What are N-Grams

An N-gram is simply a sequence of N words.
the N-gram model. Basically, an N-gram model predicts the occurrence of a word based on the occurrence of its N – 1 previous words.
By leveraging these statistics, n-grams fuel the development of language models, which in turn contribute to an overall speech recognition system.

How N-Grams Work

A model is trained on the data, learns to pick up on patterns, and then applies that “knowledge”. As always, more data is better. Just like we learn through experience, practice, and repetition, an ML model learns to discern patterns by observing many, many instances.

Looking specifically at n-grams, the pattern here is the relative frequency counts of words that follow any given string of words. Let’s take, for example, the following sentences as our training data:
She earned a degree in computer science.
We taught the computer to think.
Computer science is my favorite subject.
He uses a computer everyday.
We’ve made a lot of advances in computer science.
In computer programming, patience is everything.
Now we’ll break it down a bit. If we’re only considering the word that comes after “computer,” we have “science” appear 3 times, and “to”, “everyday”, and “programming” each appear once. This means “science” will have a frequency count of 50%, while the other three words will share the remainder for a little less than 17% each.
However, if we look at the string of words “in computer”, we have two instances of “science” and one of “programming” that follows. That equates to a 66% score for “science” and 33% for “programming.” Dan Jurasky explains this concept well here:
There is also N-gram smoothing, which is crucial for speech recognition systems. Basically, you still want to be able to predict the unforeseen data… so you need to allow for a certain likelihood density to be shared with unforeseen words.
What’s different between these two examples is the number that we assign to n in our n-gram. Essentially, an n-gram model looks at n-1 preceding

words. So a bigram, for instance, will only look at one word, such as computer. A trigram, on the other hand, considers the previous two words, like in the case of “in computer”.
Larger values of n are almost always more accurate than lower ones, but it comes with a trade-off: more computations means we need to use more resources for a longer time, giving us a more expensive and slower language model. In some cases, less is more. For instance, we often only need a bigram to figure out the next word after “Merry”. Other times, higher values for n are necessary.
Basically, what this comes down to is approximating a probability with just a few words rather than the entire history of what came before.
Mathematicians may recognize this as an application of Markov chains. Without going into too much detail about the underpinnings of this technique, a Markov model uses statistical analysis to guess an unknown variable (or word) based on a series of known values (or preceding words).

How N-Grams Are Used in Language Modeling

The most obvious first application of n-gram modeling for natural language processing (NLP) is predicting the next word, such as we commonly see in text messaging and email. Since the entire use case revolves around predicting what comes next based on what came before, the n-gram model is often the only ML technology that we need to achieve this task. This is very much not the case with many other uses of n-grams, which rely on a mixture of tech coming together to form one coherent engine.
For example, auto-correct systems, including the ones that correct grammar and spelling in word processors, also use n-gram models. Identifying whether to use “there”, “their”, or “they’re” in a given sentence, for instance, depends on the context in which they appear, and, thus, it makes sense to use an n-gram model. Applying the rules of grammar, however, requires us to look at sentences as a whole, and so what comes next may factor in just as much as what came before. This is still a pattern recognition problem that’s solvable by ML applications such as Deep Learning.

Even more complex speech recognition tasks, including voice assistants and automated transcription, use n-grams as just one small part of an overall tech stack to convert the spoken word into text. Before we even get to the n-gram, an acoustic model will analyze the sound waves that constitute speech and convert them into phonemes, the basic building

blocks of speech. Another type of language model will then translate those sounds into words and the characters that constitute them.
In this case, n-grams often serve a supporting role by performing what's known as "search space reduction." Essentially, the n-gram will look at the previously transcribed words to narrow down the possibilities for the next word. That way, instead of having to search through a large lexicon, the computer "gets an idea of where to look" and can accomplish its task more quickly and efficiently.

Final Thoughts on N-Grams and Rev AI

By combining n-grams models with other speech recognition algorithms like end-to-end (E2E) Deep Learning, Rev's team of data scientists and speech engineers have created one of the most advanced automated speech recognition (ASR) systems on the market. In addition to the algorithms themselves, our immense datasets—derived from the hard work of our over 60,000 human transcriptionists—gives us unparalleled access to high quality data that we use to train them.
The result is language models that are proven more accurate than those from major companies like Microsoft, Google, and Amazon. If you're a developer who wants to bring speech recognition into your product, you can do so instantly with rev.ai, our speech-to-text API. Get started now and see results in a matter of hours instead of weeks.