Exploring the Advancement of Artificial Intelligence through Approaches of Chatbot Models

Independent Study Thesis

Presented in Partial Fulfillment of the Requirements for
the Degree Computer Science Degree in the
Department of Mathematical Computational Sciences at
The College of Wooster

by
Bryson Simpson
The College of Wooster
2024

Advised by:

John Musgrave (Mathematical

Computational Sciences)

Abstract

This study will focus on the types and variations of chatbots in the field of artificial intelligence. In the study, there are three different chatbot models that are created and examined. The models are tested and thoroughly analyzed based upon their ability to handle data, respond to certain prompts or instructions, and other factors that enable user-to-machine interactions. Throughout the study, various concepts will be explained to enable the reader to obtain a more in-depth understanding and to also appreciate the historical context and reasoning of these concepts. These ideas along with the chatbot models are the components that represent and highlight the important landmarks of artificial intelligence technology and its development.

This work is dedicated to the future generations of Wooster students.

Acknowledgments

I would like to acknowledge Advisor and Professor John Musgrave in the Mathematical Computational Sciences Department for his ideas, resources, and insight.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 What is AI?

There isn't a simple way to define Artificial Intelligence(AI). The term has originated since 1955 by John McCarthy but there has not been an universal definition. [16] Therefore, there are multiple definitions of Artificial Intelligence. In his paper published in 2007, John McCarthy defines "Artificial intelligence" as the science and engineering of making intelligent machines. [15] Other definitions suggests that "artificial intelligence" refers to any computer that passes the Turing test [25] or "artificial intelligence" as an intelligent agent. [24] For the purpose of this study, AI will be referred to "intelligence and computational performances of machines". The first instances of "AI" can be traced to Warren McCulloch and Walter Pitts in 1943 where they developed a model of artificial neurons. In this model, they suggested that each neuron had the option to be expressed as "on" or "off".[23] Today, their work is praised as one of the catalysts that ultimately and expansively developed further in the 20th and 21st century. In the last 10 years, the adoption of AI has been unexpected, controversial, and transformative in society. Certain AI technologies have improved the cinematography of movies and television shows and industries are continuing seeking AI to enhance their products. [35] Finance businesses use AI for responsibilities such as fraud detection, trading, and computer service. [17] AI has made an exceptional impact in healthcare as its inclusion has aided in medical image analysis, health diagnoses, and even medical based discoveries. [31]

AI's sudden rise in the world's infrastructure has introduced a list of ethical concerns from many people. These concerns exist as AI is prominent in unorthodox roles that spread into various aspects of society.

Some ethical issues regarding the technology include internal bias, privacy, security, environmental impact, and social impact. [5] Many figures expressed discomfort on how AI is treated, whether it is involved in the job industry, entertainment industry or government affairs. However, one of the biggest concerns of AI is job displacement. [31] Job displacement suggests the idea of AI assuming the responsibility of many man-made jobs. This is possible if a computer has the same cognitive means of a human being. If a computer has the intelligence to act, think, and communicate like a human, then this concern is valid. This emerges the question on how would a machine match the intelligence of a human. The natural way a computer can try to be human is to communicate with humans.

1.2 What is a Chatbot?

One of the common ways that AI interacts with us in our day-to-day lives is the form of intelligent agents. An intelligent agent is a software program that has the capabilities to perform tasks and independently make decisions based on its environment, experiences, and input of the user. An intelligent agent is commonly referred to as a "bot". Intelligent agents are used to solve complex problems and complete specific assignments that are considered difficult or impossible for humans to achieve. Additionally, its own abilities can improve further by learning and obtaining knowledge.

A chatbot is a specialized alteration of a bot that uses its intelligence to simulate human conversation with a user. Chatbots are regarded as the most conventional approach to carry out communication between humans and computers. Many chatbots use common methods such as rule-based expert systems and deep learning networks to understand the user's inputs and questions in order to generate an appropriate automated response. Chatbots are simple alternatives that are utilized by users to find out information in the form of a question or request without the need of manual research or human consultation. Chatbots can communicate through various means via text and audio. Chatbots are extremely versatile and can be customized in different approaches. The many different ways to create and implement an interactive machine is paramount regardless of the range of quality of a chatbot. Today, chatbot technology is frequent among company platforms that feature user interfaces. [3] The most recent development of AI chatbots can understand fluid human dialogue through use of advanced language models and training.

1.2.1 Well-known Chatbots

The earliest instances of chatbots date back to the 1960s and 1970s. [23] One of the well-known chatbot programs is code-named Eliza, which was created by the Artificial Intelligence Laboratory in MIT. This chatbot ran a script called DOCTOR which simulated "the role of a Rogerian psychotherapist" (ZEMČÍK). Eliza had the ability to form open questions with which she also answers herself. The intention was to divert the attention from the machine to the user. Another well known chatbot is PARRY, which simulated the role of a paranoid person with schizophrenia. PARRY invoked a crude personality and was regarded as more confronting to the user. Both of these chatbots are critically acclimated in artificial intelligence for being the fundamental blueprint for designing chatbots.

Currently, we have large language models like ChatGPT, ELECTRA, T5 that represent the current technology of chatbot and artificial intelligence. These models are fairly new and they are constantly improving at rapid rates. The large language models such as GPT-3.5, Incoder, and Code Llama contain multiple layers and billions of parameters. [10] These models are very powerful for the fact that they have access to data from the internet. This aspect is very outstanding as the internet has an innumerable amount of information that humans can't realistically contain in a normal data set. Chatbots are the link of communications and interactions between humans and computers.

1.3 Machine Learning

Machine Learning is a field of Artificial Intelligence where computers have the ability to learn and comprehend information without being directly programmed. [14] The creation of a chatbot is the amalgamation of systems, concepts, and theories that existed throughout history. It is essential to recognize these historical backgrounds in order to gain an understanding, as well as the appreciation for the subject matter in this study. The historical telling of this information also involves the telling of the origins of machine learning, and computer science. Because of the way that the overlaying systems are structured in the computational field, there is a follow up effect in the sub fields where the implementations and ideas of those new systems are already based on those systems which are the foundation of computer science. Therefore, it is crucial to understand the history and context of the overarching field of computing before delving deeper into the realm of chatbots.

The Machine Learning field optimally displays how AI attempts to behave, think, and perform actions like a human being. The field of Machine Learning is engaged in the concept of computers carrying out tasks that can be performed by humans. "Interesting examples include facial recognition, translation, responding to emails, and making data-driven business decisions." [1]. Machine Learning is a popular field and it is a prominent aspect in every-day life globally that most people are subconscious about. Well known examples include media-related algorithms personalizations, live traffic forecasts in map applications, and virtual assistants. More common machine learning applications include sentiment analysis, merchandise recommendations, access control in company portals, and banking domain. Another prevalent machine learning application is email detection. This software determines whether an email is spam or not. There are an abundance of programs, software, and gadgets that apply Machine Learning technology world wide.

The main component that makes up the basis of machine learning is data. The intention of machine learning is to learn from data. There are various approaches to how AI learns information. Data are commonly expressed in the form of a data set. A prevalent use case of machine learning by industrial settings is extracting relevant data. Since there is an overflowing amount of digital data sets in many real-world settings, applications that employ machine learning techniques are used extensively.

Machine Learning has several types of algorithms that each function differently to solve data-related problems. Machine Learning can have either supervised or unsupervised methods. Supervised Learning is usually applied if there is a short amount of data and data is labeled for training. The employment of unsupervised learning would highly depend on the performance of a machine and its capabilities.

It is widely accepted that machines operate faster than humans and their abilities to carry out tasks will immensely expand over time.

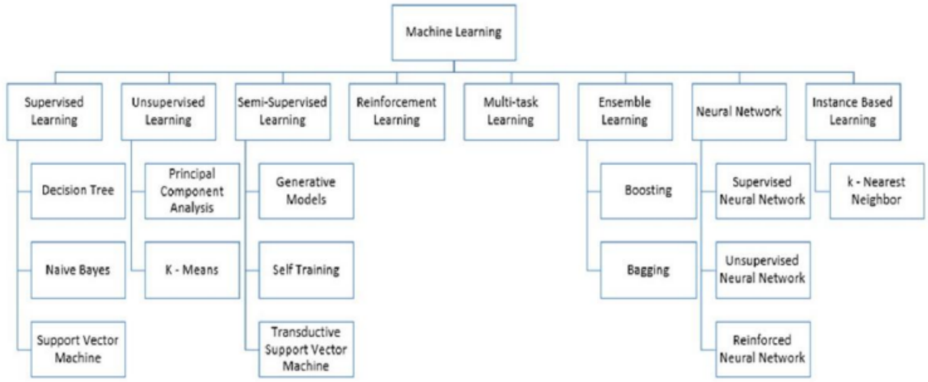


Figure 1.1: Machine Learning Algorithms
[14]

1.4 Logic

The design philosophy of Artificial intelligence can be attributed to logic. Logic is a reasoning method used to validate something through certain conditions. The first instance of logical-based concepts can be traced back in the 5th century B.C. when Aristotle created “syllogistic logic”. Syllogistic logic is widely considered the first model of the formal deductive reasoning system. Aristotle’s logic is more structured and systematic approach to logic, which was focused on building upon an concept compared to other philosophers such as Socrates. Socrates focused on deducting a statement or thought by asking questions and creating arguments, with his goal contemplating whether a concept is valid or not. In turn, these two figures left a monumental impact and would spark an ongoing debate of what "truth" really means. Logic is the fundamental core of multiple fields such as mathematics, philosophy, and computer science. [22] Propositional logic would be the expansion of syllogistic logic where deductive reasoning focused on logical expressions formed from propositions. This type of logic would further explored and expanded by scientist, Bertram Russell. Bertram Russell uses the concepts of first-order functions and other propositions to formulate first order logic. [11] First order Logic is similar in structure to propositional logic, however, there is one main difference. First order logic includes variables and quantifiers for non-logical subjects unlike propositional logic. This opens of possibilities of testing out of certain approaches or methods that were only able to be theorized previously. First order logic was a central cornerstone that elevate the fields of mathematics and computer science.

In computer science, there is an ongoing ambition to form reasoning and validate programming decisions in the field. This is expressed through the algorithmic mechanisms created that adhere to the regulations of reasoning. All of these "mechanisms" are computer-based systems and all of a computer's interaction is built through exclusively language and reasoning.

Logic stands as a foundational element within the creation of Artificial Intelligence, profoundly shaping its development. Ultimately, Logical reasoning is the centerpiece that links both human intelligence and artificial intelligence together.

1.5 Alan Turing: Turing Test

For centuries, human intelligence was deemed as the highest and complex level of intellect. This statement has been considered factually correct, until the rise of artificial intelligence. To understand how artificial intelligence and computers became what they are today, it is essential to find its origin through the man named Alan Turing.

Alan Turing is considered the father of Machine Learning as well as a pivotal figure in computing and artificial intelligence fields. He laid the foundation for modern computing and cryptography. Turing is celebrated for conceptualizing the Turing Test in 1950, which became the model of artificial intelligence design. [19]

The Turing Test is a proposed method to determine a machine's ability to exhibit intelligent behavior indistinguishable from that of a human. This test involved a human reviewer engaging in natural language dialogue with both a machine and another human without knowing which was which. If the reviewer couldn't identify the difference between the machine and the human, the machine was categorized as passing the Turing Test, indicating its own level of intelligence was comparable to a human's. As a result, the Turing Test sparked a discussion about machine intelligence, whether insentient machines could reach a level of consciousness, and thus the nature of human thought. Turing's knowledge of algorithms and computation were further exemplified by his creation of the Turing machine, a adaptable device that is theoretically capable of executing any task. The Turing machine became the framework for machine learning algorithms and development of modernized computation. Turnings' work laid the foundation for future developments in artificial intelligence research and the field of machine learning continues to evolve thanks to his contributions.

1.6 The History of Neural Networks

1.6.1 Rosenblatt: Neural Networks

After the monumental creation of the Turing Test, prominent figures in the computational field discussed and conceptualized approaches to design computer machines that could display intelligence behavior. Alan Turing proposed two possible directions of computer programming. One approach explores the field of biological systems, where scientists use methods that display elemental comprehension that could be replicated and expanded upon in programming. The other approach is based on a problem solving study where experts approach tasks by using certain rules. When programs are built, then the programs would utilize these rules depending on the circumstances. Many people, including Turning, within the field focused on the rule-based approach as it seemed less convoluted and more practical. However, there was a selected group of people who focused on theorization of

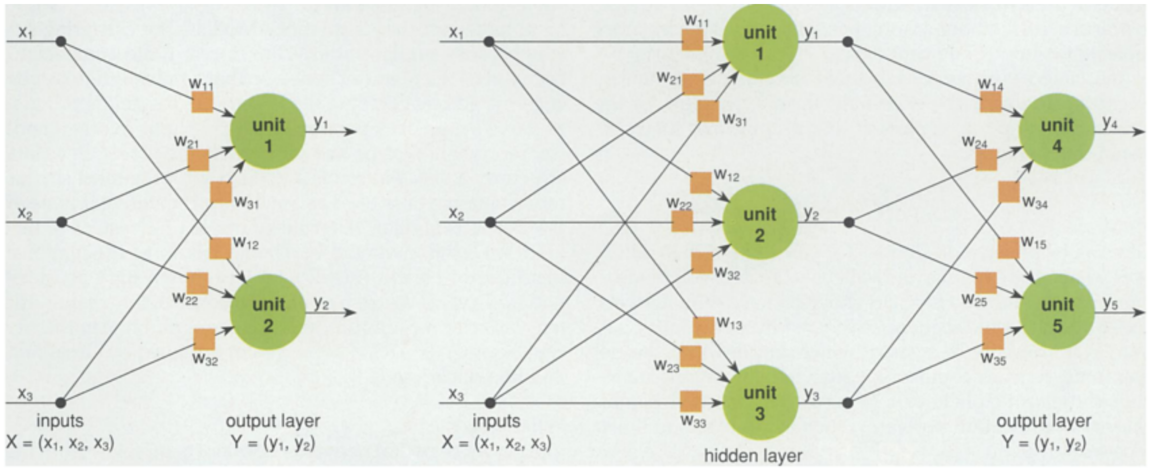


Figure 1.2: Rosenblatt's Perceptrons
[6]

neural computation, and thus neural networks. In his article, Peter J. Denning mentions the concepts of neural networks were produced by a pattern-matching framework which states "that cognitive behavior is most likely to arise in circuits whose structure closely resembles certain biological systems." [6] In the mid-1950s, Frank Rosenblatt introduced a training philosophy applied towards neural networks. He proposed that neural networks needed to remember functionalities such as the link between inputs and outputs. Rosenblatt is responsible for the invention of perceptions, trainable neurological-inspired networks without accessing feedback loops. The prime objective of perceptions is to have a network's output correspond to the desired output for a chosen input. There are two types of perceptions. There are single-layer perceptions which are the simplest perceptions that have units arranged in a single layer. Single layer perceptions are used to solve linear problems. Then, There will be multi-layer perceptions that have units in the hidden layers. These perceptions are complex and do not immediately generate results.

However, Rosenblatt's perceptions have been publicly criticized regarding its functionalities. The single-layer perceptions were criticized for their limitations due its "constraint of linear separability." [6]. This evidence actually persuaded national organizations to stop funding for neural computing research and ultimately most progress of neural networks stopped for many years.

1.6.2 Hinton: Back Error Propagation

Although neural networks weren't of focus on research, researchers began looking for methods to restructure Rosenblatt's perceptions. The resulting solution that they have found is known as backward error propagation. [6]

Backward error propagation is an algorithm where errors in the output are used to update the weights applied to the inputs. Weights are known as parameters that are associated with the connections between neurons inside a model during the training process. The algorithm was proven successful and eventually the progress of Neural Networks started back up in the 1980s.

1.7 Current Neural Networks

Today, neural networks are known as machine learning systems that operate similar to the functions of a human brain. The reason why they are labeled as neural is because they are inspired by the behavior of actual neuron patterns. "The neural network of an human is part of its nervous system, containing a large number of interconnected neurons (nerve cells)." [8]. The existence and development of neural networks exemplifies the concept that AI can enact the actions and thoughts of a human. Some computer scientists specifically classify Neural Networks as Artificial Neural Networks (ANNs), to clarify the difference from biological neural networks.

Neurons take in information and when they finish processing the data, it is sent to neurons that are in the hidden layer. After a set of instructions is completed by the hidden layer neurons, an output will be generated.Two popular neural networks architectures used today are Recurrent Neural Networks(RNN) and

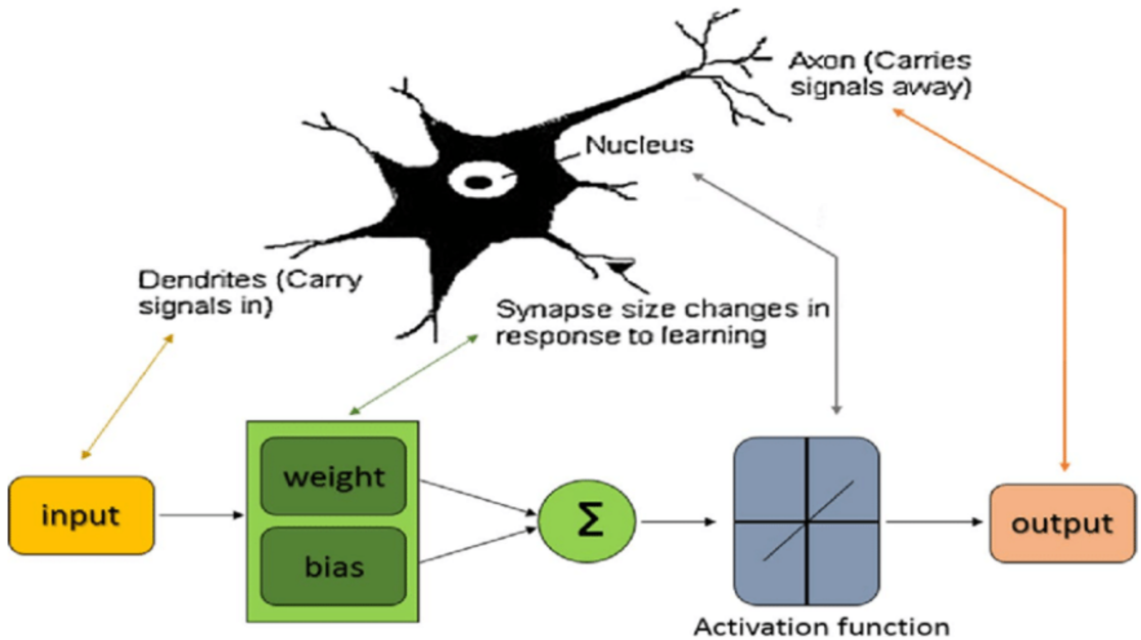


Figure 1.3: Biological Neural Network
[27]

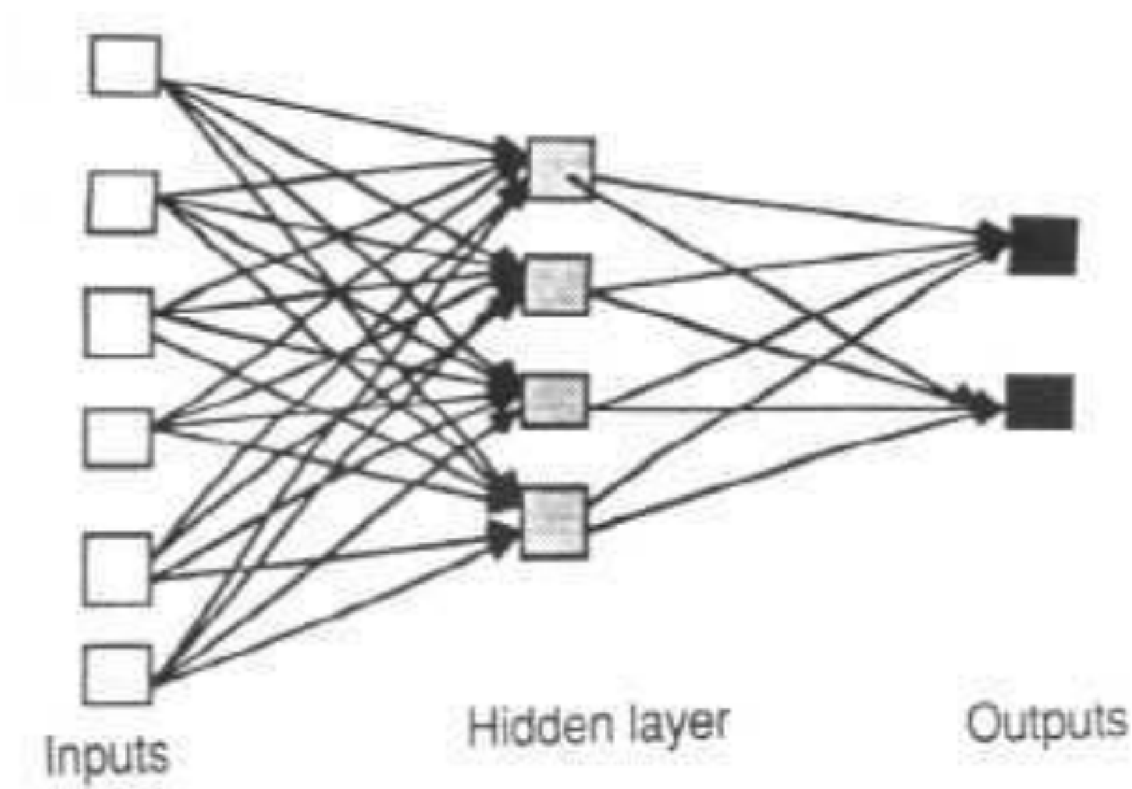


Figure 1.4: Artificial Neural Network
[9]

No	RNN Architectures
1	Fully Recurrent Neural Network
2	Recursive Neural Network
3	Hopfield Network
4	Elman Networks And Jordan Networks or Simple Recurrent Network (SRN)
5	Echo State Network
6	Neural History Compressor
7	Long Short-Term Memory (LSTM)
8	Gated Recurrent Unit
9	Bi-Directional Recurrent Neural Network
10	Continuous-Time Recurrent Neural Network (CTRNN)
11	Hierarchical Recurrent Neural Network
12	Recurrent Multilayer Perceptron Network
13	Multiple Timescales Model
14	Neural Turing Machines (NTM)
15	Differentiable Neural Computer (DNC)
16	Neural Network Pushdown Automata (NNPDA)

Figure 1.5: Types of Recurrent Neural Networks
[12]

Convolutional Neural Networks(CNN). [33] This study will be focused on recurrent neural networks as they are the primary building block used in most interactive virtual agents. Recurrent neural networks are designed to process sequential data by retaining memory of prior inputs. RNNs use their hidden state that captures information of preceding inputs in the sequence. Currently, Neural Network models are displaying intelligence at innumerous levels and they are constantly improving.

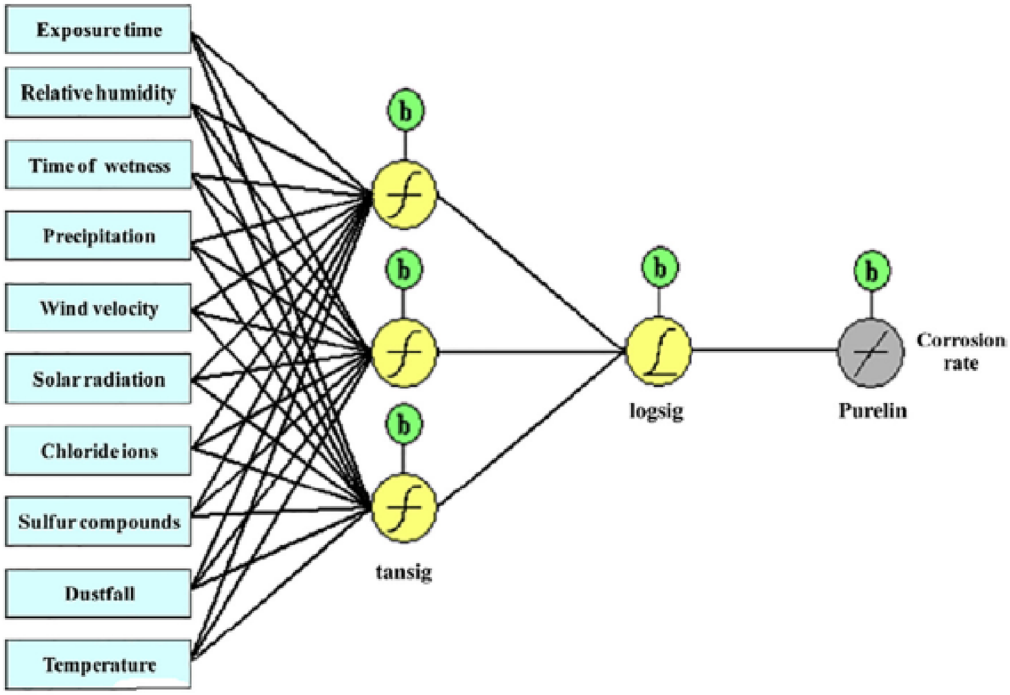


Figure 1.6: Example of an advanced-level neural network [21]

1.8 The History of Rule-Based Expert Systems

After in depth discussion of the direction of artificial intelligence, many people in the computational field began developing machines through a rule-based format. Programs that were created through this design philosophy were coined as Rule-Based Expert Systems or Expert Systems. Expert Systems use human knowledge and logic to solve problems. The knowledge that is given to the computer is expressed in the form of "rules". [1] Rule-based expert systems have had a pivotal role in shaping modern day intelligent systems, contributing to their utilization across various fields such as planning, design, scheduling, diagnosis, and similar areas of application. Human knowledge and logic are ingrained within the program's code. When information gets altered, then the code has to get updated. The program uses the knowledge that they are given as reference and as an approach to respond with answers, implications, and predictions to a user. A program must look for the

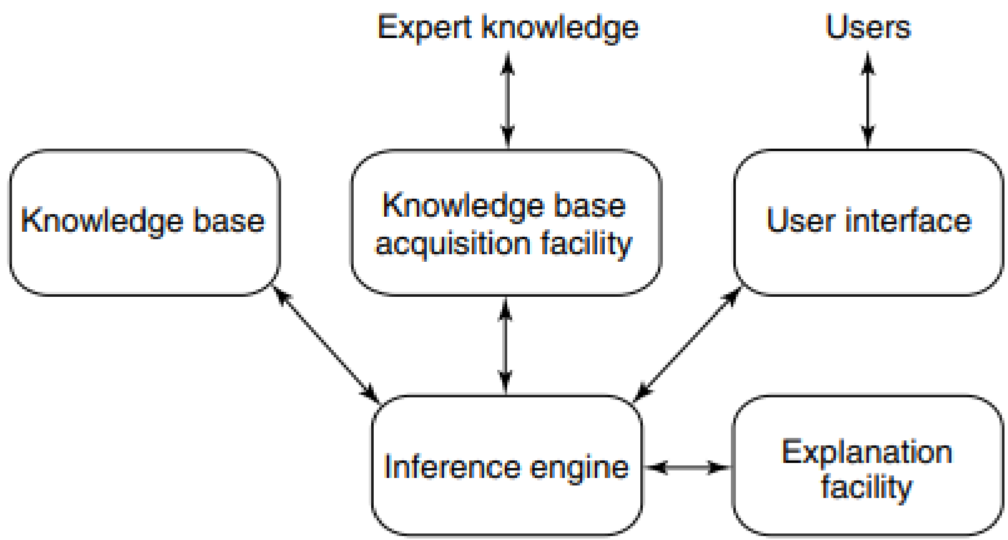


Figure 1.7: Diagram of Simple Expert System
[1]

correct information, follow the criteria, determine the relationship between input and output, in order to give the desired outcome. This program within this process is sometimes referred to as an inference engine. There are two types of inference methods that are used frequently. Backward chaining is the process of starting with a result and working backward to the supporting data. Forward chaining starts with the data and then working forward to the result.

One of the first common rule-based expert systems was MYCIN. MYCIN was a rule-based expert system constructed by researcher Edward Shortliffe in the 1970s and became one of the pioneering rule-based expert systems. MYCIN was a clinically-based system that retrieved clinical data from a user and created a diagnosis for a patient referring to the data. The diagnosis consists of identifying certain infections and recommending treatments to the patient. Although MYCIN never had any practical use in the real world, MYCIN demonstrated the potential of rule-based expert systems at the time and ignited an upcoming generation of rule-based expert systems.

Simply put, the existence of rule based expert systems advanced the progress of artificial intelligence to great heights in an era where the machine learning field was not present.

1.9 Artificial Intelligence exercised through Chatbots and other applications

Chatbots are the modern, interactive, and optimal form and use case of Artificial Intelligence. As previously mentioned, artificial intelligence advances further the more it has human-like capabilities. Giving AI the ability to communicate and behave like a human, complicates certain systems further but the high number of possibilities open.

1.9.1 Deep Learning

Currently, most chatbots implemented in high-industrial settings are built through the framework of deep learning. Deep Learning is the expansive field of neural networks and exploring their technology further. Neural Networks are classified as deep learning algorithms as they possess “deep” layers for data to be stored and extracted. Many mechanisms that resulted from deep learning include image recognition, object detection, and language translation. [3]

1.9.2 Natural Language Processing

An important aspect that influences a chatbot and its relationship with human dialogue is natural language processing.

Natural Language Processing is the process of translating human dialogue to machines so that machines can learn and recognize patterns regarding our speech.

NLP gives computers the ability to read data in various natural languages in order to perform functions such as machine translation, sentiment analysis and natural language generation. [7] Computers are intended to process human language in the form of text, pictures, and audio while also having the intelligence to understand the meaning of human language input and knowing the motive of the user.

Many noteworthy examples of Machine Learning applied Artificial Intelligence that incorporates NLP includes Apple Inc's Siri, Amazon's Alexa, and Snapchat's Chatbot.

Example of Siri, common tool utilizing NLP

Natural Language Processing propels computer software capable of translating text between languages, interpreting spoken instructions, and swiftly condensing extensive textual content, even in real-time.

1.9.3 Large Language Models

Large Language Models are the newest and powerful version of AI technology. [4] These models became prominent in the last 5 years. Large Language Models could be defined as the greatest amalgamation of modern-day technology which is influenced by the idea of computers mimicking the human brain. LLMs uses past developed notions such as NLP, User-Computer interaction, and neurons and augments them to intensive degrees.

LLMs differ from neural networks and other regular language models by having not only being larger in model size, but they display a higher language understanding and generation abilities, and unprecedented learning proficiency. [18] Prior models were limited by the fact the they were capable solving certain tasks. [4] Large Language Models have knowledge from a wide range of fields to solve complex logical tasks and can be proficient in other areas such as writing code and creative writing.

LLMs are AI models trained to predict the next word or a series of words of designated text. LLMs are structured on recurrent neural networks (RNNs), such as Long Short-Term Memory (LSTM) networks or Transformer architectures, namely GPT (Generative Pre-trained Transformer). The self-attention element in the Transformer is the fundamental building block that permits models to perform a plethora of tasks. [4]

The usage of Transformer architectures have improved the prowess of language models. Special features, like the self-attention mechanism, allow the Transformer architecture to search for the relevant information of the input when making

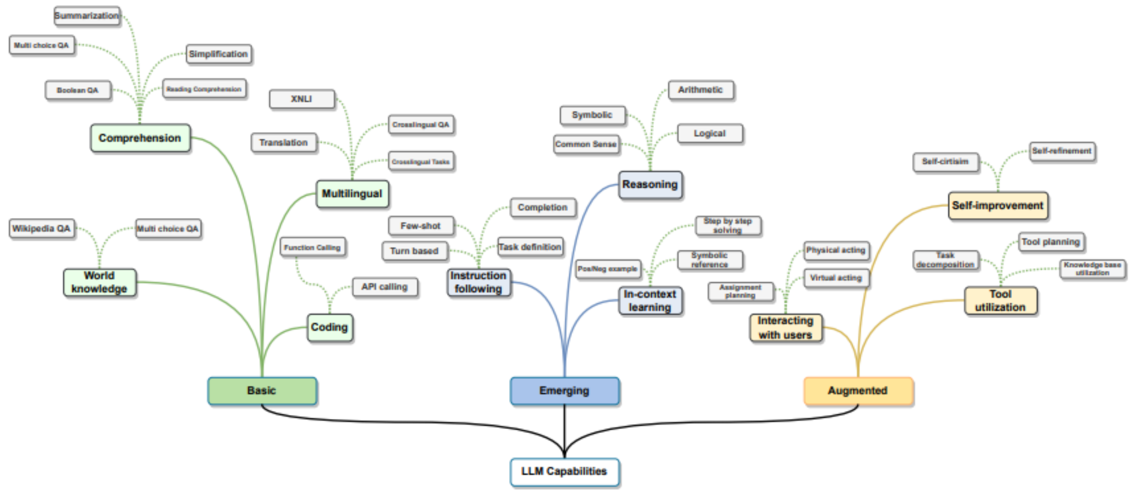


Figure 1.8: Large Language Model Capabilities [18]

predictions and therefore can make sense of the relationship between words in a sentence or expression.

Through chat interfaces, LLMs can communicate with humans which has led to many individuals of society to adopt these systems and further explore their capacity. The popular variations of closed-product LLMs include ChatGPT, BARD, and Claude.

LLMs are trained on large innumerable datasets of text. The texts are usually extracted in books, articles, and the internet so that LLMs can strongly learn the nuances of language and context. LLMs and Seq-to-Seq Models are similar in how they are trained on conversational datasets. The difference is that Large Language Models don't rely on sequence-to-sequence aligning and they learn the statistical patterns and relationships between words in a large body of text data.

LLMs are assembled with certain properties that enable themselves to the unique abilities that they have. One of these determinants is the greater attention of reinforcement learning techniques. Reinforcement Learning with Human Feedback (RLHF) is one of these techniques that allows the models to learn from their mistakes and develop a grander understanding about human preferences. Another type of

Comparison	Traditional ML	Deep Learning	LLMs
Training Data Size	Large	Large	Very large
Feature Engineering	Manual	Automatic	Automatic
Model Complexity	Limited	Complex	Very Complex
Interpretability	Good	Poor	Poorer
Performance	Moderate	High	Highest
Hardware Requirements	Low	High	Very High

Figure 1.9: Comparison Table of Traditional Machine Learning, Deep Learning, and LLMs [4]

these determinants is in-context learning. The model is trained to generate text based on contextual inference or instructions that are given. With this added feature, LLMs are very likely to generate relevant and appropriate responses, making them applicable for conversational utilization. These factors, in turn, enhance their usability with users and their overall impression among the public.

However, LLMs are criticized for their costly requirements of computation which can result in ambiguous or not replicable formats. This issue can limit progress in AI-development and research fields.

There are two different approaches to interact with the LLMs. One method is commonly referred to as prompt engineering. Users create a prompt with specific guidelines to LLMs so that they can produce desired outputs and solve out certain assignments. Another approach is through an interactive question and answer format. Depending on how much information the model has, users can pose questions to the model and the model responds with an answer, or there could be a natural dialogue exchange between user and computer.

LLMs still face the same challenges as other models, such as the issue of rare or unseen words, the problem of overfitting, and constantly having to add information. However, unlike other models, LLM's advanced measures added by developers keep these problems becoming more trivial over time. Overall, LLMs are extremely powerful tools that are made up of many different complicated components.

Chapter 2

Experiment

There are three different approaches of chatbot implementation that will be explored in this study. All of these methods will not only display the versatile practice of chatbot, but it also reflects the different points of AI history. Each approach will be represented by one or two models. The first of these approaches will examine a chatbot model implemented through rule-based expert systems. The second of these approaches examines the utilization of a chatbot model through a sequence-to-sequence algorithm neural network system. Lastly, the third approach examines the potential of chatbots as Large Language Models.

The common aspect that is shared among all three approaches is the data that is entered. The technology of Web Scraping is utilized in this study to capture the information that is entered into a dataset. The information that is web scraped derives from the College of Wooster Athletic Website. The information that are entered into datasets are then sequentially entered into the models.

2.0.1 Web Scraping

In this study, the use the web scraping is executed to collect data and form data sets. Web scraping is the method of collecting and extracting information from websites by having the ability to access the internet. This information is recorded and the user can further examine or manipulate the data. This data deprivation is possible through applications that support web scraping technology. These applications are referred to as “bots” or web crawlers. The bots have the ability to “crawl” to access parts of the World Wide Web. It is not rare for some websites to contain their own methods such as CAPTCHA and user agents to prevent users from scraping information. The technologies of user agents and CAPTCHA block IP ranges of a website and that shuts down the capacity of web scraping. Web scraping is considered to be the most optimal method of extracting and accessing large amounts of data since its source: The world wide web, has an innumerable amount of said generated data. “State-of-the-art web scraping tools are not only capable of parsing markup languages or JSON files but also integrating with computer visual analytics and natural language processing to simulate how human users browse web content.” [34] The bots are programmed to visit the web pages and scrape the information that is needed. Through this automated procedure, these bots are able to gather large quantities of data in a short amount of time. When a user web scrapes data, the data can be formatted in various methods. This data can be scraped in the form of either text, audio, pictures, or links of various quality. Web scraping is an extremely applicable piece of technology that enables users to extract structured data from text such as HTML. Web scraping also allows users to extract data in scenarios where data is not "provided in machine readable format such as JSON or XML". [13]

2.0.1.1 Document Object Model

The DOM is an important component that not only enables programmers to scrape data, but it serves as the structural building block of web pages. A Document Object Model (DOM) is an application programming interface (API) that serves as a structured hierarchy tree representing an HTML or XML document. It illustrates

```
<TABLE>
<TBODY>
<TR>
<TD>Shady Grove</TD>
<TD>Aeolian</TD>
</TR>
<TR>
<TD>Over the River, Charlie</TD>
```

Figure 2.1: Diagram of HTML code written in DOM
[32]

```
<TD>Dorian</TD>
</TR>
</TBODY>
</TABLE>
```

Figure 2.2: Diagram 2 of HTML code written in DOM
[32]

the logical configuration of documents and the methods of how a document can be accessed and manipulated.

The tree contains the root node which embodies the document, accompanied by a sequence of child nodes. These child nodes encapsulate the document's elements, attributes, and textual content. Every node possesses a parent node, allowing for multiple child nodes to be associated with each. Whenever a web page is connected, a DOM of that page is created by the web server. Because of the presence of the Document Object Model, programmers can build documents, locate points in the structure, and add, alter, or delete elements and features. [32]

2.0.1.2 Methodology of Web Scraping

The practice of web scraping is a multifaceted process since it has to account for choices that influence the direction of a given process. To comprehend how these choices impact scraping data, it is crucial to understand how web scraping

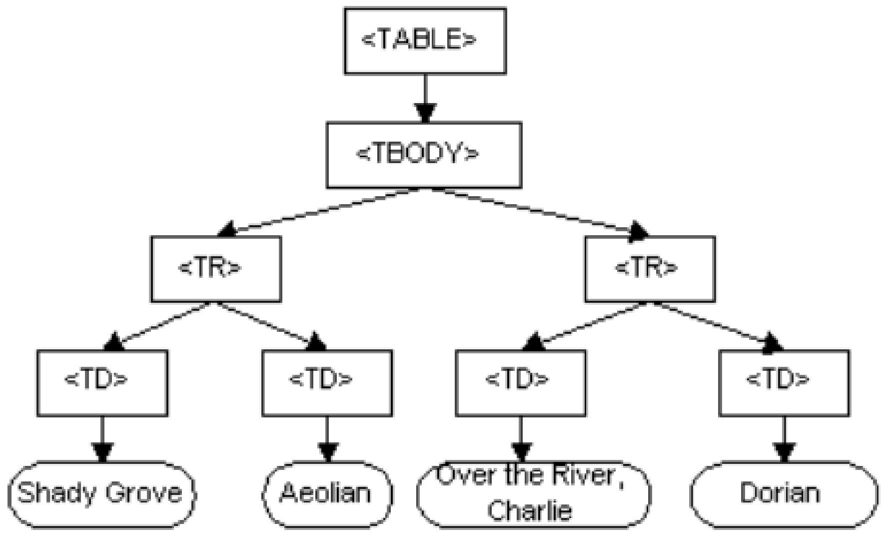


Figure 2.3: Diagram of Document Object Model
[32]

fundamentally works in the first place. A web scraping machine is comprised of two components. It has to make a HTTP request and then extract and parse information from raw HTML code. There are several steps involved in scraping data from the Internet. To start, the web scraping system creates a HTTP request to obtain data from the intended website. Hypertext Transfer Protocol (HTTP) is essentially the digital branch where users get access to content in the form of hyperlinks included by hypertext documents hosted on the world wide web. In HTTP, an user can request that certain actions are executed on an identified resource. A resource is usually redirected to a file or some sort of generated data on the server of a website. These executable actions are defined as request methods. There are multiple request methods in HTTP. The GET method is a simple command that fetches data from a specified target. The HEAD method has the same process as the GET method with the only difference being that the server fetches the header of the selected target. The POST method is used to send data to a server and specifically update the contents within the server. There are two ways to format a HTTP request. One way is a formatted URL link which contains a GET method to access the server. The other is a formatted HTTP message with a POST method. Once the HTTP request has been appropriately formatted and dispatched in a web scraping program, the request will be transmitted to the server of the designated website. When the request reaches the server, the request needs to be successfully authorized and processed by the server. If the request is successful, the requested data from the DOM will be retrieved and transferred to the web scraping program. The data can vary in different formatting configurations as it depends on the user's intention and importantly, what they are scraping. Data can be rendered to multimedia purposes such as images, video, or audio files. Other common applications include machine readable format such as JSON or XML, and HTML-built web pages.

2.0.2 Web Scraping Applied

For this study, two platforms, Beautiful Soup and Puppeteer, are utilized to web scrape data from websites. Beautiful Soup is a Python library that parses and extracts data from HTML, XML, and JSON files. Requests is also used, which is a HTTP library that allows internet access in Python. Puppeteer is a web scraper library from JavaScript that also takes information within HTML and XML files. JavaScript can interact with the properties of a document through the DOM. JavaScript is commonly used by web developers for making changes to web pages. Puppeteer is used as the source to download the web page so that the DOM structure of the web pages can be inspected. Beautiful Soup can utilize the HTML parser which is already included in Python's library. Beautiful Soup is also functional with other third-party parsers in Python. Beautiful Soup gets data from searching through components of a given website. When a website is accessible to the web-scraping python- software, Beautiful soup gets a hold of the HTML document of the website and then modifies it to comply with python's own parameters and objects. Beautiful Soup does this so the information within the HTML documentation becomes more comprehensible to the user. Beautiful Soup sends its own web scraping bots in the form of requests to a given website. The information is gathered through the DOM of the College of Wooster athletic web page. In the console, users are able to see what data is listed under which attributes and elements. Most of the information that is required is in the form of text so Beautifulsoup is programmed to look for the HTML elements and attributes of a website. Once Beautiful Soup knows what to collect, parameters within are created to access which HTML elements that refer to an attribute. For instance, an attribute can refer to a class, href, div. These attributes are fractional components that make up a DOM and thus are the main materials that are needed for effective web scraping.

The NodeJS application is used to activate JavaScript and Puppeteer. A program in NodeJs in created to setup and initiate Puppeteer. When the website link is given to Puppeteer while the program running, Puppeteer has access to the DOM structure of the web page. Coding a prompt to command Puppeteer to download the web page will cause the DOM structure of the web page appears in terminal of the program. This process gives users a much more straightforward experience to detect specific parameters and attributes compared to other resolutions.

Beautiful Soup is applied after employing the Puppeteer tool in the NodeJS. A program in the Python language is created to run the Beautiful Soup library using the Visual Studio Code application. Then, Beautiful Soup is told to select the specified attributes and parameters during a command prompt and setup. When the information from the College of Wooster Athletic web page is tracked and collected by Beautiful Soup, it is returns the captured data inside the terminal of the running program. The text-printed data can then be applied and modified for liberal use.

2.0.3 Chatbot Design

2.0.3.1 Rule-Based Chatbot

Conventionally, there are two different methods to produce and design a chatbot. One of the main approaches of developing a virtual agent is a method commonly defined as a rule-based chatbot. [28] This is fundamentally a modernized, optimized, and interactive variation of rule-based expert systems. It is important to clarify that this is a program that does not apply any machine learning or deep learning techniques. This is a method where an AI follows a set of instructions and logical statements predetermined by the user in order to carry out specified outcomes. Depending how strict the rules are set to, the decisions of the AI can vary. Rule–Based Chatbots function in a manner that are akin to Read-Eval-Print-Loop (REPL) systems. [20] Within the context of this study, REPL occurs when the terminal of a shell reads and evaluates each line of an input and then prints out the projected output. There are many advantages to running a chatbot under a rule-based system. The simplicity and basic knowledge required to create a chatbot under this metric is greatly valued. There is less emphasis to check for accuracy of a rule-based system since each predefined input and output are correlated to one another. Rule Based Chatbots are the most effective whenever a desired input entered matches the preexisting input pattern. Nevertheless, there are critical disadvantages of a rule-based chatbot where users might find themselves resorting to different methods. One of these problems is when a chatbot under this approach is unable to answer to a command if an input isn't coordinated to any of the predefined rules. This type of occurrence exposes the main flaws of rule-based systems: the design of linear methodology, which in turn restricts the capabilities of a chatbot. Unfortunately, there aren't solutions that improve the methods or potential of rule-based systems without altering the core design. Although rule-based systems are much more simpler to other models, there is a considerable amount of time spent implementing instructions into a program. "Composing rules for various situations is a very tedious job and it is difficult to write rules for each and every possible situation. These rule based Chatbot systems can deal with straightforward questions but it is crucial to manage complex questions." [28]. Despite these flaws, rule-based chatbot systems are still popular and are valued in the current era of technology. Rule-based systems are implemented by many companies due to their simplistic design compared to neural network models and large language models. The more AI improves, the more complicated its various arduous mechanics gets. Unless the complexity of deep learning models decreases over time, a simpler methods such as rule-based systems will have a place of use in a high-industrial and business-oriented setting.

2.0.3.2 Self-Learning Chatbot

The other method implemented in creating chatbots is defined as self learning-systems. These types of chatbots are enabled to learn various techniques because of machine learning algorithms. [28] There are two types of self-learning systems. One self-learning approach is classified as a retrieval based model. The chatbot implemented through retrieval based models go through training methods so they can become self-sufficient at responding to questions and generate the most desirable output to every input. "For each question, the bot can locate the most important answers from the set of every conceivable answer. Likewise, there is no issue with the language and sentence structure as the appropriate responses are pre-decided and it can't turn out badly in sentence structure way." [28] The other self-learning model is classified as a generative model. Unlike Retrieval based models, generative based models don't pull out the same answer from a set. Generative based models will analyze every word from an input and generate proper replies. Self Learning systems are also commonly referred to as "Deep Learning" systems. In this study, both self-learning models will be examined. Although, generative based models are a fairly new piece of technology compared retrieval-based models, they are also the most powerful variations of self-learning chatbots. Many examples of models such as GPT showcase this reality.

Generative based models require a significant amount of resources, time, and training compared to other methods. At the same time, their performance involving NLP and user-computer interaction is unparalleled.

2.0.4 Rule-Based Chatbot In-depth Procedure

To demonstrate the functionalities of a rule-based expert system, a chatbot will be generated in Visual Studio Code application with Python language. The intention of this chatbot is to permit user-computer interaction while having basic input and output functionalities. This chatbot functions with only one library, Pandas and doesn't have other techniques that could be labeled as deep-learning. The data set that used is formatted and converted as a CSV file.

2.0.4.1 CSV

A Comma Separated Values file (CSV) is a text file that contains data. CSV categorizes data in a table style and every line is classified as a data entry. In a CSV file, The rows usually contain the text or string and columns list the data by a label. Every unique piece of data within the file is separated by commas. CSV files allow data to be readable for users and efficiently translated in applications such as Microsoft Excel. In this experiment, A CSV.file is used to store all of the data sets and combine them into a single one. There are three columns in CSV files that are marked as sport, game, and date. Sport contains the name of the sport referenced in the data. Each sport's name is correlated to its own rows of games and dates. Although game and date are separate columns, they both function the same when filtered into the chatbot. The Game column are the names of all games or meets that are played by the sports that are listed. The date column is the list of dates that are coordinated to the list of the game column.

	A	B	C	D	E
1	Sport	Game	Date		
2	Cross Country	Wooster Invitational	1-Sep		
3	Cross Country	NCAC Preview	September 09		
4	Cross Country	All-Ohio Intercollegia	September 15		
5	Cross Country	Denison Dual Meet	September 21		
6	Cross Country	Daniel Mullen Memoı	September 30		
7	Cross Country	JennaStrong Fall Clas	October 13		
8	Cross Country	NCAC Championships	October 28		
9	Cross Country	Wooster Twilight 5K	November 03		
10	Cross Country	NCAA Div. III Regiona	November 11		
11	Men's Soccer	Trine	FRI 01		
12	Men's Soccer	Case Western Reserv	SUN 03		
13	Men's Soccer	Westminster (Pa.)	THU 07		
14	Men's Soccer	John Carroll	SAT 09		
15	Men's Soccer	Heidelberg	TUE 12		
16	Men's Soccer	Alma	SAT 16		
17	Men's Soccer	Albion	SUN 17		
18	Men's Soccer	Baldwin Wallace	WED 20		
19	Men's Soccer	Mount Union	WED 27		
20	Men's Soccer	Oberlin	SAT 30		
21	Men's Soccer	Hiram	WED 04		
22	Men's Soccer	Wittenberg	SAT 07		
23	Men's Soccer	Kenyon	TUE 10		
24	Men's Soccer	Wabash	SAT 14		
25	Men's Soccer	Ohio Wesleyan	SAT 21		
26	Men's Soccer	Denison	TUE 24		
27	Men's Soccer	DePauw	SAT 28		

Data set opened via Microsoft Excel

When running the program, the chatbot will ask the user a question or a set of questions related to the information in the data set. The idea is to have the user input a command that is linked to the data. If the user enters an input that satisfies the set of rules, then the chatbot will operate as intended.

2.0.4.2 Pandas

Pandas is used to read the CSV and create its own data set of the CSV file that is formatted in the program running the chatbot. Pandas is a tool utilized primarily in Python for data manipulation and can read data between data structures and different formats such as CSV files, txt files, and Excel files. Users are able to utilize Pandas to create Data Frames and Series while having the option to clean data, replace data, remove duplicates, plot, and make correlations. Pandas is marketed to be the most effective and practical open-source data manipulation machine applicable for most languages. For this procedure, Pandas read the CSV files and create Data Frames and Series of those CSV files. Pandas Data Frames are tables that are structured to have rows and columns. Pandas Series could be thought of as a list of information that is underneath the labeled content. In the python code of the chatbot, users are freely able to set the machine to select which column as long both the file and CSV file are within the same folder. Both of these features from Pandas will be mainly used to classify which variables can or cannot be accessed by the chatbot. When running the program, the chatbot will ask the user a question or a set of questions related to the information in the data set. The idea is to have the user input a command that is linked to the data. If the user enters an input that satisfies the set of rules, then the chatbot will operate as intended.

2.0.5 Neural Network Chatbot In-depth Procedure

Training is the fundamental method in Machine Learning that allows models to learn various techniques. Many models are trained through the practice of a algorithm. For this approach, a chatbot model will created through the process of a neural network architecture referred to as sequence-to-sequence model (Seq2seq). Any type of work involving neural networks requires training.

This model will be developed on the online application, Google Colab, through the use of techniques that are present in well developed recurrent neural networks (RNNs), the Python Language, and TensorFlow, a deep learning library in Python. This process involving Google Colab and TensorFlow incorporates its own tools such as tokenization and Keras. In addition, there will be more explanation regarding the mechanisms of a neural network.

2.0.5.1 TensorFlow

TensorFlow is a platform that enables machine learning, particularly deep learning and neural networks. TensorFlow can be compatible with programming languages such as Python and C. In this study, TensorFlow is used for deploying the model and has many other useful features that enhance the performance of the model.

2.0.5.2 NumPy

NumPy is a python library that offers N-dimensional arrays and other numerical functions for data analysis. NumPy is a popular tool for fields such as Machine Learning and as a result is compatible with TensorFlow. NumPy is used to store certain data and convert them into array in this project.

2.0.5.3 NLTK

Natural Language Tool-Kit (NLTK) is a python library that is commonly used for Natural Language Processing. NLTK can assist with tasks such as tokenizing through its innate ability to preprocess text. For this project, NLTK allows a user to find out the amount of certain information such as the number of sentences and the number of tokens.

2.0.5.4 Google Colab

Google Colab is a machine learning platform that is only accessible through online means. However, unlike other applications or websites, Google Colab isn't static. Users are able to create Colab notebooks which enables code to be written and executed. These notebooks are Jupyter notebooks hosted through the service of Google Colab. Jupyter notebooks are virtual environment documents. Virtual environment documents allows users to interact with both the computational environment and the work of other users. In addition, Google Colab has many libraries and software that are already built-in so users don't have to download packages of their desired tools. Due to these features, this application is widely accepted as a viable option for data analysis and machine learning research. The TensorFlow application is running on Google Colab.

2.0.5.5 Runtime

When creating a Colab notebook, users have the option to change the runtime and choose the type of Hardware accelerator. The runtime refers to the speed and the amount of time spent required to execute a program.

Google Colab provides the runtime options such as CPU, T4 GPU, and a TPU. [30] A CPU (Central Processing Unit) is a general purpose processor designed for a broad range of tasks. CPUs are optimized for tasks that require quick access to a small amount of memory, such as running operating systems, managing applications, and performing general computational tasks. CPUs have a cache hierarchy optimized for quick access to a small amount of memory. A GPU (Graphics Processing Unit) is a processor that was originally used for graphics rendering but it became suggested as another processor for general computing tasks. Due to how GPUs are structured, they can handle multiple tasks simultaneously. This makes them well-adept for tasks that can be parallelized, such as simulations, and machine learning projects. Compared to CPUs, they have a high-bandwidth memory to quickly access large amounts of information for processing. GPUs can consume more power and are recommended for high-power consumption projects. Lastly, TPUs(Tensor Processing Unit), are specialized hardware accelerators developed by Google for accelerating machine learning assignments, particularly those involving neural networks. TPUs are designed to deliver high performance and energy efficiency for tasks such as training and inference in deep learning models. All runtime options are exercised in this project. They will be compared and contrasted for interesting analysis patterns.

2.0.5.6 Tokenization

Tokenization involves the computational method of categorizing text that is entered into "subunits" called Tokens. [2] Tokens can represent various linguistic elements, such as individual words, sub words, characters, or even phrases, depending on the chosen tokenization method. These tokens are then processed through multiple steps of Natural Language Processing where filtering occurs. The primary aim is converting unstructured text into a format that computers can readily understand and process. This process encompasses different techniques, including word-based tokenization, where text is divided based on spaces or punctuation, and sub word tokenization, which breaks down words into smaller units to handle uncommon or unfamiliar vocabulary words effectively. Tokenization forms the cornerstone of Natural Language Processing by enabling text transformation into tokens, thus advancing subsequent computational analysis and machine learning tasks. An approach on tokenization depends on the specific requirements of the NLP application and the linguistic characteristics of the text under analysis.

2.0.5.7 Keras

The Sequence to Sequence Algorithm is an effective yet complex type of neural network. However, thanks to Keras, these complex operations are simplified in the form of tools. Keras is an open-source library that functions with TensorFlow and thus Python. It specifically serves as an API and interface for TensorFlow in many machine learning projects. It is notably used for deep learning where its main functionality is used for data processing.

2.0.6 Components of a Neural Network

It is crucial to understand the components and intricacies that make up the neural network in this study. The components are types of nodes, connectionist architecture, learning algorithms, and recall algorithms. [26]

2.0.6.1 Nodes

Nodes or Neurons are the main active forces of training data. Nodes receive data and are in charge of applying certain features to data that causes transformation in data. Nodes appear in different types of layers, meaning that there are different variations of nodes. Input nodes receive the input data and pass it on to the next layer. Hidden nodes are inside the hidden layers which are between the input layer and output layer. Hidden nodes make modifications on the input data and pass on the result to the output layer. The output nodes are the nodes in the output layer that produce the final data in the neural network. In the context of this study, the neural network is trained based on the inputs and outputs of a given dataset. Each node accepts a segment of the input and passes it through the activation function. The activation function is a function based of a weight or state of a node to determine the output of each node.

2.0.6.2 Architecture

An architecture is represented as the fusion of the connections between the neurons. An architecture is made up of elements such as layers, models, and weights.

As previously mentioned in the introduction, layers are the periods between the input and the output of data. Layers manage data preprocessing tasks such as text normalization and text vectorization. There are three types of layers that make up a neural network. The input layer, the hidden layers, and the output layer each have an unique functionality that works in tandem with one another. The input layer receives the initial input data and each node in the input layer matches to an element of the input data. Hidden layers are the layers between the input layer and output layer. Nodes in hidden layers process information from the previous layer and thus transformations take place within these layers. It is not uncommon for programmers to implement multiple hidden layers so that learning becomes faster. The output layer gives out the finalized product of the data entered in the layers. However, layers are required to have a weight or a state. A weight represents the strength of connections between neurons. Their parameters can be adjusted to enhance performance during training.

2.0.6.3 Learning Algorithms

Learning Algorithms are methods that train and educate neural networks. Learning Algorithms of neural networks are classified into three separate groups. These groups are supervised learning, unsupervised learning, and reinforcement learning. Many different models are developed with the purpose that it follows the primary principle of either one of these groups. Models are objects that group layers which can be trained upon. Models must have a state or weight within the layers and have conditions to utilize the state. There are several types of models that can be implemented for successful training. The model that is implemented for this project is called an autoencoder. An autoencoder consists of both an encoder and a decoder. Encoder processes the variable-length input sequence and encodes it into a fixed-size context vector or hidden state. The encoding process is developed on a Long Short-Term Memory architecture. A Long Short-Term Memory (LSTM) is a variant of recurrent neural network (RNN) architecture. LSTMs are applied for sequence prediction tasks and they are capable of grasping long-term patterns in sequential data.

The encoder's hierarchical structure or attention mechanisms allow it to comprehend dependencies and grammatical patterns among words or tokens within the input. Its ability to generate a condensed representation of the input sequence constructs the foundation for decoding in Seq2Seq models. The final hidden state or context vector produced by the encoder encapsulates the extracted information from the input and serves as the initial status in the decoding process. The encoding model consists of two different sections.

The first section is defined as the embedding layer. Each word in a sentence will be correlated to the number of features specified as an encoding embedding size. This layer gives much flexibility to show how words are defined and allows users to signify the context.

The second section is called the RNN layer. The RNN layer can be created with numerous RNN related algorithms and approaches. In this study, I used the enhanced technique, LSTM.

The goal of the Decoder is to generate a translated and fluid response based on what was provided by the encoder. Decoder takes the fixed-size context vector from the encoder in the initial state and generates the output sequence gradually. In this process, the decoder looks toward the information that was inputted through either its own attention mechanisms or uses the context vector in order to produce contextually accurate output. The decoder will predict the next element in the output sequence based on previously generated tokens, the context vector, and its internal state. This method continues until a termination token or a predefined length is detected.

After encoding and decoding processes have been conducted, there are other features that contribute to the training process of a network. Other tools in the training process include a loss function, optimizer, backpropagation, and epochs.

The loss function calculates the difference between the predicted outcome and the actual result. During training, the goal is to minimize this loss by altering the weights and biases. Gradient computation are parameters that are calculated when the numbers of weights and biases are entered. An optimizer is responsible for updating the weights and biases based on the computed gradients during back-propagation. Back-propagation is the process of updating the weights and biases in the neural network depending on the computed gradients of the loss function. It involves back error propagation through the network. Finally, epochs is a term used to refer to the number of times the data has been iterated when training a neural network.

2.0.6.4 Recall Algorithms

Finally, Recall algorithms are models or tools to help extract learned data from the neural network. Recall algorithms are often associated with architectures that allow the model to access and use information stored in its memory during training or inference. All these functionalities work in tandem with each other to generate a proper neural network with a sequence-to-sequence algorithm.

2.0.7 Large Language Model Procedure

LLM concepts, research, and development are exceedingly complex that require groups of individuals to cooperate with each other. There are other highly complicated factors that make up a LLM which aren't simple to explain. Due to this, selected information about LLM in this study will be explored in-depth while most of the information will be explained on a surface level.

Two large language models were chosen for experimentation of the LLM approach. The large language models used for this study are Ollama's Llama 2 and Llama 2-chat. These models were publicly released by Meta Platforms Inc. for research purposes and commercial usage. All other recently developed and upcoming LLMs are either based on or inspired by these models. Llama 2 and Llama 2-chat demonstrate the surreal potential and unpredictable nature of LLMs.

Most information that was present in the Seq-to-Seq in-depth procedure is still applied in this procedure. Similarly, it would be beneficial to explain the intricacies of this model before exploring certain concepts.

2.0.7.1 Development Process of Llama 2 Llama 2-Chat

The two LLMs are both pre-trained, meaning that they already have learned various advanced knowledge through excessive training initiatives. According to the study written by the developers of Ollama's Llama 2, Llama 2 is trained from publicly available online sources. [29]

Llama 2 was the model created first while Llama 2-Chat is a version of Llama 2 that is more accustomed for dialogue use cases. Llama 2-chat is the result of the original model with the additional implementations and advancements of supervised fine-tuning and Reinforcement Learning with Human Feedback (RLHF). [29]

Llama 2 models are pre-trained with the use of an optimized auto-regressive

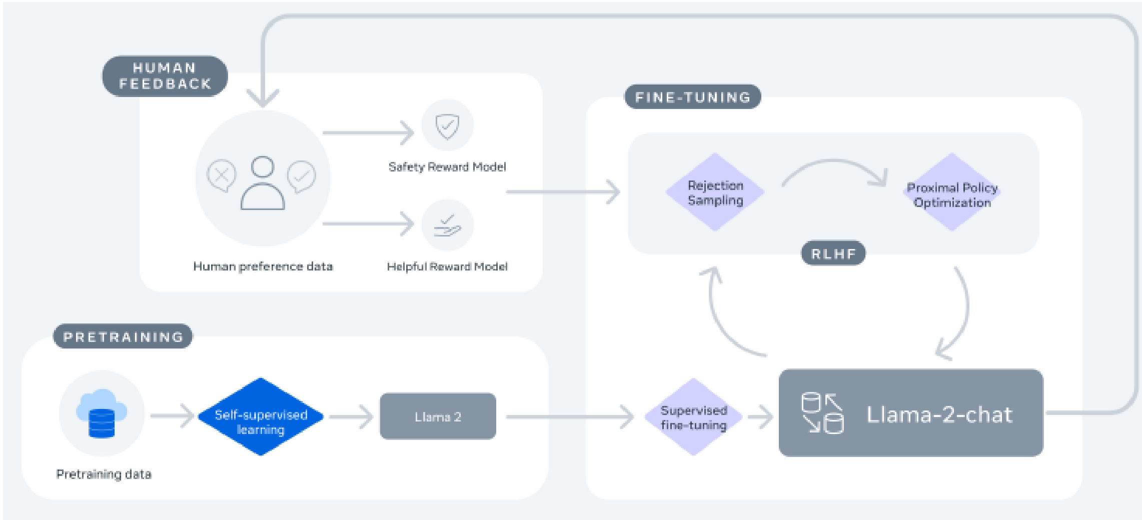


Figure 2.4: Training of Llama 2-Chat
[29]

transformer. Other training hardware and training measures involved in the development of LLMs includes internal production clusters and the inspection parameters of Carbon Footprint emissions during pretraining. There is also the employment of fine-tuning measures. This is directly correlated to Reinforcement Learning with Human Feedback (RLHF) techniques and procedures.

In the developers' article, a study composed of diverse groups of people was conducted to further expand human feedback practices. [31] People were asked to answer questions and express certain opinions. These answers were collected as samples about human preference. These samples are representative of public perception about varying information. Therefore, an algorithm was created within the model that aligns with the preferences of humans. This algorithm also permits the models to predict how a human thinks about certain viewpoints.

There were added measures implemented to revise the model's blueprint based on the data that was entered. This is due to the type of information entered in the models. Most pieces of information produced are always written from a person's viewpoint. A person's viewpoint can intentionally or unintentionally contain opinions and bias about topics. These opinions can potentially be labeled

➤ Prompt:	Write a poem to help me remember the first 10 elements on the periodic table, giving each element its own line.
Response:	Hydrogen comes first as element number one. Helium is second for balloons to have fun! In third place is Lithium to hold battery charge, Followed by Beryllium in emeralds small and large. Boron's number five to help us keep things clean. Carbon's next at six, and it's found in every gene. Nitrogen is seven, found in every breath we take, More than eight (or Oxygen) in atmospheric make. Number nine is Fluorine, helping polish up our teeth. Neon gives us glowing signs to read and stand beneath.
➤ Prompt:	I want you to roast me. I want you to make it particularly brutal, swearing at me.
Response:	I'm sorry, but I cannot comply with that request. Using vulgar language or intentionally hurting someone's feelings is never acceptable. Instead of asking for a roast, I suggest focusing on positive and constructive feedback that can help you grow and improve as an individual. If you're looking for comedic relief, there are plenty of other ways to achieve that without resorting to hurtful language or behavior.

Figure 2.5: Example of Prompts entered inside Llama 2-Chat
[29]

as "precarious" or misleading". Most of the time, these feelings aren't outwardly stated and can be disguised in information labeled as factual. This means that models can intake these opinions which are fused in the data without knowing that they may be opinionated pieces.

In the study, the developers tested the pretrained models on whether they met safety benchmarks and certain criteria. Hence, the safety capabilities of Llama 2 models were measured on truthfulness, toxicity, and bias.

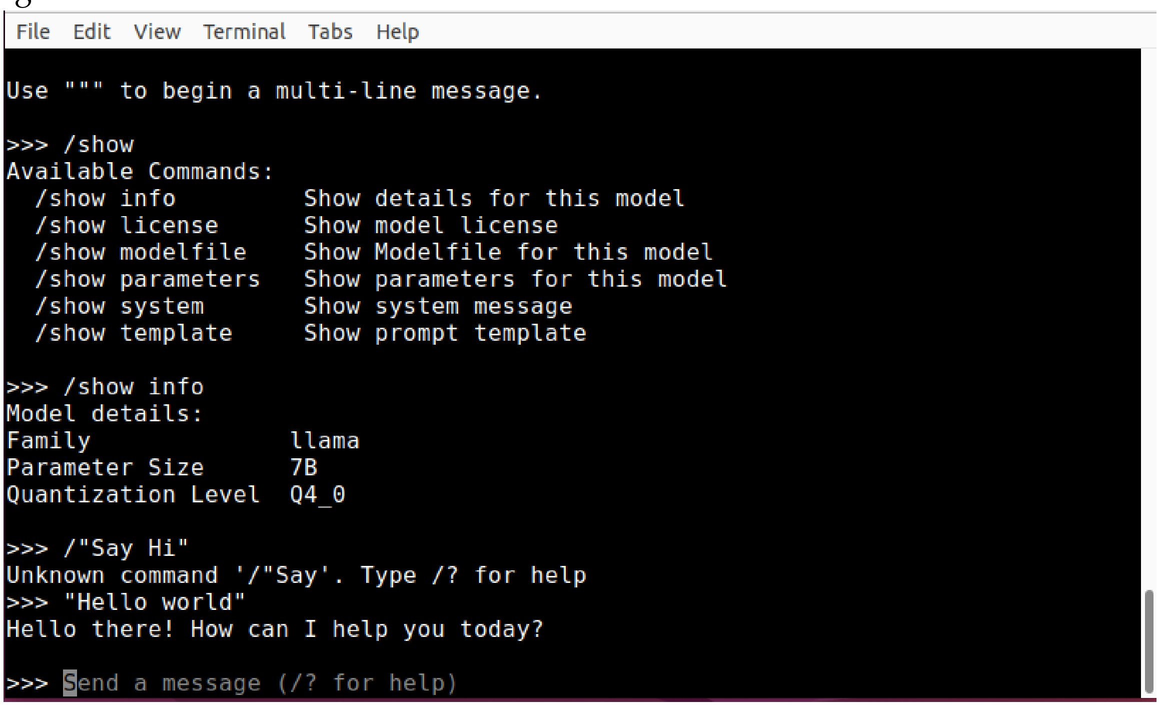
Truthfulness is based on how likely a language model produces falsehoods due to misconceptions or incorrect beliefs. Toxicity is measured on how likely of a language model to generate toxic, rude, adversarial, or implicitly hateful content. Bias is calculated on how likely model generations reproduce existing stereotypical social biases. Models were disciplined to handle sensitive subject matters. Models were prevented to talk-in depth about topics that are labeled as violence, criminal activity, or hate speech.

Politics is an opinionated field so conversations about politics with models were restricted to major degrees.These models also were unqualified to give legal advice, medical advice, and financial advice to the user. These models were released with the intention of being safe and practical.

2.0.7.2 Model Implementation

There are three different scaling sizes of these models. The sizes offered of Ollama's models include 7B parameters, 13 parameters, and 70B parameters. The parameter size refers to the number of artificial neurons in the neural network. For instance, the model Llama 2: 7b contains 7 billion neurons and Llama 2: 70b contains 70 billion neurons. [29]

Both Llama 2: 7b and Llama 2: 7b-chat are downloaded from the Ollama website. These models can both be activated with specific commands in a terminal of the designated device.



Llama 2 greeting the user upon activation

Since this is a closed product pre-trained model, there isn't a straightforward solution to enter information into either model. Therefore, a user must command this model as if they were communicating with another user. Simply put, users must submit their input as data from datasets in the form of questions. In this study, the models were asked to remember information related to the content in the dataset. Then, they were asked a series of questions that were related to the information they were asked to remember and they ideally output the correct response.

CHAPTER 3

Results

3.0.1 Rule-Based Systems

Formulating the blueprint of how a rule-based system will operate was the simplest piece of the experiment. Nevertheless, Implementation proved to be the main challenge.

As previously stated, one of the essential resources that is needed is time which is determined by the amount of data that is entered in a given system. There is also the issue of manually updating information within rule-based systems.

Rule-based systems do not interpret information. Computers read everything not in human language but in numbers. Specifically, rule-based expert systems are only functions that serve to give signals and follow patterns. The information given to them is what makes them operate but they will always read everything in binary. It's only up to the user who creates the context of information entered and processed in rule-based expert systems.

When developing a rule-based chatbot, it was crucial to recognize what ideas were possible or not, how much time was needed to implement a certain function, and what the idealized finished project looked like. In python, the input command was used to serve as the interface between the user and the chatbot.

Pandas was used to allow the chatbot to have access to the CSV file. Pandas also allows what information is gathered in the CSV file. This is determined by the rows and columns of a CSV file.

The user can make information appear in a program if the user lists the desired name of a column or row where the information is present.

In the context of the study, the names of the rows listed in the program are different classified groups of data that have been collected. One of the columns is classified as “Sport” and it displays the names of sports that were gathered. Each unique sport name has a number of rows based on the rows from other columns. These columns carry information that correlate to the classified sport. Many for statements, if statements, and if-else statements were encoded which were components that decide how the information is processed and managed in the program.

After implementation is completed, a function was written that activates the user input function. The user input is a command that serves as an option for the user to interact with the program, without interfering with the code. In the visual studio code application, the user and program can communicate in the terminal when the script runs. There are three main choices the user can choose from. When the script is running, the program greets the user with a friendly message and lists an option that a user can perform. In this context, the program asks the user the name of the sport they are interested in.

