

Virtual Teaching Assistant for Undergraduate Students Using Natural Language Processing & Deep Learning

by

Sadman Jashim Sakib
18101635

Baktiar Kabir Joy
18301018

Zahin Rydha
18301042

Md. Nuruzzaman
18301126

Khaled Ahmmed Anik
18301083

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
May 2022

© 2022. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



Sadman Jashim Sakib
18101635



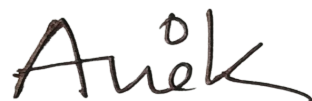
Baktiar Kabir Joy
18301018



Zahin Rydha
18301042



Md. Nuruzzaman
18301126



Khaled Ahmmed Anik
18301083

Approval

The thesis project titled “Virtual Teaching Assistant for Undergraduate Students Using Natural Language Processing & Deep Learning” is submitted by

1. Sadman Jashim Sakib (18101635)
2. Baktiar Kabir Joy (18301018)
3. Zahin Rydha (18301042)
4. Md. Nuruzzaman (18301126)
5. Khaled Ahmmmed Anik (18301083)

Of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on May 24, 2022.

Examining Committee:

Supervisor and Program Coordinator:
(Member)

**Annajiat
Alim
Rasel** Digitally signed by
Annajiat Alim Rasel
DN: cn=Annajiat Alim
Rasel, o=Brac University,
ou=CSE Department,
email=annajiat@bracu.ac.
bd, c=BD
Date: 2022.06.02 08:05:23
+06'00'

Annajiat Alim Rasel
Senior Lecturer
Department of Computer Science and Engineering
Brac University

Secondary-Supervisor:
(Member)



Matin Saad Abdullah, PhD
Professor
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Online education's popularity has been continuously increasing over the past few years. Many universities were forced to switch to online education as a result of COVID-19. In many cases, even after more than two years of online instruction, colleges were unable to resume their traditional classroom programs. A growing number of institutions are considering a hybrid approach to education, in which some face-to-face teaching is augmented with online learning. Nevertheless, many online education systems are inefficient, and this results in a poor rate of student retention. In this paper, we are offering a primary dataset, a virtual teaching assistant named VTA-bot, and its system architecture. In addition, we are showing a first implementation of the suggested system, which consists of a chatbot that can be queried about the content and topics of the 'Programming Language I' course, an introductory programming language course offered by the CSE department of Brac University. Students in their first year of university will benefit from this strategy, which aims to increase student participation and involvement in online education.

Keywords: Dataset, Chatbot, Tokenization, Classifier, Bag of words, Stemming, Lemmatization, Word Embedding, Prototype, Neural Network

Acknowledgement

First and foremost, we owe a debt of gratitude to the Almighty Allah, for whom our thesis was finished without any serious setbacks.

Secondly, we would like to thank our honourable supervisor Annajiat Alim Rasel sir and secondary supervisor Matin Saad Abdullah sir for giving us their valuable time whenever we needed it. In the beginning, we lacked extensive expertise. But they never turned their backs on us and provided us with complete direction at all times. We are immensely grateful to them for their unwavering assistance.

And finally, we would like to express our gratitude to our parents. They supported us through all of life's ups and downs. We would not be where we are now without their unwavering encouragement and prayers.

Table of Contents

Declaration	i
Approval	ii
Abstract	iv
Acknowledgment	v
Table of Contents	vi
List of Figures	viii
List of Tables	ix
Nomenclature	x
1 Introduction	1
1.1 Background and Motivation	1
1.2 Research Aim	2
1.3 Research Problems	2
2 Literature Review	4
3 Problem Statement	6
4 Dataset for VTA-Bot	7
4.1 Methodology	7
4.1.1 Objective	7
4.1.2 Book Selection	8
4.1.3 Topic Selection	9
4.1.4 Questions & Answer Selection	9
4.1.5 Tagging the Questions	9
4.1.6 Review	9
4.2 Data Hierarchy	9
4.3 Scale	11
4.4 Data Pre-processing	12
4.4.1 Importing the Dataset from Google Sheet	12
4.4.2 Preparing Dataset for Preprocessing	12
4.4.3 Case Folding	15
4.4.4 Punctuation Removal	16

4.4.5	Removal of Stopwords	16
4.4.6	Stemming	17
4.4.7	Lemmatization	17
4.4.8	Emoji Removal	18
4.5	Fitting dataset on basic classifiers	18
5	System Design	26
5.1	System Architecture	26
5.1.1	User Interface	26
5.1.2	Backend Process	26
5.2	Design Data Flow	27
5.2.1	Conceptual Queries	27
5.2.2	Factual Queries	28
5.2.3	Predefined Questions	28
5.2.4	Support Seeking	28
6	Prototype Development	29
6.1	NLP Techniques	29
6.2	The Neural Network Model	30
6.3	Prototype Implementation	31
6.4	Prototype Training and Testing	32
7	Future Work	35
8	Conclusion	36
	Bibliography	39

List of Figures

4.1	Workflow	8
4.2	Data Hierarchy	10
4.3	Number of questions under the tags (%)	11
4.4	Dataset from Google-sheet	12
4.5	Eliminate empty rows	13
4.6	Reset row numbers after eliminating null rows	13
4.7	qSet	14
4.8	aSet	15
4.9	Case Folding	16
4.10	Punctuation Removal	16
4.11	Removal of stopwords	17
4.12	Stemming	17
4.13	Lemmatization	18
4.14	Emoji Removal	18
4.15	Classifiers accuracy on whole dataset	19
4.16	Classifiers accuracy on refactored dataset	20
4.17	Accuracy comparison between whole, refactored dataset	21
4.18	Classifiers accuracy on further refactored dataset	22
4.19	Accuracy comparison between whole, refactored and further refactored dataset	23
4.20	F1 Score on Whole Dataset	24
4.21	F1 Score on refactored Dataset	24
4.22	F1 Score on further refactored Dataset	25
5.1	System Architecture	27
5.2	Design Data Flow	28
6.1	Data preprocessing	29
6.2	NLP Techniques	30
6.3	Feed Forward Neural Network	31
6.4	Model Training Loss Curve	33
6.5	Accuracy Curve	33
6.6	Conversation Between VTA-bot and Student	34

List of Tables

4.1	Different Sites Containing Python Queries	7
4.2	Dataset Overview	11
6.1	Different Libraries and Functions Used	32

Nomenclature

The following list describes several symbols & abbreviation that will be later used within the body of the document

AI Artificial Intelligence

AIML Artificial Intelligence Markup Language

aSet Answer Set

JSON JavaScript Object Notation

MLF Multi-Layer Feed-Forward Network

NLG Natural Language Generation

NLP Natural Language Processing

NLTK Natural Language Toolkit

NLU Natural Language Understanding

qSet Question Set

URL Uniform Resource Locator

VTA Virtual Teaching Assistant

Chapter 1

Introduction

Students are unable to grasp computer programming due to a lack of resources. To improve the teaching of any subject, available innovations must be utilized. One of the most important skills for the Fourth Industrial Revolution is programming[21]. The majority of students avoid programming because they perceive it to be difficult. Additional learning and teaching techniques may be employed to aid in the learning process. Undergraduates have less concerns about development environments, and the majority believe that hands-on learning is more valuable than studying theory-based subject matter [27]. Our research aims to solve the issue of a significant number of new learners who become stuck while learning the fundamentals of the Python programming language and may not obtain appropriate assistance through the online-based education systems of many institutions.

1.1 Background and Motivation

With the rapid development of modern technology, the use of automated objects is expanding exponentially. The introduction of intelligent agents is greatly facilitated by Artificial Intelligence (AI). NLP-based chatbot help is a subset of artificial intelligence (AI). NLP-based chatbots are used in several areas, including health, e-commerce, finance, and food production. Nearly 1.4 billion people use chatbots, which may save up to 30% on operating costs, and 27% of people are enthusiastic about artificial intelligence assistance systems [17]. Unfortunately, the education system has still not utilized AI-based assistant technology to the extent that it should, resulting in a scarcity of helpful tools in the area. The education system is a promising use for Chatbots, notably if an interactive educational environment is required. Introduction to programming courses has significant dropout rates, with up to 50% of students dropping out [3]. It is possible to use chatbots to assist students with programming language-related questions as a partial replacement for teaching assistants, given that programming languages are a crucial module of Data Science and Computer Science. This paper presents an assistive teaching Chatbot that uses NLP techniques to aid students in learning the Python programming language in a participatory manner at any time.

1.2 Research Aim

Our study aims to assist a significant number of first-year students who struggle with a variety of issues and oftentimes become disoriented when adapting to programming-related concepts. Moreover, at certain universities, there may not be enough support to assist students with the online-based education system. Thus, the purpose of this research is to introduce a virtual teaching assistant (VTA) chatbot that will facilitate student learning and decrease the possibility of failure. Again, due to the shortage of adequate data in this field, we aim to create a primary dataset for our system. So now the chatbot can automatically respond to a variety of inquiries pertaining to fundamental programming and avoid serving as a conduit for cheating by guiding students to solve their unique challenges without providing exact answers. We will demonstrate the chatbot's initial implementation, which will be developed using Natural Language Processing (NLP) and deep learning approaches [22].

1.3 Research Problems

1. Users have a limited amount of time to get responses to their inquiries and expect lightning-fast responses. It's challenging to build chatbots that maintain users' attention till the end.
2. When dealing with user input, such as slang, misspellings, tone, humour and grammar, it is tough to get it to work successfully.
3. Another significant obstacle is chatbot testing, which accounts for the bulk of the difficulties. Chatbots are constantly evolving as a result of advances in natural language modeling. As a consequence, testing and operationalizing the chatbot is vital to ensuring its correctness.
4. Students will always receive assistance from assistive bots whenever they become stuck in the learning process, even during exam time. As a result, it is a significant problem to design the system in such a manner that it does not become a conduit for copying.
5. Finding a good dataset is also a huge problem. We couldn't discover a dataset that matched our system criteria in our scenario.
6. Making an appropriate dataset for our system required processing a substantial amount of raw data.
7. Finding the right framework for our bot was a difficult decision. It was possible to create a bot quickly using pre-built frameworks. Because of this, there are several drawbacks, such as the loss of flexibility and complete control.
8. The annotation process is one of the most difficult components of this research, since the majority of queries, whether programming-related or not, look confusing.
9. Choosing relevant questions from a given topic proved to be a daunting problem.

10. Selecting books that corresponded well with the course material was challenging for us.
11. For better training, we must exclude frequent words during pre-processing; however, most of the frequent words in our dataset are keywords. Therefore, we do not exclude these common keywords.
12. Finding appropriate classifiers to determine the quality of our dataset based on its characteristics.

Chapter 2

Literature Review

Since ELIZA [41], an early chatbot, there have been significant advancements in chatbot technology. Using NLP and machine learning components, chatbots are artificially intelligent systems designed to simulate interactions with a single user or a group of users. Chatbots have both academic and commercial applications. In addition to non-academic applications, experts at Stanford University say that chatbots are better for students than other ways to talk [17].

The integration of education and AI will be aided by the introduction of big data, the constant emergence of digital campuses and online learning platforms [20]. Memon proposed an effective, simple, easy, and low-cost framework approach for designing a text-based multi-interactive chatbot. This will support the development of a multi-interactive chatbot's system for an educational area using AIML 2.0. This will also facilitate the students a personalized learning environment for their learning towards an outcome-based education domain[14].

