Jake Kurlander

Sentence Generation (Linguistics)

5/2/2016

FUNCTIONAL OVERVIEW

My project will be a series of algorithms that (attempt to) create sentences given a sample input. It will run as a command line, printing its output to the console. Its input will be a text document, either a normally-read thing like a book or a speech, or a dictionary. I will probably use different inputs for different algorithms, and need both of these kinds of input. The output will be sentences, which should be at least somewhat similar to English. Some of these will have errors, and I hope to be able to compare the algorithms I use based on their output.

This program isn’t really a super helpful thing in general. It is more of a research project for me, so I can learn about how to implement some tricky new algorithms I’ve never tried to put into a program before. Also, I believe I will end up with LOTS of experience with trees.

DESIGN OVERVIEW

The program will probably just ask “What algorithm do you want to use to make a sentence?” and “What input file would you like to base your sentences off of?” then spit out a sentence. A sample interaction is below.

What algorithm do you want to use to make a sentence?

>>Markov Chain

What input file would you like to base your sentences off of?

>>Dracula.txt

Output Sentence: Towards me so I had to make certain amount of horses.

PRIORITIZATION OF FEATURES

The first thing I want to get done is the general stuff – the user input and abstract Generator superclass. My first algorithm will probably be the Markov Chain, since I understand it the best, and it seems like the simplest algorithm. Basically, it just takes a document, figures out what words tend to follow other words, and makes a generic series of words. You can also have Markov Chains of different “degrees”, where you pay attention to what three-or-four-word-phrases are common. I would also like to do a syntax tree, where the program decides what parts of speech it needs for each phrase, which phrases it needs for each clause, and what clauses it will use for a sentence, then fills in the part-of-speech spots with random words. If I have time, I will move on to other algorithms, and maybe even come up with my own (if I have a LOT of time…)! I’d also like to have a couple “basic” algorithms like random words and taking real sentences from a document.

DESIGN DETAILS

I will be using Jgrasp for my development environment. I will probably use the IO and Util packages a lot, but not much else. I don’t think I’ll need to use anything other than java for development.

This won’t be all that similar to anything I’ve done before as far as the way the algorithms work. It’s very similar to Empirical.java when you look at input and output, since I’m making some algorithms and testing them, but the vast majority of the work I’m going to do and the code I will write will feel new and different.

Pseudocode

Print introduction

Loop as long as the user wants to keep using this program

Print “What algorithm do you want to use?”

Get response

Create new Generator object. Make it of type whatever the user just asked for (MarkovGenerator)

Print “What input file do you want to base everything off of?”

Read in file using FileReader

Ask if the user wants to print another sentence with this generator

Give the file to the Generator.

Generator analyzes the file, prints a sentence.

Ask if the user wants to print another sentence (loop).

Ask if the user wants to use another generator.

Generator

private String[] data;

//Not 100% sure about if I want the tree in the superclass; I might change it as I learn more.

private Tree[] Analysis;

Public String getSentence();

Public Generator (String [] library);

//Each generator will also have a lot of individual methods that make it work better.

TESTING

My program is pretty straightforward; if it produces English-looking sentences, it’s doing it right. Since my goal is abstract (“imitate a language”), I don’t have a super concrete way of saying “This algorithm works. This one does not.” I hope to have some time (maybe during the presentation?) to get people to review my algorithms, and decide how good they are at making sentences. I’d like to know how good these algorithms are at the following:

* Producing grammatically correct sentences
* Producing “focused” sentences (“The boy in the park kicked the ball” vs “Crunchy and fresh fruit find to carry the finish.”) –Does the sentence seem like random words or a sentence?

I would like to ask people what they think of the sentence’s meaning, it’s grammar, and maybe if I make a good enough algorithm I would put generated sentences and complete, copied, human-written sentences together and try to get people to tell the difference.

There is only one situation where I think I could run into serious trouble: if the dictionary it creates sentences from isn’t complete. I have no way to create sentences from an incomplete or badly formatted dictionary, so the only thing I can do is print an error message and make sure no exceptions get thrown.

GRADING RUBRIC

10 points – External Correctness

Does the program create some sort of sentence approximations (7)? Does it throw errors when it shouldn’t/is user input robust (3)? Note: This shouldn’t be graded on whether or not it makes grammatically correct sentences, it should be graded on whether or not it uses the algorithms correctly. Some of these algorithms don’t make good sentences all the time.

20 points – Internal Correctness

Does the program correctly use the algorithms specified in the program’s introduction (12)? Does it use the right data structures to store information (3)? Is there redundant code (2)? Is it efficient/Does it take longer than a few seconds to generate a sentence (3)?

10 points – Style and documentation

Is the program readable (4)? Without prior knowledge of how these algorithms work can you understand how this program works (6)?

PROPOSED IMPLEMENTATION SCHEDULE

5/15 – Complete infrastructure – Abstract Generator superclass, user input structure, file IO, and super basic algorithms (random words, random sentence).

5/20 – Work on Syntax tree structure, meet with Grammar teacher sometime this week. Have one of the two algorithms done by now.

5/22 – Work on other algorithm, have both Markov chain and Syntax tree done by the end of this weekend!

Beyond… -- Work on whatever algorithms seem interesting, better, or I design.

POTENTIAL SHOWSTOPPERS

I can’t think of anything that would *completely* derail my project. Even if it turns out that something I planned to do is far too complicated, I could still just find some easier way to generate sentences, and try that instead.

RESOURCES

I am currently using some linguistics things I find on the internet, and plan to meet with either a Latin teacher or an English teacher to talk about grammar.