Xing Su
CompSci 201
2/19/13

Markov Analysis

Part A

1. **How long does it take using the provided, brute force code to generate order-5 text using Romeo and Juliet (romeo.txt) and generating 100, 200, 400, 800, and 1600 random characters. Do these timings change substantially when using order-1 or order-10 Markov models? Why?**

# Characters	order-1 Markov Time (s)	order-10 Markov Time (s)
100	0.047	0.034
200	0.091	0.058
400	0.179	0.124
800	0.354	0.248
1600	0.718	0.491

Yes the timings change significantly when changing from order 10 to order 1 Markov models. Because for an order 1 model, for each character generated, the model is going through the entire text to produce the distribution arraylist and randomly generating a character. This in turn will produce much more iterations than the order 10 Markov, where 10 characters are taken at the same time.

2. **Romeo has roughly 153,000 characters. Hawthorne's Scarlet Letter contains roughly 500,000 characters. How long do you expect the brute force code to take to generate order-5 text when trained on hathorne.txt given the timings you observe for romeo when generating 400, 800, 1600 random characters? Do empirical results match what you think? How long do you think it will take to generate 1600 random characters using an order-5 Markov model when the King James Bible is used as the training text --- our online copy of this text contains roughly 4.4 million characters. Justify your answer -- don't test empirically, use reasoning.**

I estimated the running times by averaging the order-1 and order-10 running times from above and multiplying it by the ratio of Scarlet Letter letter count to Romeo letter count. Below are the running results:

# Characters	Estimated Time (s)	Actual Time (s)
400	0.495	0.399
800	0.984	0.762
1600	1.975	1.623

The running times are correct in the orders of magnitude but are reasonably smaller than the estimates. This is because the running time of a order 5 model should be closer to order 10 because of the amount of operations it saves compared to order-1.

For the King James Bible, the running time should be 14.607 seconds. (4.4 mil is about 9 times the size of Scarlet Letter, thus running time should be 9*1.623 = 14.607)

3. **Provide timings using your Map/Smart model for both creating the map and generating 200, 400, 800, and 1600 character random texts with an order-5 Model and romeo.txt. Provide some explanation for the timings you observe.**

   Each of the following cases was run separately (executed upon running MarkovMain.java without prior load/computations). I did this because for the same 5 model, the HashMap will only be created once, and the creating map time will be 0 for any trials after the first one.

# Characters	Creating Map (s)	order-5 Markov Time (s)
200	0.303	0.001
400	0.215	0.001
800	0.284	0.003
1600	0.223	0.007

The amount of time needed to create the maps is determined by the size of the given text. Since for each case, the same text is loaded, the creating map time is fairly consistent, as shown above. Because of the fact that after the hashmaps are built, each following character generated is done through looking the N-gram in the HashMap (which is an O(1) operation), the text generation process is O(N) and inherently faster than the brute force.

Part B

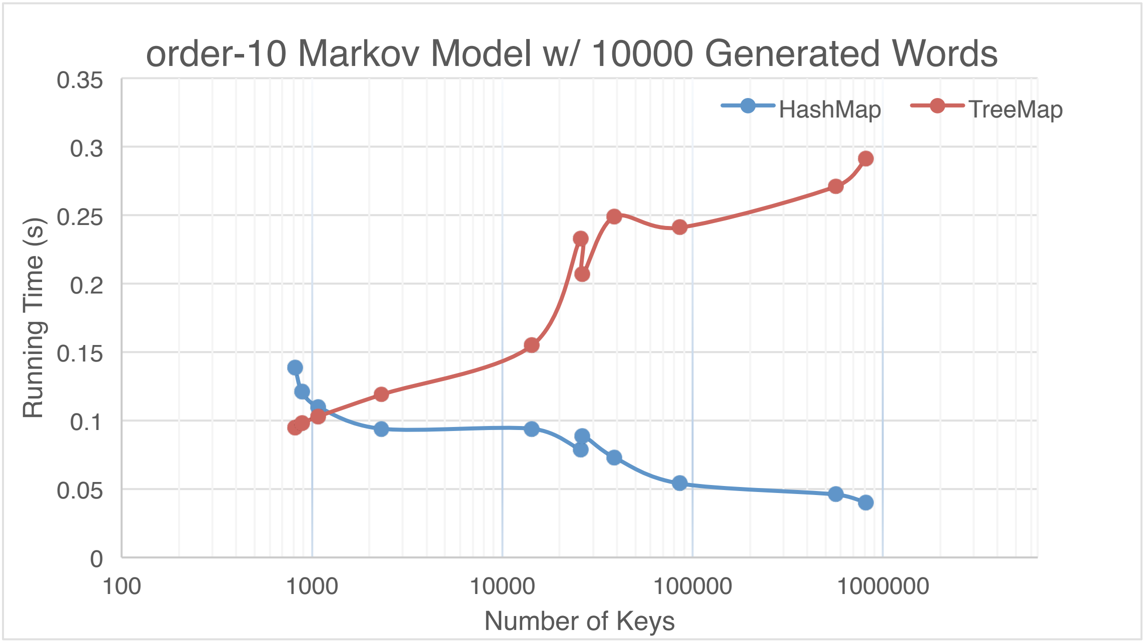
1. **The goal of the second part of the analysis is to analyze the performance of WordMarkovModel using a HashMap (and the hashCode function you wrote) and a TreeMap (and the compareTo function you wrote). The main difference between them should be their performance as the number of keys (that is WordNGrams as keys in your map) gets large. So set up a test with the lexicons we give you and a few of your own. Figure out how many keys each different lexicon generates (for a specific number sized n-gram). Then generate some text and see how long it takes.**

Graph the results. On one axis you'll have the number of keys, on the other you'll have the time it took to generate a constant of words (you decide...choose something pretty big to get more accurate results). Your two lines will be HashMap and TreeMap. Try to see if you can see any differences in their performance as the number of NGrams in the map get large. If you can't, that's fine. Briefly write up your analysis (like 1 or 2 paragraphs) and include both that and the graph in a PDF you submit.

The following data was generated with an order-10 Markov model for 10000 words. I used all of the files provided plus Candide by Voltaire and War & Peace by Leo Tolstoy.

File Name	Number of Keys	Running Times (HashMap)	Running Times (TreeMap)
edwards-nh.txt	814	0.139	0.95
clinton-nh.txt	887	0.121	0.098
obama-nh.txt	1079	0.11	0.103
poe.txt	2324	0.094	0.119
melville.txt	14353	0.094	0.155
romeo.txt	25787	0.079	0.233
alice.txt	26411	0.089	0.207
candide.txt	38839	0.073	0.249
hawthorne.txt	85752	0.054	0.241
warandpeace.txt	565388	0.046	0.271
kjv10.txt	813486	0.04	0.291

The following graph is then generated (Keys is on logarithmic scale).



As we can see from above, our results indicate that as the number of keys increase, the running time for the TreeMap-based Markov model increases logarithmically, while the HashMap-based Markov model decreases. This makes sense because HashMaps look up values using hashcodes, which means that the more unique keys/hashcodes the map has, the faster the lookup operation will take. This is consistent with what we discussed in class regarding the O(1) running time that hashcodes facilitate. On the other hand for TreeMaps, the more keys we have, the more branches it will have to dig through to find a specific value using the given key. As the number of keys gets larger and larger, more and more operations will need to occur for each lookup operation, which in turn increases the running time. This is also consistent with the O(lnN) running time that TreeMaps have.