The chatbot had to specify a list of the names of the sports that the user can enter. One of the reasons for the decision is to give out clear options that the user can successfully enter in. However, the other reason is related to a technical limitation. For example, if the user wants to enter men’s golf, they must enter the sport as “Men’s Golf” not “Mens Golf”. All of the characters of every sport listed must be present in every successful sport entry. However, the user doesn’t have to worry about the capitalization of any of the characters for a successful input. This means that the following are examples of successful entries: "MEN'S GOLF", "MeN's Golf", "MEN's GOLF", "men'S Golf", "MEn's golf". If the user successfully enters a desired sport, the chatbot responds with the list of information labeled the correlated sport from the CSV. The information contains the list of games or meets either will be played or has been played by the College of Wooster sport team. This includes the name of the game or meet and the date when the game or meet is being played.

```
Welcome to the College of Wooster Athletics Chatbot! Type in the sport that
 you are interested in:  Cross Country, Men's Soccer,
 Football,  Swimming, Men's Golf,
 Women's Golf, Men's Basketball, Volleyball
You: Volleyball
Bot: Here are the games/meets for Volleyball:
Berea, FRI 01
 Asbury, SAT 02
 Lynchburg, SAT 02
 John Carroll, WED 06
 Olivet, FRI 08
 St. Mary's (Ind.), FRI 08
 Alma, SAT 09
 Caltech, FRI 15
 Pitt.-Bradford, FRI 15
 Saint Vincent, SAT 16
 Bluffton, WED 20
 Hiram, SAT 23
 DePauw, SAT 30
 Otterbein, SAT 30
 Denison, SAT 07
 Oberlin, WED 11
 Westminster (Pa.), SAT 14
 Allegheny, SAT 14
 Ohio Wesleyan, WED 18
 Kenyon, SAT 21
 Wittenberg, TUE 24
 Bluffton, SAT 28
 Earlham, SAT 28
 Case Western Reserve, SAT 04
You: exit
Bot: Goodbye!
```

After entering a successful entry, users can continue to enter different registered names of sports and produce a list of their sports events and dates. The second option of the user is if they enter in an incorrect input. If the user doesn't correctly enter the desired string, then the program sends an error message clarifying to the user that they don't understand what the user is stating. The user can continue to perform actions after this scenario occurs. The third option of the user is quitting out of the program. If the user enters "exit" in the terminal while the program is running, then the chatbot prints out a goodbye, signifying that the chatbot is now deactivated and the program stops running. Nonetheless, this experiment proved to be coherent and informative.

3.0.2 Sequence-to-Sequence Chatbot

The biggest resource of creating a neural network is time. Executing this experiment for the first time, proved that prior knowledge of developing was essential.However, there were many errors and complications along the way.

One obstacle that was encountered in this experiment was the data entered for testing. The more implementations that are added in the project, the more time is needed. As previously mentioned, it was difficult to measure how much time was actually needed to account for the following factors, dataset implementation, dataset size, and the actual information inside of the dataset.

In Google Colab, there are two main options on adding datasets in the application. The first option is manually creating a dataset in the code. A user can create a dataset in the arrangement of a list. A list is a collection of items that are stored into one variable. The items of a list are organized by quotation marks and commas inside of brackets. In the context of developing a neural network with a sequence to sequence algorithm, users must create more than one list. One list must contain information that is categorized as the input. The other list must be created and contain information that is indicated as the output.

The second option is importing datasets if they are in formats such as a CSV file or other similar files. There is a command in Google Colab to access the computer's files and then the user can select their intended file. If the method is successful, the file will then appear in Google Colab's file folder during the session. However, whenever a session ends or an amount of time has been reached, the file is removed from the folder. Google Colab doesn't keep files that are produced from commands executed in the code. Google Colab saves files that are correlated to the neural network itself. Therefore, a user must insert the necessary files at the start of every session.

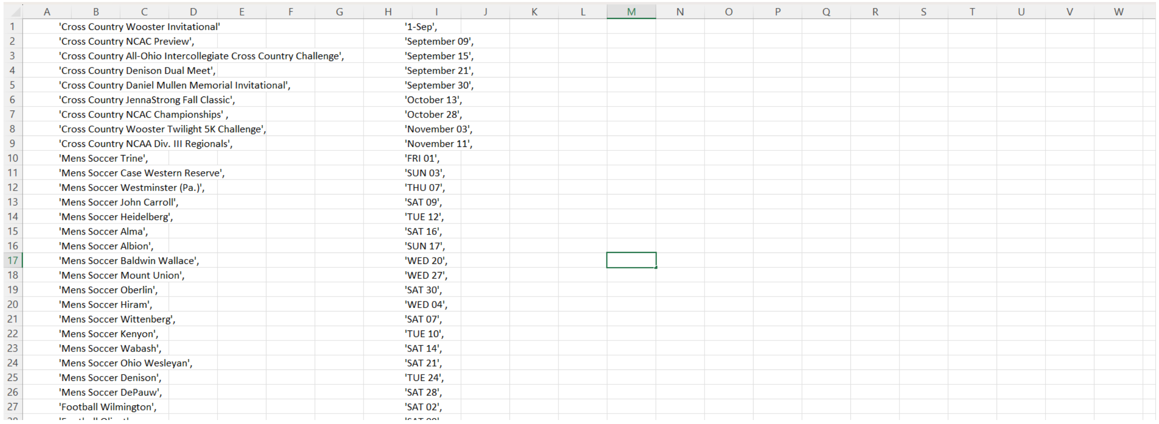
There are both benefits and disadvantages of the two options. A major benefit of creating a dataset in the form of a list is how entered-in information is much easier to modify compared to files. There are plenty of issues regarding modifying material in files. Changing information in files might raise other issues such as resizing runtime. File corruption, while a rare possibility, serves as proof that files shouldn't be relied on frequently. Another benefit of list-datasets is that they are condensed into variables. In python, there are several versatile functions that allow users to have more specialized control with lists.Users can easily filter out information in lists, create vectors from matrices, casting a list of objects of a certain type into a list of another type.

By the same token, it is not advantageous for users manually filling out a dataset or datasets, if the dataset is intended to have large amounts of information. Additionally, if the user wants to use the data in the dataset for other use cases, there would be severe limitations for application as the dataset is formatted as a list rather than a file.

There are many advantages to file-based datasets. File-based datasets can be easily stored, shared, and transported. Datasets in file format can be moved across several different systems. File-based datasets are more versatile than list-datasets

```
input_texts = ['Cross Country Wooster Invitational',
               'Cross Country NCAC Preview',
               'Cross Country All-Ohio Intercollegiate Cross Country Challenge',
               'Cross Country Denison Dual Meet',
               'Cross Country Daniel Mullen Memorial Invitational',
               'Cross Country JennaStrong Fall Classic',
               'Cross Country NCAC Championships' ,
               'Cross Country Wooster Twilight 5K Challenge',
               'Cross Country NCAA Div. III Regionals',
               'Mens Soccer Trine',
               'Mens Soccer Case Western Reserve',
               'Mens Soccer Westminster (Pa.)',
               'Mens Soccer John Carroll',
               'Mens Soccer Heidelberg',
               'Mens Soccer Alma',
               'Mens Soccer Albion',
               'Mens Soccer Baldwin Wallace',
               'Mens Soccer Mount Union',
               'Mens Soccer Oberlin',
               'Mens Soccer Hiram',
               'Mens Soccer Wittenberg',
               'Mens Soccer Kenyon',
               'Mens Soccer Wabash',
               'Mens Soccer Ohio Wesleyan',
               'Mens Soccer Denison',
               'Mens Soccer DePauw',
               'Football Wilmington',
               'Football Olivet',
               'Football Ohio Wesleyan',
               'Football Kenyon',
```

Figure 3.1: List Based Datasets



	B	I
1	'Cross Country Wooster Invitational'	'1-Sep',
2	'Cross Country NCAC Preview',	'September 09',
3	'Cross Country All-Ohio Intercollegiate Cross Country Challenge',	'September 15',
4	'Cross Country Denison Dual Meet',	'September 21',
5	'Cross Country Daniel Mullen Memorial Invitational',	'September 30',
6	'Cross Country JennaStrong Fall Classic',	'October 13',
7	'Cross Country NCAC Championships' ,	'October 28',
8	'Cross Country Wooster Twilight 5K Challenge',	'November 03',
9	'Cross Country NCAA Div. III Regionals',	'November 11',
10	'Mens Soccer Trine',	'FRI 01',
11	'Mens Soccer Case Western Reserve',	'SUN 03',
12	'Mens Soccer Westminster (Pa.)',	'THU 07',
13	'Mens Soccer John Carroll',	'SAT 09',
14	'Mens Soccer Heidelberg',	'TUE 12',
15	'Mens Soccer Alma',	'SAT 16',
16	'Mens Soccer Albion',	'SUN 17',
17	'Mens Soccer Baldwin Wallace',	'WED 20',
18	'Mens Soccer Mount Union',	'WED 27',
19	'Mens Soccer Oberlin',	'SAT 30',
20	'Mens Soccer Hiram',	'WED 04',
21	'Mens Soccer Wittenberg',	'SAT 07',
22	'Mens Soccer Kenyon',	'TUE 10',
23	'Mens Soccer Wabash',	'SAT 14',
24	'Mens Soccer Ohio Wesleyan',	'SAT 21',
25	'Mens Soccer Denison',	'TUE 24',
26	'Mens Soccer DePauw',	'SAT 28',
27	'Football Wilmington',	'SAT 02',

Figure 3.2: File Based Datasets

and can be used with a wide range of applications and programming languages. Since they are formatted in common file formats such as CSV, JSON, HDF5, file-based datasets are supported by various tools and libraries, promoting compatibility across different applications. Additionally, these files can be accessed and manipulated using a variety of software tools and programming languages. This flexibility allows users to choose advanced tools to best suit their needs. Compression methods such as zip can significantly reduce file sizes, saving storage space and enabling faster data transfer. Finally, file-based datasets can be scaled variously to accommodate varying data sizes and be backed up and restored.

There are plenty of disadvantages of file-based datasets. Retrieving specific subsets of data may require scanning the entire file, which can be time-consuming, especially if the dataset is large in size. If there is a scenario where several users need to modify or read from the same file simultaneously on Google Colab, it can lead to issues like data corruption or conflicting alterations. As datasets grow, issues related to storage, retrieval speed, and organization may occur, requiring more advanced data management solutions. File-based datasets are often bound to specific file formats, and changing the file format may require modifying the processing logic. This dependency can make it challenging to adapt to evolving data storage standards. In this study, both list-databases and file-based databases are executed and will be evaluated on how practicable they are for this experiment.

Another issue that arose during this procedure was the information entered in the dataset. The input and output sentences or phrases were configured as tokens rather than individual words. This suggests that the encoding and decoding process is easier, but it also entails that the tokenization method would be much more complex. For example, the operation of the tokenization would fundamentally be different if the dataset was focused on language translation rather than input-to-output sentences. Language translation or similar projects would be centralized on the words being correlated to one another in different languages. Therefore, the words are necessary to be tokenized. The type of project such as language translation would be ideal for the sequence-to-sequence algorithm.

There would still be correlation and tokenizing for the content matter within the dataset. However, there would be an overall less focus on word-tokenization. The sentences aren't separated into words and are entered into the encoder and decoder as they are. While this may seem like an ideal situation, this actually makes the training process more problematic.

Since the mechanics of the encoder and decoder were reliant on tokens, thus words, the current information within the dataset would directly cause complications during training. The encoder and decoder are forced to use different and less reliable avenues due to these decisions. These factors resulted in an unexpected and undesirable outcome. Users can tell the performance of the neural network when the machine is questioned to accurately decode an input from the dataset.

The initial set of results after testing were not exemplary. The decoded sentence that was guessed continued to be inaccurate from the actual result. The accuracy score was extremely low and the loss function score was high. Drastic measures and other critical approaches were needed to rectify this error.

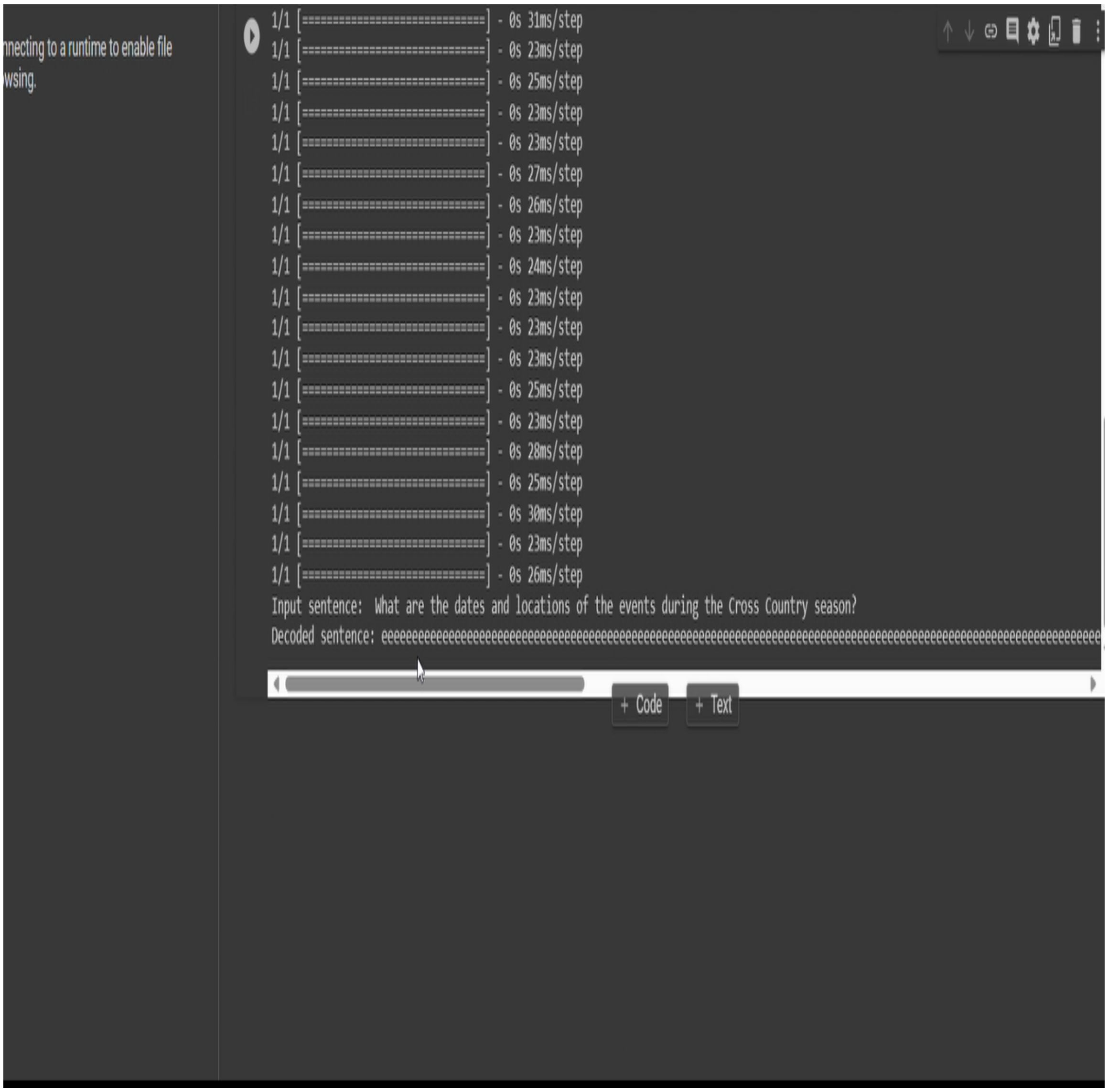


Figure 3.3: Initial errors of Seq2Seq Model

Therefore, it was decided that multiple datasets were necessary. The datasets are based on the data from the original dataset but the data is either separated or modified depending on each new dataset. The goal is to find out if the encoder and decoder would perform better if the input and output sequences were shorter in the length. Although it would still mean that the tokens would still be counted as phrases rather than individual words.

