N-grams are sequences of items of n-length derived from a corpus of text. They are based on the idea of conditional probability, in that the probability of an item being last in an n-gram is dependent on all of the probabilities that the sequence before it was seen. For example, a bigram consists of two items, where the probability of the last item is dependent on the probability of the first item. Since an item is in actuality dependent on the entire sequence before its occurrence, not just the prior sequence in the n-gram, the n-gram model is a generalization motivated by the need to get adequate counts of text to calculate probabilities.

N-grams are then used to build a language using conditional probability. Given that we see an item, and it is dependent on all the other items in the n-gram, we get P(A | B), where A is w(i) and B is (w(i-1), w(i-2), …, w(i - n + 1). We can then transform this using conditional probability to P(A, B) / P(B), which allows us to simply get the counts of such n-grams and (n-1)-grams in the corpus and estimate the probability using MLE. However, because we do not want division by 0 or any 0 probabilities (because probabilities are multiplied together), we need to perform smoothing, which is at its core adding one to the count of each n-gram in the vocabulary. In this way, these probabilities and counts form a language model capable of predicting the most likely next word in a text and in such a way can either classify text or generate new text. Applications for n-gram models are mostly in NLP, where n-gram models could be used to generate text, classify text, translate text, etc.

As such, source text (or a corpus) are vitally important to n-gram models, and all language models in general. For the n-gram model, source text is needed to learn anything about language. If there is no source text, all words are equally likely and therefore entirely random. For models based on neural-networks, weights are initialized for nodes. Without source text, training cannot be run, and because backpropagation is the method by which neural-networks learn, the language model is just the initialized weights (which are often initialized randomly).

As described previously, n-gram models can generate text by predicting the most likely next word given n-1 previous items. Although this works reasonably well for shorter sequences, because n-gram models are bad at remembering context (because they only take into account a narrow window of context), the generated text can quickly become nonsensical.

Language models can be evaluated using simple accuracy metrics (such as number correctly predicted / number of predictions), which is an extrinsic metric derived from annotated data from humans, or by intrinsic metrics like perplexity, which is essentially just cross entropy. Perplexity = 2^cross entropy. I'm honestly not sure why this is done, but it might be because perplexity is somewhat easier to read than cross entropy.

Google's n-gram viewer is essentially a large database containing counts for n-grams throughout history. For example, if I put in an n-gram 'Hello world' into the n-gram viewer, I see that the n-gram is not found frequently before 1980, where the frequency explodes over the course of 20 years. I think this is funny because it is likely this jump in frequency correlates to

the widespread distribution of computers and people learning how to code ‘Hello world!’ programs.