Numerous academic institutions have integrated AI in a variety of methods, from online education systems to web-based virtual chatbots [12]. An example of such a system employs natural language processing (NLP) to evaluate and retrieve student-entered text [26]. This technology may provide students with an interactive learning environment. As a result of the newly adopted online-based teaching technique, students often feel disoriented when transitioning to programming principles and may lack sufficient assistance. We have not been able to fully duplicate the formal and informal learning that happens in person and through interactions with other people [9].

Attempting to recall code line by line is not the proper method for comprehending a programming language [19], because students' learning capacities vary. Heller et al. (2005) created Freudbot, a chatbot, to collect insights from psychology students in order to establish an environment in which students may participate and, therefore, overcome the difficulty of responding to questions [11]. Sadhasivam presented in his study "Implementation of Chatbot That Teach Programming Language" a chatbot named Progbot that can educate an underused programming language easily and intelligently [19]. Apprentices who are uncertain where to start can still inquire the chatbot to start the lesson, and it begins with the fundamental topics and steadily advances to additional troublesome material. The request is replied to by the sys-

tem's built-in artificial intelligence. The user has to choose the invalid reply button, which is able to alarm the system administrator [15], if they find any response that they are not looking for. The foremost related work to our paper could be A. Goel's presentation of Jill Watson [16]. Though it is not a personalized system [25], Jill Watson released the lecturers from responding to students' inquiries. The objective of this work is to provide a customized and intelligent virtual teaching assistant (VTA) chatbot designed to make students' learning experiences more fun and reduce the risk of failure.

The VTA in ProTracer 2.0 will figure out if a student's answer is right or wrong by comparing it to the teacher's answer [28]. Responding to basic programming queries and aiding students in overcoming obstacles will be the main capabilities of the Chatbot. The extraction of intent and entities is a crucial stage in the process of conversation management. Rasa NLU is an adaptive mechanism that recognizes intent and entities in human speech [13]. The Rasa NLU model is passed to the NLU Interpreter after training, which decodes trial intents to determine whether NLU can correctly classify and retrieve entities [13]. A smartphone-based chatbot system named Alpha is executed in the Python programming language and uses the Dialog Flow framework, with a machine learning model to pull out the intents and a cloud-based database to store data [17]. The primary objective of virtual teaching assistants is to improve both the learning experience and the coordination mechanisms in the classroom. The university's academic cloud stores lessons for students' mobile terminals. The VTA facilitates additional effective learner understanding by providing feedback instantly [24]. 'Coding Tutor' [23], 'e-Java' [18], 'ProgBot' [19] and 'PythonBot' [21] are available with affiliated literature. Regardless, most of them include extremely undersized content and concentrate on a particular programming language. This paper is related to existing literature to some extent still mainly focuses on support in the python programming language related courses for first-year Undergraduate students.

Chapter 3

Problem Statement

Our system detects the intents of the queries as a supervised classification problem where the set of intents, $I = \{\text{"greetings"}, \text{"method"}, \text{"class"}, \text{"loop"}, \text{"if-else"} \dots \dots \text{"goodbye"}\}$. Given a set of question patterns, $T = \{t_1, t_2, t_3, t_4 \dots t_n\}$ and their responses, $R = \{r_1, r_2, r_3, r_4, \dots, r_n\}$ and other metadata information, our goal is to predict the correct intent label, $Y = \{y_1, y_2, y_3, \dots, y_n\}$ for generating the appropriate response. After processing the input set, our system generates a probability set, $P = \{p_1, p_2, p_3, p_4, \dots, p_n\}$ through a matching algorithm, which represents the matching of queries with intents. The output with the maximum probability value indicates the correct intent [2].

Chapter 4

Dataset for VTA-Bot

4.1 Methodology

The digital revolution has resulted in a massive increase in the amount of data that can be stored. Various databases hold the answers to students' questions about the Python programming language. Stack Overflow, Stack Exchange, Python.org, Programmers Heaven and Find-Nerd are among the question and answer sites for professional and hobbyist programmers (Table-4.1). For the ultimate collection of student questions and answers, there are free and open-source techniques. There are also Python-related queries from a number of well-known programming books. In addition, the concise note supplied by the Brac University teachers offered adequate information. Models and methods for indexing, retrieving, organizing and interacting with this data may be developed by leveraging these datasets. Again, some available Python datasets gave the advantage to create a new dataset for the VTA-bot. The workflow of creating the dataset is shown in Table-4.1.

Websites	No. of Python Queries
Stack Overflow	1,870,130
Programmers Heaven	546
Stack Exchange	5,452,235
Python.org FAQ	1,050
FindNerd	200

Table 4.1: Different Sites Containing Python Queries

4.1.1 Objective

Initially, we intended to compile a database of 2,000 questions and answers. After evaluating the relevant books, websites, concise notes, and accessible datasets, we determined that 2,000 questions would approximately cover all of the course's contents. At first, our objective was to create a flowchart of our work Figure-4.1. We started sample collection accordingly.

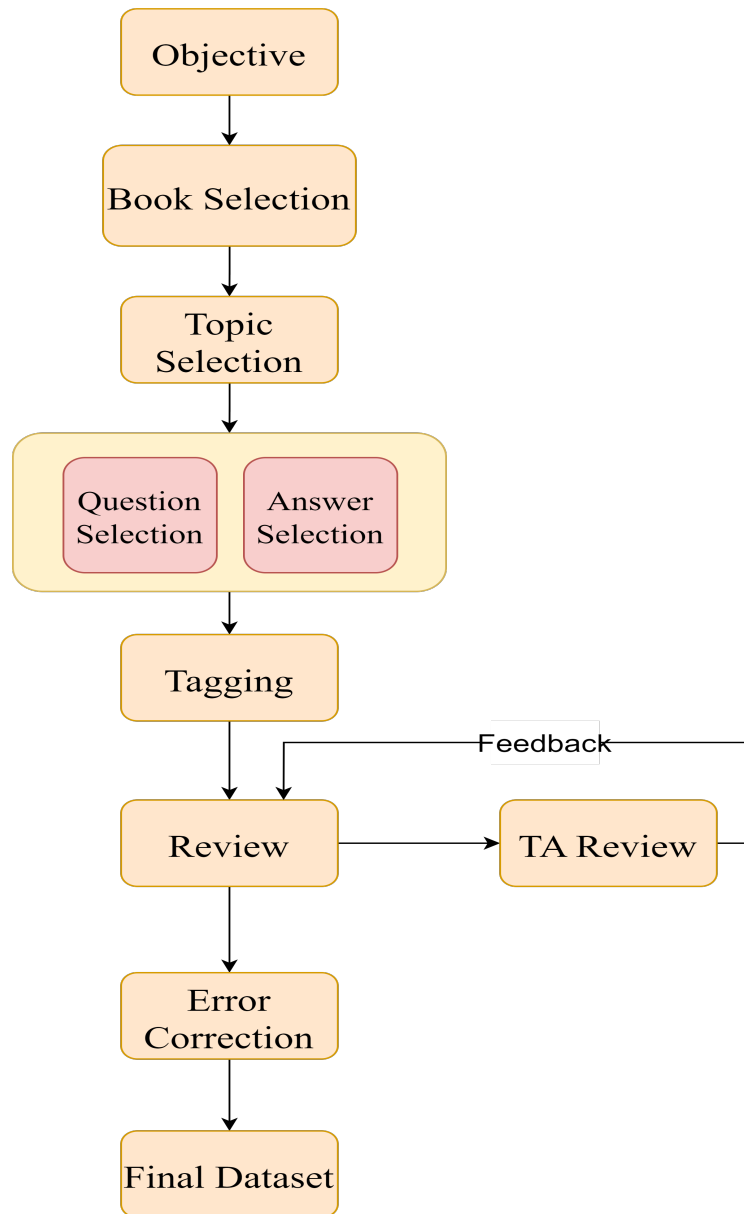


Figure 4.1: Workflow

4.1.2 Book Selection

We first chose ten books for the course [29], [30], [32]–[39]. We formed a group of two members to choose books from among them. They picked the following three books:

1. Alex Martelli’s “Python in a Nutshell” 1st Edition [37].
2. The second edition of Mark Lutz and David Ascher’s “Learning Python” [36].
3. Mark Summerfield’s “Programming in python 3: a complete introduction to the python language” [33]

After selection, the whole group evaluated the books and agreed to gather samples. We discovered that these three books cover almost all of the course’s content.

4.1.3 Topic Selection

We assigned topic selection to each member of the group. We initially compiled a list of chapter-specific topics. Then, we eliminated all repeated topics and compiled a list of unique ones. These topics encompass the entire course.

4.1.4 Questions & Answer Selection

This was one of the hardest tasks to select questions and answers. We first distributed the unique topics to each member of the group. Each group member then compiled all potential questions and answers for each topic.

4.1.5 Tagging the Questions

The process of tagging the questions was one of the most challenging aspects of this study because arranging the questions according to a similar pattern, appeared to be perplexing. This is also a major purpose of the project, since identifying requests as a different intent may result in wrong output, which we are trying to avoid. We gave a tag to every similar patterned question.

4.1.6 Review

To verify the completeness of the data we obtained, we consulted a teacher's assistant for this course. According to the feedback we received, we modified our dataset. We included several essential questions acquired from the students' queries to our dataset. After fixing all the errors in the dataset, we finally obtained our desired dataset.

4.2 Data Hierarchy

Our Python programming language data collection is organized hierarchically. The root is set to "intent". We saved "topics" inside of "intent". Each "topic" includes "tags". Again, each "tag" comprises questions of a similar kind. Lastly, each question corresponds to a certain response.

