N-grams is a technique to create a list that contains the subset of all N number groups of grammatical characters and words in a text such that they are adjacent to each other. For example, a unigram is a list of 1 grammatical character and word such as [cat, dog, mouse, .]. The bigram and trigram have a list of groups of 2 and 3 grammatical characters and words respectively. For instance, [mouse in, in type] and [hype house in, house in type]. Note that while the list for the n-gram and the number of words/characters are defined, we have not specifically talked about the contents yet. N-grams utilize the sliding window technique across a sentence. Take the sentence: “My mother ate spaghetti.” The bigram generated from this sentence would be [My mother, mother ate, ate spaghetti] (sometimes stop symbols are included, not in this case).

With that in mind, n-grams have definitive applications in creating language models. By utilizing the n-gram to generate a probability model, we can create an algorithm that takes a word or sentence as input and outputs either the likelihood of whether the input belongs to a specific n-gram dataset or the most probable next n – len_input word. N-grams can be used to generate sentences to easily describe a particular subject or piece of data depending on already inputted associations. Additionally, they have capacity of recognize certain languages or potentially, more abstractly, characteristics of language that may inherently exist in a piece of text.

Because n-grams are linked with probability models, it is important to discuss how it is that the n-gram has its application in manufacturing a probability model. To briefly explain, the algorithm takes an input of text and then generates an n-gram from the input text. Then, the algorithm decidedly uses an n-gram and unigram (simply just a list of tokens) provided dataset for use to grab the count for each n-gram group that are in the n-gram dataset. Afterwards the algorithm divides it by the total number of tokens from the dataset. Finally, we do this continuously for every n-gram of the input text and apply Bayesian probability to generate an appropriate probability.

As I previously mentioned, the dataset helps to be able to calculate the probability. Thus, the source text or dataset having a notable and defined characteristic that is meant to be intrinsically drawn from the source material. If the probability model were to use source text with no real discernible discretion, the output would most likely be unusable unless the intention was to gain insight on patterns that may exist in the text over continuous sloven runs.

Additionally, an important step in coming up with an appropriate language is smoothing the probability. This means to apply some methodology such that the probability distribution for n-gram inputs are significantly smaller or larger than any other. A simple approach to smoothing is to add one to all the inputs regardless of whether it exists or not in the n-gram dataset. To compensate for this, the algorithm adds the total number of unigram token to the total vocabulary size.

Language models can easily be used to text generation if the dataset is chosen in a manner that's consistent to bring out a certain piece of content. However, one of the stark limitations of this is that the semantic definition and contextual clues of the sentences tend to be lost. For example, take the phrase, "Jack Be Nimble, Jack Be Quick". The text generator has the possibility to generate "Jack Be Fast" which is not contextually and semantically correct.

To evaluate language models, it comes under the scrutiny of the algorithm creator to tell the algorithm whether the language model's outputs is appropriate. It is most likely favorable to say that language models that produce/recognize sentences that have make grammatical and semantic sense.

Google's n-gram viewer works by taking a string and searches for the probability that the string appears from a large corpus of printed material published between 1500 and 2019. An example of this is the phrase "follow your passion" comes out to a very small percentage that appears to have a steady increase from 1986 to 2008.