Generating Random Text

Word N-Grams

Review

There is a mistake in Owen's getFollows method! Think about where you want to start searching after the follow word for each key word is found.

Continue

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

Interactive Transcript

Search Transcript

English

0:03
Hi, we'll develop and test the code for MarkovWordOne. We'll copy the helper method, getFollows for MarkovOne. We generally don't like to copy code but you haven't learned yet the right way to abstract this out to avoid duplication. You'll learn a better way than copying the code soon. Since myText is now an array of strings, rather than an array, we'll need to make some changes. Some changes will be easy. We'll need to change the string method, .length with parentheses to just .length without the parentheses to find the number of strings in the array myText. We'll have harder changes, too. Since we'll need to replace the string .indexOf method with a helper method to return the index of a search string key starting at a specified index in an array. Let's get started. Hi, in this coding demo, we're going to look at the MarkovWordOne class, that uses an order one Markov process but with words rather than characters or letters. To see how this works, I'm going to use our MarkovRunner class, and create an object of type MarkovRunner, on my object workbench, and invoke the runMarkov method. This is using an order one text, I'm gonna use confucius, and see what kind of random text we have.

1:20
We've seen this before, and we can see here that it printed out, I'm using the Markov one class. This is text that's a little difficult to understand, because each letter predicts one letter that follows it. We're going to use the same code and the same ideas as you've already seen, but with words instead of letters. The way I'm going to do that is to modify the code in runMarkov to use MarkovWordOne rather than MarkovOne. There's MarkovWordOne, MarkovWordOne, now I'm going to compile this code. It compiled with no syntax errors.

2:01
I'm going to come to my object workbench and make an object. I have to make a new one since I just recompiled it, and now I'm going to use the runMarkov method again with confucius, and I see that I get one random word of text, rather than the 200 that I asked for. Notice I am using MarkovWordOne, it prints the name of the class. Courtesy, whether, and children. I'd like to get 200 words rather than 1. Let's see where that code is letting me down in the Markov class. So in MarkovWordOne, we've already seen in previous examples that the setTraining method is done for us. I'm using an array of strings rather than one string.

2:47
I've written the getRandomText method, and that works just as the method worked in MarkovOne for letters by calling getFollows, and it's the getFollows method that I'd like to look at. Not surprisingly, this method returns an empty array list, since I haven't written it yet. I have to fill the follows array list with each word that follows my key. The way I'm going to start that is to take the code in MarkovOne for follows, and copy and paste that into the function that I am writing. Now copying and pasting is not really a great idea. But until we learn about abstract base classes in a future lesson, I don't have much choice. So I'm going to come into my MarkovWordOne class, make some space for this code I just copied and paste it.

3:46
Forgot one letter there. Now if I try to compile this it's likely not gonna compile. Cannot find symbol- method length(). In my new version, my myText variable is an array. In the previous version with letters, it was a string. Strings have a .length method and arrays have simply a .length. That's two places that I'm replacing the length method with the length field that's in the array class. When I compile this, cannot find symbol IndexOf. Strings have an IndexOf method that allows me to find the location of this key starting at a given position. I'm going to have to replace the string method IndexOf with my own helper method that's inside the class. We can see here that it has three parameters, the array of strings we're searching for, the target we're searching for in that array, and the location. I'm gonna simply replace myText.indexOf with the helper method, indexOf myText. Now myText is the parameter, rather than being the object on which the indexOf method is invoked, as it was with strings.

5:06
Cannot find method substring. In strings, to find a specific character, I used substring and said, start here and take this many characters. In my new model, because this is MarkovOne, I simply need to get

5:23
the text that starts at start+1. The reason I'm using start+1 is that I have a one character look ahead. So I just need the next character, rather than the general code that I had here, to make that clearer in case It's not quite clear enough. In a word one model, I'm using one word to predict the next word. So, since I found the word at index start, I take the word after that and add it to my follows array.

6:00
My class is compiled, and there are no syntax errors. If I now compile MarkovRunner again,

6:12
And then make a new object, and invoke the runMarkov method, on confucius, I still get one word, a different word because it was random. And the reason I get only one word is because although I've made my follows method work, at least compile, which gives me high confidence it's gonna work. My IndexOf methods, which searches through the array of string words for this particular target, starting at a given position, simply returns -1, meaning the string wasn't found. So I need to search for every string and return the location. I'm going to start my search at this location, given by the parameter start.

6:59
I'm going to search through all the words,

7:02
which I use the length of the array to tell me when the end is. And I'm gonna increment my variable each time.

7:08
If the word I'm looking for is my target, I found it.

7:14
If words sub k is equal to, now if equal to were strings I need to remember to use .equals, if that's equal to my target,

7:24
I'm simply gonna return the index at which I found it, which is k. So, I've taken this for loop, which is a standard loop over all the variables, the values inside words, starting at start, and going through the length of the array. If I find the word I'm looking for, which means words[k].equals(target), I return the index because this returned the indexOf the first occurrence of target that occurs on or after location start. If I make it all the way through this for loop without finding it, the loop exits and I'll return -1, meaning it wasn't found.

8:04
This code compiles. I'll now compile MarkovRunner.

8:10
I'll create a MarkovRunner object on the object workbench.

8:15
I'll invoke its runMarkov method with confucius and now, I have 200 random words. Vassals are jade is past. The Master said, when the mad-head of the bounds of Ch'i told how about the project, this is nonsensical text but each word is real because I chose words at random. As they say, in the house of lot, vexes, for five seeds. Enjoy the code.

Downloads

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

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