Open in app

Following

591K Followers

A Lite Introduction to Markov Chain

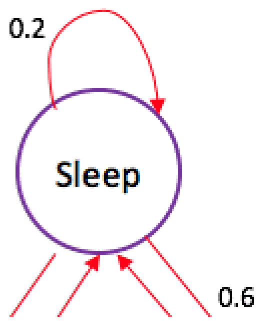
Vonn N Johnson May 17, 2019 · 5 min read ★

Previously I wrote a lite introduction to Latent Dirichlet Allocation as part of a deep dive into what makes Natural Language Processing (NLP) so great. NLP will be picking up steam over the next couple of years (until 2025 to be exact) as Artificial Intelligence continues to grow. It is one of the key drivers, and as it continues to be a part of our daily we are only going to understand one on a fundamental level, and that is Markov Chains.

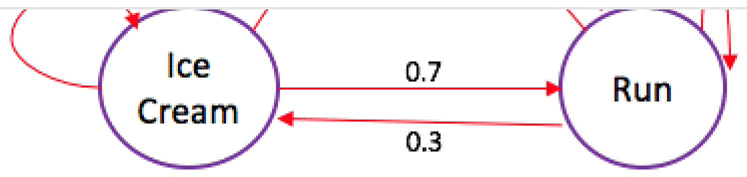
To start Markov Chains is not only used for NLP. There are other applications for it with numbers, but we will solely focus on how it relates to NLP for the moment. Also, there is a lot of terminologies associated with Markov Chains (e.g., Transition Diagram, Transition Probability Matrix, and Initial State Distribution Matrix) and mathematics that we will not discuss. This article is a lite introduction to Markov Chains. In later series, we can do a deep dive but first, let's get a basic gist of it.

Markov Chains, its namesake is the Russian mathematician Andrey Markov. Defined as a "…stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event." A stochastic model is one of randomness. We are trying to discover based on this variable and its associated variables what will happen next. This model is, in fact, incredibly simple to understand, but fundamental to this understanding is the Markov Property.

The Markov Property states that we do not know all of history, just the previous history only, we are memoryless. Making a prediction requires just the last event and the likelihood of future events. Imagine today is raining, and you do not know what other weather events took place in the past. However, you do know that tomorrow there is a 50% chance it will rain, a 30% chance it will be sunny and a 20% chance it will be cloudy. Choose your adventure. Let's go with the highest probability. Now moving on to the next day we again forget what has happened the previous day and know that today it's raining, for simplicity, the probabilities are the same, and we pick that it will be sunny this time, a new state, which means new probabilities for something to happen. What if the probability of more sun is 10%, the probability of rain is 20%, but the probability of snow is 70%?



Open in app



If you look up Markov chain, you'll probably see images like this floating around. Let's break this down, starting in ice cream. It's a bit dizzying so bear with me. If you have ice cream, there is a 10% chance you will have ice cream again, a 20% you will go to sleep, and a 70% chance you will go running. If you run, there is a 60% chance you will run again, a 30% chance you will eat ice cream and a 10% chance you will sleep. If you sleep, there is a 20% chance you will go back to sleep, 20% chance you will eat ice cream and a 60% chance you will run. It looks like the universe, aka Markov Chain, is trying to tell you something: Run!!

Applied to Natural Language Processing, we can use this to predict words. Given the word "I," what words would come after it? Well in a corpus of text we can find all the word s that do come directly after I and put them in a dictionary with "I" as the key and all the words associated with it as the values. If a word shows up more than a couple of times, then its probability is raised, too few and its probability drops. From there, a word is selected, and we move on to the next key: value pair to make another decision. This process goes on and on as long as you see fit through the function you create in your code.

The problem with using this with NLP is the understanding of how well the model is doing. This model is what is known as a compartment model where we specify the different states under consideration, and we determine the probabilities of moving to the next step no matter how many there may be. Were we working with numbers this could be easy to evaluate how it does. A two-state model of Alive or Dead illustrates this. Let's say we have 1000 people alive, and you have a 5% chance of dying and a 95% chance of living. With these strict numbers, we can create expectations for ourselves for each iteration. In the first iteration, 50 people die, and 950 live, second year 50 more die and 900 are alive. We can keep going and going to set the expectation, and then we can run our model to see how close it gets to those numbers. This example makes it easy to used it for text generation, and the only way I could evaluate it was to determine that it didn't sound grammatically correct, just a jumble of words. The single actual evaluation I could do was compare it to a sentence that the Markov chain created. In the future, if you are going to use Markov Chain to do text generation, which it is primarily used for in NLP, try comparing it to the text generated by a Recurrent Neural Net model and see how well the Markov Chain made version holds up.

Machine Learning Probability Explained Markov Chains