CS 4395 - Assignment 5: Narrative

N-grams are groups of words found adjacent to one another in a text with N words. Hence, N-grams can be Unigrams, Bigrams, Trigrams, and so on. N-grams are used to build language models by analyzing which words are used in conjunction with other words. This is then used to train a model that can assign probabilities to the appearance of each N-gram in a training set.

N-grams could potentially be used in applications with spelling and grammar correction functionality which can be seen in applications such as Microsoft Word and Google Docs. It can also be used for language detections software by comparing the probability's of different N-gram models of different languages.

The probabilities for unigrams and bigrams are calculated by using a training file which contains a large volume of text (a corpus) from which you can then count the appearances of individual tokens for a unigram and adjacent tokens for bigrams. To calculate the probability of a unigram you simply divide it's appearances by the total number of unigrams (tokens). For bigrams, you divide the number of times a bigram appears in the training set by the number of unigrams.

The source of the text used in the building a language model is extremely important, as this model is going to be what is used to calculate different probabilities of each N-gram. The bigger the size of the training set, the more useful the language model will be. A training set should also have a highly varied vocabulary, since a training set with insufficient unique tokens will be unable to provide accurate probabilities for N-grams when using the model to analyze patterns in text for an entire language.

Smoothing is important to prevent an N-gram with a probability of zero to make the total probability of a text zero. When calculating the probability of a bigram in a text, using LaPlace smoothing can be used to this purpose, by adding +1 to the bigram count and adding +V to the unigram count in the denominator, where V = size of the vocabulary which results in the equation probability=(b+1)/(u+V).

Language models can be used for text generation by using the different probabilities for various N-grams to find the word or words with the highest probability of coming after a given word or words. This has limitations however, as this can create extremely long sentences that make no sense, create sentences with no clear subject, or create sentences that are grammatically incorrect based on how the text generation is implemented.

The efficacy of a a language model can be evaluated by how well it is able to evaluate real languages, predict text, and create understandable grammatically correct sentences. This is all determined by the training corpus used to map out the n-grams and their corresponding probabilities to occur in a text. The corpus must also ideally be very large. It must also be grammatically correct and have an extremely varied vocabulary to be able to adequately model a real language's complexity and nuance.

Google has a tool that can be used to find the probabilities of various n-grams using texts training texts from different periods of time. To use it, you simply type words or phrases into the search bar, separating them with a comma and press Enter. A line graph will appear that will plot the different probabilities of each n-gram over different periods of time, where the x-axis is the time in years and teh y-axis is the probability. For example: