

THE USE OF MACHINE LEARNING IN CREATIVE WRITING: COMPUTER GENERATED SENTENCES IN THE CONTEXT OF CLASSICAL MYTHS

INDEPENDENT STUDY THESIS

Presented in Partial Fulfillment of the Requirements for
the Degree Bachelor of Arts in the
Department of Computer Science at The College of Wooster

by
Scott Stoudt

The College of Wooster
2019

Advised by:

Nathan Sommer



THE COLLEGE OF
WOOSTER

© 2019 by Scott Stoudt

ABSTRACT

The purpose of this project is to explore the field of machine learning and to utilize ML techniques to create a program designed to produce sentences containing a desired word, given by a user. The sentences are to mimic the style and content of the data being used to aid the user in their creative writing process. Machine learning is gaining a lot of attention in the tech world, as buzz words like “AI” and “deep learning” are seen tacked onto any and everything. Big tech companies are starting to invest a lot of time, money, and resources in machine learning research and AI implementation into their products. The goal of investing in this field is to improve the user experience by programming a computer to think like a human in order to speed up a given task. This can be seen in text prediction, search engine results, and self-driving features in newer vehicles. Using existing myths as training data, this study explores the viability of machine learning in creative disciplines, more specifically, in creative writing. The resulting generated myth-focused sentences will be analyzed against the criteria of what makes up a myth to determine if computers can replicate the same authentic and creative works written by human authors.

In memory of Rebecca A. Stoudt.

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Nathan Sommer for helping me throughout this project; particularly in helping me work through the data processing portion of the application. I would also like to thank Dr. Jacob Heil for serving as a great mentor and friend throughout my college career. I owe a great deal to him for introducing me to the world of digital humanities, which has allowed me to combine my love for technology with my passion for work in the humanities. In addition, I would like to thank the rest of the professors in the Computer Science, History, and Classics departments who I have learned from during my time at the College.

Most of all, I would also like to thank my father, Gary, my sister, Sara, and the rest of my family for supporting, motivating, and believing in me over the years. I would not be where I am today without their support in the paths I have chosen in life. Lastly, I would like to acknowledge my friends from Wooster and back home for their feedback and support: Connor, Jacob, Colby, Joe, Philip, Kz, Dylan, Thanh, and Gianni. Their friendship and support kept me motivated throughout this project and in general over the years.

CONTENTS

Abstract	v
Dedication	vii
Acknowledgments	ix
Contents	xi
List of Figures	xiii
List of Tables	xv
List of Listings	xvii
CHAPTER	PAGE
1 Introduction	1
1.1 Project Goals	3
1.2 Data Collection	5
1.2.1 The Content	5
2 Mythology	7
2.1 What is a Myth?	7
2.2 Classical Greco-Roman Mythology	8
2.2.1 Structure of Greco-Roman Myths	11
2.2.2 Sentence Styling Within Myths	15
3 Machine Learning	17
3.1 Introduction to Machine Learning	17
3.1.1 Markov Chains	19
3.1.1.1 Recurrent States	21
3.1.1.2 Transient States	21
4 Myth Generation Using Markov Chains	23
4.1 Writer's Block Application	23
4.1.1 General Overview	24
4.1.2 Data Cleaning Process	25
4.1.3 Data Processing	27
4.1.3.1 The Word Dictionary	28
4.1.3.2 Text Classification	29
4.1.3.3 Sentence Structure List	31

4.1.4	Text Generation	32
4.1.4.1	User Input	32
4.1.4.2	Sentence Generation Process	33
4.2	Analysis of Generated Output	34
5	Viability of AI in Creative Disciplines	37
5.1	Can It Be Done?	37
5.1.1	Machine Generated Text in the Real World	38
5.2	Should It Be Done?	38
5.2.1	Should It Be Done?: Writer's Block Application	39
6	Conclusion	41
6.1	Writer's Block Tool	41
6.1.1	Future Work	42
6.2	Computer Generated Myths	45
6.2.1	Future Work	45
APPENDIX		PAGE
A	Text Files Referenced	47
	References	51

LIST OF FIGURES

Figure		Page
1.1	The increase in funding for AI by the U.S. government between 2011 and 2018. [3]	2
2.1	Quintessential journey made by a hero in myths that is described in Campbell's book [7].	14
3.1	Simple Markov chain consisting of two states. [13]	20
3.2	Complex Markov chain with probabilistic movement between states. [2]	21
3.3	Markov model with both recurrent and transient states.	22
4.1	A flowchart for the Writer's Block application.	25
4.2	Unicode 'Replacement Character'	26
4.3	Removing unwanted text from data using RegEx.	26
4.4	A flowchart for the Writer's Block data processing approach.	27
4.5	A flowchart for the Writer's Block text generation process.	32
4.6	Screen shot of the prompt the user receives to use the Writer's Block application.	33
4.7	Screen shot of outputs produced by an early version of the Writer's Block application.	35
6.1	Example tweet from the user to the Writer's Block Bot.	43
6.2	Example tweet from the Writer's Block Bot to the user with a positive response.	44
6.3	Example tweet from the Writer's Block Bot to the user with a negative response.	44

LIST OF TABLES

Table		Page
2.1	Comparison between the Greek and Roman versions of deities in the Greco-Roman world.	10
3.1	Demonstrating fruit classifications used in a supervised learning model. [12]	18
4.1	Text classification for NLTK's pos_tag.	30

LIST OF LISTINGS

Listing	Page
4.1 A segment of our dictionary that keeps track of each word in the data, along with the words that follow it and the frequency in which the pairs occur.	28
4.2 One sentence converted to its POS to show a valid sentence structure.	31
A.1 Myths in Data	47

CHAPTER 1

INTRODUCTION

Over the last decade, the technology industry has seen a large increase in consumer interest and monetary investments. As shown in Figure 1.1 below, the U.S. has increased its Artificial Intelligence investments from \$282 million in 2011 to \$4.2 billion in the first two quarters of 2018. As data comes in for the remainder of the year, we can assume that this number will be roughly doubled. We see a peak in U.S. investments at around \$5 billion in 2017. Of course, the information we have only considers investments in the public sectors and therefore does not take private investments into consideration, which would further inflate these totals considerably if recorded. This has led to rapid technological advancement across a multitude of disciplines, an increase in research and development funding, as well as an increase in the demand for computer scientists to conduct this research and development. One of the areas within computer science that has seen an increase in interest is Artificial Intelligence (AI). The idea behind AI, which is the practice of simulating intelligence in machines, is that intelligence can be mapped out and written in terms that can be fed to a computer and be “programmed to “think” like a human and mimic the way a person acts” [8]. Through training, either by feeding a machine large amounts of data to process to gain an understanding of text and language problems or via a trial-and-error method of learning, machines can learn how to process and manipulate information similar to how humans do.

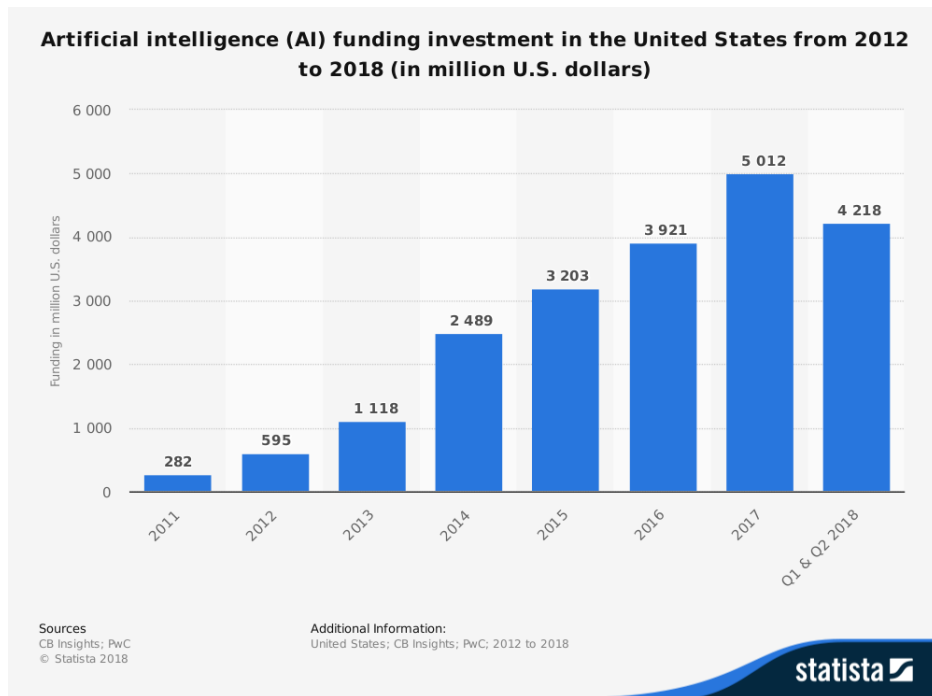


Figure 1.1: The increase in funding for AI by the U.S. government between 2011 and 2018. [3]

AI can be found through everyday use of technology such as the text prediction features of your phone and Google’s auto-complete search bar. Both of these are examples of machines constantly learning from the data it is fed to understand sentence structure and how humans think to make their experience with technology faster and more efficient. The key to both of these examples is that, as the machine trains with more and more data over large periods of time, they get better. Big tech companies have access to a global scale’s worth of data, which makes their algorithms so impressive. The task of training becomes increasingly more challenging as the data set shrinks or is less complete.

In addition to the quality and quantity of data you have to train the machine on, the task you give the computer also plays a huge factor in how well a machine can be programmed to complete it. The task of teaching a computer to make educated guesses about what words users would want next is relatively simple in the world

of AI, as language and grammar rules remain mostly unchanged each year. It is obvious that language evolves over time with new words being created annually, old words and phrases being updated, and language trends following a natural ebb and flow; however, these changes are gradual and the programs are able to include these changes in their training as they are made.

When the task changes to having computers generate new sentences that have meaning, while also arranging these sentences to create coherent full bodies of text, the complexity increases significantly. The algorithms change, the training becomes much more involved, and the success rate goes down because these tasks can be difficult for people as well. If you are tasked with filling in a blank or continuing a sentence, you will probably find that to be pretty easy. However, if you were asked to come up with a bunch of sentences off the top of your head and have them make sense together, you might find this task more challenging and thought intensive. It takes a certain kind of person with a lot of creativity to come up with original work, whether it be music, art, or stories. Logically, you would imagine that having AI create original works would be an even more daunting task. But who's to say that it can't be done? For this project, we explore the idea of computer generated text with a creative lens. Specifically, we develop a program that aims to generate unique, coherent sentences within the mythological context from a data set of existing stories.

1.1 PROJECT GOALS

The goal of this project is to design a program that aids a writer during the writing process by offering up sentences pertaining to a given word, provided by the user. For the purpose of this project, the program will generate sentences in a mythological setting. The program will be trained using existing myths of similar content, but

varying in length. What the program hopes to be able to discern from the other stories is how to understand the new words, places, and ideas of an ancient culture, as well as gaining a rudimentary understanding of the concept and structure of myths as a whole. This second goal of having the program understand the structure of the generic myth is important because we would like to see if a computer is capable of producing sentences matching the writing style of a typical myth without direct input from a user. In this case, we, as the programmer, should remain as hands off as possible, in regards to the output. The intention is to have the program generate sentences that require very little effort on the user's part to try and piece together the sentences to make sense. Since the point of this program is to aid the writer in the writing process, the goal is to give the user a sentence or two to reignite their creativity and jump right back in to creating their myth. Some of the tweaks the user might be expected to make would be to place punctuation, such as commas, into the appropriate place so that the flow of the sentences makes sense as well as minor grammatical modifications to make the sentence connect with prior sentences and overall tone of the writer's piece. Ideally, the computer should be expected to make some decisions itself, such as, what a good sentence is and whether or not the sentences being generated are of an appropriate length. Our goal to not produce a "mad lib" style sentence generator where the structure is fully provided and the computer is simply tasked with filling in the blanks with the user provided work or other related terms. The structure of each sentence produced should be similar to the ones from the training data, but should contain unique differences to set each sentence apart from the originals, as to not create parody or rewrite of common classical myth sentences.

1.2 DATA COLLECTION

An important first step in this project was setting up the data set that would be used to train the computer to learn how to generate myths. Luckily, the desired data is easily accessible and available in raw text form, due to these ancient stories being part of the public domain. This accessibility allowed the collection process to be simplified down to finding and downloading the desired text files and skip right into the data cleaning process, which is discussed later in Chapter 4. By using these online text databases, the process of converting physical text to a digital format can be bypassed. This process would have involved many hours of scanning pages, using OCR (optical character recognition) on the scans, and combing through lines of text to make sure sentences are in order and formatted appropriately and words were properly identified and correctly translated.

1.2.1 THE CONTENT

The training data set chosen for this project was classical myths, which is discussed in greater detail in Chapter 2. In short, these are myths from ancient Greece and Rome. We use these myths specifically because they are similar in structure, word usage, and involve the same names and places. The data set consists of 74 myths of varying length from Homer's *Iliad* and *Odyssey*, which are large epics, to *The Epigoni*, which is roughly 500 words. During the collection process, each text was evaluated to see if the proposed "myths" were true myths or simply small descriptions of various gods, heroes, or places from the ancient world, which were present in texts of collections of myths that were pulled from. It was important for the data set to be able to be modified as the project developed, as different stories had to be added and removed to guide the software into generating appropriate text.

CHAPTER 2

MYTHOLOGY

Before we can begin talking about what it takes to teach a program how to write a myth, we must first establish a proper definition for ourselves. Throughout the field of classical studies, there are many takes on what a myth is and what it consists of; with some definitions taking religious, cultural, or literary standpoints (even a combination of the three). Despite the disagreements, some commonalities can be found that provide a general consensus of what it means to be a myth. This section discusses what it means to be a myth and what kind of structure, design, and content can be found in a typical mythological story.

2.1 WHAT IS A MYTH?

Mythology, which comes from the Latin word *mythologia* is simply, the “telling of story”. Therefore, the simplest answer to “What is a myth?” is that it is a story. To most, a story is a very generic, broad term, so let’s break this word down a bit more. In my experience, at the start of every classical studies class that deals with myth, we begin by playing with the definition of myth, with each student offering up what myth means to them. Through this exercise, I’ve learned that there is no one, textbook definition of the word. Rather, it is a combination of many components.

A myth is typically defined as a traditional story that deals with early historical events of of a culture. These stories are created and preserved to explain various

natural events or socioreligious values a culture wishes to instill in their people, passed down orally or in writing. This is why many, like Joseph Campbell, author of *A Hero with a Thousand Faces* [7], think that myth in the ancient world served as a “proto-science”, explaining things such as day/night cycles, season changes, weather, and much more to the uneducated culture. Obviously myth served as the source material of ancient religions just like stories in the Bible explaining life during the early founding of Christianity. Lastly, myths served as a form of entertainment either through performance or literature. A key component of myth is that there is typically some level of inclusion of supernatural events and/or beings.

2.2 CLASSICAL GRECO-ROMAN MYTHOLOGY

For the purpose of this project, we will be dealing with classical Greco-Roman myths, which, as the name suggests, are myths that belong to ancient Greek and Roman culture. Many scholars, including authors Carl Ruck and Danny Staples emphasize in their book, *The World of Classical Myth* insist that the term “Classical Mythology” solely refers to the ancient Greek myths without the inclusion of the Latin (Roman) myths that branched from the Greeks as the Roman civilization developed [16]. To avoid confusion, “Greco-Roman” is added to the term to clarify that the inclusion of Roman myths is intended. For clarification, throughout this paper, the use of the term “classical myth” will be used to refer to the group of both Greek and Roman myths.

A common misconception about the ancient world is that the Roman civilization and its culture was entirely stolen and appropriated from Greek culture, which is not entirely true. The Greek and Roman peoples lived independently from one another in and around modern day Italy, each with their own “history, customs, stories” [16] and cultural difficulties. Ancient Greek society quickly grew to surpass

their neighbors in the Mediterranean in many ways. Scholars believe that the ancient Greeks were ahead of their time, that is the Classical Age (8th century BCE - 5th century CE). The Greeks became recognized for their advancements in writing, philosophy, and the arts. Even as science and philosophical thought started to gain popularity as the new way of thinking about the world, the Greek myths remained to serve as a part of the Greek culture and history. Because of this cultural superiority, the Greeks received a lot of envy from the surrounding, less advanced groups like the Romans. Lacking a real, clear written history of their own, Roman writers, artists, and poets began to study Greek culture to elevate their own works. Since the Romans did have their own stories and beliefs, it is unfair to say that they stole from the Greeks. Like today, cultures are always learning from each other and modifying their own ways to include pieces of other culture that they value and respect. There is typically no mal-intent, and some sort of homage is payed to the founding culture. Just as today, the Romans sought to “adopt the Hellenic (Greek) tradition” by comparing Greek deities with their own. This process lead to a sort of redesign within their religion where gods and important figures from Roman culture were assigned to their closest Greek counterpart. This is why you may have heard that ancient Greek gods and goddesses have a Greek and Roman version.

Greek Name	Roman Name
Zeus	Jupiter
Hera	Juno
Athena	Minerva
Poseidon	Neptune
Aphrodite	Venus
Ares	Mars
Artemis	Diana
Apollo	Apollo
Hephaestus	Vulcan
Hermes	Mercury
Hades	Pluto
Dionysus	Bacchus
Demeter	Ceres

Table 2.1: Comparison between the Greek and Roman versions of deities in the Greco-Roman world.

As you can see from Table 2.1 above, each of the Greek gods were given a new name and also updated personalities. These changes did not stray too far from the original, however some traits of the gods were updated to better fit the values and traits of the Roman people. The purpose of the Romans updating these things is to blend their history and myths to fit in the Greek story line and their characters. Hence, we still see important character like Heracles (Greek) or Hercules (Roman) in both mythologies, even though the stories may differ across cultures.

One of the fascinating things about myth is that contradiction is a generally accepted element. The way myth works is that each story is viewed independently. So while the stories can overlap and reference each other, events can also be altered

to create a kind of alternate or parallel universe within the canon of the culture. This is why we see multiple myths detailing the location of the oldest olive tree in the ancient world (one being a sanctuary of Zeus in Olympia and the other on Athens' Acropolis) [16]. This quality of myth ties in perfectly with the goal of this project since the computer is tasked with creating its own myth story based on vocabulary from existing classical myths. This means that the story that is generated should be made up of the same people, places, and ideas of the cultures it is trying to replicate. Therefore, as long as the generated myth has a somewhat clear and logical flow, the 'accuracy' of the terms should not matter. If the generated myth involved Zeus giving birth to Hera (typically described as his wife) it would not follow the typical order of things, but would still be considered a valid myth with its own take on that portion of the mythology.

2.2.1 STRUCTURE OF GRECO-ROMAN MYTHS

When it comes to the content of myths, there are two main approaches that the story can take: an origin story or "the hero's journey". The first approach, the origin story, is probably the most simple of the myth structures as there are no complex character arcs and the content, whether it is dialogue, setting, or character development is much more simplified and to the point. The purpose of these origin stories is to explain to the reader how the many gods and heroes came into being. These types of myths were also used to explain how the natural world operated, with myths telling why the seasons change, how day and night passes, and why the world experiences things like storms, drought, and famine. These types of stories vary in length and complexity, but generally, the origin stories will try to remain short and concise by introducing the setting and main characters briefly. The overall plot of these stories are intended to be very linear and easy to remember, as these stories

are often referenced throughout the overall mythology of the culture within the larger, more complex story types.

The other type of story we will discuss is what some, like Joseph Campbell [7], refer to as “The Hero’s Journey”. Illustrated in Figure 2.1 below in greater detail, the hero’s journey consists of three main stages: the departure, the initiation, and the return [7]. While these three stages contain an assortment of sub-parts, the heroic journey should contain these main components.

DEPARTURE

Every myth begins with the hero’s call to adventure; something that will motivate the character to leave the comfort of their familiar homeland. This ranges anywhere from a tale of lost treasure to the disappearance of a beloved item or person. Whatever the case may be, the hero is called upon either by the adventure itself or by a godly figure. From here, it is up to the hero to accept the call. Some answer the call eagerly, while others must be persuaded and encouraged by others. Once the call is answered, the hero ventures out in to the world and begins to make the important shift from the mortal/known world to that of the supernatural/unknown world. Typically, the departure within the story involves a physical transition between these worlds, where the hero crosses their first threshold on their journey. For example, in book 6 of the Latin epic, *Aeneid*, Aeneas, with help from a priestess, descends into the shadowy underworld by crossing the river Acheron. The river, in the story, serves as a physical barrier between the two worlds, with a mystical guardian in charge of controlling who may enter and leave the underworld. The departure section, in summary, sets up characters, setting, and situations and leads all the way up to the supernatural places and events of the story.

INITIATION

The initiation section of a myth deals with the main events of the story. This includes the various trials, interactions, struggles, and person transformation that the main character deals with. During the initiation, the hero learns how to operate in their new surroundings and interact with mystical beings that they probably have not had to deal with before. It is a time of personal growth, learning, and other complex character dynamics. This section usually includes a turning point in the hero's journey where they reach a realization or turn of events that leads to the climax of the story. The initiation section leads up to the final phase, the return.

RETURN

After the goal has been completed, whether that be a personal achievement, retrieval of an item, or rescuing an individual from a dangerous situation, the hero must make their return from the unknown world back to their own realm. The process of returning home can vary in difficulty depending on the condition of the individual and the current state of the relationship between the hero and the gods. One scenario that we see is that when it comes time for the hero to return home, the hero may not want to return on account of what they have seen and learned on their journey. The hero's journey is a time of great personal growth, both physically and mentally. Because of this, the hero may be faced with a period of reflection where they convince themselves that returning home is the right thing to do. However, sometimes, the return home may shift into a relocation to a place where the character feels they need to be.

The other scenario has to deal with the hero's relationship with the gods, since the physical side of the return usually is affected by them one way or another. Depending on the goal of the journey and the methods the hero uses to complete it,

the hero will either receive the approval of the gods and secure aid on their passage home or they will go against the wishes of the gods and be presented with a difficult return “complicated by marvels of magical obstruction and evasion”[7]. Finally, myths can become very socially complex, like in the *Odyssey* where the goddess Athena is actively trying to help Odysseus return home to Ithaca while other gods, such as Poseidon, work against them to delay his return. Even though Odysseus has the support of his patron goddess, the situation is made more difficult due to his actions with the cyclops Polyphemus, son of Poseidon, and the side Odysseus fought on during the Trojan War.

Regardless of the path chosen, the return portion of the myth means a return to reality and to the world that the readers can relate with. Here, we lose a lot of the magical elements from the story. The only sort of magic or supernatural involvement that remains in the story is down to the magical reward the hero receives or the spiritual connection the hero has to their patron god or goddess.

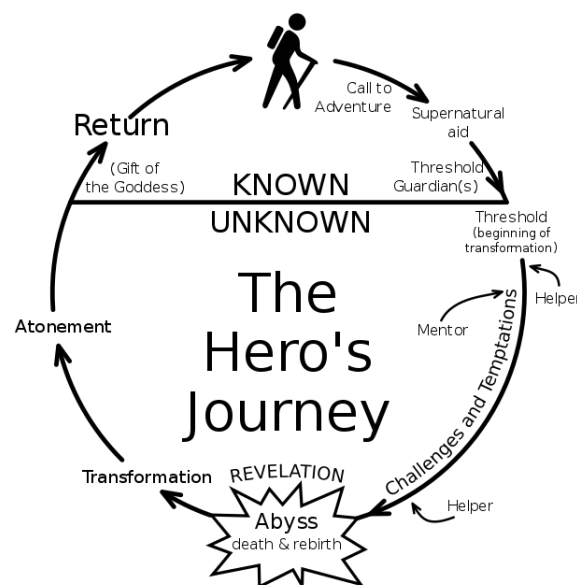


Figure 2.1: Quintessential journey made by a hero in myths that is described in Campbell’s book [7].

As you can see, there’s a lot of story elements that go in to making a myth. Such

elements would be difficult for a computer to replicate without a large amount of data and a complex training system. However, for our purposes, we can ignore a lot of the story components necessary in a myth, as keeping these pieces straight will be left to the individual writers. Instead, our program just needs to understand how to create independent sentences within the context of classical myths.

2.2.2 SENTENCE STYLING WITHIN MYTHS

In order to discuss the sentence structure of myths and their individual sentences, we must first understand the situation in which we are dealing with myths for this project. In their original forms, classical myths were written in ancient Latin or Greek. Under these particular writing systems, the text would surely follow one of the writing styles and sentence structure of the language. However, for this project, we are dealing with versions of the original text that have been translated to English, where a lot of these nuances have been removed through the translation process.

For our purposes, we are mainly looking for the style choices we see in the English translations of classical myths. Apart from normal English grammar rules, there are no clearly defined structural patterns within individual sentences of the myth. In most translations of the stories, the words are kept as close to the originals with most changes made to make the sentence flow properly or to explain a word, concept, or idea that existed in ancient times that may not map to our modern world. However, there are translations that diverge from the original text to the point where the translation becomes more of a parody to the original, such as when writers place the story within a modern setting. The story and characters remain the same, but character features, scenery, and other details are pulled out the ancient context. When we look to have a computer program generate sentences, we are looking to make sure that the sentences contain appropriate words of the classical period and that the ideas within the new sentences line up with the overall mythology. This

simply means that any future sentences made for the purpose of being used in a myth should be void of words that do not fit the classical era and any mythological content should be consistent with the rest of the works that have been produced.

In later sections, we will discuss how these components are monitored when designing a system capable of creating new sentences for the mythological context.

CHAPTER 3

MACHINE LEARNING

In this section, background information is given for the machine learning aspects of the project. This section sets up the techniques that will be used in designing the software portion of the project, which are needed before the software's process can be elaborated upon in a later section.

3.1 INTRODUCTION TO MACHINE LEARNING

Within the field of computer science lives an exciting research area known as **Artificial Intelligence (AI)**. Established in 1956, the area experienced fluctuating support and interest from scholars. In recent years, we have seen a sharp increase in the interest of AI, particularly by big tech companies, like Google. The study of AI deals with the “intelligence” that machines show, where the machine analyzes its environment and makes decisions that will ultimately help it complete its given goal in the most efficient way possible based on the parameters set by the programmers.

Machine learning (ML) is considered to be a sub-field of artificial intelligence, which focuses on the study of the algorithms and models that a computer actually uses to execute tasks set by a programmer. Through training on relevant training data, a machine can “learn” the most efficient way of completing a task, without being told the step-by-step instructions, allowing the program to navigate through problems of varying difficulty and compositions. Through various ML methods,

such as supervised, unsupervised, and reinforcement learning a machine can work out a given problem using the provided data.