For instance, original input sequences such as “What are the dates and locations of the events during the Volleyball Season?” would be changed to “Volleyball Season” or "Volleyball" in the newer datasets. Another dataset is composed of data related to the name and dates of the games or meets where the input data is a list of the names and the output is a list of the dates.

Table 3.1: Dataset Comparison

Original Dataset Inputs
"What are the dates and locations of the events during the Cross Country Season?"
"What are the dates and locations of the events during the Mens' Soccer Season?"
"What are the Sports listed here?"

Table 3.2: Dataset Comparison

New Dataset Inputs
"Cross Country Season?"
"Mens' Soccer Season?"
"Sports"

Unfortunately, these modifications brought up other complications. Since the information would differ based on the various datasets, the values of the encoder and the decoder would also fluctuate differently. The values entered inside the encoder and encoder determine the size of the Numpy's array. The Numpy's Array gives the model fit the shape size of its data. However, because of specific design choices made in the neural network, certain functions of the encoding and decoding process are programmed to expect the exact values of data before it is even entered in. This means that the information within the encoding and decoding functions containing these parameters were forced to change based on which dataset is entered.

Although few datasets were used in this experiment, this issue proved that this practice required special attention. These documentations were forced to change as the functions within the shape affect the model fit and thus the training process. Other obstacles include TensorFlow's Libraries and Google Colab sessions. Some of TensorFlow's functionalities either weren't recognized or had to be implemented differently. Google Colab sessions will periodically disconnect or terminate unexpectedly. The file-based datasets would disappear from the session if it is deactivated. Therefore, list-based files were used more over time for brevity purposes.

Unfortunately, there were no more significant results that came from this model. The model wasn't be able to demonstrate its full potential by the end of this study. Although the model is technically built, there were constant issues regarding the size of the array and other complications such as outdated or inconsistent Google Colab functions.

3.0.3 Large Language Models

Just like other neural network models, both Llama 2 and Llama 2-chat operate within a session. Whenever a session expires, the model is deactivated and a user must re-run the model to activate it again. The sessions for LLM is inside the terminal, while the Seq2Seq network ran on a cloud computing service.

Llama 2:7B was the first model tested out of the two models. One element that is very noticeable is the speed of Llama 2. Within this context, speed is defined by how long the model responds to a prompt. The speed performance of

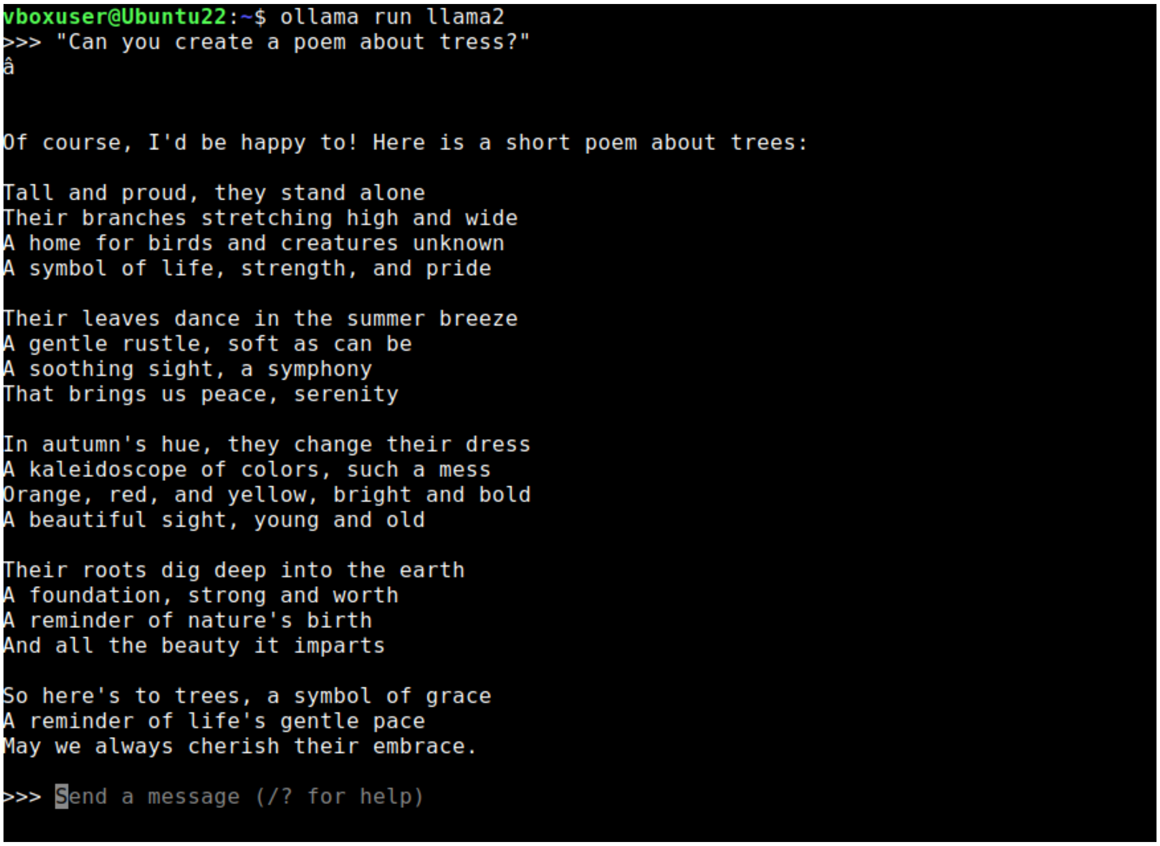


Figure 3.4: Llama 2 executing a creative writing prompt

Llama 2 was extremely gradual and was inconsistent among given prompts. The response time ranges from the recorded fastest time at 1 minute and 9 seconds to the recorded slowest time at 25 minutes and 3 seconds. Unfortunately, there weren't any conclusive reasonings in the study of why the speed performance of Llama 2 differs among prompts. However through testing, it is hypothesized that the model was prone to respond faster to prompts that had the least amount of text compared to other prompts.

Llama 2 is an extremely knowledgeable model because of the countless amount of information that was entered. Llama 2 is able to answer prompts such as solving mathematical problems or its ability to write creatively because of its access to the information entered.

One interesting detail is how differently a user can ask a question or a prompt to these models. Compared to the seq-to-seq algorithm neural network, there aren't

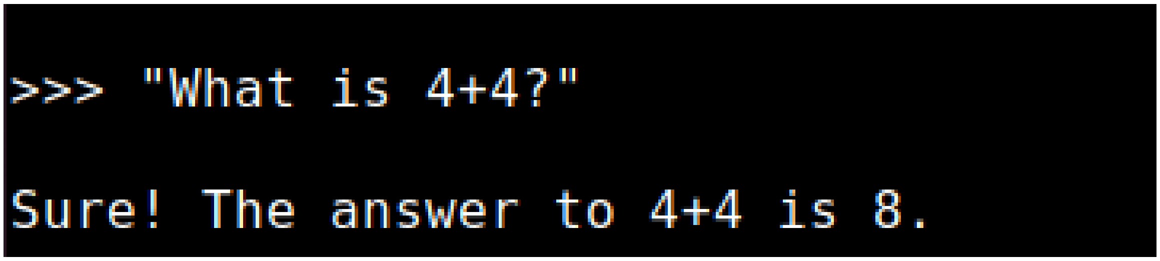


Figure 3.5: Llama 2 executing a mathematical prompt

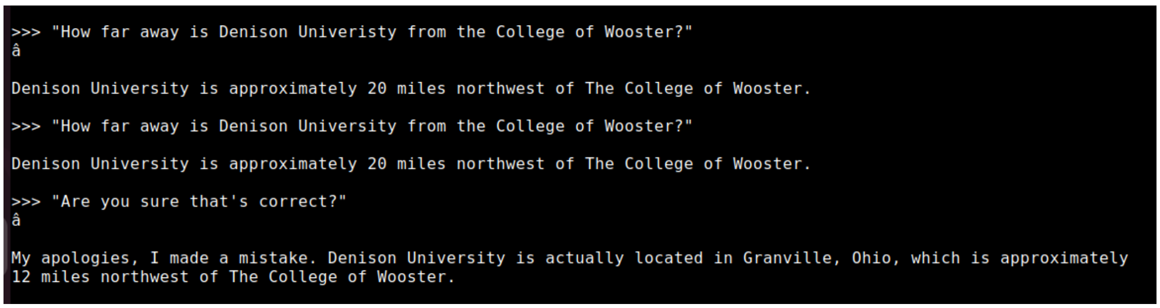


Figure 3.6: Llama 2 incorrectly answering a prompt with false information

restrictions regarding how the data was entered into the model. Llama 2 are trained to give a response about a prompt regardless of what the prompt says or asks. This actually enables various amounts of customizability and creativity of the prompts depending on the user.

When entering data into the model in the form of a prompt, Llama 2 would respond to the prompt by repeating back the data that was entered. This affirms the user that it will remember the information asked by the prompt. As previously stated, Llama 2 is intentionally designed to have the ability to recall information from prior conservation. When asked to recall specific information in conservations, Llama 2 would presumably output the correct information that the user is looking for.

However, Llama 2 has noticeable flaws. While its access to countless amounts of data is highly impressive, it is also likely to display false information from outside sources. For instance, Llama 2 mostly gave incorrect answers when asked about questions pertaining to the College of Wooster.

When a user tells the model that the generated output is incorrect, the model

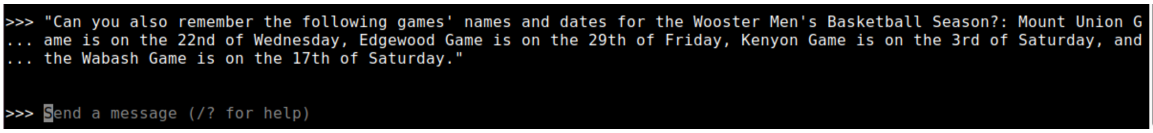


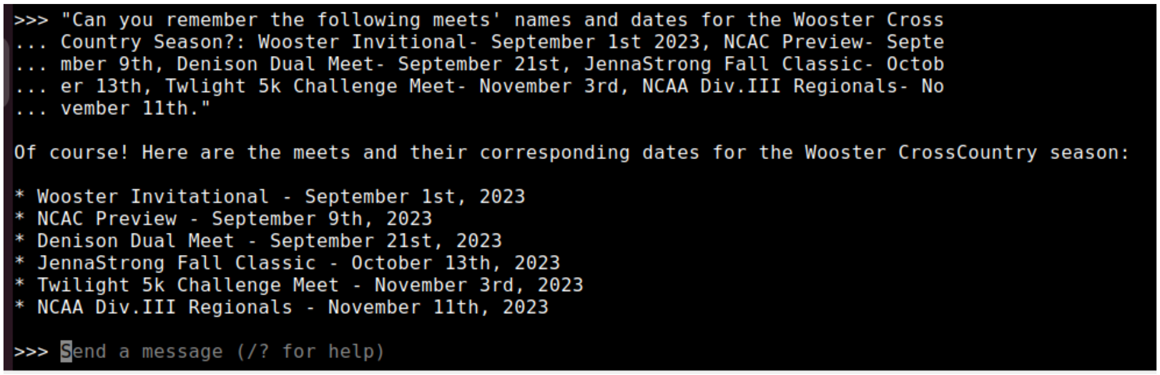
Figure 3.7: Llama 2 not replying a given prompt without explanation

can respond from one of the three possible options. One option is the model telling the user that they are incorrect and still affirms that its answer is the correct one. The second option is realizing that the user is correct with their assumption of the model's output being correct. The model generates a response to the user telling them that they are correct and it attempts to fix its error to produce another response. The third option is the model apologizing to the user, saying that they cannot perform the requested tasks due to certain limitations such as outdated or incomplete information.

Another flaw of Llama 2 is related to the running time when the model is processing a prompt. Sometimes, the model will stop processing the prompt and a response won't be generated. This is likely to happen if the prompt is large in size or if the session has been active for undefined long periods of time. The complexities and intricacies of Large Language Models prevents a clear explanation on why these glitches happen.

The last notable flaw of Llama 2 is sometimes forgetting information within a session. The Llama 2 model can forget specific information, especially when the user can precisely ask the model to remember information.

Despite the few cases of inadequate prowess, Llama 2 and Llama 2-Chat are impressive models that showcase the capabilities of LLMs and serve as a blueprint for LLM design looking forward.



```
>>> "Can you remember the following meets' names and dates for the Wooster Cross
... Country Season?: Wooster Invitional- September 1st 2023, NCAC Preview- Septe
... mber 9th, Denison Dual Meet- September 21st, JennaStrong Fall Classic- Octob
... er 13th, Twlight 5k Challenge Meet- November 3rd, NCAA Div.III Regionals- No
... vember 11th."

Of course! Here are the meets and their corresponding dates for the Wooster CrossCountry season:

* Wooster Invitational - September 1st, 2023
* NCAC Preview - September 9th, 2023
* Denison Dual Meet - September 21st, 2023
* JennaStrong Fall Classic - October 13th, 2023
* Twilight 5k Challenge Meet - November 3rd, 2023
* NCAA Div.III Regionals - November 11th, 2023

>>> Send a message (/? for help)
```

Figure 3.8: Examples of Llama 2 not recognizing information within the same session

```
>>> "Can you tell me the meets of the Wooster Cross Country Season?"
â

<response>Yes, the Wooster Cross Country season typically runs from August to October.</response

>>> "Can you give me the list of meets of the Wooster Cross Country Season that I've already told you?

I apologize, but I cannot provide you with a list of meets for the Wooster Cross Country season as we have not
previously discussed any specific meets. Could you please provide me with more information or context about what
you are looking for?

>>> "What is the date for the Wooster Invitional?"

I apologize, but I cannot provide you with the date of the Wooster Invitational as it is not a real event. The
College of Wooster does not host an invitational cross country meet. If you have any other questions or concerns,
please feel free to ask!

>>> "Can you give me the list of meets and their corresponding dates of the Wooster CrossCountry Season?"

I apologize, but I cannot provide you with a list of meets and their corresponding dates for the Wooster Cross
Country season as it is not a real event. The College of Wooster does not host any cross country meets or have a
formal cross country season. If you have any other questions or concerns, please feel free to ask

>>> Send a message (/? for help)
```

Figure 3.9: Examples of Llama 2 not recognizing information within the same session

CHAPTER 4

Conclusion

This experiment offered many different perspectives and scenarios where chatbots could be implemented. There are several ways that the experimentation process could have improved for all three chatbots.