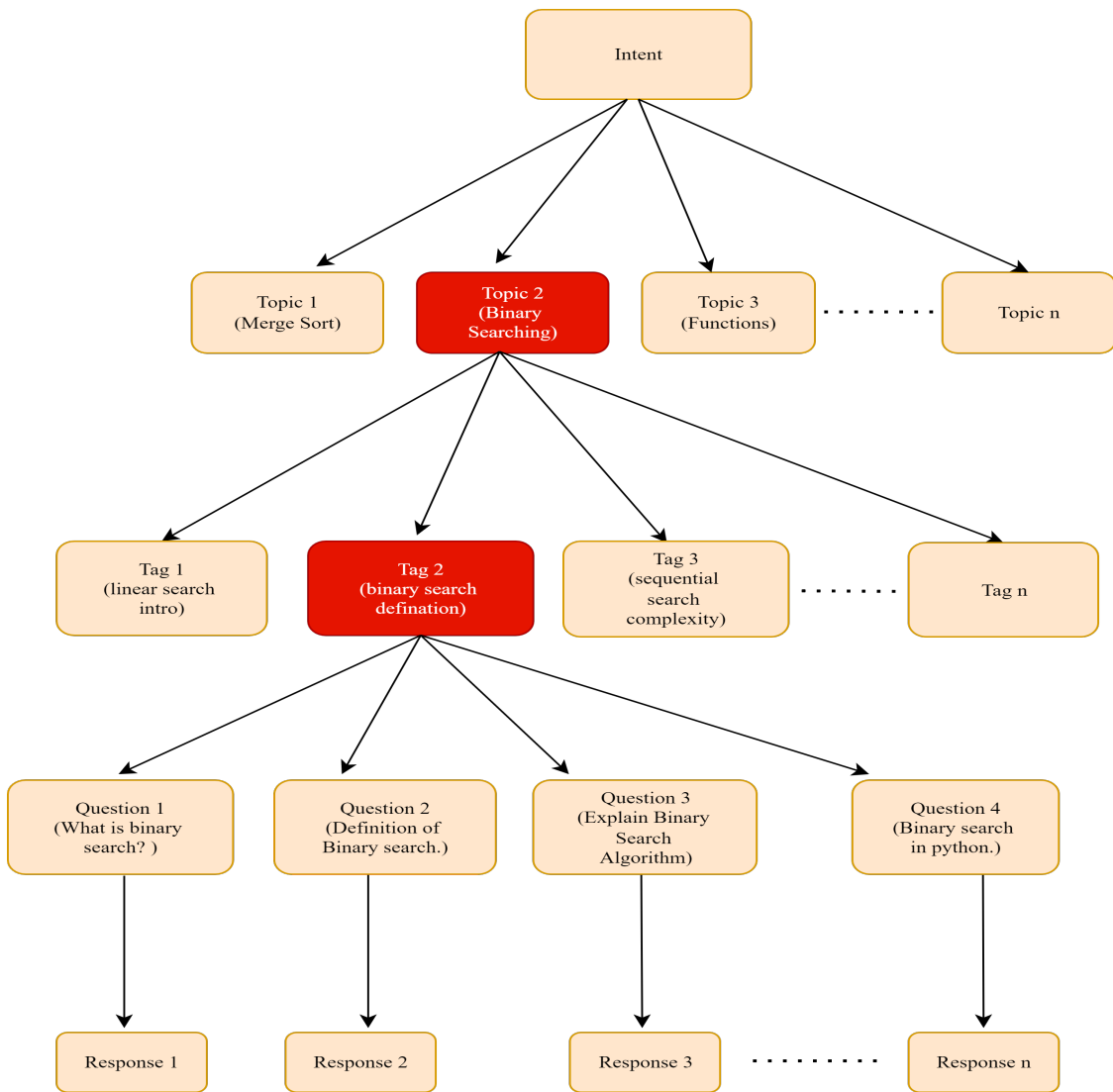


Figure 4.2: Data Hierarchy

4.3 Scale

The dataset has been arranged according to the above procedure and is ready for preprocessing and training-testing [31]. Currently, our dataset has a total of 214 unique tags, 1,006 questions, and 254 unique responses. Below are the percentages of questions associated with each category [Figure-4.3]. 18.7% of all tags have two or less questions, 18.2% contain three questions, 22.4% contain four questions, and the remainder contain five to ten questions. On average, each tag comprises between five and six questions.

No of Tag	No of Pattern	No of Response
214	1,006	254

Table 4.2: Dataset Overview

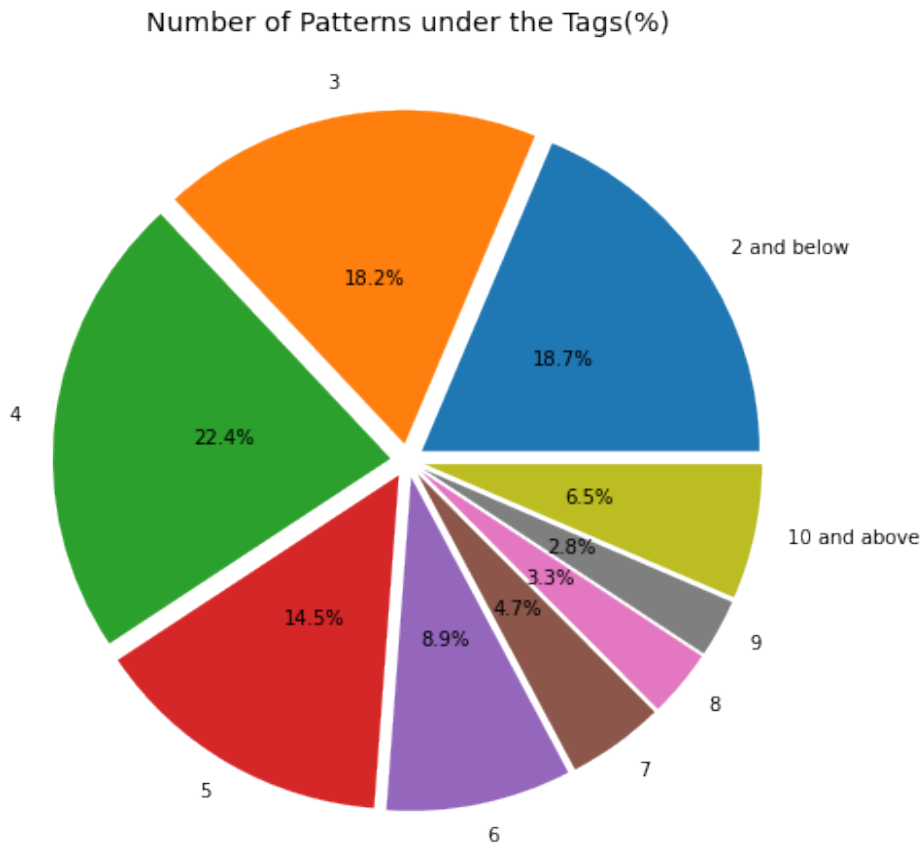


Figure 4.3: Number of questions under the tags (%)

4.4 Data Pre-processing

4.4.1 Importing the Dataset from Google Sheet

To begin the preprocessing of our dataset, we must first import the dataset. The dataset was imported by copying it directly from the Google Sheet (Figure-4.4). Therefore, we will always be able to use the most current dataset. If we do not import our dataset, we will be compelled to do so, and every time we modify the dataset, we will need to import it again, which will be a time-consuming procedure. In contrast, if we loaded our dataset directly from the Google sheet, we can instantly get the updated dataset whenever there is a change to the dataset itself. In addition, it will save us time, which is a significant benefit.

	Tag	Pattern	Response
0	Greetings	Hello	Hello
1	Greetings	Hi	Hello there!
2	Greetings	Greetins	Hi
3	Greetings	Whats up	I am good. How are you?
4	Greetings	Hey	Hello
...
1021	Insertion Sort tutorial	Insertion sort algorithm in Python tutorial	English:\nhttps://www.youtube.com/watch?v=IEA3...
1022	Insertion Sort tutorial	Insertion sort algorithm in Python bangla tut...	English:\nhttps://www.youtube.com/watch?v=IEA3...
1023	Insertion Sort tutorial	Insertion Sort in Python bangla tutorial	English:\nhttps://www.youtube.com/watch?v=IEA3...
1024	Insertion Sort tutorial	insertion sort algorithm bangla tutorial	English:\nhttps://www.youtube.com/watch?v=IEA3...
1025	Insertion Sort tutorial	Insertion Sort tutorial	English:\nhttps://www.youtube.com/watch?v=IEA3...

1026 rows × 3 columns

Figure 4.4: Dataset from Google-sheet

4.4.2 Preparing Dataset for Preprocessing

After the dataset has been imported, a series of operations must be performed to make it ready for pre-processing.

Eliminating empty rows

Our dataset contains a number of empty rows. During the process of making modifications to the data, one of the jobs that must be completed is the removal of any null rows from the dataset. If these rows are included in the dataset, the accuracy and performance of any machine learning system will decline. Before any machine learning approach can be properly applied to a dataset that contains null rows, it is essential that all null rows be deleted. This is a precondition for all machine learning methods [4]. We began with 1026 rows prior to removing the null rows. After removing rows containing null values from our dataset, we are left with 1006 rows (Figure-4.5).

	Tag	Pattern	Response
0	Greetings	Hello	Hello
1	Greetings	Hi	Hello there!
2	Greetings	Greetins	Hi
3	Greetings	Whats up	I am good. How are you?
4	Greetings	Hey	Hello
...
1021	Insertion Sort tutorial	Insertion sort algorithm in Python tutorial	English:\nhttps://www.youtube.com/watch?v=IEA3...
1022	Insertion Sort tutorial	Insertion sort algorithm in Python bangla tut...	English:\nhttps://www.youtube.com/watch?v=IEA3...
1023	Insertion Sort tutorial	Insertion Sort in Python bangla tutorial	English:\nhttps://www.youtube.com/watch?v=IEA3...
1024	Insertion Sort tutorial	insertion sort algorithm bangla tutorial	English:\nhttps://www.youtube.com/watch?v=IEA3...
1025	Insertion Sort tutorial	Insertion Sort tutorial	English:\nhttps://www.youtube.com/watch?v=IEA3...

1006 rows × 3 columns

Figure 4.5: Eliminate empty rows

Reseting row numbers after eliminating null rows

Following the removal of empty rows from our dataset, the row numbers in our dataset will need to be reset. Figure - 4.5 clearly demonstrates that the index for the very last row is 1,025. However, there are 1,006 rows overall here. After removing the empty rows from our dataset, the row number must be reset in order to avoid this issue. Following the resetting of the row numbers, there are 1,006 rows remaining, with the last row having an index of 1,005 (Figure-4.6).

	Tag	Pattern	Response
0	Greetings	Hello	Hello
1	Greetings	Hi	Hello there!
2	Greetings	Greetins	Hi
3	Greetings	Whats up	I am good. How are you?
4	Greetings	Hey	Hello
...
1001	Insertion Sort tutorial	Insertion sort algorithm in Python tutorial	English:\nhttps://www.youtube.com/watch?v=IEA3...
1002	Insertion Sort tutorial	Insertion sort algorithm in Python bangla tut...	English:\nhttps://www.youtube.com/watch?v=IEA3...
1003	Insertion Sort tutorial	Insertion Sort in Python bangla tutorial	English:\nhttps://www.youtube.com/watch?v=IEA3...
1004	Insertion Sort tutorial	insertion sort algorithm bangla tutorial	English:\nhttps://www.youtube.com/watch?v=IEA3...
1005	Insertion Sort tutorial	Insertion Sort tutorial	English:\nhttps://www.youtube.com/watch?v=IEA3...

1006 rows × 3 columns

Figure 4.6: Reset row numbers after eliminating null rows

Creating Response and Pattern Set

After eliminating the null rows, we get two distinct sets for response and pattern. This is an essential pre-processing step. We extract the pattern and its tag from the dataset and constructed qSet (Figure-4.7). qSet will be utilized to train our model. Using aSet (Figure-4.8), we separate the answers and its corresponding tags. This will be implemented in the prototype.

	Tag	Pattern
0	Variables Defination	What is the defination of Veriables?
1	Advantage Disadvantage of Dictionaries	Tell me Advantages of Dictionary 🧑‍🔬
2	numeric datatype classification	How many data types are in there in python?
3	Data Type Conversion	How to do converting of One data type to another?
4	%Operator	Can someone give the mod example in Python? 😊 😊
5	Linear search list	Can linear search be used on an unsorted list?...
6	Sequential search working technique	Does sequntial search can work with unsorted d...
7	Sorting	How can I fix 'Need to sort Python list on the...
8	Sort and replace list item	How do I replace all items in a list with thei...
9	count() method	how to count, how many times a value exist in ...

Figure 4.7: qSet

	Tag	Response
0	Greetings	Hello
1	Greetings	Hello there!
2	Greetings	Hi
3	Greetings	I am good. How are you?
4	Greetings	How are you?
...
249	tuple tutorial	English:\nhttps://www.youtube.com/watch?v=GstQ...
250	String tutorial	English:\nhttps://www.youtube.com/watch?v=Ctqi...
251	Function tutorial	English:\nhttps://www.youtube.com/watch?v=NSbO...
252	Dictionary tutorial	English:\nhttps://www.youtube.com/watch?v=2IsF...
253	Insertion Sort tutorial	English:\nhttps://www.youtube.com/watch?v=IEA3...

