Markov Chain: Shakespearean Text Generation

Brian Standage, Christian Ferguson, Noah Matlock

bstandag@iu.edu
fergusch@iu.edu
nmatlock@iu.edu

Abstract

Generating text that recreates a certain writing style can be achieved through a Markov Chain. For this project, a Markov Chain is created by analyzing *The Complete Works of Shakespeare* and assigning probabilities to different combinations of words. Through this process, the Markov Chain is able to produce plausible text that recreates the writing style of Shakespeare. The purpose of this project is to see the accuracy of text generation with Markov Chains.

Index Terms: Markov Chain, natural language generation, stochastic models, Python

1. Background

1.1. What is a Markov Chain?

Given a set of states, $S = \{s_1, s_2, s_3, ..., s_n\} \; \forall \; n \in \Re$, the Markov process starts at a random state s_n, and moves in steps from one state to another. A Markov chain is described as memoryless. The memorylessness of Markov chains means that the it does not matter how the chain got to the current state, nor does it matter how long the chain has been at the current state. Thus, given a Markov chain at the current state s_i, the probability the chain moves to the next state, s_j, is denoted by p_{ij}. The memoryless feature of Markov chains is called the Markov property[1].

1.2. Transition Matrix

The *transition matrix* is comprised of the *transition probabilities,* or the probabilities p_{ij}. The transition matrix describes the transition probabilities of the Markov chain moving from one state to the next, p_{ij}, or remaining in the same state, p_{ii}. The transition matrix can be denoted as follows[2]:

$$P = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,j} & \cdots \\ p_{2,1} & p_{2,2} & \cdots & p_{2,j} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \ddots \\ p_{i,1} & p_{i,2} & \cdots & p_{i,j} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \ddots \end{pmatrix}$$

Figure 1: *Representation of the transition matrix.*

The probability of going from state *i* to state *j* is calculated by:

$$P(j \mid i) = P_{ij} \qquad (1)$$

A very important characteristic of the transition matrix is that the sum of the rows is equal to 1. In other words, the probability of going from state *i* to any other possible state is 100%.

1.3. Markov Chain of Higher Order

A Markov chain of higher order is a unique in that it does not have the same Markov property as a regular Markov chain. The Markov chain not only depends on the current state, but also the $n-1$ previous states where a finite number n represents the order of the Markov chain. Thus, a Markov chain with order 3 is dependent on the two previous states to the current state.

2. Problem Statement

The overall goal of our project was to generate plausible text in the writing style of Shakespeare, and use this text to determine if a one-key Markov chain is sufficient for generating coherent and meaningful text. Natural language generation is a relevant problem that has clear advantages within society. Some of these advantages include saving time and money when writing product descriptions and data summaries, and also improved clarity of charts and graphs through personalized data analysis. With this project, we aim to test the quality of a one-key Markov chain in natural language generation.

Our motivation for producing text in the style of Shakespeare has to do with the availability of data and the ease of analyzing the results. *The Complete Works of Shakespeare* is available for free online, allowing us to easily access mass amounts of relevant data. Additionally, Shakespeare has a very specific and unique writing style. Because Shakespeare's writing style is so unique, analyzing the output of our Markov chain becomes much more objective. This allows us to classify our results as either successful or unsuccessful with less subjectivity.

3. Data Collection

3.1. Collecting the Data

In order to collect as much relevant data as possible to use in the Markov chain, we decided to use *The Complete Works of Shakespeare*[3]. *The Complete Works of Shakespeare* provide the Markov Chain with expansive data that is as representative as possible of Shakespeare's writing style. The more data available the more closely the output can truly resemble work produced by William Shakespeare.

In addition to the comprehensive nature of *The Complete Works of Shakespeare,* the data is also available online as a free text file. The format is advantageous in that the entire collection of data can be easily downloaded, cleaned, and processed on the average machine.

3.2. Cleaning the Data

Although the data was easily available online, the raw data was not appropriate to use in terms of our output goals. Given so, we had the particularly challenging task of cleaning the data. Because of our specific goals, cleaning the data involved removing any data that was not part of Shakespeare's actual sentence structure. For the sake of time, we split *The Complete Works of Shakespeare* into two categories: sonnets and plays. Our decision to split the data into these specific categories was based on the ease of cleaning.

The sonnets were very easy to clean as they contained little extraneous data. In order to clean the sonnets the only work necessary was removing line numbers and multiple line breaks that had been added for human legibility between each sonnet and inserting special "END" characters at the end of each line.

Alternatively, cleaning the section comprised of plays proved to be a rather challenging task. The raw play data contained large amounts of extraneous data that would cause our intended output to be flawed. For example, the raw play data contained speaker's names, dates, scene descriptions, special characters, etc.. This information would obviously flaw the data by including words that are not part of Shakespeare's typical sentence structure.

After cleaning the data we were left with useable and relevant data that could be used with the Markov chain. As a result of cleaning the data, we were left with 20,114 words from the sonnets and 1,037,819 words from the plays. With a total of 1,057,933 useable words, our Markov chain is sure to be a comprehensive representation of Shakespeare's writing style.

4. Implementation

To train our model, we cleaned a considerable portion of the completed works of Shakespeare and tweaked the data to contain a special tag that indicated the end of a line. This helped to discourage run-on sentences, as well as giving us the option to parameterize our output to a certain number of lines.

