



Faculty of Computer Science and Information Technology

***WEB-BASED ARTICLE SUMMARIZATION WITH MACHINE
LEARNING TECHNIQUES***

Lim Wu Tong

Bachelor of Computer Science with Honours (Software Engineering)

2023

UNIVERSITI MALAYSIA SARAWAK

THESIS STATUS ENDORSEMENT FORM

TITLE Web-based Article Summarization with Machine Learning Techniques

ACADEMIC SESSION: 2022/2023

LIM WU TONG

(CAPITAL LETTERS)

hereby agree that this Thesis* shall be kept at the Centre for Academic Information Services, Universiti Malaysia Sarawak, subject to the following terms and conditions:

1. The Thesis is solely owned by Universiti Malaysia Sarawak
2. The Centre for Academic Information Services is given full rights to produce copies for educational purposes only
3. The Centre for Academic Information Services is given full rights to do digitization in order to develop local content database
4. The Centre for Academic Information Services is given full rights to produce copies of this Thesis as part of its exchange item program between Higher Learning Institutions [or for the purpose of interlibrary loan between HLI]
5. ** Please tick (✓)

- | | | |
|-------------------------------------|--------------|--|
| <input type="checkbox"/> | CONFIDENTIAL | (Contains classified information bounded by the OFFICIAL SECRETS ACT 1972) |
| <input type="checkbox"/> | RESTRICTED | (Contains restricted information as dictated by the body or organization where the research was conducted) |
| <input checked="" type="checkbox"/> | UNRESTRICTED | |



(AUTHOR'S SIGNATURE)

Validated by

(SUPERVISOR'S SIGNATURE)

Permanent Address

1, TAMAN KEMUDI
JALAN KUALA KEDAH
06600 ALOR SETAR, KEDAH

Date: 01/07/2023

Date: _____

Note * Thesis refers to PhD, Master, and Bachelor Degree

** For Confidential or Restricted materials, please attach relevant documents from relevant organizations / authorities

**WEB-BASED ARTICLE SUMMARIZATION WITH MACHINE LEARNING
TECHNIQUES**

LIM WU TONG

This project is submitted in partial fulfilment of the requirements for the degree of Bachelor
of Computer Science with Honours (Software Engineering)

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

2023

**RINGKASAN ARTIKEL BERASASKAN WEB DENGAN TEKNIK
PEMBELAJARAN MESIN**

LIM WU TONG

Projek ini merupakan salah satu keperluan untuk Ijazah Sarjana Muda Sains Komputer
dengan Kepujian (Kejuruteraan Perisian)

Fakulti Sains Komputer dan Teknologi Maklumat

UNIVERSITI MALAYSIA SARAWAK

2023

DECLARATION

I hereby declare that this project is my original work. I have not copied from any other student's work or from any other sources except where due reference or acknowledgement is not made explicitly in the text, nor has any part had been written for me by another person.



.....
(LIM WU TONG)

01 JULY 2023

Matric No: 72789

ACKNOWLEDGEMENT

Firstly, I would like to express my deep appreciation to my supervisor, Dr Mohammad bin Hossin, who has provided invaluable assistance, guidance, and advice throughout the development of this project. His expertise and support have been instrumental in helping me to complete my work. At the same time, I would also like to thank my examiner, Prof. Dr Narayanan Kulathuramaiyer, for his valuable comments and suggestions that have helped to improve the quality of my project.

I would also like to acknowledge the help provided by Prof. Dr Wang Yin Chai, the final year project coordinator, who has given me the necessary information and guidance to complete my final year project. In addition, I would like to thank all my friends and classmates who have supported me and provided me with the information needed to finish this project.

Finally, I am especially grateful to my parents and siblings for their unwavering support and encouragement. Their love and encouragement have given me the strength and motivation to complete this project. Without their help, this project would not have been possible.