254 rows × 2 columns

Figure 4.8: aSet

4.4.3 Case Folding

Lower casing refers to the process of converting the text of a sentence or paragraph to lowercase. In the first stage of the pre-processing phase, we will convert all the values in our dataset to lowercase. There are no constraints on the data entry format. The data entry field accepts both capital and lowercase characters [5]. Therefore, if we do not use letters in the same case, the same words may sound differently, which might create confusion. We will need to use lower case to discover a solution to this challenge. By analysing Figure-4.9, we can get an idea of what the lowercase version of the dataset will look like. The term “pattern_lower” will be transformed to “pattern” after the usage of lower case. Our dataset will now adhere to the new pattern, which is all lowercase.

	Tag	Pattern	Pattern_lower
0	Variables Defination	What is the defination of Variables?	what is the defination of variables?
1	Advantage Disadvantage of Dictionaries	Tell me Advantages of Dictionary 🧨	tell me advantages of dictionary 🧨
2	numeric datatype classification	How many data types are in there in python?	how many data types are in there in python?
3	Data Type Conversion	How to do converting of One data type to another?	how to do converting of one data type to another?
4	%Operator	Can someone give the mod example in Python? 😊😊	can someone give the mod example in python? 😊😊

Figure 4.9: Case Folding

4.4.4 Punctuation Removal

After the lower case conversion is complete, the punctuation marks must be removed from the texts. The patterns in the collection may or may not include any punctuation at all depending on their particular design. It is necessary to eliminate any punctuation that may be included in the patterns in order to ensure that the patterns do not include any punctuation [7]. When we remove the punctuation from the patterns in Figure-4.10, we are able to observe how the patterns appeared before and after the removal of the punctuation. After removing the punctuation, we will now begin working with the modified patterns that we have obtained.

	Tag	Pattern	text_wo_punct
0	Variables Defination	what is the defination of variables?	what is the defination of variables
1	Advantage Disadvantage of Dictionaries	tell me advantages of dictionary 🧨	tell me advantages of dictionary 🧨
2	numeric datatype classification	how many data types are in there in python?	how many data types are in there in python
3	Data Type Conversion	how to do converting of one data type to another?	how to do converting of one data type to another
4	%Operator	can someone give the mod example in python? 😊😊	can someone give the mod example in python? 😊😊

Figure 4.10: Punctuation Removal

4.4.5 Removal of Stopwords

After the punctuation has been eliminated, we will next eliminate the stop words. Stop words are any words included inside a stop list that are eliminated either before to or after the processing of natural language data[6]. There might be a large number of stopwords in the pattern of our dataset, or none at all. Stopwords are insignificant or almost insignificant words. Therefore, we must eliminate these words. Figure-4.11 illustrates the approach for removing stop words from the patterns contained in our dataset.

	Tag	Pattern	text_wo_stop
0	Variables Defination	what is the defination of variables	defination variables
1	Advantage Disadvantage of Dictionaries	tell me advantages of dictionary 🧯	tell advantages dictionary 🧯
2	numeric datatype classification	how many data types are in there in python	many data types python
3	Data Type Conversion	how to do converting of one data type to another	converting one data type another
4	%Operator	can someone give the mod example in python 😊 😊	someone give mod example python 😊 😊

Figure 4.11: Removal of stopwords

4.4.6 Stemming

Model training will be more successful if the redundant words in the patterns are eliminated, after deleting the stop words. Text normalization can be improved by reducing the inflection of words to their root forms using the natural language processing approach called stemming. Typically, the term “stemming” refers to a rudimentary heuristic procedure that removes derivational affixes from words in an attempt to achieve this aim as accurately as possible. Consequently, stemming is used to reduce words to their fundamental form or stem, which may or may not be a genuine word in the language [40]. Examples include “connect” as the root of the following words: connections, connected and connected. When it comes to “troubl”, the root of “troubles”, “troubled” and “trouble” is not a recognized word. The technique we use while stemming our patterns is depicted in (Figure-4.12). Our new revised pattern will be stemmed-text when it has been stemmed.

	Tag	Pattern	text_stemmed
0	Variables Defination	defination variables	defin variabl
1	Advantage Disadvantage of Dictionaries	tell advantage dictionary	tell advantag dictionari
2	numeric datatype classification	many data type python	mani data type python
3	Data Type Conversion	converting one data type another	convert one data type anoth
4	%Operator	someone give mod example python	someon give mod exampl python

Figure 4.12: Stemming

4.4.7 Lemmatization

Lemmatization is the next step after stemming. Although lemmatization is derived from the term, its meaning is preserved. As the word’s meaning diminishes, a predefined dictionary used by lemmatization keeps track of the word’s meaning [8]. While Lemmatization clearly recognizes “troubled” as the base form of “trouble”, Stemming removes the “ed” element and converts it into “troubl”, which has a

different connotation and spelling problems. This is an illustration of the difference between Lemmatization and Stemming. Our data is subjected to stemming and lemmatization in order to extract all of the words' roots. Our model training will benefit greatly from this (Figure-4.13)

	Tag	Pattern	text_lemmatized
0	Variables Defination	defination variables	defination variables
1	Advantage Disadvantage of Dictionaries	tell advantages dictionary 🍌	tell advantage dictionary 🍌
2	numeric datatype classification	many data types python	many data type python
3	Data Type Conversion	converting one data type another	converting one data type another
4	%Operator	someone give mod example python 😊😊	someone give mod example python 😊😊

Figure 4.13: Lemmatization

4.4.8 Emoji Removal

Text pre-processing is a critical step when working with Natural Language Processing (NLP). In the text cleaning phase, we must do a variety of text preprocessing operations, such as handling stop words, special characters, emoji, emoticons, punctuation, spelling correction, URL, etc. This stage involves removing emojis from our pattern in order to make the data both clean and informative (Figure-4.14).

	Tag	Pattern	text_emojiRemoved
0	Variables Defination	defination variables	defination variables
1	Advantage Disadvantage of Dictionaries	tell advantage dictionary 🍌	tell advantage dictionary
2	numeric datatype classification	many data type python	many data type python
3	Data Type Conversion	converting one data type another	converting one data type another
4	%Operator	someone give mod example python 😊😊	someone give mod example python

Figure 4.14: Emoji Removal

4.5 Fitting dataset on basic classifiers

After all of the pre-processing, we must now evaluate how well our dataset responds to user inputs. To determine this, we utilized four distinct classifiers and compared their respective accuracy gains. After evaluating the accuracy, we will be able to predict the accuracy of the neural model of our prototype. Our chosen classifiers for training and testing are the Naive Bayes Classifier, the Decision Tree Classifier, the Linear Support Vector Machine and the Logistic Regression classifier.

Initially, we conducted tests on the entire dataset by dividing it 8:2 for training and testing purposes. As input, we use the patterns to anticipate the tag. Tags are assigned to every comparable question pattern.

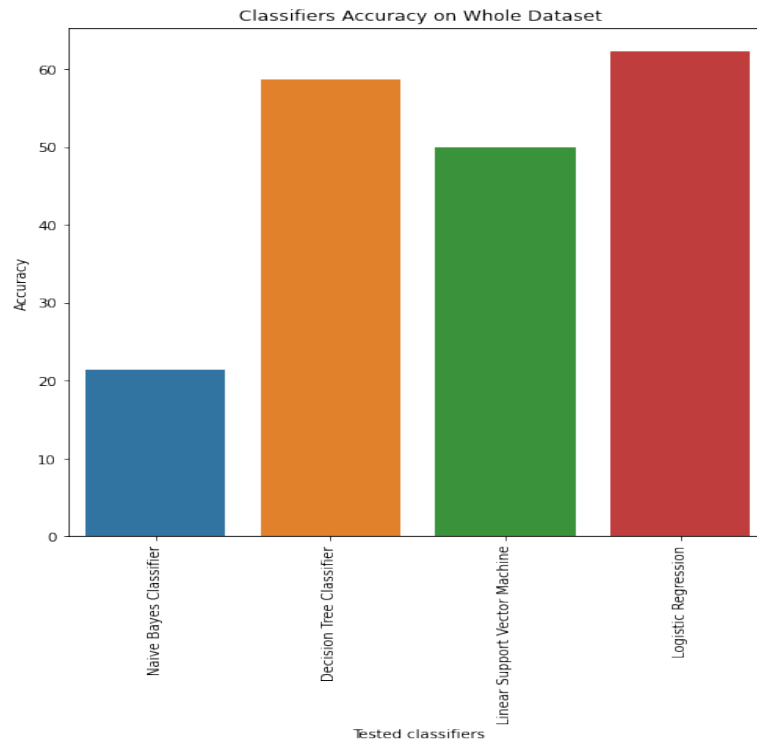


Figure 4.15: Classifiers accuracy on whole dataset

After training and evaluating the models with the entire dataset, different classifiers achieve varying degrees of accuracy. In Figure-4.15, we can see the accuracy of all classifiers over the entire dataset. The accuracy for Naive Bayes Classifier is 21.42 percent, Decision Tree Classifier is 58.67 percent, Linear Support Vector Machine is 50 percent and Logistic Regression is 62.24 percent. On the entire data set, Logistic Regression provides the highest level of accuracy, 62.24 percent.

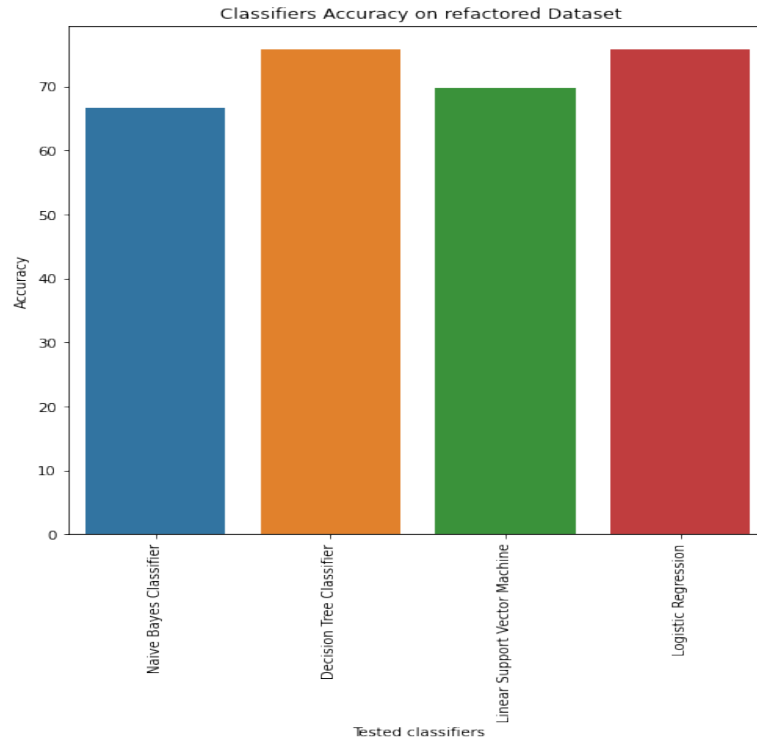


Figure 4.16: Classifiers accuracy on refactored dataset

We are not entirely satisfied with the overall accuracy of the entire dataset. Due to this, we omitted certain tags with less training patterns. We refactored data while preserving the patterns of tags with at least 10 patterns. On rerunning these classifiers on the refactored data, the accuracy of each classifier improves. Figure-4.16 depicts the classification accuracy of each classifier for the refactored dataset. Now, the accuracy of Naive Bayes Classifier is 66.67 percent, Decision Tree Classifier is 75.75 percent, Linear Support Vector Machine is 69.70 percent and Logistic Regression is 75.75 percent. On the refactored dataset, Logistic Regression and Decision Tree Classifier get the highest accuracy, 75.75 percent.

