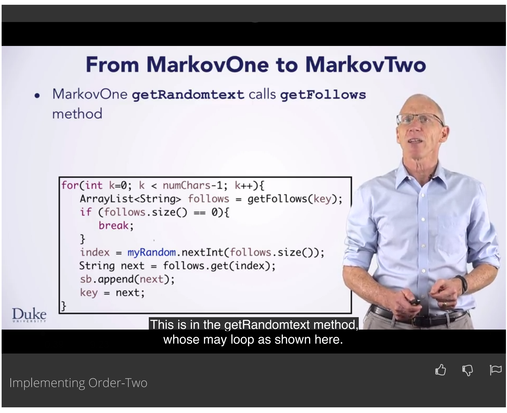
Generating Random Text



Implementing Order-Two

Have a question? Discuss this lecture in the week forums.

Interactive Transcript

Search Transcript English

0:03
Hi. Now you've written a MarkovOne class which generates random text based on what follows the most recent letter. That's doing a bit better at making realish seeming text but it's not doing great. What if you made it look at the previous two letters? I bet it would be even better. You're going to start with a MarkovOne.java class as the basis for implementing MarkovTwo.java, to make this improvement. And you could easily extend this MarkovTwo class to MarkovThree. Or more generally to an order end Markov class. The MarkovOne class you developed earlier, calls a getFollows helper method. This is in the getRandomtext method, whose may loop as shown here. We'll see the entire message on the next slide. The getFollows method uses the algorithm we developed with the seven step process earlier. It returns a list of all characters that follows the key. We're using one character strings, here rather than characters. Depending on the value of the array list returned by getFollows, the loop exits when no following characters are found. Or the list returned as used as the source of a randomly chosen character, as the Markov process generates random text. Here the .netInt method chooses a string at random from the array list. The random string is added to the string of random text we're building, one single character at a time. Then the loop continues with the random string as the next one character key, and its follow list will be found the next time through the loop.

1:32
We'll look some of the details of getRandom text, in preparation for developing Markov Two, and then the more general Markov model classes. Here is the entire getRandomText method, with the four loop from the previous slide not shown. The key, is a one-character string, chosen at random from all valid indexes of the training text. We don't treat the last character of the text as valid, since it doesn't have have a follow character, and that's why we used myText.length minus 1 as a value for generating a random index. The index is used to create a one character string as the key. And that's why we used index and index+1 as the arguments to the substring method. As we get ready to write code from MarkovTwo, think about what changes we'll need to make this code work for a two-character key instead of a one-character key. Let's get coding. I'm using the MarkovBasics project, and I'm gonna go from MarkovOne to MarkovTwo. And we'll see that that makes a pretty big difference in the quality of the random text being generated. Here I'm in the MarkovRunner program that creates a MarkovOne object, and then generates three different random texts from it. So, I compile that class, and it compiles just fine. Now I'm going to make a new MarkovRunner object, and then I'm going to invoke the method in it called runMarkov.

2:59
I'm gonna pick some random text and generate from it. I think I'll use Romeo and Juliet by Shakespeare, and when I generate that random text, we can see this doesn't exactly look like Romeo and Juliet. It's kind of hard to see from lisullagonghe w eayomyon anooff bencasel, that it looks like Romeo and Juliet. This is a MarkovOne text generation, which means that each character Is used to predict the next one.

3:25
As you may remember from the lesson, if I'm in my MarkovRunner class, and I simply change MarkovOne to MarkovZero, And run this program, by compiling it, and it had no syntax errors. Now when I run it, by creating a new object

3:46
using the run Markov method, and it's generating another random Romeo text. This looks even less like Romeo. OtrshFrn f so t ehn o. Now, this does have a lot of Es and As, because it chose text at random, based on the frequency with which it occurred in the training text.

4:05
One of the features of our design is that using MarkovZero in place of MarkovOne, still allowed my client program to work, because both MarkovZero and MarkovOne rely on a getRandomText method and a setTraining method. That's all they rely on. We'll be able to catch that commonality with an interface in a later lesson. For now, we're going to create the MarkovTwo class using the two characters of prediction, and run that. When I compile it, it says cannot find symbol method setTraining. So the MarkovTwo class doesn't have these methods. Lemme open it up and see what's going on. Whoa. MarkovTwo is just the shell of a program. It has no methods in it. In fact, the comment says, copy MarkovOne and make changes to it. So I'm going to get rid of this comment.

5:00
I'm going to open up my MarkovOne class.

5:05
I'm gonna copy the guts, the complete guts of MarkovOne, by dragging all the way to the bottom. BLANK_AUDIO] Copying that.

5:16
And then pasting it to MarkovTwo.

5:21
Now, I know, that the name of the constructor must be MarkovTwo rather than MarkovOne.

5:28
Now when I compile it, this class compiles fine. I don't need MarkovOne anymore.

5:35
And when I go to MarkovRunner, which now uses MarkovTwo as the name of the class, when I compile this class, it works just fine. Because I copied the guts of MarkovOne into MarkovTwo. In general, copying code from one class to another, is not a great software design technique, but it's what we have available to us at this point, until we develop an interface to capture the common code.

6:02
As I look through this code, I wanna understand the differences between MarkovOne and MarkovTwo.

6:09
The get training text method, the setTraining method is the same. The getRandomText method is slightly different, because I'm going to use two characters to predict the next one. Rather than just using a single character predict the next one. So as we can see here.

6:29
This picks one substring as my initial key, and that substring has a length one, going from index to index+1. I'll change that to go from index to index+2, and that will mean that the value of a valid index could only go up to two before the end. I need to be able to pick any index that allows for a two character substring. So I need to change this one to a two, this one to a two, and I'll need to change this one to a two as well, because this group has already had two characters generated and stored here as the key. So I only need to generate num chars minus two. When I compile this program, it works fine with no errors. I'm gonna use MarkovRunner, compile it, create a new MarkovRunner object, Run the Markov method. Use Romeo.

7:28
And we can see here, that the text looks a little more realistic like English. I can see words like Tyed, which might be the Montagues and the Capulets somehow, home is here, pandr is here, this cade. These words look more, even right here Roman instead of Romeo. So I have a little more English quality in my text, by simply changing the ones to twos. I might go one step further and change the twos to threes, just to see that I'm on the right track here.

8:04
Now I will be generating an order three Markov text. If that works, I'll have pretty good confidence that my changes were correct.

8:18
And I'm hoping that this looks even more like Romeo.

8:26
That seems a little more like Romeo, wherefore art thou Romeo? I can imagine that this is like that, so I'm reasonably happy and satisfied that my changes worked as they were supposed to. Because I'm writing MarkovTwo and not MarkovThree, I'm gonna change my threes back to twos. That was me simply verifying that my changes were correct.

8:54
As a reminder, because getFollows, the helper method that you see here, works regardless of the size of the key I'm using, because it uses things like key.length. And key.length. My changes, in changing the ones to twos were good enough. I've got MarkovOne, MarkovTwo, and I could easily make Markov n for a n character Markov generation. I'll wait until we do interfaces for that. Happy programming!

Downloads

Lecture Video mp4
Subtitles (English) WebVTT
Transcript (English) txt

Would you like to help us translate the transcript and subtitles into additional languages?