Joshua Rosen

Alexandra Bertagnolli

Linear Algebra Final Project

Using Markov Chains to Procedurally Generate Text

Table of Contents:

1. Abstract

Markov chains can be utilized to reveal patterns in text and human speech. In this project, we will program a Markov chain which generates statistically probable text based on its input. When the user gives the program a text file, the program calculates the likelihood of which words follow each other. Each word is then assigned a transition matrix, in which each element of the matrix denotes the probability of which word occurs next. The program chooses and displays the most likely series of words to occur using the elements of these transition matrices. We intend to use this program to analyze patterns and bias in text.

2. Description of Markov Chains

Markov Chains provide a way of modeling the probability of future events based solely on the probability of past events. Markov chains are memoryless; that is, they can determine the probability of the next state occurring based solely on the current state.

A Markov chain is created by forming a transition matrix, wherein each i, j entry in the matrix represents the probability that the jth state will follow the ith state. For text, these probabilities are found by summing up the number of times the jth letter follows the ith letter, and then dividing by the number of times the jth letter exists in the text. Given a starting point, Markov chains use the transition matrix to create a probabilistic model of what happens next.
For example, in the word "Mississippi," the transition matrix would look as follows:

	M	**I**	**S**	**P**
M	0	0	0	0
I	1	0	.5	.5
S	0	.5	.5	0
P	0	.5	0	.5

The first column vector corresponds to the letter M and indicates a 100% probability that the next “state” to follow the letter M will be the letter I. The second column vector corresponds to the letter I and indicates a 50% probability of transitioning to the letter S and a 50% probability of transitioning to the letter P. As an example, generating 10 letters from this transition matrix (starting with the letter S) could give the result “SIPISIPPISI.” Another possible result could be “SSSIPIPISI.”

3. Properties of Transition Matrix

The elements of each column vector in the matrix indicate the probability of transitioning to a different letter. Because the elements in each column vector will always sum to 1, the column vectors are known as **stochastic vectors**, or **probability vectors**. Thus, a square matrix is formed from these column vectors and indicates the transition probabilities for each letter. This square matrix is known as a **left stochastic matrix**[1] because each of its columns are stochastic

[1] https://en.wikipedia.org/wiki/Stochastic_matrix

vectors. Because every element represents a transition probability, all possible states are represented in the stochastic matrix; that is, it is possible (though not always probable) to recreate the original text by following the constructed transition matrix.

A 26 *x* 26 matrix represents the probability of each letter occurring, and forms a **first order** Markov chain. A **second order** Markov chain would analyze the text by pairs of letters rather than by individual letters, and would be represented by a 26^2 *x* 26^2 matrix. A **third order** Markov chain analyzes the text by groupings of three letters, and so on. While looking at higher ordered Markov chains can reveal patterns in the structure of words and frequency of groupings of letters, a first order Markov chain can also apply to words, rather than letters. Instead of a 26 *x* 26 matrix representing the transition between letters, the matrix can represent the probability of transitioning from one word to the next within a group of text. With this view, a second order Markov chain would then represent the probability of groupings of two words, a third order groupings of three, etc.

A third order markov chain operating on individual letters can show differences in language structure. A small sample of 200 groupings of 3 letters generated from an English dictionary and a German dictionary, while not comprehensive, reveals a difference in word structure for the two languages. The English set contained many more groupings of consonant/vowel/consonant, like “cad,” “beh,” “deg,” and “bas” than the german set. The german set contained many more groupings of vowel/vowel/consonant like “auf,” “ein,” and “ier.” The German set did contain some consonant/vowel/consonant groupings, and the English set contained some vowel/vowel/consonant groupings, but looking at the sets as a whole, you are probabilistically more likely to encounter vowel/vowel/consonant groupings in German than in English. While looking at one sample set from a Markov chain is far from a comprehensive study or review, this example shows that markov chains can be used to reveal underlying patterns.

4. Description of Project process

Our project closely follows the process detailed above, however instead of counting the occurrences of characters, our program counts the occurrence of unique words. The final square matrix has the dimension n x n where n is the number of unique words in the given text. Words are determined by splitting the text on whitespace[2].

The basic algorithm is as follows:

1. Determine the number of unique words in the text.
2. Count occurrences of each word, incrementing the appropriate position in the appropriate column vector.
3. Transform each column vector into a stochastic vector.
4. Create the probability matrix out of these vectors.
5. Generate a random word using the corresponding weight in the probability matrix.
6. Repeat Step 5 as many times as the user requests.

[2] A notable limitation of our program is that it does not account for punctuation: the text “hello” and “hello!” are treated as different words.

Note that before Step 5 can occur, we must choose a word to start with, as the transition matrix only gives the probability of *transitioning* from one state to the next, but does not give us a starting state. Our program simply chooses a random starting word in an attempt to yield interesting results.

5. Example of Generated Text and Analysis

More complete experiments and analysis would have to be done with actual text generated from our Markov chain before any definitive results could be concluded. However, the generated text from our Markov chain *sounds* like the input text. For example, given all of Pride and Prejudice by Jane Austen, the Markov chain generates text that has Austen's distinct style and voice, even though what is generated is not always grammatically correct english, and doesn't always make sense. Some samples include:

> "Darcy may cough as he finally seated himself; and arranging such an evening without ostentation or any complacency."
>
> "at your opinion of ourselves, vanity is passing."
>
> "Such amiable qualities must comprehend the apothecary, who lived a
> truth in anything with quickness is a disadvantage to do justice to provide for."
>
> "make his rank, and reprehensible, according to what passed, by imitation."
>
> "Miss Bingley succeeded no longer to catch their original design for the next morning"

Markov chain generations from Franz Kafka's The Metamorphosis behave in the same way - the generated text sounds like Kafka.

> "But then was held back, calmed down, convinced that he lay there eating their leader."
>
> "Mrs. Samsa appeared with no surprise for careful aim, threw himself from work straight away."

> "half-rotten vegetables; bones from troubled dreams, he was getting up at his daughter was still hurriedly thinking all these things and if from hunger,"

In conclusion, the text generated from our Markov chain exhibits many properties of normal literature and does an admirable job of emulating the original artist's distinct voice and style. Notably, the main limitation of text Markov chains is their inability to truly grasp the nature of human language; however their simplistic design and applications to linear algebra make them a noteworthy subject of interest.

6. Bibliography

Purdue University: http://www.stat.purdue.edu/~mdw/CSOI/MarkovLab.html

Markov chain PHP text generator: http://projects.haykranen.nl/markov/

Markov Chain article: https://blog.codinghorror.com/markov-and-you/

Markov Chain basics: http://www.sosmath.com/matrix/markov/markov.html

Wikipedia article on Markov Chains: https://en.wikipedia.org/wiki/Markov_chain

Wikipedia article on Stochastic Matrices: https://en.wikipedia.org/wiki/Stochastic_matrix

Pride and Prejudice and The Metamorphosis retrieved from Project Gutenberg:

https://www.gutenberg.org/