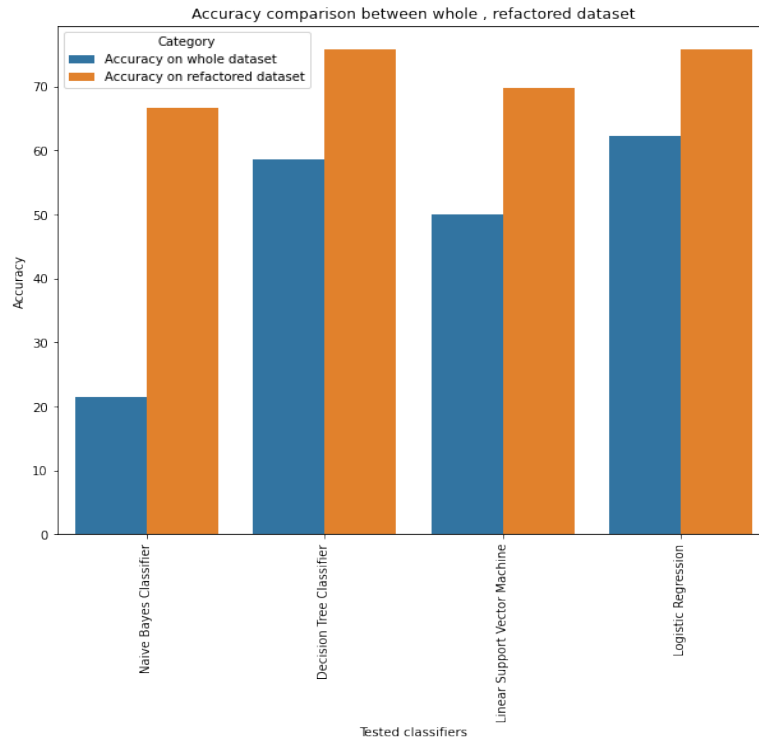


Figure 4.17: Accuracy comparison between whole, refactored dataset

From the comparison in Figure-4.17, we can observe that the refactored dataset has greater accuracy than the original dataset. For the refactored dataset the accuracy of the Naive Bayes Classifier increases by 17.08 percent, 19.7 percent for the Linear Support Vector Machine and 13.51 percent for the Logistic Regression algorithm. This is because the refactored dataset contains tags with a greater number of patterns, which helps classifiers train more effectively, resulting in improved accuracy.

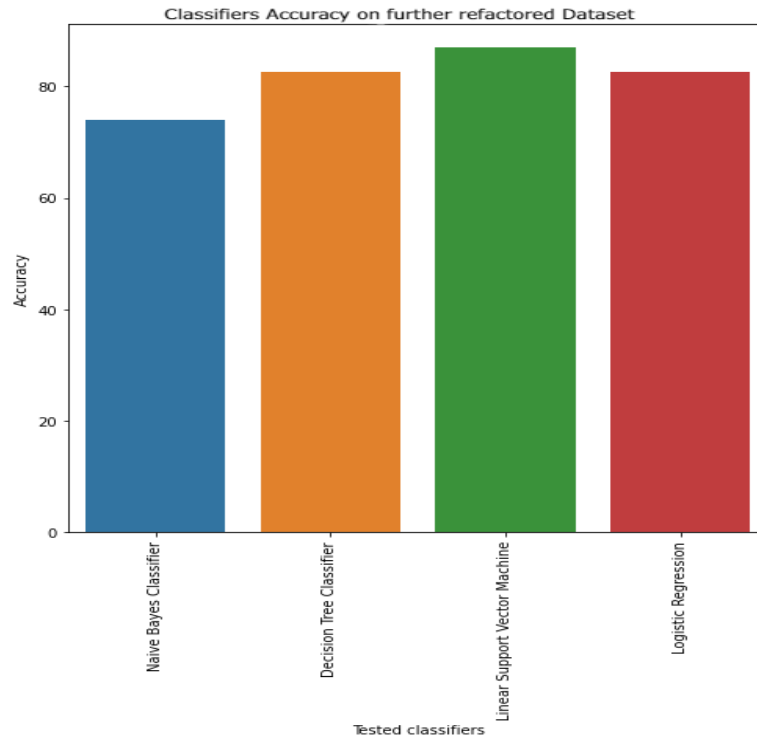


Figure 4.18: Classifiers accuracy on further refactored dataset

To get a clearer idea, we rerun all four classifiers using a further refactored dataset to determine if eliminating tags with fewer patterns yields a substantially higher degree of precision. Now the dataset contains only tags with at least 10 patterns. We then rerun these classifiers on the newly refactored data and observe a significant increase in accuracy across the board. Figure-4.18 depicts the classification accuracy of all classifiers for the refactored dataset. Now, the accuracy for Naive Bayes Classifier is 73.91 percent, Decision Tree Classifier is 78.26 percent, Linear Support Vector Machine is 86.95 percent, and Logistic Regression is 82.60 percent. With the further refactored dataset, we achieve the highest accuracy with Linear Support Vector Machine and Decision Tree Classifier, 86.95 percent and 82.60 percent, respectively.

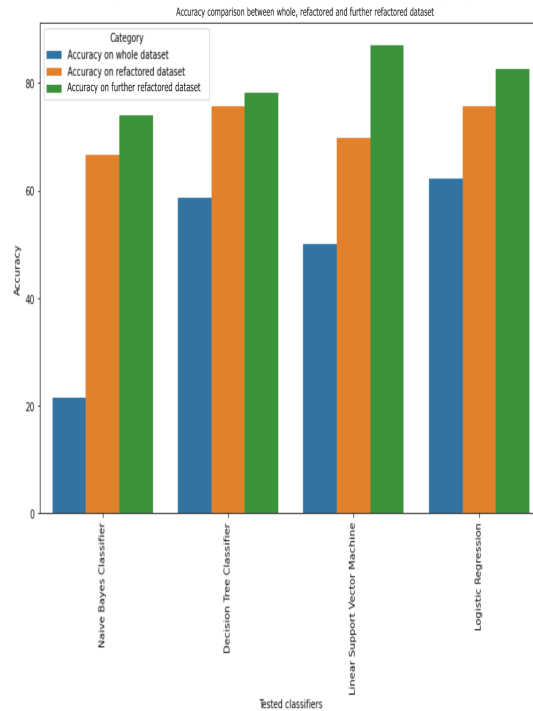


Figure 4.19: Accuracy comparison between whole, refactored and further refactored dataset

Comparing the accuracy of all classifiers on the entire dataset, the refactored dataset, and the further refactored dataset in Figure-4.19, we can see that the more tags with a higher number of patterns we add, the more balanced our dataset becomes. After testing the accuracy, we can anticipate that the neural model of our prototype will have an accuracy of more than 90 percent. Our goal is to increase the accuracy of the entire dataset in the future by incorporating additional patterns for each tag, with the assistance of relevant faculties, student tutors and also ourselves.

F1-Score

In model training, F1-score is one of the most important evaluation metrics. It elegantly summarizes the predictive performance of a model by combining precision and recall.

Initially, we test our four classifiers on the entire dataset and obtain an average f1-score (Figure-4.20) of 0.16 for the Naive Bayes classifier, 0.56 for the Decision Tree classifier, 0.46 for the Linear Support Vector Machine(SVM) and 0.60 for the Logistic regression.

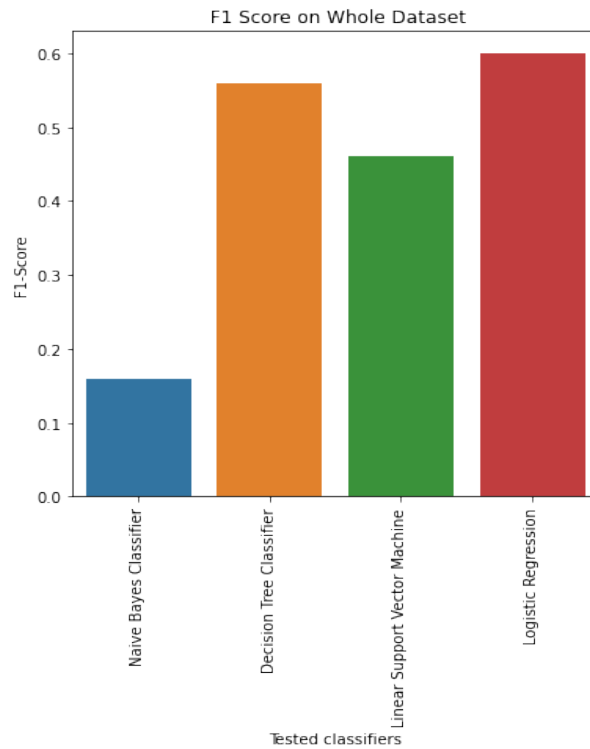


Figure 4.20: F1 Score on Whole Dataset

Second, we retest all four classifiers on the refactored dataset and obtain an average f1-score (Figure-4.21) of 0.64 for the Naive Bayes classifier, 0.75 for the Decision Tree classifier, 0.65 for the Linear Support Vector Machine (SVM) and 0.75 for the Logistic regression.

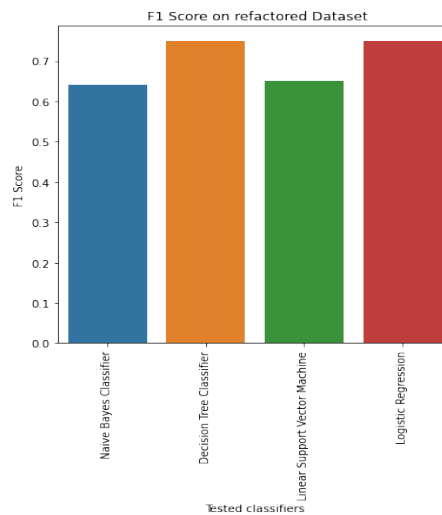


Figure 4.21: F1 Score on refactored Dataset

When we finally retest all four classifiers on our further refactored dataset, we obtain an average f1-score (Figure-4.22) of 0.71 for the Naive Bayes classifier, 0.82 for the Decision Tree classifier, 0.85 for the Linear Support Vector Machine (SVM) and 0.81 for the Logistic regression classifier [10].

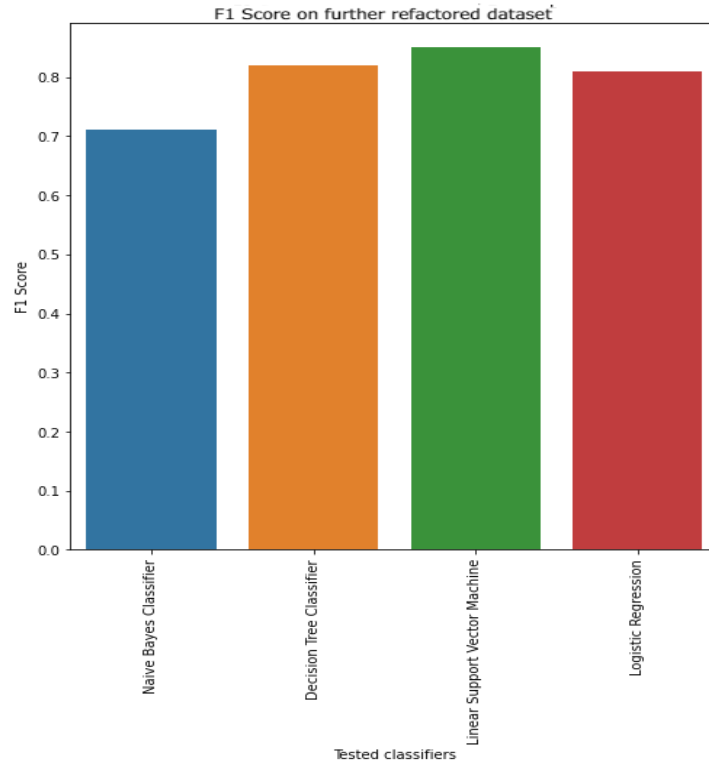


Figure 4.22: F1 Score on further refactored Dataset

Chapter 5

System Design

5.1 System Architecture

We implemented our VTA-Bot system Figure-5.1 as a web application. As a result, students have easier access from their mobile devices and desktop browsers.

