

# Assignment 7: Hash Tables and Markov Text Generation

Assigned: Thursday, November 20, 2014  
Due: Friday, December 5, 2014, 11:59 P.M.  
Type: Individual

## Problem Description

The Infinite Monkey Theorem (IFT) says that if a monkey hits keys at random on a typewriter it will almost surely, given an infinite amount of time, produce a chosen text (like the Declaration of Independence, *Hamlet*, or a script for *Star Trek the Next Generation*). The probability of this actually happening is, of course, very small but the IFT claims that it is still possible. Some people intent on testing this hypothesis have set up virtual monkey sweatshops where poor cyber-monkeys type relentlessly in less than humane conditions. After billions and billions of simulated years, one monkey in the group was able to type out a sequence of 19 letters that can be found in Shakespeare's *The Two Gentlemen of Verona*. (See the April 9, 2007 edition of *The New Yorker* if you're interested.)

The IFT might lead to some interesting philosophical discussions, but from a practical point of view—who cares? A more promising line of inquiry might go like this: Could a monkey with prior knowledge of Claude Shannon's information theory and Markov chains produce a reasonable imitation of a known text in a reasonable length of time? Now that might actually be interesting. It's so interesting, in fact, that we should give it a name—the Markov Monkey Question (MMQ)—and we should spend some time thinking about it.

It turns out that the answer to the MMQ is "maybe." Here's an excerpt from a Markov Monkey version of the script for the 1967 *Planet of the Apes*.

"PLANET OF THE APES"

Screenplay by Michael Wilson

Based on Novel By Pierre Boulle

DISSOLVE TO: 138 EXT. GROVE OF FRUIT TREES - ESTABLISHING SHOT - DAY

Zira run back to the front of Taylor.

The President, I believe the prosecutor's charge of this man.

ZIRA Well, whoever owned them was in pretty bad shape.

He picks up two of the strain.

You got what you wanted, kid. How does it taste? Silence.

Taylor and cuffs him.

Over this we HEAR from a distance is a crude horse-drawn wagon is silhouetted-against the trunks and branches of great trees and bushes on the horse's rump. Taylor lifts his right arm to ward off the blow, and the room and lands at the feet of Cornelius and Lucius are sorting out equipment falls to his knees, buries his head silently at the Ranch).

DISSOLVE TO: 197 INT. CAGES - CLOSE SHOT -

FEATURING LANDON - FROM TAYLOR'S VOICE (o.s.)

I've got a fine veterinary surgeons under my direction?

ZIRA Taylor!

ZIRA There is a small lake, looking like a politician.

TAYLOR Dodge takes a pen and notebook from the half-open door of a guard room. Taylor bursts suddenly confronted by his original pursuer (the dismounted cop coming up with a cigar butt and places it in the drawer beside them.

TAYLOR What's the best there is a. loud RAP at the doll was found beside the building. Zira waits at the third table.

TAYLOR Good question. Is he a man?

CORNELIUS (impatiently.

DODGE Blessed are the vegetation.

These SHOTS are INTERCUT with: 94 WHAT THE ASTRONAUTS

They examine the remnants of the cage.

ZIRA (plunging on) Their speech organs are adequate.

The flaw lies not in anatomy but in the back of his left sleeve.

TAYLOR (taking off his shirt. 80 DODGE AND LANDON

You don't sound happy in your work.

GALEN (defensively) Gorilla hunter stands over a dead man, one fo

Besides a few spelling errors and some rather odd things that make you wonder about the author, this passage is surprisingly human-like. It turns out that Markov monkeys can apply a basic idea first discussed by Claude Shannon to produce pretty good imitations of real text. (And by the way, this basic idea has very important applications in image processing and the automatic generation of realistic terrains and surfaces in computer graphics.)

## Approach

So, here's the basic idea: Imagine taking a book (say, *Tom Sawyer*) and determining the probability with which each character occurs. You would probably find that spaces are the most common, that the character 'e' is fairly common, and that the character 'q' is rather uncommon. After completing this "level 0" analysis, you would be able to produce random *Tom Sawyer* text based on character probabilities. It wouldn't have much in common with the real thing, but at least the characters would tend to occur in the proper proportion. In fact, here's an example of what you might produce:

### Level 0

rla bsht eS ststofo hhfosdsdewno oe wee h .mr ae irii ela iad o r te u t mnyto onmalysnce,  
ifu en c fDwn oee iteo

Now imagine doing a slightly more sophisticated *level 1* analysis by determining the probability with which each character follows every other character. You would probably discover that 'h' follows 't' more frequently than 'x' does, and you would probably discover that a space follows '.' more frequently than ',' does. You could now produce some randomly generated *Tom Sawyer* text by picking a character to begin with and then always choosing the next character based on the previous one and the probabilities revealed by the analysis. Here's an example:

### Level 1

"Shand tucthiney m?" le ollds mind Theybooure He, he s whit Pereg lenigabo Jodind alllld  
ashanthe ainofevids tre lin-p asto oun theanthadomoere

Now imagine doing a *level k* analysis by determining the probability with which each character follows

every possible sequence of characters of length  $k$ . A level 5 analysis of *Tom Sawyer* for example, would reveal that 'r' follows "Sawye" more frequently than any other character. After a level  $k$  analysis, you would be able to produce random *Tom Sawyer* by always choosing the next character based on the previous  $k$  characters (the "seed") and the probabilities revealed by the analysis.

At only a moderate level of analysis (say, levels 5-7), the randomly generated text begins to take on many of the characteristics of the source text. It probably won't make complete sense, but you'll be able to tell that it was derived from *Tom Sawyer* as opposed to, say, *The Sound and the Fury*.

