams

The probability of a unigram occurring in the text is given as P(w1) = count of the unigram w1 / total number of unigrams. The probability of a bigram is given as P(w1, w2) = P(w1) * P(w2|w1). The probability P(w2|w1) is given by dividing the count of the bigram w1, w2 by the count of the first unigram w1 in the bigram.

The Importance of Source Text

The source text is important to determine word suggestions for the text, which means that if the text was in English, then the model would suggest English words, or if the text was in Spanish, it would suggest Spanish words. Also, the context of the corpus is important to determine words for the model, meaning if the text was related to medicine, it would suggest medical words.

The Importance of Smoothing

The probability of an n-gram, assuming that it only looks at the previous word, is the product of the probabilities of each bigram in the n-gram. For example, the probability P(w1, w2, w3) = P(w1) * P(w2|w1) * P(w3|w2). Each probability itself is given as P(w) = count of words / total number of unigrams. However, if a word isn't present in the text, its count and subsequently its probability and the probability of any bigram containing it will equal to 0, thus making and the overall product multiply to 0. This sparsity problem is solved by smoothing, which smooths distribution by replacing any 0 probabilities with some other value in the overall probability. A simple approach to smoothing is Laplace or add-1 smoothing, which adds 1 to every count to ensure none of them are 0. To balance out this out, the denominator (the total number of unigrams) gets added to the vocabulary size, which is the number of unique tokens.

Text Generation Using A Language Model

A text generator can use the probabilities of n-grams from some source text to predict sequences of words to concatenate into generated text. If given a unigram to start off with, a text generator can cycle through n-gram probabilities to find which n-grams have the highest probability of starting with that unigram. From there, n-grams can be concatenated together. N-gram probabilities for a higher *n* value will return more accurately generated language (eg. trigrams are more effective than bigrams).

Evaluating Language Models

Language models can be evaluated via extrinsic or intrinsic evaluation. Extrinsic evaluation entails human annotation of the generated text based on a predefined metric. This form of evaluation is time-consuming and is typically saved for the end of an NLP project. Intrinsic evaluation compares models using a predefined metric, such as perplexity, which measures how well the language model can predict text in some amount of test data.

Google's N-Gram Viewer

Google's n-gram viewer (https://books.google.com/ngrams) is a search engine for the frequencies of n-gram phrases in Google's books corpora. The viewer displays a graph showing the distribution of the phrase's frequency over time in years. Different smoothing options can be chosen to smoothen the distribution. The following example shows a graph for the frequencies of "natural", "language, and "processing" between 1800 and 2019.