Supervised learning is when an input is mapped to an output based on input/output pairs that are created using the training data. These pairs establish a generalization of the data that allow the machine to make an educated guess on the output that should follow the given input. With supervised learning, the categories that are used to distinguish the data are previously known by the machine, usually given by the programmer. In the table below, the categories size, color, and shape would be predefined, in addition to the names and category descriptions for all possible fruits. From there, the machine would receive an input of an “unknown” fruit; then by analyzing the fruit using the defined categories, the program would map the fruit to a specific fruit name given as an output.

Size	Color	Shape	Fruit Name
Big	Red	Rounded shape with depression at the top	Apple
Small	Red	Heart-shaped to nearly globular	Cherry
Big	Green	Long curving cylinder	Banana
Small	Green	Round to oval, Bunch shape cylindrical	Grape

Table 3.1: Demonstrating fruit classifications used in a supervised learning model. [12]

Unsupervised learning, like the name suggests, is similar to supervised learning except that the data is processed from scratch, without help from the programmer. The task of an unsupervised learning program also differs in that instead of dealing with a specific chunk of data and mapping to an output, data is processed as a whole and broken up through an unknown categorization method. Unsupervised learning methods therefore do not use training data to help make their decisions for their outputs. Instead, information, such as a group of unknown fruit, is taken in

and separated by various labels that are unknown to the user, but understood by the program. This method is much more hands off and black-box like in nature.

The last of the generally accepted machine learning techniques is reinforcement learning. With reinforcement learning, an input is received in the form of a start state, along with a overall goal for the model to try and achieve. Essentially, starting and ending points are given, but the computer has no knowledge on how it is going to efficiently make it between those points. Unlike supervised learning, there is no training data for the computer to learn the rules from, therefore the program must learning through a series of attempts and some sort of validation on how well each attempt went given after each try by a user. Also in contrast with supervised learning there are no correct answers given, therefore each answer, or output, the computer spits out needs to be evaluated by an outside agent. During this evaluation process, the computer will receive a response from a positive to negative spectrum. Based on this response, the program will return to the start state and try again with adjustments based on the type of response it was given.

3.1.1 MARKOV CHAINS

Perhaps one of the simplest machine learning techniques to understand are Markov chains. Markov chains are a system of nodes that each represent a state that can be traversed between, moving from one to another. As the Markov model requires data to be converted into state pairs before the chain can be used to generate an output, Markov chains are considered to be a form of supervised learning. The program will use a set of training data to learn from and build its “key” in the form of state pairs. Each state can represent any bit of data, whether it be numerical data or some real situation. For instance, consider a stoplight. At any given time, the light will either be red, yellow, or green. We could represent each light color as its own node and have connections between the nodes to match the pattern of a typical

traffic light sequence (i.e. moving from green, to yellow, then to red, and finally returning to green). Figure 3.1 below shows a simple two state Markov chain that has 4 possible transitions. Both states have the option of staying in their current state or moving over to the other. For this example, each option is equally likely to occur [1]. When dealing with Markov chains, the **Markov property** is assumed. The Markov property, for the purpose of this project, states that future states in the chain are only dependent on the current state, not the previous events that may have occurred. What this means is that if the chain has just progressed from state *A* to state *B*, when deciding to move to the next state *C*, the program is only concerned with the different options available to move from *B* to *C*. The computer will forget all about the information of state *A*.

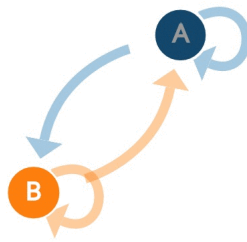


Figure 3.1: Simple Markov chain consisting of two states. [13]

However, simple Markov chains like the one shown above are pretty impractical and are only really good at explaining things at a basic level. Markov chains typically involve a great number of states that are all connected to one another in different ways. If we look at Figure 3.2, we can see a Markov chain that is comprised of 6 states (still a very low number of states). If you look closely, you will notice that each node is connected to each other and that each arrow between the two nodes varies in color. The color maps to a value that represents the likelihood that the transition from node *A* to *B* will occur. When moving between a given node, all of the possible next states are gathered and a new state is selected based on the

probabilities. This is a more realistic example as we rarely see equal chances when deciding which piece of data comes next.

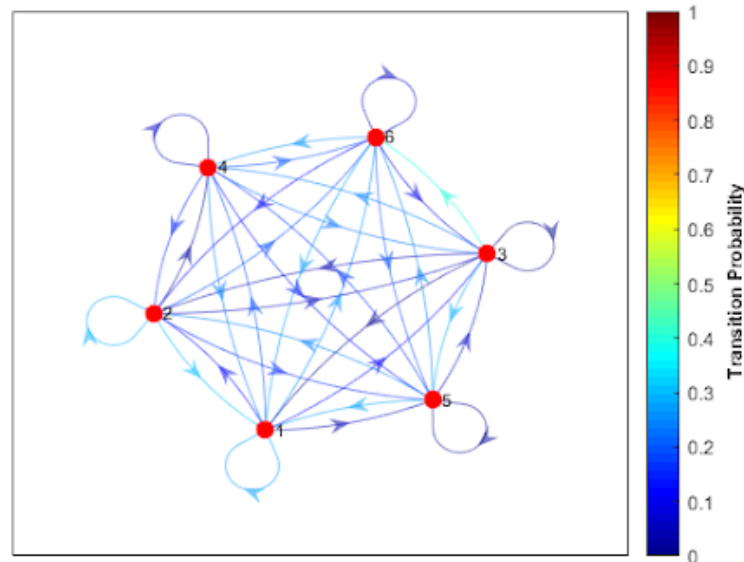


Figure 3.2: Complex Markov chain with probabilistic movement between states. [2]

Within a Markov chain, each state can be classified in to one of two states: recurrent or transient.

3.1.1.1 RECURRENT STATES

For any given state A in a Markov chain, the state is considered **recurrent** if, for any possible next state B there is at least one path that leads back to A . In Figure 3.3 below, states 2 and 3 are recurrent because they each have a self-loop and a path back after moving to the nearby state.

3.1.1.2 TRANSIENT STATES

In contrast with recurrent states, for any given state A in a Markov chain, a state is considered **transient** if, there exists a state B where we can come from A , but not go

back. In our example below, state 1 is transient because we can move from state 1 to state 2, however there is no way we can navigate between the remaining two states that would move us back to state 1.

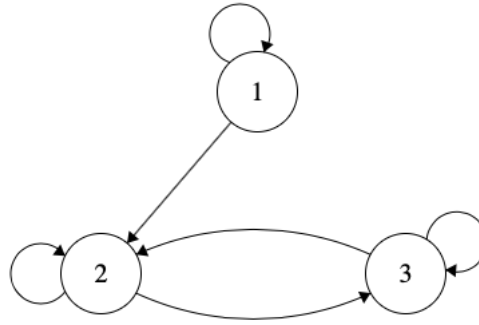


Figure 3.3: Markov model with both recurrent and transient states.

In the following section, we will discuss exactly how the data is processed and how the Markov chains are built by the program, including how weights are applied to the state transitions to allow for a probabilistic state transition.

CHAPTER 4

MYTH GENERATION USING MARKOV CHAINS

In this section, the overall process of the software portion of the project, the Writer's Block application, is drawn out and explained in detail using charts, diagrams, and explanations of the designed functions to understand the important steps that are occurring behind the scenes to produce the outputted sentences. The chapter concludes with an explanation of how the resulting text is evaluated in between iterations of the text generation trials.

4.1 WRITER'S BLOCK APPLICATION

The Writer's Block application idea came as a result of wanting to expand on the functionality and usefulness of a program that generated classical myth themed sentences. Rather than generating random sentences, the focus of the software was shifted from an independent, machine generated text process to a collaborative creative writing tool that combines the creative talents and know-how of the English language that a individual writer possess with the unfiltered and endless flow of text that a computer program is able to generate without fear of experiencing symptoms of "Writer's Block" that arises in humans.

Through this combination of human and machine processes, the project aims to create a simple, useful, low barrier to entry digital tool that a writer can use to help stimulate their creative thoughts and aid in the writing process. It is important

to understand that because the Writer's Block application is a tool, it is still up to the individual to verify and expand upon the ideas that arise through the machine generated text. The text that is outputted is a result of a machine's uninformed processing of a large amount of text where little to no contextual information is pulled from the data; rather, the words are treated as valueless strings. For the purposes of this application, the text generated might make sense grammatically or as a stand-alone sentence, but it is up to the user to determine if the sentence fits in the context of the rest of their work or, in relation to the focus of classical myths, in the context of the given mythological canon.

4.1.1 GENERAL OVERVIEW

The Writer's Block application, from start to finish, consists of four parts: data collection/cleaning, data processing, text generation, and text analysis. Before any text generation can be done by the machine, a great deal of work needs to be done to collect and clean the textual data that will be used to train the program and inform it on how to generate new text. This is discussed in Section [4.1.2](#). After this, the program can run through its two processes, data processing and text generation, where the training data is analyzed and then used to create new sentences. Finally, the text that was generated requires some level of analysis from a human user to determine the usefulness and level of correctness of the text. This process is important in evaluating and improving the generation process as each run of the software exposes new shortcomings of the software that can be fixed in subsequent versions. The following diagram maps out the general overview of the application.