5.1.1 User Interface

The user interface is designed with only the most essential functions. There is an input field for users to enter their queries and an output field to receive the system's response. There is also a decision module for users to select the desired functionality. They can select between conceptual and factual queries. This is essential for guiding the user in the right direction.

5.1.2 Backend Process

All processing and matching to generate a response is performed in the system's backend.

Pre-Processing

In the backend of our system, received data is preceded by preprocessing. As users can provide any type of unwanted data as input, we will need to process these inputs and extract only the required data. We perform pre-processing on user input by first tokenizing it, then removing stop words, and finally stemming and lemmatizing all remaining inputted values.

Tier-2 Processing

After preprocessing is complete, we do the word-embedding and place it in a bag of words. We call it tier-2 processing.

Matching and extracting data from server

Using our neural network model, we compare the processed data with our dataset stored on the database server following all processing.

Generating Output

After comparing the processed data to our dataset, we produce the most optimally matched output. This output is then displayed on the user interface.

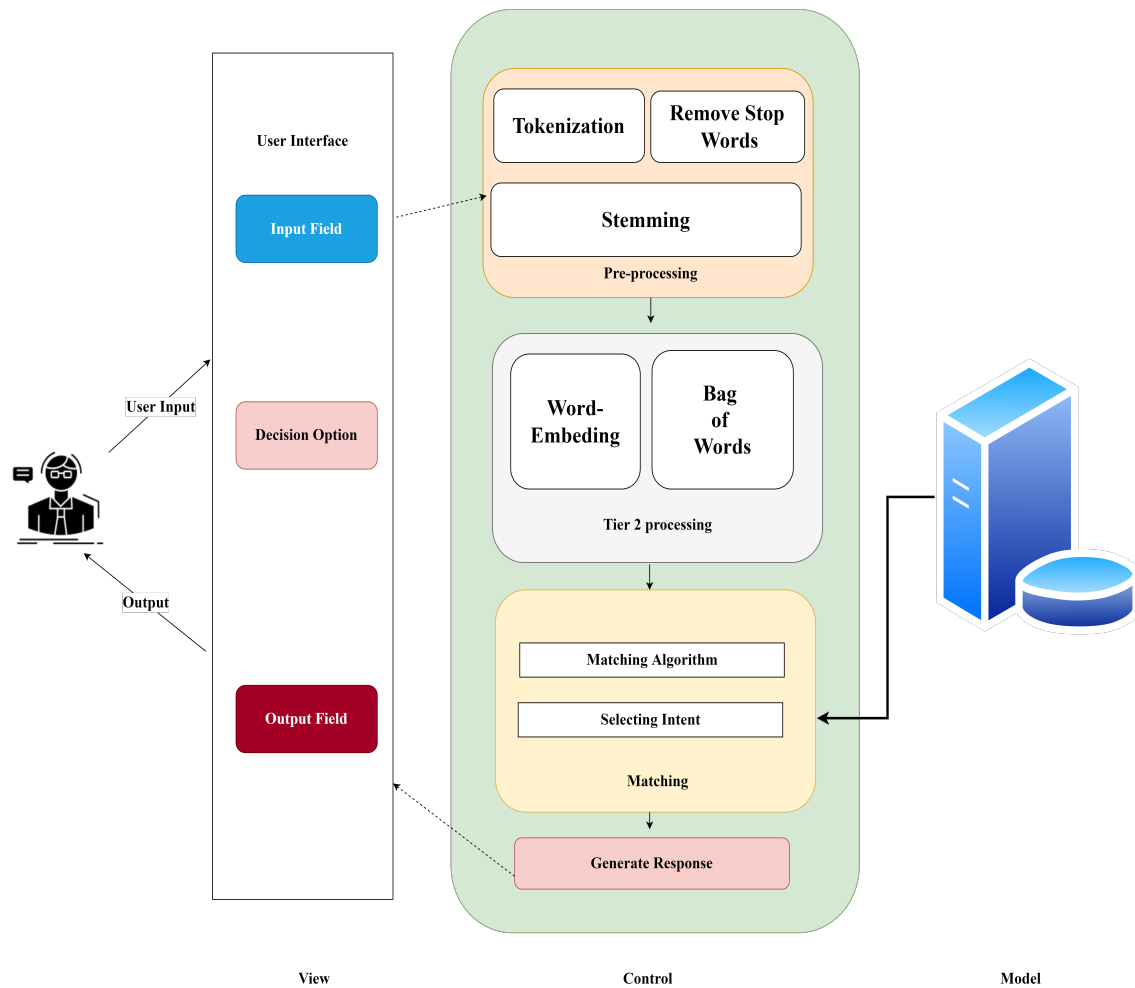


Figure 5.1: System Architecture

5.2 Design Data Flow

VTA-bot has a different support mechanism. To begin, students will choose either course-related support or mentorship-related support. The mentorship support will provide basic course guidelines, suggestions for further courses and live support appointments (Figure-5.2). On the contrary, the course-related section has four subsections:

5.2.1 Conceptual Queries

VTA-bot provides sequential concept-related support through conversation with the flexibility of choosing a particular concept or subsequent concept or redirecting to the dedicated live support appointment.

5.2.2 Factual Queries

This section contains all deadlines, faculty or TA contact information, quizzes and assignment marks.

5.2.3 Predefined Questions

Frequently asked questions with solutions are included, as well as instructions for setting up a live-support session.

5.2.4 Support Seeking

Dedicated support area to schedule live discussions with faculty or TA.

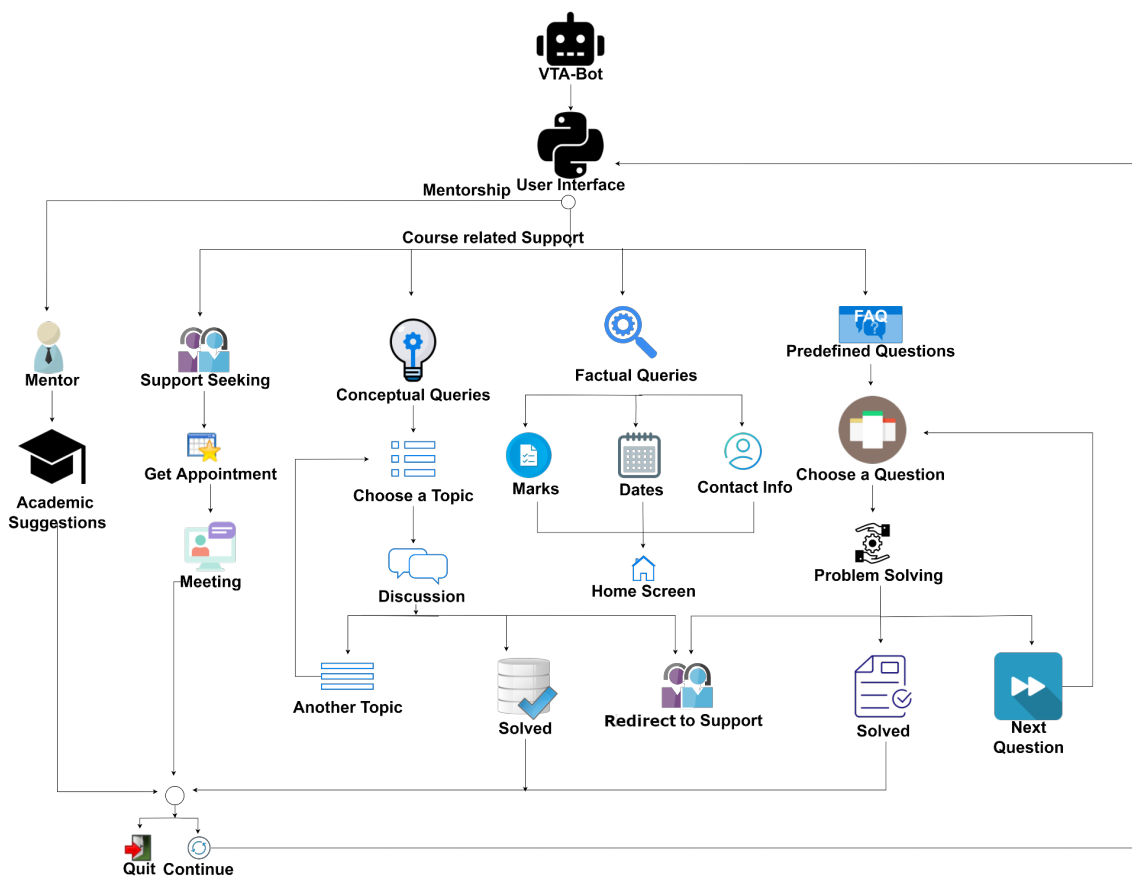


Figure 5.2: Design Data Flow

Chapter 6

Prototype Development

6.1 NLP Techniques

We cannot just feed the input phrase to the neural network in its existing form. We need to translate the pattern strings into numbers that the network can interpret in some way. This is accomplished by transforming each statement into a “bag of words”. To do this, we collected training words, or all the terms in the training data that our VTA-bot can look at. The bag of words for each new sentence was then calculated using all of these keywords. A bag of words has the same size as the “all words” array (Figure-6.1), and each slot contains a 1 if the word appears in the incoming phrase, or a 0 if it does not. Prior to calculating the bag of words, we used two more natural language processing techniques: Tokenization and Stemming (Figure-6.2).

- **Tokenization:** It splits a string into meaningful units (e.g. words, punctuation characters, numbers)
- **Stemming:** It creates the word’s root form. It is an imprecise heuristic that removes the endings of words.

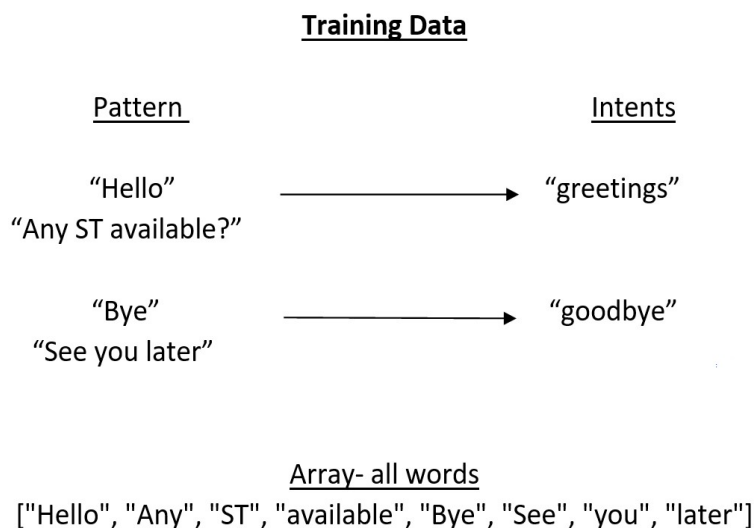


Figure 6.1: Data preprocessing

Our NLP Preprocessing Pipeline

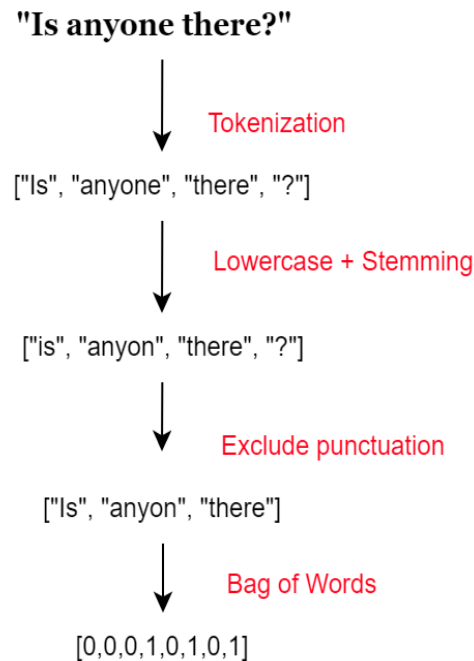


Figure 6.2: NLP Techniques

6.2 The Neural Network Model

A MLF neural network consists of neurons that are ordered into layers (Figure-6.3). The first layer is called the input layer, the last layer is called the output layer, and the layers between are hidden layers. A Feed Forward Neural Net (Figure-6.3) with two hidden layers is used to create our neural network model. Training and prediction are the two modes of operation supported by the Feed Forward neural network. Two data sets are required for the training and prediction of this neural network: the training set and the set to be predicted (test set). It takes the input, passes it through various layers one by one, and then finally gives the output. This feed-forward Neural Network is constructed using the “torch.nn” package. The “torch.nn” is a python package provided by “PyTorch” that assisted us in developing and training the neural network.