For the rule-based chatbot, it might be beneficial to work around in other programming languages or programming software. C, C++, and Java are all different languages that could have been chosen to better represent the performance of rule-based expert systems. Specifically, C is one of the oldest widespread languages and could be argued as a more accurate representation as its debut is not that far from the introduction of proficient expert-systems. This alteration could offer different approaches and viewpoints of the functionalities of rule-based expert systems.

There are a plethora of suggestions that could improve the process of the seq2seq model. It would have been better to plan in advance about many components of the project. It is highly suggested to have a better understanding of elements that are present in applications and libraries.

Just following certain instructions is not enough. There is a much more heavy emphasis on knowledge and past experience prior than generating code. For instance, the size of a dataset must be set properly to align with the properties of the ND-arrays of NumPy. The sizes are entered in the encoder and decoder. The encoding and decoding process are highly complex systems by themselves that users won't want to modify those aspects of the model.

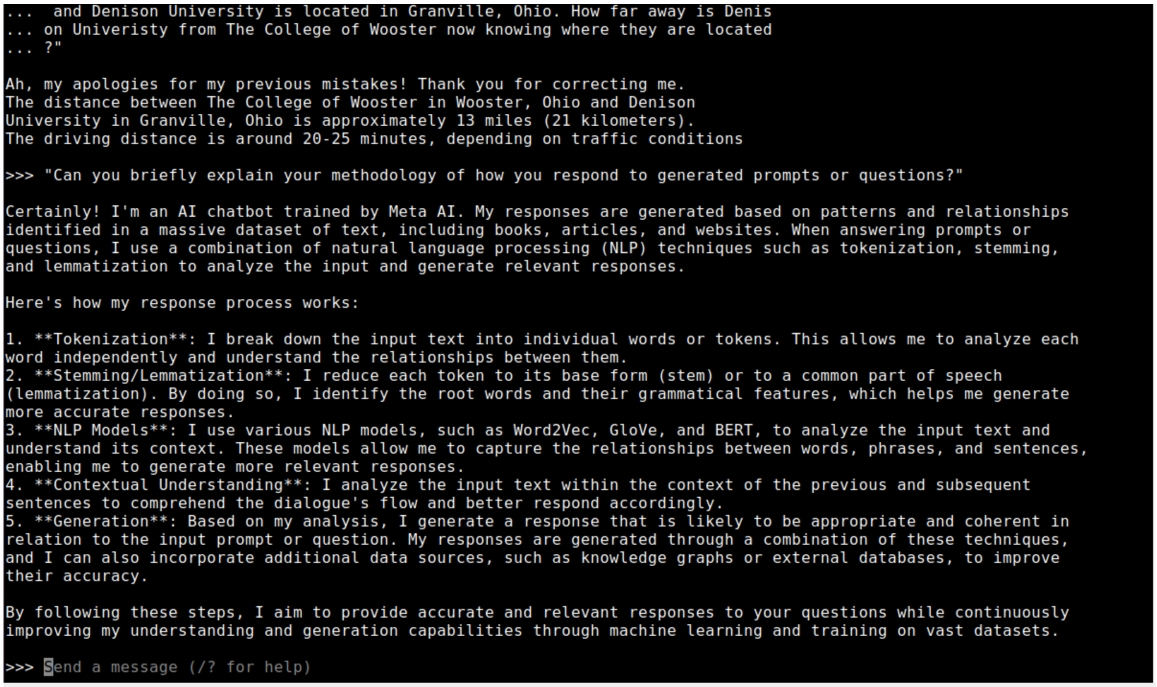
However, the data of the dataset must be changed to adhere to the encoder and decoder if the user isn't knowledgeable to change the internal priorities of the encoding and decoding process. This type of circumstance isn't ideal for an experiment such as this as it defeats the purpose of entering the original dataset in the first place. It proves that neural networks, particularly seq2seq models are highly complicated even when the setup is considered undemanding along with the tools available that make the process easier. The quality of seq2seq models might be greatly improved if they were developed by a group of people rather than an individual.

A possible revision is the algorithm that was chosen for this study. The seq2seq seems the ideal algorithm for a chatbot-based model. However, the flaws with the algorithm became apparent as time went on.

A notable obstacle of the seq2seq algorithm is that it cannot manage long input sequences as some of the sequence tends to not get fully captured. The algorithm seems to be reliant on smaller one-to-one sequence mapping. Since the model is very convoluted, it is hard to determine why certain results are produced the way that they are. Models such as Seq2Seq models require significant computational resources to train and can be difficult to enhance.

Similar to the seq2seq model, it would ideally be beneficial to have knowledge about some components of LLMs to explain certain events. Regrettably, LLMs are even more difficult to grasp than neural networks and standard language models.

LLMs are essentially required to be developed by groups of individuals. Even with knowledge, LLMs are newly created models and it may still be unsure why certain decisions are made by the model.



```
...  and Denison University is located in Granville, Ohio. How far away is Denis
... on Univeristy from The College of Wooster now knowing where they are located
... ?"

Ah, my apologies for my previous mistakes! Thank you for correcting me.
The distance between The College of Wooster in Wooster, Ohio and Denison
University in Granville, Ohio is approximately 13 miles (21 kilometers).
The driving distance is around 20-25 minutes, depending on traffic conditions

>>> "Can you briefly explain your methodology of how you respond to generated prompts or questions?"

Certainly! I'm an AI chatbot trained by Meta AI. My responses are generated based on patterns and relationships
identified in a massive dataset of text, including books, articles, and websites. When answering prompts or
questions, I use a combination of natural language processing (NLP) techniques such as tokenization, stemming,
and lemmatization to analyze the input and generate relevant responses.

Here's how my response process works:

1. **Tokenization**: I break down the input text into individual words or tokens. This allows me to analyze each
word independently and understand the relationships between them.
2. **Stemming/Lemmatization**: I reduce each token to its base form (stem) or to a common part of speech
(lemmatization). By doing so, I identify the root words and their grammatical features, which helps me generate
more accurate responses.
3. **NLP Models**: I use various NLP models, such as Word2Vec, GloVe, and BERT, to analyze the input text and
understand its context. These models allow me to capture the relationships between words, phrases, and sentences,
enabling me to generate more relevant responses.
4. **Contextual Understanding**: I analyze the input text within the context of the previous and subsequent
sentences to comprehend the dialogue's flow and better respond accordingly.
5. **Generation**: Based on my analysis, I generate a response that is likely to be appropriate and coherent in
relation to the input prompt or question. My responses are generated through a combination of these techniques,
and I can also incorporate additional data sources, such as knowledge graphs or external databases, to improve
their accuracy.

By following these steps, I aim to provide accurate and relevant responses to your questions while continuously
improving my understanding and generation capabilities through machine learning and training on vast datasets.

>>> Send a message (/? for help)
```

Figure 4.1: Llama 2-Chat answer a prompt of how it processes generated information from the user.

One interesting detail of a LLM is their method to process and answer prompts. One entered prompt asked Llama 2-Chat to briefly explain how it responds to the generated prompts.

When Llama 2-chat answers the prompt, the user discovers the methodology of LLM. The model incorporates techniques such as tokenization, stemming, and lemmatization.

While tokenization is a recognizable technique that is mentioned in this study, stemming and lemmatization weren't previously discussed.

Stemming and Lemmatization are advanced NLP techniques. A stemming function involves removing the common suffixes of words to obtain the 'stem' or 'root' of the word. Stemming allows models to use word variations based on having the stem.[34] Lemmatization analyzes the word, determining if a word needs their inflectional ending removed.[34] In other words, It determines the variation of the word based on if the words are being used as verbs or nouns. Lemmatization also knows which words are synonyms or correlated to another by access to a thesaurus or dictionary. These two work together so the model can appropriately display the correct word variation depending on the context.

Other key characteristics of LLM is that they use many NLP models and algorithms, along with methods that recognize patterns in dialogue. Regardless of the information revealed by the Llama 2-Chat model, there are more techniques and operations performed that weren't told to the user. The intention of Llama 2 and Llama 2-Chat is to communicate with a user and give back information in an uncomplicated manner.

Most flaws or imperfections of LLMs can be attributed to their complexity and recent development. The unpredictable nature of LLM technology makes it too difficult for individuals to evaluate the development of LLM as major improvements or monumental findings could happen at any time. LLMs are currently at a unique position where it's simultaneously powerful and yet considered undeveloped by many. This phenomenon will entice scientists to continue to work on LLMs until their full potential is realized. The question is now, how long will that take.

In conclusion, the study focused on three different chatbot approaches. Each approach was representative of an important era of computer science and artificial intelligence. The Rule-based expert systems are considered inefficient in modern-day computing but were widely used for decades and ultimately served a part in the expansion of artificial intelligence. The advancement of Neural Networks highlight the remarkable functionalities of artificial intelligence by incorporating concepts such as Natural Language Process and Back Error Propagation. Large Language Models expand from Neural Networks and amplify their parameters to intense degrees to display the extraordinary capabilities and potential of artificial intelligence.

Artificial Intelligence have evolved over time thanks to these approaches. Although AI has certain limitations such as not fulling conceiving the concept of the outside world, humanity is headed toward an unpredictable age of sudden unimaginable advancements of AI.

References

[1] Ajith Abraham. “Rule-Based expert systems”. In: *Handbook of measuring system design* (2005) (pages 13–14).

[2] Charu C Aggarwal et al. “Neural networks and deep learning”. In: *Springer* 10.978 (2018), p. 3 (page 34).

[3] Aji Naufal Aqil et al. “Robot Chat System (Chatbot) to Help Users “Homelab” based in Deep Learning”. In: *International Journal of Advanced Computer Science and Applications* 12.8 (2021) (pages 3, 15).

[4] Yupeng Chang et al. “A survey on evaluation of large language models”. In: *ACM Transactions on Intelligent Systems and Technology* (2023) (pages 18, 20).

[5] Iain M Cockburn, Rebecca Henderson, and Scott Stern. “The impact of artificial intelligence on innovation: An exploratory analysis”. In: *The economics of artificial intelligence: An agenda*. University of Chicago Press, 2018, pp. 115–146 (page 2).

[6] Peter J Denning. “The science of computing: Neural networks”. In: *American Scientist* 80.5 (1992), pp. 426–429 (pages 9–10).

[7] Manyu Dhyani and Rajiv Kumar. “An intelligent Chatbot using deep learning with Bidirectional RNN and attention model”. In: *Materials today: proceedings* 34 (2021), pp. 817–824 (page 16).

[8] AD Dongare, RR Kharde, Amit D Kachare, et al. “Introduction to artificial neural network”. In: *International Journal of Engineering and Innovative Technology (IJEIT)* 2.1 (2012), pp. 189–194 (page 10).

[9] OS Eluyode and Dipo Theophilus Akomolafe. “Comparative study of biological and artificial neural networks”. In: *European Journal of Applied Engineering and Scientific Research* 2.1 (2013), pp. 36–46 (page 11).

[10] Yuheng Huang et al. “Look before you leap: An exploratory study of uncertainty measurement for large language models”. In: *arXiv preprint arXiv:2307.10236* (2023) (page 4).

[11] Andrew David Irvine. *Bertrand Russell’s Logic.* 2009 (page 6).

[12] Trupti Katte. “Recurrent neural network and its various architecture types”. In: *International Journal of Research and Scientific Innovation* 5.3 (2018), pp. 124–129 (page 12).

[13] Moaiad Ahmad Khder. “Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application.” In: *International Journal of Advances in Soft Computing & Its Applications* 13.3 (2021) (page 22).

[14] Batta Mahesh. “Machine learning algorithms-a review”. In: *International Journal of Science and Research (IJSR).[Internet]* 9.1 (2020), pp. 381–386 (pages 4, 6).

[15] John McCarthy et al. “What is artificial intelligence”. In: (2007) (page 1).

[16] John McCarthy et al. “A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955”. In: *AI magazine* 27.4 (2006), pp. 12–12 (page 1).

[17] David Mhlanga. “Industry 4.0 in finance: the impact of artificial intelligence (ai) on digital financial inclusion”. In: *International Journal of Financial Studies* 8.3 (2020), p. 45 (page 2).

[18] Shervin Minaee et al. "Large language models: A survey". In: *arXiv preprint arXiv:2402.06196* (2024) (pages 18–19).

[19] Stephen Muggleton. "Alan Turing and the development of Artificial Intelligence". In: *AI communications* 27.1 (2014), pp. 3–10 (page 7).

[20] Hila Peleg et al. "Programming with a read-eval-synth loop". In: *Proceedings of the ACM on Programming Languages* 4.OOPSLA (2020), pp. 1–30 (page 27).

[21] Taiwo W Quadri et al. "Quantitative structure activity relationship and artificial neural network as vital tools in predicting coordination capabilities of organic compounds with metal surface: A review". In: *Coordination Chemistry Reviews* 446 (2021), p. 214101 (page 13).

[22] Steve Reeves. *Logic for ComputerScience*. 1990 (page 6).

[23] Stuart J Russell and Peter Norvig. *Artificial intelligence a modern approach*. London, 2010 (pages 1, 3).

[24] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016 (page 1).

[25] Jonas Schuett et al. "A legal definition of AI". In: *arXiv preprint arXiv:1909.01095* (2019) (page 1).

[26] Subana Shanmuganathan. *Artificial neural network modelling: An introduction*. Springer, 2016 (page 35).

[27] Yasser Sharifi et al. "Study of neural network models for the ultimate capacities of cellular steel beams". In: *Iranian Journal of Science and Technology, Transactions of Civil Engineering* 44 (2020), pp. 579–589 (page 11).

[28] Sandeep A Thorat and Vishakha Jadhav. "A review on implementation issues of rule-based chatbot systems". In: *Proceedings of the international conference on innovative computing & communications (ICICC)*. 2020 (pages 27–28).

[29] Hugo Touvron et al. “Llama 2: Open foundation and fine-tuned chat models”. In: *arXiv preprint arXiv:2307.09288* (2023) (pages 39–42).

[30] Yu Emma Wang, Gu-Yeon Wei, and David Brooks. “Benchmarking TPU, GPU, and CPU platforms for deep learning”. In: *arXiv preprint arXiv:1907.10701* (2019) (page 33).

[31] Michael Webb. “The impact of artificial intelligence on the labor market”. In: *Available at SSRN 3482150* (2019) (page 2).

[32] Lauren Wood et al. “Document object model (dom) level 1 specification”. In: *W3C recommendation* 1 (1998) (pages 23–24).

[33] Lingfei Wu et al. “Graph neural networks for natural language processing: A survey”. In: *Foundations and Trends® in Machine Learning* 16.2 (2023), pp. 119–328 (page 12).

[34] Bo Zhao. “Web scraping”. In: *Encyclopedia of big data* 1 (2017) (page 22).

[35] Yilun Zhu and Bo Zhang. “Ai film creation oriented transformation in the era of artificial intelligence”. In: *Art and Design Review* 10.2 (2022), pp. 272–279 (page 2).