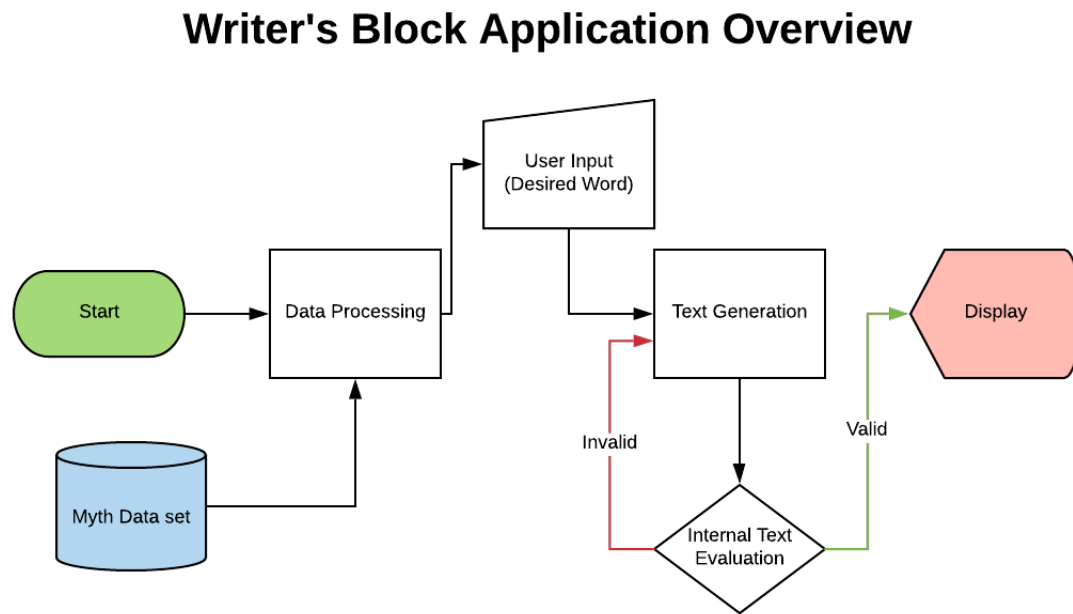


Figure 4.1: A flowchart for the Writer's Block application.

4.1.2 DATA CLEANING PROCESS

The process of creating and cleaning the data for this project begins with Project Gutenberg [4], which is an online database of over 57,000 free eBooks available to the public thanks to the public domain. Here began a search of the database for individual classical myths and entire collections of Greek and Roman stories. Once a particular text had been found to work with, the raw text version was downloaded, which is the best format to easily work with and manipulate the data. The process of cleaning the data involved handling the Greek/Latin letters within the text, stripping in-text citation and image references, and removing the front and end matter from each of the stories. For the most part, cleaning the data was simple and could be done through simple scripts, regular expressions (see Figure 4.3 below), and

separating myths into their own text files. The challenging part of the data cleaning process involved making sure the Greek and Latin letters were properly represented in the text files. For some of the stories, the conversion from the original text to the raw text file was fine and the letters were showing up correctly. This was not the case for all of the stories, with accented letters and compound letters (such as ï and Æ) all being converted to the same replacement character (Figure 4.2).



Figure 4.2: Unicode 'Replacement Character'

This meant that the text files had to be sifted through individually and the corrupted character had to be manually replaced with the appropriate Latin/Greek letter.

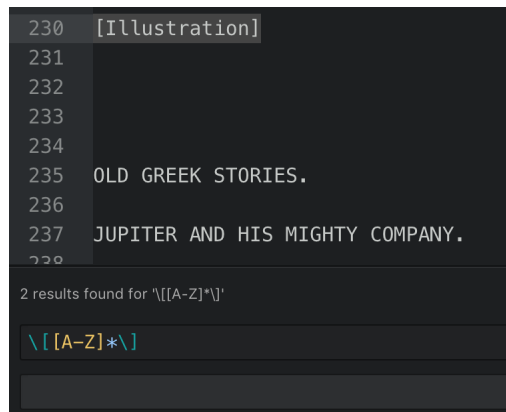


Figure 4.3: Removing unwanted text from data using RegEx.

The data cleaning process for this project was much different from others due to the way the data was collected. Since material was pulled from different texts, there was very little uniformity in the way the text files were organized. Another major issue that arose during the data collection and cleaning process of existing text files

was the fact that an overwhelming amount of text was being dealt with that had not been previously read through in their entirety. Because of this, there were a lot of text formatting and styling issues that caused complications in the data processing portion of the software, which is discussed further in the next section. Overall, this made the cleaning process more involved and prevented me from automating a majority of the work through scripts, which would have been possible if we had used our own, consistent data.

4.1.3 DATA PROCESSING

As shown in Figure 4.4, there are three main steps the program must execute before proceeding to the text generation phase: creating our word dictionary, classifying words in our data, and determining a list of possible sentence structures.

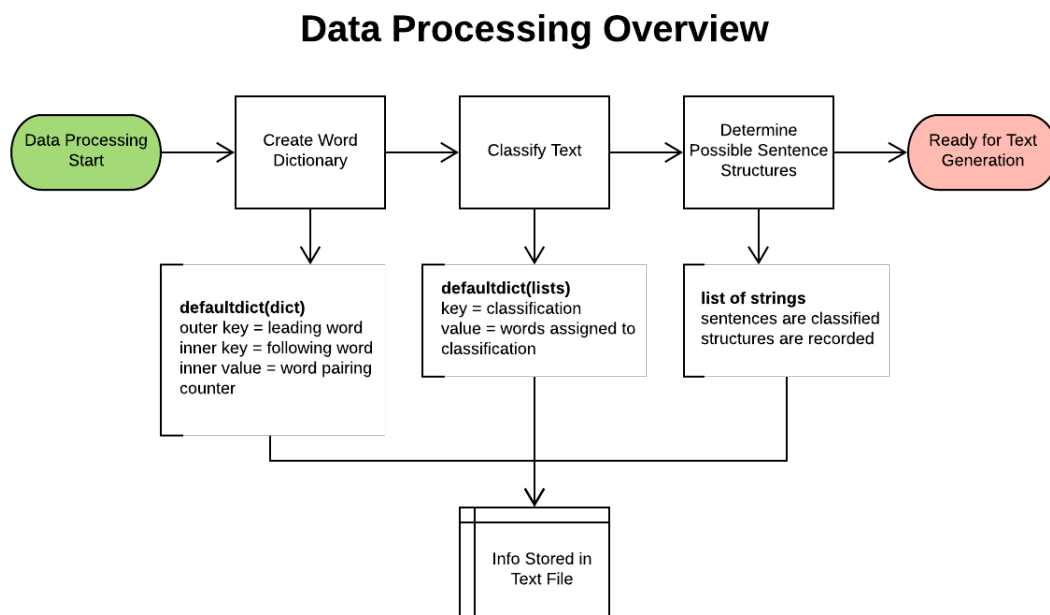


Figure 4.4: A flowchart for the Writer's Block data processing approach.

4.1.3.1 THE WORD DICTIONARY

To prepare our data for processing, each text file was modified to add the words "START" and "END" to the beginning and ends of the stories. This modification does not affect anything for the purpose of individual sentence generation, but was added in case the scope of the project changed and larger chunks of text was to be generated. This would inform us of how our myths tending to begin and end their stories.

After that, we set up our Markov chains by creating a dictionary of word pairings and the frequency in which these words appeared together throughout the data. To do this, we utilize the `defaultdict` Python library, which allows us to easily create complex dictionaries. For our program, we initialize a dictionary of dictionaries, each with their own key/value pairs. The outer keys represents words in the text with a value of a dictionary containing words that follow that word, as well as the frequency in which the the word pairings occur. Listing 4.1 below shows how this would look for the word "name".

Listing 4.1: A segment of our dictionary that keeps track of each word in the data, along with the words that follow it and the frequency in which the pairs occur.

```
1 "name": {"given": 1, "of": 35, "the": 2, "was": 42, "
    became": 1, "him": 2, "in": 5, "them": 1, "Pluto ,": 1,
    "to": 3, "is": 14, "has": 1, "means": 1, "a": 1, "as":
    1, "and": 6, "into": 1, "for": 4, "nor": 1, "will": 6,
    "every": 4, "might": 1, "among": 1, "by": 1, "whatever ,
    ": 1, "that": 1, "at": 1, "I": 2, "lives": 1, "would":
    1, "implies ,": 1, "she": 2, "which": 1, "Agamemnon": 1,
    "had": 1, "them": 1, "Astyanax ,": 1}
```

As we can see, the key "name" has a value pair of another dictionary with words

like “given”, “of”, “them”, etc. A number follows each of these inner words, which represents the frequency of a particular word pairing. The pair (“is”: 14) means that we see “name is” 14 times throughout all the myths in our data set. After our dictionary is created, it is placed into a text file. This is done to speed up the process later when our text generation functions want to use this dictionary. Creating this dictionary is initially very expensive, so we do not want the program to have to do this more than once during a single run of the program. Barring changes to the data set, the dictionary only needs to be created once. After the initial creation the rest of the program only needs to open and read the text file that contains the dictionary to access the information.

