N-grams are a sequence of n items from a given text. They are stored as a series of tuples with n items in each. For example, a unigram is a sequence of single tokens, a bigram is a sequence of tuples with two tokens in each, and so on and so forth. They are used as a way of calculating the most likely word to come after a given word, based on the sequence of words in the ngram. With a bigram, a model can find the most likely word to come next, and as you get longer tuples you can start to predict farther and farther ahead. Some applications of n-grams are grammatical autocorrect, sentence prediction like on mobile keyboards, and AI novel generators like NovelAI and other such applications.

The way to calculate the probability for unigrams and bigrams is as follows. Take the number of bigrams plus one, then divide it by the number of times the first token in the bigram appears plus the total number of tokens. The importance of the source text in building a language model is that the model's entire knowledge of the language is from the text. If, for example, the source text only uses a narrow subset of the entire language, then the model would only be able to recognize words and phrases from said subset. On the other hand, with a good source text that covers more of the language, the model becomes more useful and more effective.

Because of this, smoothing becomes important in making language models effective. What smoothing addresses is the imbalances that are caused by the fact that any given source text isn't going to be a generalized sample of the given language, it's going to be specific to whatever domain the text is about. Thus, smoothing flattens out the peaks in the probability distribution, converting the specialized model into a more generalized one. Without this, if say the model were trained on a medical textbook, it would not be able to recognize a physics textbook as the same language, even if it were to happen to be so.

Due to the way n-grams record the sequence of tokens as well as the tokens themselves, language models can use that to, given a starting set of words, calculate which word is most likely to make sense to come next. For example, if it has the word "of", it can look at the bigrams and know that the most likely word to come next is "the". The main limitation of this is that in order to ensure that the generated text continues to make sense after a few words, longer and longer n-grams are required, meaning that the same piece of source text can end up requiring more and more space to store it, and taking longer and longer to calculate the next word, as more and more factors need to be considered.

The most common factors that language models are judged by are perplexity, or how many possible options it can come up with for the next word, and cross entropy, or how much information is produced with each generated word. In general, a good model has minimized cross entropy loss and high perplexity.

Google N-gram viewer is basically a search engine that lets you look up the frequency of a given term or set of terms over time. For example, when I entered the phrase "Sherlock Holmes", the N-Gram viewer tells me that the Ngram first appeared in 1761, before reappearing in 1885, increasing over time to a peak in 2019, with no sign of slowing down. His most recent jump in popularity started in 2004, as a series of movies about him were produced over the next few years with varying success, as well as more recently him appearing as a playable character in a fairly popular mobile game.