Then, we set up a parser to read through the text, separating by word. We created a list of words that appear in the text, each with a list of next occuring words as well as the frequency associated:

```
FOR word in text:
    if not (word in word list):
        add word to word list with empty next word list
    if (next word in next word list):
        increase frequency of next word in next word list
    else:
        add next word to next word list with frequency 1
```

After scanning the entire text, we have a list of words and the possible words that may follow them. The special end tag from before has a **next word list** consisting of words that begin lines in the original text, and is treated as a word for the algorithm, also appearing in the **next word list** of words appearing at the end of a line of text. By rolling a weighted random word from the list of next words available to the current word, and stringing those together until we roll the special end tag again, we generate a line of text.

To generate a "new" sonnet, specifically, we repeat that line generation process fourteen times to match the line length of the poetic form.

For the actual coding itself, we used raw Python scripts and standard libraries, and no specialized third party libraries or technologies for the scripts.

5. Future Work

As we were wrapping up, we had a bit of a roundtable about what future modifications (beyond the immediate scope of this project) could be done to increase the overall usefulness of this experiment, either as entertainment or as a research exercise. We landed on four key components that would elevate our output to a higher level:

- Fully cleaning and priming the million-plus words of play data would give us the opportunity to generate possibly richer output, given the number of already existing connections existing in the play text. Some of the words in the sonnet text are only ever followed by a single specific word, so while the abundance of words available in the sonnet text discourages runs of the same specific words, making connections over nearly fifty times that amount of data would further discourage those runs. Additionally, cleaned play data would be the first step into segueing into a side project of performing procedurally generated Shakespearean plays on a stage.
- The introduction of an efficient word hashing function or key-text pair indexing system would greatly speed up the actual generation time as iterating through the lists would not be so time consuming as jumping to the correct word. At present, the play text generation takes an average of 4.5 seconds per line, which could be reduced to mere milliseconds with a strong structural rework and hashing function. This would likely increase training time, but as the input data would never change (no new true Shakespeare will ever be written), retraining time is not a major concern. Generation runtime is where optimization would have the greatest impact overall.
- Because one of the most iconic aspects of the Shakespearean style (though one we expected to not be able to emulate within our parameters going in) is the use of iambic pentameter, a way to color or bias our output to try to force ten syllable lines, as well as a syllabic flagging system would overall enhance the quality of the output and bring it closer to a performable status. This requires deviation from the Markov chain model in a practical sense, in that every word in a line beyond the first must always rely contextually on those that come before it, negating the entire practical purpose of the Markov chain in the first place.
- Lastly, if we chose to shortcut the context continuation system mentioned in point one of this section, we could opt for two-key or three-key Markov models instead, basing the next word on the preceding two or three words. The difference

between this and the pseudo n-key model discussed in point three is that the integrity of the Markov model is tied to the use of a fixed length key grouping, so basing the next word off of the last three or four or five or six words in the same model no longer functions as a Markov model. A fixed key length of two words or three words for every connection however would still function as a Markov model and perhaps provide some additional context. Despite the size of the sonnet data set, a three key model might introduce overfitting in the form of the generator returning exact lines from the training data instead of similar lines. We are confident however that the sheer mass of the play data set would mostly avoid this issue thanks to its size if nothing else.

6. Conclusions

First and foremost, this project has reaffirmed that quality of data vastly exceeds quantity of data in terms of importance. Despite having roughly fifty times the raw input data for the plays as compared to the sonnets, building a model that allowed us to generate performable, deliverable content was exponentially easier with the smaller, cleaner data sample. While the Markov chain model differs in many ways from other, perhaps more intensive machine learning approaches, they all share a weakness to noisy data and a need to account for it if it is what is available.

To establish context, here is an example of the output of our one-key Markov chain:

“Against the west,

Be wise world to call,

Since my bootless cries,

Wound me not be as a devil:

Whilst that your self, and anon

To thy truth, thy sweet argument,

Reserve their age in hope, my wailing chief,

And soon to thy cruel eye is she stores, to thy unworthiness raised love doth my self with thy parts,

Why should blunter be taken.

The basest jewel from variation or shape which still green.”

While the goal of this project was to produce text in the *style* of Shakespeare and not necessarily text with coherence in terms of comprehension and meaning (and therefore a success by our measure), we have effectively demonstrated that a one-key Markov model is insufficient for generating text with comprehension and meaning throughout. On a line by line basis, a two-key model may produce stronger results, but overall coherence throughout a passage would require an altogether different strategy.

7. Acknowledgements

Our team extends a special thank you to our professor, Dr. Donald Williamson, for providing us with an understanding of the topics covered in this project as well as a template for this report.

In addition, our team would also like to thank Project Gutenberg for providing free and publicly available ebooks, allowing us to easily access *The Complete Works of Shakespeare*.

8. References

[1] “Markov Chains.” *Grinstead and Snell's Introduction to Probability*, by Charles M. Grinstead and J. Laurie. Snell, Editore, 2000.

[2] Klacksell, Gustaf and Sundberg Jesper “Markov Chain. Can You Describe the Stock Index with a Markov Chain?” KTH Royal Institute of Technology at Campus Valhallavägen KTH Matematik / Matematisk Statistik

[3] Shakespeare, William.. *The Complete Works of Shakespeare*. Urbana, Illinois: Project Gutenberg. Retrieved April 15th, 2018, from http://www.gutenberg.org/ebooks/100?msg=welcome_stranger