Table of Contents

ACKNOWLEDGEMENT	i
Table of Contents	ii
LIST OF FIGURES	vii
LIST OF TABLES	ix
ABSTRACT	x
ABSTRAK.....	xi
Chapter 1: Introduction.....	1
1.1 Preliminaries.....	1
1.2 Problem Statement	2
1.3 Project Objectives	4
1.4 Scope	4
1.5 Significant of project.....	4
1.6 Project Outcome	5
1.7 Thesis Organization.....	5
1.7.1 Chapter 1: Introduction	6
1.7.2 Chapter 2: Literature Review	6
1.7.3 Chapter 3: Methodology	6
1.7.4 Chapter 4: Implementation	6
1.7.5 Chapter 5: Evaluation, Testing and Result	7
1.7.6 Chapter 6: Conclusion and Future Works.....	7

1.8	Project Schedule	7
Chapter 2: Literature Review		10
2.1	Introduction	10
2.2	Text Summarization Approach	10
2.2.1	Machine Learning	11
2.2.2	Natural Language Processing	12
2.3	Text Summarization method	13
2.3.1	Naive Bayes method	13
2.3.2	Artificial Neural Network method	16
2.3.3	Decision Tree method	20
2.4	Existing Web-based Text Summarization Software	22
2.4.1	QuillBot Summarizer Tool	22
2.4.2	Scholarcy.....	25
2.4.3	Comparison between the existing web-based text summarization software	27
2.5	Summary	28
Chapter 3: Methodology		29
3.1	Introduction	29
3.2	Project Methodology	29
3.2.1	Step 1: Literature Review	30
3.2.2	Step 2: System Architecture.....	31
3.2.3	Step 3: System Development Life Cycle	32

3.2.3.1 Phase 1: System requirement	33
3.2.3.2 Phase 2: System Design	35
3.2.3.3 Phase 3: System Implementation	45
3.2.3.4 Phase 4: System Testing	46
3.2.3.5 Phase 5: System Deployment.....	46
3.2.4 Step 4: Evaluation The proposed system	47
3.2.5 Step 5: Documentation.....	48
3.8 Summary	48
Chapter 4: Implementation	50
4.1 Introduction	50
4.2 Hardware and Software Environment	50
4.3 Environment Setup.....	51
4.3.1 Django Framework	51
4.3.2 Visual Studio Code	53
4.3 System Modules	55
4.3.1 Graphical User Interface (GUI) System	55
4.3.2 User class	56
4.3.3 Pre-processing Text	57
4.3.4 Naïve Bayes Summarizer.....	58
4.3.5 Neural Network Summarizer	59
4.3.6 Decision tree Summarizer.....	60

4.4	Deployment	61
4.5	Summary	64
Chapter 5: Evaluation, Testing and Result.....		65
5.1	Introduction	65
5.2	Evaluation.....	65
5.2.1	Summarization Quality Evaluation.....	65
5.2.1.1	ROUGE Metrics Evaluation	66
5.2.1.2	Expert Evaluation.....	68
5.2.2	Preferredness Evaluation	69
5.2.3	Time Efficiency Evaluation	70
5.3	Testing.....	71
5.4	Result.....	72
5.5	Summary	74
Chapter 6: Conclusion and Future Work.....		75
6.1	Introduction	75
6.2	Achievements	75
6.3	Limitations	75
6.4	Future Works.....	76
6.6	Summary	76
References		77
APPENDIX A		83

APPENDIX B..... 84

APPENDIX C 105

APPENDIX D 106

LIST OF FIGURES

Figure 1. 1 Gantt Chart Week 1-8	7
Figure 1. 2 Gantt Chart Week 9-16	8
Figure 1. 3 Gantt Chart Week 17-24	8
Figure 1. 4 Gantt Chart Week 25-32	9
Figure 1. 5 Gantt Chart Week 33-40	9
Figure 2. 1 Architecture of neural network (Wang, 2003).....	17
Figure 2. 2 Feed-forward neural network model after training (Kaikhah, 2004).....	17
Figure 2. 3 Sample decision tree (Mirza, Mittal & Zaman, 2018).....	21
Figure 2. 4 QuillBot Summarizer	23
Figure 2. 5 QuillBot Summarizer in Key Sentences mode	24
Figure 2. 6 QuillBot Summarizer in Paragraphs mode	24
Figure 2. 7 Scholarcy Article Summarizer	25
Figure 2. 8 Scholarcy Highlight	26
Figure 3. 1 Steps of Project Methodology.....	30
Figure 3. 2 System Architecture of Web-based Article Summarization system with Machine Learning Techniques	32
Figure 3. 3 Waterfall model	33
Figure 3. 4 Use Case diagram of Web-based Article Summarization system with Machine Learning Techniques	37
Figure 3. 5 Class Diagram of Web-based Article Summarization system with Machine Learning Techniques	39
Figure 3. 6 Sequence Diagram of Web-based Article Summarization system with Machine Learning Techniques	41

