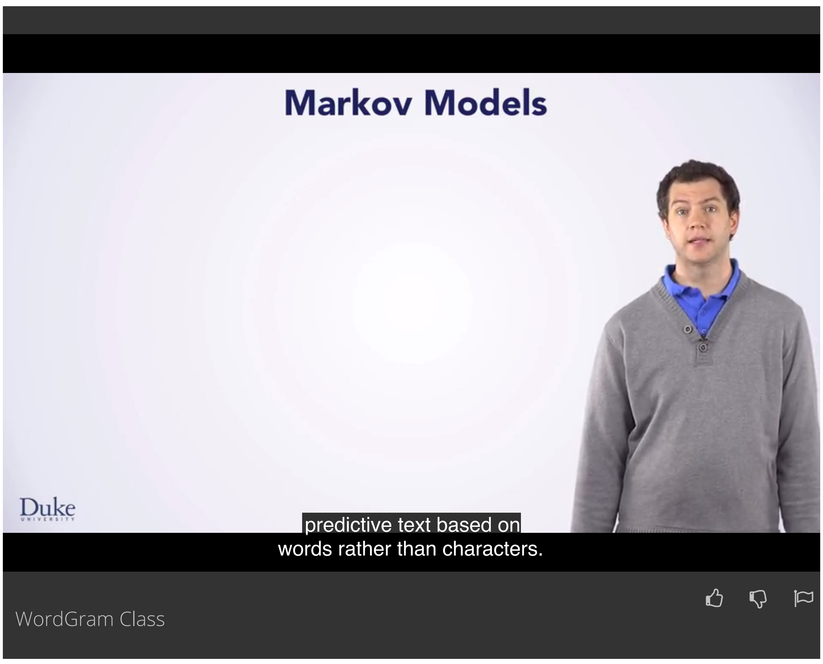
Generating Random Text

Word N-Grams

Review



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

Interactive Transcript

Search Transcript English

0:03
Okay, now we're going to look at the design and implementation of a class, WordGram, which will let us create Markov random and predictive text based on words rather than characters. When we used characters and strings, it was not too difficult to go from MarkovOne to MarkovTwo, and then to creating interfaces in an abstract class. In the getRandomText method, changing from MarkovOne to MarkovTwo required changing the constant 1 to the constant 2 in three places. Changing to any number, not just 2, requires the same three changes. My order is used here instead of using a 1 or a 2. My order is used here to get a substring of the right length, not just length one or two and my order is used here to indicate we've already generated letters and stored them before we start looping. The code and helper method getFollows does not change, not at all, in moving from order one to order two, to any order. Changes to the MarkovWordOne class that uses words to generate predictive text at random are not as simple or straightforward. As we'll see in a minute when using characters and strings, we relied on the >substring method to get any subsequence of a string. We need an analog of this .substring method for a string ray arso we can extend MarkovWordOne to a class with any number of string or words can predicting a new word. Let's look at the method getFollows for an N character Markov class. Then we'll extend that to to N words.

1:34
This is the getFollows method for the character and string version of Markov. Instance variable myText stores the string that represents the training data for the Markov class. The key for the randomText is also a string. We use this key to find all following characters to predict random text, and extend the random sequence we're returning.

1:56
The method getFollows relies on .indexOf and .substring to do the work. .indexOf returns the index of the first occurrence of key, starting the search at pos. .substring returns a sequence of char values, the substring starting at a specific location. How can we extend these ideas from N characters to N words? The code and version that use characters made it straightforward to change from MarkovOne to Two, to any number. As we said, strings can be thought of as a sequence of characters. Strings can easily be one, two or three characters long. We saw String variable key and the method .indexOf using sequences of chars.

2:41
We'll have to give some thought in going from the already tested MarkovWordOne class to a general class that can use two or N words to generate text.

2:51
We'll need to move from a one-word key, which can be represented as a string, to an N-word key, which can't easily be represented as a single string. We'll need to search for N-words, not just one, in creating the follows helper method. We'll design and implement a word sequence class to handle order 2 or 3 or even order 20 random text based on words. This will be a sequence of strings just as a string is a sequence of chars. It will represent a sequence of strings stored in an array, the analog of the Markov class that treats a string as a sequence of chars.

3:29
This new class will be called a WordGram. Let's look more closely at that idea. Our WordGram class will represent a sequence of strings, not characters. While strings can be a sequence of letters, like P-L-A-N-T to represent plant, or the sequence that is the string dinosaur, a WordGram object will be a sequence of strings like the sequence, the dinosaur eats plants.

3:55
Designing this class will take a little thought, because the building block of a string is the primitive type char, whereas the building block of a WordGram is a string, not a primitive type. Internally, a WordGram class will be an array of string references, as you can see here. Now that we've got the concepts thought out, the next thing we want to do is think about how to design the class. We'll see that in the next video.

Downloads

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

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