4.1.3.2 TEXT CLASSIFICATION

As we did with creating our word dictionary, we used `defaultdict` to keep track of the words in our database and which part of speech they fall under. To do this, we used NLTK's `pos_tag` function, which, given a word, would return which part of speech (POS) it is classified under. The table below shows the different classifications that NLTK recognizes and the abbreviation used within the code.

Key	Part of Speech		
CC	Coordinating conjunction	PRP\$	Possessive pronoun
CD	Cardinal number	RB	Adverb
DT	Determiner	RBR	Adverb, comparative
EX	Existential there	RBS	Adverb, superlative
FW	Foreign word	RP	Particle
IN	Preposition or subordinating conjunction	SYM	Symbol
JJ	Adjective	TO	to
JJR	Adjective, comparative	UH	Interjection
JJS	Adjective, superlative	VB	Verb, base form
LS	List item marker	VBD	Verb, past tense
MD	Modal	VBG	Verb, gerund or present participle
NN	Noun, singular or mass	VBN	Verb, past participle
NNS	Noun, plural	VBP	Verb, non-3rd person singular present
NNP	Proper noun, singular	VBZ	Verb, 3rd person singular present
NNPS	Proper noun, plural	WDT	Wh-determiner
PDT	Predeterminer	WP	Wh-pronoun
POS	Possessive ending	WP\$	Possessive wh-pronoun
PRP	Personal pronoun	WRB	Wh-adverb

Table 4.1: Text classification for NLTK's pos_tag.

We store the classification information within a dictionary of lists where each key within the dictionary represents a POS and the values are a list of words that fall under that classification. The classification function will work through the text files, classify the words that have not already been added to the dictionary, and update the appropriate list. Just as before, with the word dictionary, the completed POS dictionary will be saved with a text file for quick, easy access later on. This process also only needs to be run once and does not need to be run again, until the training data changes.

4.1.3.3 SENTENCE STRUCTURE LIST

The final text processing action that needs to be done is creating a list of acceptable sentences based on their POS orders. This step is very important to the sentence generation process, as it is our way of reducing the amount of junk sentences the program produces. At this point in the development, there was a need for a Python library that could analyze a sentence and determine if it was grammatically correct. However, after such a library could not be found, the decision was made to create, in some form, our own way of sentence verification. It quickly became apparent that creating this "miracle" code was not feasible, so a patchwork solution was designed. We knew that the sentences that appeared in our myths were legitimate, grammatically correct sentences, therefore we used them as examples to model our new sentences from. For that reason, a function was created to make a list of lists, navigating sentence by sentence through our text files. Each word for a given sentence would be converted to its POS and added to a list. Once a sentence was analyzed, the list would be added to a master list that would hold all the possible sentence structures within our training data. Just as before, once the function completed, the final list would be stored in a text file. The example below shows how one of these analyzed sentences looks within the text file.

Listing 4.2: One sentence converted to its POS to show a valid sentence structure.

```
1 " [ 'NNP' , _ , 'NNP' , _ , 'VBD' , _ , 'DT' , _ , 'NN' , _ , 'VBN' , _ , 'TO' , _ , 'DT' , _ , 'JJ' , _ , 'NN' , _ , 'IN' , _ , 'NNP' , _ , 'VBG' , _ , 'NN' , _ , 'IN' , _ , 'DT' , _ , 'NNP' ] "
```

While the list we ended up with no where near represents the complete list of possible sentence structures, this method served as an adequate technique to control the sentences that were being generated and eliminate a significant amount of unusable sentences, as we discuss in Section 4.2.

4.1.4 TEXT GENERATION

Figure 4.5 shows the work flow of how our program produces its generated text from the user input to the actually text generation process and internal checks.

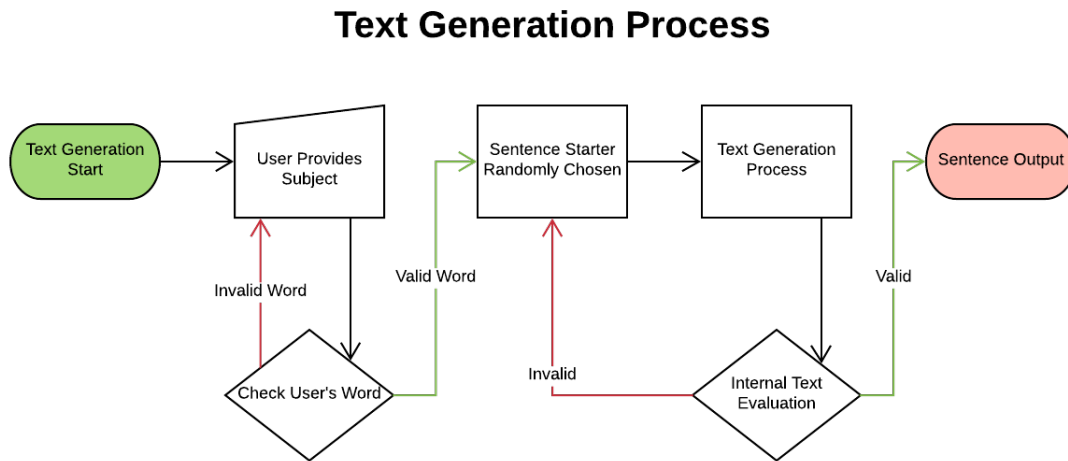
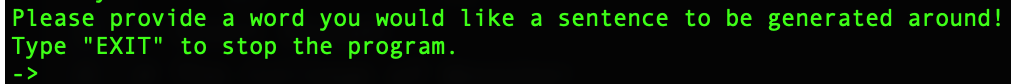


Figure 4.5: A flowchart for the Writer’s Block text generation process.

4.1.4.1 USER INPUT

The program begins by asking the user to provide a word to generate a sentence for (Figure 4.6). From here, the given word is checked with the word dictionary to see if a sentence can be created. Since we are using Markov chains which contain word pairings for words within the training myths, only words that exist in our chosen myths can provoke a result. Should the user provide a word not in the word dictionary, such as “television”, which would not appear in the classical context, the program will simply inform the user that the word cannot be used and to provide it with another word.



```
Please provide a word you would like a sentence to be generated around!  
Type "EXIT" to stop the program.  
->
```

Figure 4.6: Screen shot of the prompt the user receives to use the Writer's Block application.

4.1.4.2 SENTENCE GENERATION PROCESS

After a valid word has been given, the generation process can begin. The way we began our text generation process was by gathering a list of words called determiners to start our sentences off. **Determiners** are words that come at the beginning of a noun phrase that make it clear what the noun refers to. From this list, a word is randomly selected and placed in front of the user's word. This noun phrase is then added to our Markov chain to serve as the beginning of our chain that the rest of the sentence will be built from. From here, the chain probabilistically chooses word after word using the choice function of Python's numpy library. The choice function takes parameters of a list of options, in this case possible words that can follow the given word, to choose from and the weight for each option. Based on these weights, a new word is chosen to be added to the chain. This process continues until one of the stop characters ('.', '!', '?') is added to the chain. When the sentence is finished. The chain of individual words and punctuation is merged into one sentence as a string. At this stage, we have a sentence, but we must check to make sure it is appropriate enough to be presented to the user. To figure this out, the sentence is put through a checker function where the newly generated sentence is broken down into its parts of speech. The sentence structure of the generated text is cross-referenced with the list of possible sentence structures we made during the data processing phase. If the generated structure cannot match with one in the list, the sentence is scraped and the process starts over. When a valid sentence

is produced, it is presented on the screen to the user. From here, further analysis may be needed from a human reader to evaluate the computer's work and give developers and opportunity to make tweaks to the generation process.

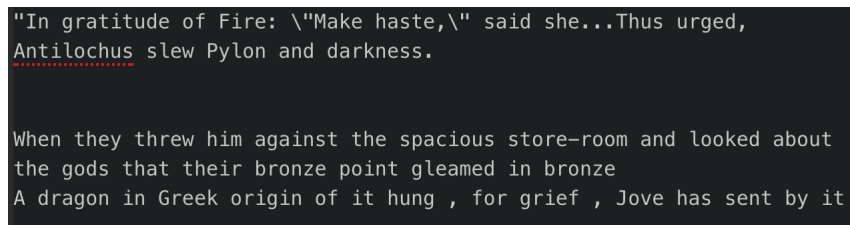
4.2 ANALYSIS OF GENERATED OUTPUT

For the program to produce the current level of output, the generation process needed to be improved over time. Thus many different outputs were generated during testing.

The first iteration of text generation was an exploration of what the program could make using the basic Markov chain method alone and the words from our training data. At the time a character limit was set and we told the program to just produce any sort of text. Unsurprisingly, the output was very messy, especially with punctuation. We would be left with a lot of unusable "sentences", if you could call them that, all of which would be left with random words connected with many commas and hyphens. An example would have looked something like this: "the,,Achilles"-I,I,'air". This output was not implementing any sort of weight or probability when moving from word to word and we realized that our program was not breaking up the words correctly. For instance, punctuation would be left connected to words so that "ship." and "ship" were filed under different words and thus, led down a different part of the chain.