Here are some more examples of text that is generated from increasing levels of analysis of *Tom Sawyer*. (These "levels of analysis" are called *order  $k$  Markov models*.)

## Level 2

"Yess been." for gothin, Tome oso; ing, in to weliss of an'te cle - armit. Papper a comeasione, and smomenty, fropeck hinticer, sid, a was Tom, be suck tied. He sis tred a youck to themen

## Level 4

en themself, Mr. Welshman, but him awoke, the balmy shore. I'll give him that he couple overy because in the slated snufflindeed structure's kind was rath. She said that the wound the door a fever eyes that WITH him.

## Level 6

people had eaten, leaving. Come - didn't stand it better judgment; His hands and bury it again, tramped herself! She'd never would be. He found her spite of anything the one was a prime feature sunset, and hit upon that of the forever.

## Level 8

look-a-here - I told you before, Joe. I've heard a pin drop. The stillness was complete, however, this is awful crime, beyond the village was sufficient. He would be a good enough to get that night, Tom and Becky.

## Level 10

you understanding that they don't come around in the cave should get the word "beauteous" was over-fondled, and that together" and decided that he might as we used to do - it's nobby fun. I'll learn you."

Once you have performed a given level of analysis, here's the basic process of generating text from an order  $k$  Markov model of some sample text.

1. Pick  $k$  consecutive characters that appear in the sample text and use them as the initial seed value.
2. Append the seed to the output text being generated.
3. Repeat the following steps until the output text is sufficiently long.
  - (a) Select a character  $c$  that appears in the sample text based on the probability of that character following the current seed.
  - (b) Append this character to the output text.
  - (c) Update the seed by removing its first character and adding the character just chosen ( $c$ ) as its last character.

If this process encounters a situation in which there are no characters to choose from (which can happen if the only occurrence of the current seed is at the exact end of the source), simply pick a new seed at random and continue.

For example, suppose that  $k = 2$  and the sample text is:  
the three pirates charted that course the other day

Here is how the first three characters might be chosen:

- A two-character seed is chosen at random to become the initial seed. Let's suppose that "th" is chosen. So, seed = th and output = th.
- The first character must be chosen based on the probability that it follows the seed (currently "th") in the source. The source contains five occurrences of "th". Three times it is followed by 'e', once it is followed by 'r', and once it is followed by 'a'. Thus, the next character must be chosen so that there is a 3/5 chance that an 'e' will be chosen, a 1/5 chance that an 'r' will be chosen, and a 1/5 chance that an 'a' will be chosen. Let's suppose that we choose an 'e' this time. So, seed = he and output = the.
- The next character must be chosen based on the probability that it follows the seed (currently "he") in the source. The source contains three occurrences of "he". Twice it is followed by a space and once it is followed by 'r'. Thus, the next character must be chosen so that there is a 2/3 chance that a space will be chosen and a 1/3 chance that an 'r' will be chosen. Let's suppose that we choose an 'r' this time. So, seed = er and output = ther.
- The next character must be chosen based on the probability that it follows the seed (currently "er") in the source. The source contains only one occurrence of "er", and it is followed by a space. Thus, the next character must be a space. So, seed = r\_ and output = ther\_, where '\_' represents a blank space.

As another example, suppose that  $k = 2$  and the sample text is:

agggcagcggcg

Here are four different output text strings of length 10 that could have been the result of the process described above, using the first two characters ('ag') as the initial seed.

agcggcagcg  
aggcaggcgg  
agggcaggcg  
agcggcggca

## Implementation Details

You are provided with two Java files that you must use to develop your solution: MarkovMonkey.java and LanguageModeler.java.

```
public class MarkovMonkey {
    public static void main(String[] args) {
    }
}
```

```

public class LanguageModeler {

    // create an order K model for text
    public LanguageModeler(int K, File text)

    // create an order K model for text
    public LanguageModeler(int K, String text)

    // return the first K characters of the sample text
    public String firstSeed()

    // return K consecutive characters from the sample text,
    // selected uniformly at random
    public String randomSeed()

    // return a character that follows seed in the sample text,
    // selected according to the probability distribution of all
    // characters that follow seed in the sample text
    public char nextChar(String seed)
}

```

The main method of MarkovMonkey must process the following four command line arguments (in the args array):

- A non-negative integer  $k$
- A non-negative integer *length*.
- The name of an input file *source* that contains more than  $k$  characters.
- The name of an output file *result*.

Your program must validate the command line arguments by making sure that  $k$  and *length* are non-negative and that *source* contains more than  $k$  characters and can be opened for reading. If *result* does not exist, your program creates it. If *result* already exists, your program overwrites it. If any of the command line arguments are invalid, your program must write an informative error message to System.out and terminate. If there are not enough command line arguments, your program must write an informative error message to System.out and terminate.

With valid command line arguments, your program must use the methods of the LanguageModeler class to create an order  $k$  Markov model of the sample text, select the initial seed, and make each character selection. You must implement the LanguageModeler methods according to description of the Markov modeling process in the section above.

A few sample texts have been provided, but Project Gutenberg (<http://www.gutenberg.org>) maintains a large collection of public domain literary works that you can use as source texts for fun and practice.

## Submission

Submit your solution via WebCAT no later than the date and time specified. Turn in MarkovMonkey.java and LanguageModeler.java. **Do not** turn in any input or output text files.

## Acknowledgments

This assignment is based on the ideas of many people, but Owen Astrachan in particular.