Arjun Venkat - asv180003
Arjun Balasubramanian - axb200075

N-grams are continuous sequences of words or symbols or tokens in a document. They can be defined as the neighboring sequences of items in a document. Essentially, they use the frequency of words or word sequences occurring in a text to predict the next word/token using probability. N-grams have many possible uses, including predictive text input, machine translation, and speech recognition. For a unigram model, for a given sequence of tokens, you just multiply the probability of each individual token appearing in the text to get the final probability of the sequence. For a bigram model, you multiply the probability of each set of two tokens in the sequence (which itself is calculated using the probability of the token given the probability of the previous token). The main importance of the source text is that it needs to have a large variety of words present roughly in the natural frequencies that they would be in a normal usage (in that context). For example, if a model is trained on a text that heavily uses one particular word, then it will be biased towards that word and predict it more often than it should.

The importance of smoothing is that is can help is with seeing trends in data. Without smoothing data can have trends, but the inherent randomness of data can obscure the trend and make it difficult to make out for humans and sometimes the model itself. One potential simple approach to smoothing would be to group data into the average of an n number of points, for example if we had 1000 datapoints after this process we would be left with 100. By taking an average we can reduce the effects of the randomness and see the trend a little more clearly.
The naïve approach for using n-grams to generate language is by starting with a start word and using the previous n-1 words to predict the most likely next word in the sentence. For example, if you're using bigrams, you would use the start word to predict the next word, then use the predicted word to predict the next word and so on. A limitation of this approach is that the sentences don't make sense grammatically since the part of speech isn't considered, another limitation is that only the previous n-1 words are used to predict the next word rather than all text generated so far.
One way you can evaluate a language model is to use human annotators to determine how well a language model is doing. However, one drawback to this is that it can be very subjective as well as expensive and time consuming to have people annotate the outputs by hand. A cheaper method would be using perplexity to determine however it does not have the complexity of human analysis.
Google N-gram viewer is a site that searches all corpuses of books in google books to looks for occurrences of specific n-grams and shows the number of occurrences over time. Below you can see an example using the words "machine learning" and "language processing".