The next iteration included the probability selection method previously described. After doing a better job of breaking up our data set and making punctuation separate, we entered another round of testing. This time instead of using a character limit, we instructed the program to produce some N number of sentences. At this point, these sentences were not perfect and the program did not understand sentences structurally, instead it just knew a sentence had ended when a period was reached.

Again, the outputs were not perfect, but we were starting to see some chunks of text that made sense or were just creative and amusing. In the image below, we see two bits of text produced by the program, the top text was produced with punctuation still not figured out and the bottom text was produced when punctuation was broken up appropriately. The text being produced here was starting to make sense, at least to the point where a human could piece together the gist of the story attempting to be told.



```
"In gratitude of Fire: \"Make haste,\" said she...Thus urged,  
Antilochus slew Pylon and darkness.  
  
When they threw him against the spacious store-room and looked about  
the gods that their bronze point gleamed in bronze  
A dragon in Greek origin of it hung , for grief , Jove has sent by it
```

Figure 4.7: Screen shot of outputs produced by an early version of the Writer's Block application.

At this point, a solution was sought after to produce better, more grammatically correct sentences. As described in Sections 4.1.3.2 and 4.1.3.3, we added in another level of text identification to help better determine what a good sentence looked like. Using this method, improved our results and the types of sentences we saw. However, since the checking process was added, the time needed to produce a result increased as sentence occasionally had to be discarded and attempted again. Some highlights from this stage of development include:

This was broken up, so that he had no thought dishonorable for she made his
hiding place over the quoit.

The Hellespont, bane of Troy as you have yet broken his son of great pain.

Finally, we implemented the system that asked a user for their desired word and had the program generate a sentence using it.

That fire flashing from distant Argos, until his hand of vantage disappeared.

This output came as a result of many requests for the program to generate a sentence using the word “fire”. Depending on the word, the program is able to create a sentence that is understandable to humans and something that could be considered thought provoking enough to create a deeper narrative around. The more common the word is in myth, the easier time the program will have in successfully generating a new sentence.

CHAPTER 5

VIABILITY OF AI IN CREATIVE DISCIPLINES

In this section, we will discuss what role, if any, should machine learning and AI have in creative disciplines; specifically in creative and informative writing. We will talk about the state of the discipline of machine learning and what can realistically be done in the present and near future, as well as some of the ethical concerns that arise when dealing with independent machines.

5.1 CAN IT BE DONE?

If we were to try and answer the question “Can AI replicate human writing?” a decade or so ago, many would probably say “No”. However, as previously discussed, over the last few years the world of machine learning has grown drastically with the increase in funding and research. Many large companies such as Google and Microsoft continue to improve text prediction software to aid users in sending texts, composing emails, and writing other documents. While these systems are far from perfect, the process of working with the users and learning how they write improves computers’ understanding over time. These systems aim to constantly improve their quality and usability for their users while simultaneously learning how to generate text in general. With this information, new products can be made to create text entirely generated by computer programs. These kinds of projects demonstrate

that, while not perfect, our goal of having AI create unique pieces of writing is not only possible, but something that could be achieved in the near future.

5.1.1 MACHINE GENERATED TEXT IN THE REAL WORLD

A perfect answer to the question “Can it be done?” is one of the latest projects by OpenAI, a non-profit artificial intelligence organization co-founded by Elon Musk. The company was created to research and develop safe and friendly AI that would hopefully benefit humanity [14]. The project was a large-scale unsupervised language model, which the company referred to as GPT-2. This model was trained on 40GB of textual data gathered from 8 million web pages. Given an input of a one-sentence prompt, the model was tasked with predicting a continuous stream of words based on the previous ones. The product of such a process was an impromptu article about the given topic. By scanning through the text from all these web pages, the GPT-2 was able to adequately complete various “language tasks like question answering, reading comprehension, summarization, and translation” [14]. However, since this is an unsupervised learning model, these tasks are learned simply through the text reading process and not from any task-specific training data that would be present in a supervised learning model. As a result, the program was able to mimic writing styles from sources like Wikipedia, novels, and news or media outlets. Assuming the user given prompt was valid and the topic was broad and/or popular enough, the model would produce convincing, well-written articles.

5.2 SHOULD IT BE DONE?

Machine learning projects, like the ones worked on by OpenAI are far from perfect and as a result, they can raise a lot of questions and concerns about the ethics of these kinds of projects. These concerns increase for projects that follow an unsupervised

model, as the programmers do not have control over how the program learns and what kind of content is produced. Only after running tests can the users view the output for the first time. It is because of this exact reason that OpenAI decided to postpone the release of GPT-2's latest build and roll back to their previous, more stable version. During testing of the current version, the team was concerned with the content that was being regularly produced as it was creating pieces that contributed to content that is "deceptive, biased, or [contains] abusive language"[14], which actively went against the company's friendly AI goals. Due to the content and sources the model was learning from, it became very good at creating misleading or fabricated "news" that the company did not want on the Internet. With so many other agents present in the online space, OpenAI did not want something they created to contribute to this already growing problem of watering down good, factual content. While OpenAI takes a step back and makes changes to their model, they chose to release a smaller, older version of the software for others to work with and continue research within the field. However, components like the dataset, training code, and model weighting were withheld to prevent other users from easily replicating the system the company chose to not release. These efforts do a great job of being conscious of the effects of a product while also keeping the research alive and accessible.

5.2.1 SHOULD IT BE DONE?: WRITER'S BLOCK APPLICATION

During the development of the Writer's Block application, a question arose if there were any foreseeable concerns with what the application could be spitting out and if so, what safeguards were thought of in preventing improper use of the application? While these points were definitely considered, no obvious concerns arose simply based on how this particular model generated the sentences. Unlike OpenAI's GPT-2, our program trains on known, trusted text from existing myths. Because

of this, every word that could potentially be used in the program's output would have to come from a myth in our data set. For the most part, the content of classical myths are fairly clean in terms of offensive language. While many ancient stories are filled with violence, slavery, and misogyny, the topics that could potentially come up in our results would be in the historical context and void of any and all charged issues of today's society. Since all words must be present in a story within our data set, should a user attempt to misuse the software and provide an inappropriate word to be used, the program is designed to reject any and all words that are not present in our myth word bank. If our program utilized a more hands off approach like with the GPT-2 model where the source material was not previously vetted, we would not be able to make the same guarantee.

CHAPTER 6

CONCLUSION

In this section, we will discuss the results of our research and development in the form of the Writer's Block application, as well as the the immediate and long-term goals for the future of the project.

6.1 WRITER'S BLOCK TOOL

The writer's block tool idea came as a result of an impasse that was reached in trying to create a Python script capable of generating myth based sentences. At some point in the research and development, a realization was reached that the original goal for the project seemed to be, although not impossible, more involved than originally imagined and that the solution to this problem required a lot more knowledge and understanding of machine learning and its techniques. As discussed in chapters [2](#) and [5](#), myths are very complex pieces of writing that require a great deal of creativity and understanding about the culture and history of a given civilization. Generating meaningful individual sentences is difficult enough for writers, let alone computers that have no contextual understanding of the words they are working with. Taking things a step further, it is not surprising to see that generating sentences to form a complete thought becomes an even bigger challenge. The work that writers do to create large bodies of text that remain not only consistent within the piece, but also within the culture's mythological canon requires constant thought and a deeper

understanding of the people, places, terms, and ideas being used at the time of story's setting. It is also the job of the writer to write in such a way that paints a scene with the words for the reader. As a writer, productivity comes in waves. Often times authors find themselves stuck on a particular point in their writing, unable to continue the flow of the story or unique ideas. They develop what is commonly referred to as 'writer's block'.

From these complications arose the idea of developing a tool for writers to help generate ideas to clear up their writer's block and progress with the story they were trying to tell. This idea became a perfect collaboration between man and machine, which is something we, as a self-declared digital humanists, value. Sometimes the answer to the problems of mankind is not "How can machines do this for us?", but rather "How can machines help us do this or make this process better?".

6.1.1 FUTURE WORK

The Writer's Block application that was created can be considered a digital tool for the use in the humanities. Two huge components of digital humanities projects are usability of a tool and open source collaboration. In order to follow these qualities, future plans for the software piece of this project were made for the near future. Firstly, in terms of the application, we can consider the current state to be in 'beta'. Before setting the program loose on the Internet for consumers to use, the code needs to be cleaned up and tweaked to perfect the outputs so that the users avoid sifting through many, less than perfect, sentences to find an acceptable result. Next, the script will need to be modified to fit the Twitter platform and work in its new role as a Twitter bot. Error handling will be needed to deal with invalid or inappropriate user inputs, as well as generating appropriate responses back to the user to present the sentence or sentences in accordance with Twitter's character limit. Lastly, in order to increase user engagement, the source code will be publicly

available on GitHub for fellow digital humanists and other interested individuals to make modifications to further the project as they see fit.