Neural Network Model

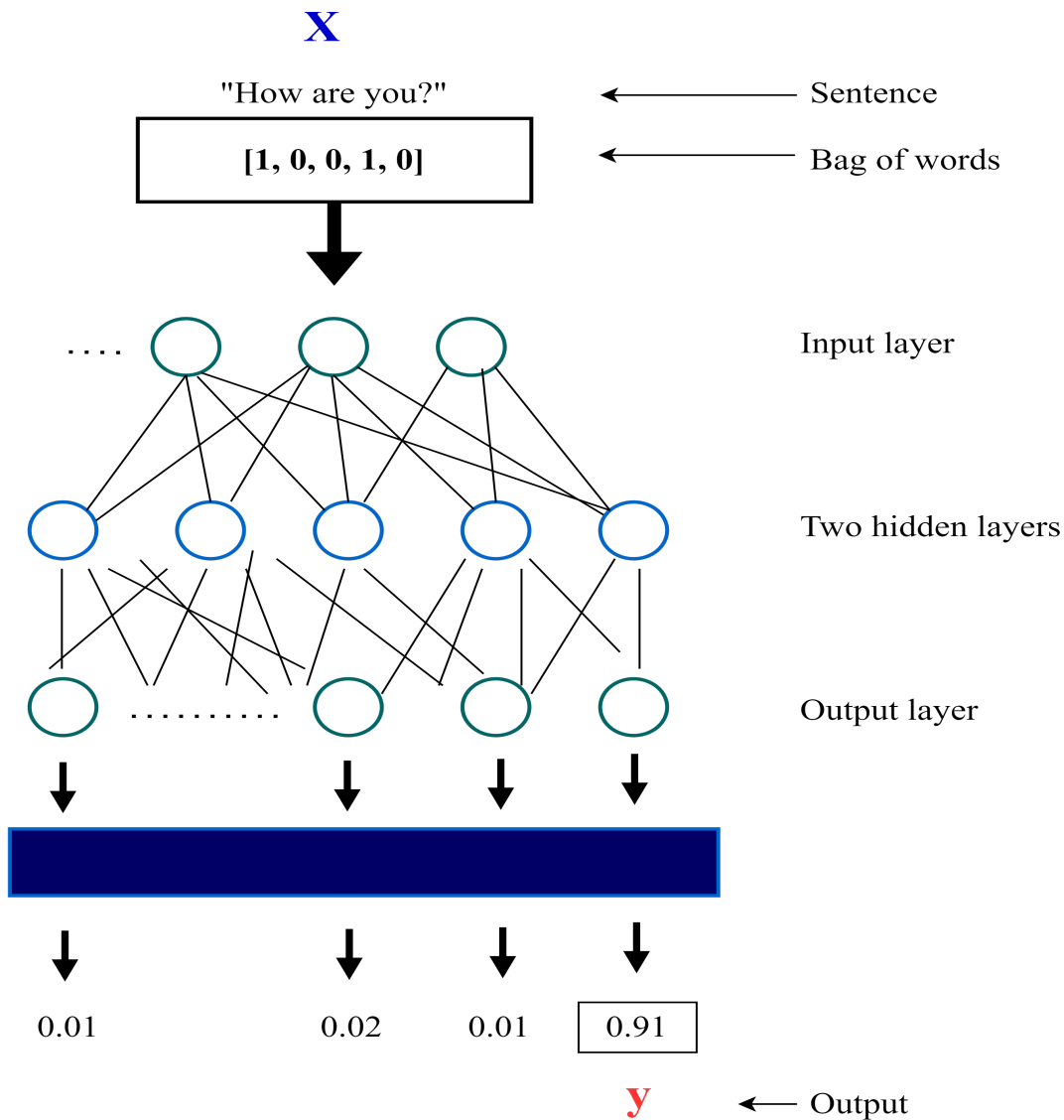


Figure 6.3: Feed Forward Neural Network

6.3 Prototype Implementation

Our VTA-bot is developed utilizing the Python programming language and deep learning algorithms. We started by installing PyTorch (6.1) and NLTK (6.1). We then prepared a JSON file containing a training data set. Afterwards, we constructed the NLP Pipeline utilizing the NLTK package of python. For this, the NLTK module was utilized. We implemented the model of a Neural Network. The torch.nn module offered by PyTorch aided in the creation and training of the neural network model [1]. Then, we designed the Training Pipeline, which is comprised of a series of sequential phases that cover everything from data gathering and data pre-processing through training the model and distribution. We developed the virtual teaching assistant (VTA-bot) by loading the learned model and predicting new phrases. According to the course materials, we may alter the JSON dataset file with potential patterns and replies and rerun the training.

Libraries And Function	Operations
PyTorch	An open source machine learning framework that accelerates the path from research prototyping to production deployment.
Torch.nn	help us in creating and training of the neural network.
Nltk	a toolkit build for working with NLP in Python.
nn.ReLU()	the activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input.
optimizer.zero_grad()	clears old gradients from the last step.
criterion(outputs, labels)	expects a class index (1 to the number of class) as target when calling forward(input, target) and backward(input, target).
loss.backward()	the whole graph is differentiated w.r.t. the loss, and all Variables in the graph will have their .grad Variable accumulated with the gradient.

Table 6.1: Different Libraries and Functions Used

6.4 Prototype Training and Testing

We load the dataset, which is in JSON format and after performing proper NLP techniques we generate the bag words for every pattern in the dataset . We trained our model with the updated data. We set the batch size to 8 and the number of epochs to 1,000. Then we calculated the loss and accuracy in each 100 epoch. The number of epochs is a hyper-parameter that controls how many times the learning algorithm runs over the whole training dataset. Loss is the error over the training set. A loss function takes the (output, target) pair of inputs and computes a value that estimates how far away the output is from the target. Throughout the training loop, in every epoch the loss is decreasing and the accuracy is increasing. In the Figure-6.4 we can see that, in the first iteration the loss was 1.1089. After our training concluded, our loss was effectively reduced to 0.0007. Again, in the Figure-6.5 , we can see the accuracy increased to 95%. Training the network is repeated for a large number of training set examples until the network finds a stable state. The most typical strategy for avoiding over-fitting is stopping early. Early stopping is based on separating data into training and validation sets and computing the validation error regularly during training. Training is terminated when the number of validation errors begins to rise. In the Figure-6.6 we can see our VTA-bot is assisting a student.