To facilitate these future plans, space on Twitter has already been reserved to house the bot, which can be found @DHWritersBlock. The flow of the Twitter bot process involves a user tweeting at the bot with a message along the lines of “@DHWritersBlock Give me a sentence using ‘x’”. The bot will only be concerned with the word, x , that is inside of the quotations. All other text will be ignored.



Figure 6.1: Example tweet from the user to the Writer's Block Bot.

After running the sentence generator program on the given word, the bot will reply to the original tweet with either a relevant, generated sentence or an error message if a sentence was unable to be created with the current data set.

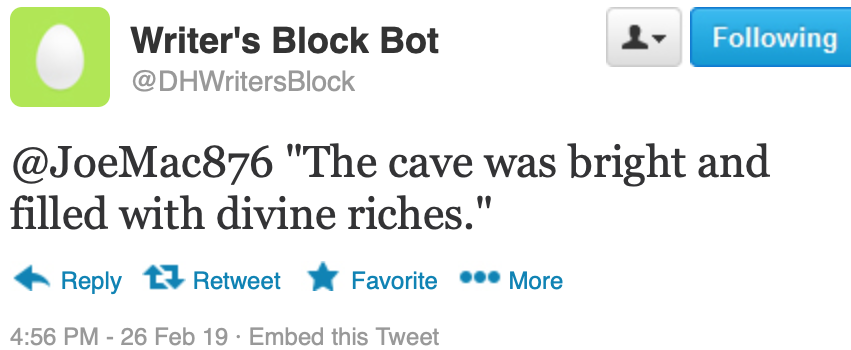


Figure 6.2: Example tweet from the Writer's Block Bot to the user with a positive response.



Figure 6.3: Example tweet from the Writer's Block Bot to the user with a negative response.

This particular bot will be running on Classical myth data, however, because the code will be available on GitHub, it will encourage individuals to fork the repository and create their own bot using their own data set, to shift the content material produced in the response tweets.

6.2 COMPUTER GENERATED MYTHS

The resulting product from this project was a tool that generated sentences in the context of classical mythology based on the key words given by a user. While this provides writers with a way of aiding the writing process, this idea of using computers to create new text could be expanded from independent sentences to interconnecting sentences, paragraphs, and potentially whole stories. For the purpose of our original goal of sentence generation, the Markov method that was used was enough. However, should this project expand to the extent of consistent paragraphs and coherent stories, the methods used would certainly need to change. Markov chains, at least by themselves, are not enough to produce such a complex output. For this future goal to be achieved, more ML techniques of varying complexity will need to be explored and utilized.

6.2.1 FUTURE WORK

After becoming better informed on more advanced machine learning techniques, this project would be revisited and exploration of the possibility of auto-generated myths would be done. Specifically, the sub-field of natural language processing, or NLP, would be a great place to start this exploration. Using **topic modeling**, an unsupervised learning algorithm capable of “discovering the abstract “topics” that occur in a collection of documents”[10] and **LDA**, an example of a topic model that classifies document text to a specified topic, the computer program potentially could learn enough about classical myths to generate one of its own creation. After hearing about projects such as OpenAI’s text generator discussed in Chapter 5, we can clearly see that these thoughts about some of the possibilities of what machine learning projects can achieve are shared throughout the community. This goal is inherently challenging and will take a great deal of understanding of machine

learning techniques. As research continues in the fields of machine learning and artificial intelligence, these goals will become more and more of a reality.

APPENDIX A

TEXT FILES REFERENCED

All text files and content referenced within this paper can be found in this appendix section.

Listing A.1: Myths in Data

```
1  A HANDBOOK OF MYTHOLOGY: THE MYTHS AND LEGENDS OF ANCIENT
   GREECE AND ROME
2  - E.M.BERENS
3  =====
4  hdem - Hymn to Demeter
5  ody - Odyssey
6  ili - Iliad
7  urag - Uranus and Gaea
8  cronus - Cronus
9  div - Division of the World
10 orman- Theories as to the Origin of Man
11 cadmus - Cadmus
12 perseus - Perseus
13 ion - ION
14 dadic - Daedalus and ICARUS
15 arg - The Argonauts
16 goldfl - Story of the Golden Fleece
17 pel - Pelops
18 herc - Heracles/Hercules
19 bell - Bellerophon
20 thes - Theseus
21 seven - The Seven Against Thebes
22 epi - The Epigoni
23 alcm - Alcmaeon and the Necklace
24 heraclidae - The Heraclidae
25 sot - The Siege of Troy
26 retgt - The Return of the Greeks from Troy
27 =====
```

```

28
29 MYTHS THAT EVERY CHILD SHOULD KNOW: A SELECTION OF THE
    CLASSIC MYTHS OF ALL TIMES FOR YOUNG PEOPLE
30 - HAMILTON WRIGHT MABIE
31 =====
32 tga - The Three Golden Apples
33 pom - The Pomegranate Seeds
34 chim - The Chimaera
35 goldt -The Golden Touch
36 gorg - The Gorgon's Head
37 dragt -The Dragon's Teeth
38 mirpit - The Miraculous Pitch
39 paradisec - The Paradise of Children
40 cycl - The Cyclops
41 =====
42
43 GODS AND HEROES OR THE KINGDOM OF JUPITER
44 - R. E. FRANCILLON
45 =====
46 sat - Saturn
47 gag - The Gods and the Giants
48 prompan - The First Man [The Story of Prometheus and
    Pandora]
49 gflo - The Great Flood [The Story of Deucalion]
50 latnio - The Stories of Latona and Niobe
51 flay - The Flayed Piper [The Story of Marsyas]
52 tmg - Too Much Gold [The First Story of Midas]
53 crit - The Critic [The Second Story of Midas]
54 laurel - The Laurel
55 hyac - The Hyacinth
56 sunf - The Sunflower
57 narci - The Narcissus
58 phae - Presumption [The Story of Phaethon]
59 orion - Diana and The Story of Orion
60 minerva - Minerva [Wisdom]
61 gof - The God of Fire
62 cuppsy - The Story of Cupid and Psyche
63 mai - Mercury and Iris
64 nep - Neptune
65 kqd - The King and Queen of the Dead
66 kingd - The Kingdom
67 orph - Orpheus and Eurydice
68 nmd - The Man Who Never Died
69 lostsec - A Lost Secret
70 champa - The Champion of Athens
71 appdis - The Apple of Discord
72 =====

```

```

73
74 OLD GREEK STORIES
75 - JAMES BALDWIN (Begins stories with I.)
76 =====
77 jhmc - JUPITER AND HIS MIGHTY COMPANY
78 ga - THE GOLDEN AGE
79 io - THE STORY OF IO
80 ww - THE WONDERFUL WEAVER
81 lotsb - THE LORD OF THE SILVER BOW
82 aa - ADMETUS AND ALCESTIS
83 qmh - THE QUEST OF MEDUSA'S HEAD
84 sa - THE STORY OF ATALANTA
85 ho - THE HORSE AND THE OLIVE
86 wa - THE WONDERFUL ARTISAN
87 ct - THE CRUEL TRIBUTE
88 =====
89
90 MAKERS OF HISTORY: ROMULUS
91 - JACOB ABBOTT
92 =====
93 saene - THE STORY OF ÆNEAS
94 faene - THE FLIGHT OF ÆNEAS
95 lil - THE LANDING IN LATIUM
96 rs - RHEA SILVIA
97 =====

```


REFERENCES

1. Markov chain vs. markov process. <https://www.hackingnote.com/en/machine-learning/markov>. 14
2. Create and modify markov chain model objects. <https://www.mathworks.com/help/econ/create-and-modify-markov-chain-model-objects.html>. viii, 15
3. Ai funding united states 2012-2018. <https://www.statista.com/statistics/672712/ai-funding-united-states/>. viii, 2
4. Project gutenber. https://www.gutenberg.org/wiki/Main_Page, Jun 2018. 19
5. Steven Bird. Natural language toolkit. <https://www.nltk.org/>, 2017.
6. Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with python*. OReilly Media, 2009.
7. Joseph Campbell. *The hero with a thousand faces*. Yogi Impressions, 2017. viii, 5, 8, 9, 10
8. Laura Dragonette. Artificial intelligence - ai. <https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp>, Apr 2018. 1
9. William F. Hansen. *Handbook of classical mythology*. ABC-CLIO, 2004.
10. Susan Li. Topic modeling and latent dirichlet allocation in python. <https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24>, May 2018. 30
11. Jain Parag. arxiv.org. <http://www.arxiv.org/>, Aug 2017.
12. Saimadhu Polamuri. Supervised and unsupervised learning. <http://dataaspirant.com/2014/09/19/supervised-and-unsupervised-learning/>, Sep 2014. ix, 13
13. Victor Powell and Lewis Lehe. Markov chains explained visually. <http://setosa.io/ev/markov-chains/>. viii, 14

14. Alec Radford. Better language models and their implications. <https://openai.com/blog/better-language-models/#sample1>, Feb 2019. 23, 24
15. Sebastian Raschka and Randal S. Olson. *Python machine learning: unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics*. Packt Publishing, 2016.
16. Carl A. P. Ruck and Danny Staples. *The world of classical myth: gods and goddesses, heroines and heroes*. Carolina Academic Press, 1994. 6, 7