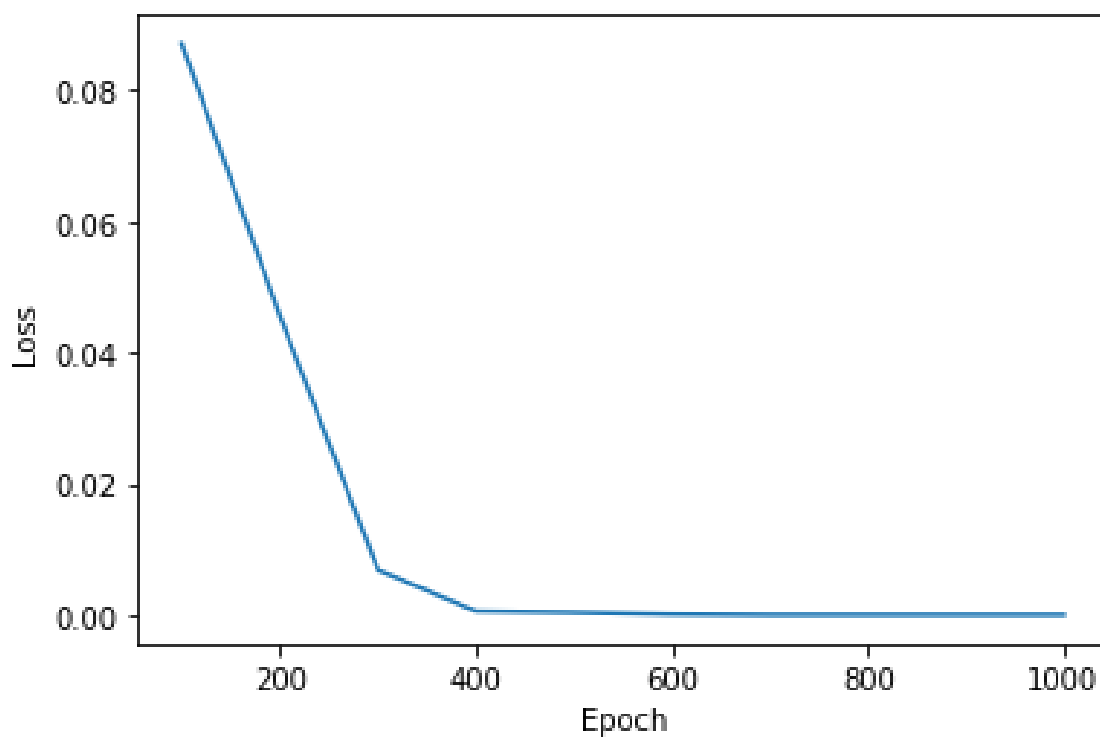


Figure 6.4: Model Training Loss Curve

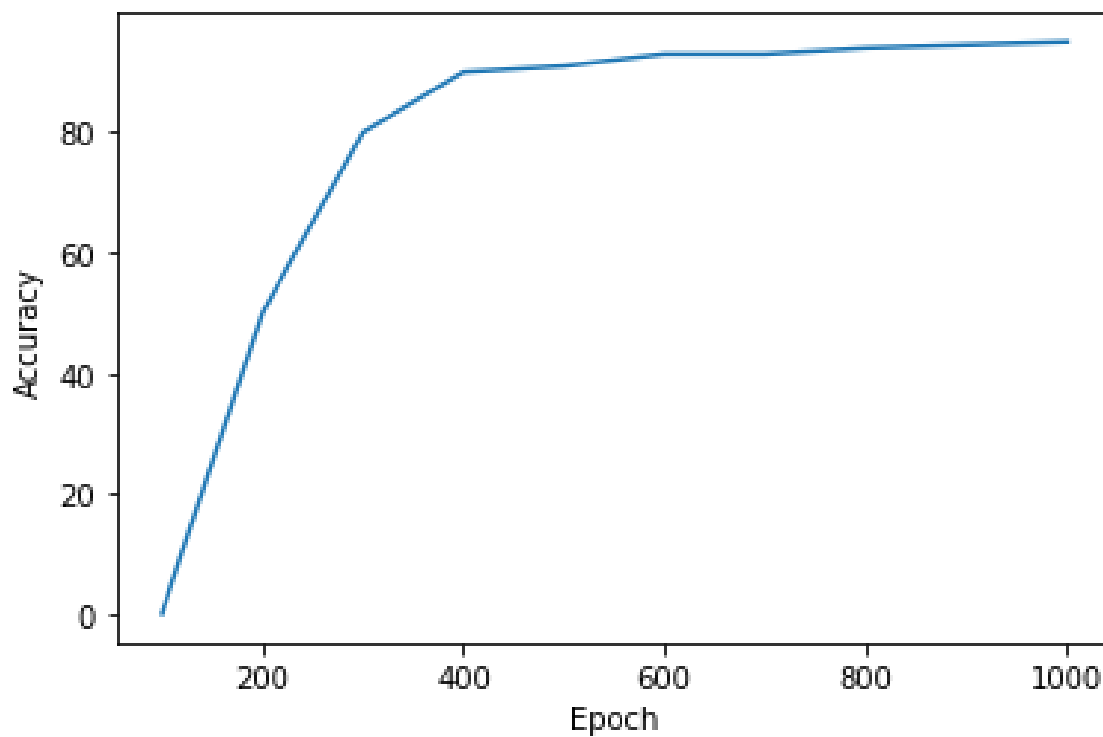


Figure 6.5: Accuracy Curve

```
Welcome to VTA-bot      (type 'quit' to exit)

You: What is tuple?

VTA-bot: Tuple is a collection of values separated by comma and enclosed in parenthesis. Unlike lists, tuples are immutable. The immutability can be considered as the ident.

You: What is tuple?

VTA-bot: A tuple is a sequence of elements of any type and is immutable. In python, they are initialized inside parenthesis '()' instead of square brackets '[]'.To create a
Code:
example_tuple = ('Banana', 'Mango', 'Apple', 'Orange','Grape','Jackfruit')
for fruit in example_tuple:
    print(fruit)
Output:
Banana
Mango
Apple
Orange
Grape

You: tuple ki?

VTA-bot: A tuple is a collection of objects which ordered and immutable. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples can

You: tuple amake bujhaw

VTA-bot: Tuples are used to store multiple items in a single variable. A tuple is a collection which is ordered and unchangeable. Tuples are written with round brackets.

You: tuple vs list

VTA-bot: 1.List is mutable but tuples are immutable
2.List iteration is slower and time-consuming but tuples iteration is faster.
3.List is performed well in insertion and deletion operations but tuples are performed well in accessing elements operations.
4.List takes more memory but Tuples take less memory.

You: capital of Bangladesh?
VTA-bot: I do not understand...
You: quit
```

Figure 6.6: Conversation Between VTA-bot and Student

Chapter 7

Future Work

This paper proposed a dataset and an architecture for a VTA-bot capable of offering several services to students enrolled in online and on-site courses, with “assisted learning” as the ultimate goal. In addition, a first implementation of such a system was introduced, consisting of a chatbot able to respond to the inquiries of students enrolled in “Programming Language I”. This chatbot is only a small portion of the VTA system we are proposing, and we are already developing other services for the system. By monitoring the interactions between students and the VTA-bot using knowledge extraction methods, we plan to continue refining and enhancing the model even after it has been deployed. We also intend to add student engagement: by analyzing student behavior, the VTA-bot will be able to determine which students are facing trouble with the course contents and which are dissatisfied with the response; it may then act proactively or send a warning to a human teacher. Again, the addition of new courses is something we want to do in the near future. Lastly, ongoing research is centered on the prospect of personalizing learning materials by recommending various contents based on the student’s interactions with the VTA-bot.

Chapter 8

Conclusion

Students are sometimes hesitant to contact with strangers, and due to the newly established online-based education system, they get disoriented when transitioning to new programming concepts and lack adequate supervision. Our goal in writing this paper was to introduce our VTA-bot as a medium of learning to make their educational experiences more enjoyable and reduce their chances of failing. Along with answering core python programming questions, our VTA-bot will be able to assist students with course-related concerns and provide real-time assistance from their teachers. We demonstrated a preliminary implementation, which consists of a Chatbot that will assist and support students through the use of artificial intelligence (AI) technologies, including natural language processing (NLP). To help students even more, we intend to continue working on improving our system in the years to come, hoping for a revolution in our education sector.

Bibliography

- [1] F. A. R. lab (FAIR). “Pytorch.” (), [Online]. Available: <https://pytorch.org/>. (accessed: 15.10.2021).
- [2] G. Cloud. “Intents.” (), [Online]. Available: <https://cloud.google.com/dialogflow/es/docs/intents-overview>. (accessed: 15.10.2021).
- [3] D. Jovic. “The future is now - 37 fascinating chatbot statistics.” (), [Online]. Available: <https://www.smallbizgenius.net/by-the-numbers/chatbot-statistics/>. (accessed: 15.10.2021).
- [4] N. Kumar. “Data wrangling: Removing null values from dataset in python using pandas library.” (), [Online]. Available: <http://theprofessionalspoint.blogspot.com/2019/03/data-wrangling-removing-null-values.html>. (accessed: 9.3.2022).
- [5] S. R. Kumar. “Getting started with text preprocessing.” (), [Online]. Available: <https://www.kaggle.com/code/sudalairajkumar/getting-started-with-text-preprocessing/notebook?fbclid=IwAR2YkpAEgIVtPuNa5543LU4IILjeFEY8kzk--6SbT8zk0Rvi1LeuZSkFtZs>. (accessed: 9.3.2022).
- [6] U. Malik. “Removing stop words from strings in python.” (), [Online]. Available: <https://stackabuse.com/removing-stop-words-from-strings-in-python/>. (accessed: 10.3.2022).
- [7] V. Rani. “Nlp tutorial for text classification in python.” (), [Online]. Available: <https://medium.com/analytics-vidhya/nlp-tutorial-for-text-classification-in-python-8f19cd17b49e>. (accessed: 9.3.2022).
- [8] “Stemming & lemmatization.” (), [Online]. Available: <https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html>. (accessed: 4.3.2022).
- [9] Wikipedia. “Apprenticeship learning.” (), [Online]. Available: https://en.wikipedia.org/wiki/Apprenticeship_learning. (accessed: 15.10.2021).
- [10] J. Cabero-Almenara, F. D. Guillén-Gámez, J. Ruiz-Palmero, and A. Palacios-Rodríguez, “Teachers’ digital competence to assist students with functional diversity: Identification of factors through logistic regression methods,” *British Journal of Educational Technology*, vol. 53, no. 1, pp. 41–57, 2022.
- [11] N. N. P. Anh and H. T. Ngan, “Artificial intelligence in mathematics education: An empirical study of using chatbot in teaching and learning mathematics at vietnamese high schools,” *5th ASIA PACIFIC International Modern Sciences Congress*, 2021.
- [12] V. GAUR *et al.*, “Chat-bot system project based on the natural language,” 2021.

- [13] A. N. Mathew, J. Paulose, *et al.*, “Nlp-based personal learning assistant for school education,” *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 11, no. 5, 2021.
- [14] Z. Memon, H. Aghian, M. S. Sarfraz, *et al.*, “Framework for educational domain-based multichatbot communication system,” *Scientific Programming*, vol. 2021, 2021.
- [15] A. Roy, D. Singh, and S. Sahana, “Educational assistance bot,” in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1797, 2021, p. 012062.
- [16] Q. Wang, K. Saha, E. Gregori, D. Joyner, and A. Goel, “Towards mutual theory of mind in human-ai interaction: How language reflects what students perceive about a virtual teaching assistant,” in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–14.
- [17] S. Abdelhamid and A. Katz, “Using chatbots as smart teaching assistants for first-year engineering students,” in *2020 First-Year Engineering Experience*, 2020.
- [18] S. H. M. Daud, N. H. I. Teo, and N. H. M. Zain, “Ejava chatbot for learning programming language: A post-pandemic alternative virtual tutor,” *International Journal*, vol. 8, no. 7, pp. 3290–3298, 2020.
- [19] M. A. Jayakumar Sadhasivam, A. B. Amitava Mazumdar, B. J.M, and V. Kumar, “Implementation of chatbot that teach programming language,” *Xi’an University of Architecture & Technology*, vol. 12, 2020.
- [20] Y. Liang, Y. Yu, and W. Ouyang, “Intelligent chat robot in digital campus based on deep learning,” in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1629, 2020, p. 012079.
- [21] C. W. Okonkwo and A. Ade-Ibijola, “Python-bot: A chatbot for teaching python programming,” *Engineering Letters*, vol. 29, no. 1, 2020.
- [22] L. Benedetto, P. Cremonesi, and M. Parenti, “A virtual teaching assistant for personalized learning,” *arXiv preprint arXiv:1902.09289*, 2019.
- [23] S. Hobert, “Say hello to ‘coding tutor’! design and evaluation of a chatbot-based learning system supporting students to learn to program,” 2019.
- [24] V. Fernoagă, G.-A. Stelea, C. Gavrilă, and F. Sandu, “Intelligent education assistant powered by chatbots,” in *The International Scientific Conference eLearning and Software for Education*, “Carol I” National Defence University, vol. 2, 2018, pp. 376–383.
- [25] A. K. Goel and L. Polepeddi, “Jill watson: A virtual teaching assistant for online education,” Georgia Institute of Technology, Tech. Rep., 2016.
- [26] M. Al Emran and K. Shaalan, “A survey of intelligent language tutoring systems,” in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, 2014, pp. 393–399.
- [27] R. Ahmed, “Elabmate: A tool for delivering programming courses effectively,” *International Journal of Advanced Corporate Learning (iJAC)*, vol. 5, no. 3, pp. 6–11, 2012.

- [28] C.-Y. Chou, B.-H. Huang, and C.-J. Lin, “Complementary machine intelligence and human intelligence in virtual teaching assistant for tutoring program tracing,” *Computers & Education*, vol. 57, no. 4, pp. 2303–2312, 2011.
- [29] P. Barry, *Head first Python: A brain-friendly guide*. ”O’Reilly Media, Inc.”, 2010.
- [30] P. Barry and D. Griffiths, *Head First Programming: A Learner’s Guide to Programming Using the Python Language*. ” O’Reilly Media, Inc.”, 2009.
- [31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.
- [32] M. L. Hetland, *Beginning Python: from novice to professional*. Apress, 2008.
- [33] M. Summerfield, *Programming in Python 3: a complete introduction to the Python language*. Addison-Wesley Professional, 2008.
- [34] W. Chun, *Core python programming*. Prentice Hall Professional, 2007.
- [35] D. M. Beazley, *Python essential reference*. Addison-Wesley Professional, 2006.
- [36] D. Ascher and M. Lutz, *Learning Python*. O’Reilly, 2004.
- [37] A. Martelli, *Python in a Nutshell*. ” O’Reilly Media, Inc.”, 2003.
- [38] M. L. Hetland, *Practical Python*. Citeseer, 2002, vol. 648.
- [39] A. Martelli, A. Ravenscroft, and D. Ascher, *Python cookbook*. ” O’Reilly Media, Inc.”, 2002.
- [40] J. B. Lovins, “Development of a stemming algorithm.,” *Mech. Transl. Comput. Linguistics*, vol. 11, no. 1-2, pp. 22–31, 1968.
- [41] J. Weizenbaum, “Eliza—a computer program for the study of natural language communication between man and machine,” *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966.