Figure 3. 7 State Diagram of Web-based Article Summarization system with Machine Learning Techniques	43
Figure 3. 8 Wireframe of User Interface of Web-based Article Summarization system with Machine Learning Techniques	45
Figure 4. 1 The webpage of Python website	52
Figure 4. 2 Screenshot of Command Prompt install Django.....	52
Figure 4. 3 Screenshot of Command Prompt Create Django project.....	52
Figure 4. 4 Screenshot of Command Prompt Run Django Project	53
Figure 4. 5 Screenshot of Web Browser for Django Project Install and Run Successfully	53
Figure 4. 6 Screenshot of VS Code Interface with Project file	54
Figure 4. 7 Screenshot of VS Code Interface with Installed Extension.....	54
Figure 4. 8 The main page of ArtSum Web-based System.....	56
Figure 4. 9 Upload_file() in User class	57
Figure 4. 10 summarize() in User class.....	57
Figure 4. 11 Pre-processing Text for Naïve Bayes and Decision Tree	58
Figure 4. 12 Pre-processing Text for Neural Network.....	58
Figure 4. 13 Naïve Bayes Summarize Process.....	59
Figure 4. 14 NeuralNetwork Class	60
Figure 4. 15 Decision Tree Summarize Process	61
Figure 4. 16 Creating Django stack VM instance from the marketplace.....	62
Figure 4. 17 Connecting VM instance via SSH	62
Figure 4. 18 Cloning the project repository	63
Figure 4. 19 Running the project and deploying the system.....	63

LIST OF TABLES

Table 2. 1 Comparison between the existing web-based text summarization software.....	27
Table 3. 1 Hardware Requirement for the proposed system	34
Table 3. 2 Software Requirement for the proposed system	35
Table 4. 1: Hardware and Software Environment.....	50
Table 5. 1 ROUGE Evaluation Results.....	67
Table 5. 2 Processing Time for Each Summarizer Techniques	70
Table 5. 3 Test Case	72

ABSTRACT

The motivation behind this project is the increasing amount of information available on the internet, which makes it difficult for people to sift through and find the relevant information they need. Text summarization can help to address this problem by condensing lengthy texts into shorter summaries that convey the main points and ideas of the original text. However, traditional text summarization methods often produce summaries that are too short or lack coherence, which can make them difficult to understand. Machine learning techniques have the potential to overcome these limitations and produce more accurate and coherent summaries. In order to develop the web-based article summarization system, various machine learning techniques were studied and compared. The Naive Bayes, Neural Network, and decision tree techniques were chosen for their ability to handle both numerical and categorical data, and their robustness to noise and missing values. These techniques were implemented using the Python programming language and the scikit-learn library. The front-end of the system was developed using the Django framework, along with HTML, CSS and JavaScript for styling and interactive elements. The performance of the system was evaluated using a dataset of articles and their corresponding summaries. The quality of the summaries was assessed using metrics such as ROUGE and expert evaluation, while the preferredness were evaluated through user surveys and time efficiency observed from the system. The results showed that the system was able to produce summaries that were of good quality, preferred by users, and efficient in terms of time. Overall, the web-based article summarization system with machine learning techniques demonstrated the potential to be a useful tool for condensing and summarizing texts in a more accurate and coherent manner.

ABSTRAK

Motivasi di sebalik projek ini adalah peningkatan jumlah maklumat yang tersedia di internet, yang menyukarkan orang ramai untuk menyaring dan mencari maklumat berkaitan yang mereka perlukan. Rumusan teks boleh membantu menangani masalah ini dengan memekatkan teks yang panjang lebar menjadi ringkasan yang lebih pendek yang menyampaikan perkara utama dan idea teks asal. Walau bagaimanapun, kaedah ringkasan teks tradisional sering menghasilkan ringkasan yang terlalu pendek atau kurang koheren, yang boleh menyukarkannya untuk difahami. Teknik pembelajaran mesin mempunyai potensi untuk mengatasi batasan ini dan menghasilkan ringkasan yang lebih tepat dan koheren. Untuk membangunkan sistem ringkasan artikel berasaskan web, pelbagai teknik pembelajaran mesin telah dikaji dan dibandingkan. Teknik Naive Bayes, Neural Network dan Decision Tree dipilih kerana keupayaannya mengendalikan kedua-dua data berangka dan kategori, serta keteguhannya terhadap bunyi dan nilai yang hilang. Teknik-teknik ini telah dilaksanakan menggunakan bahasa pengaturcaraan Python dan scikit-learn. Bahagian hadapan sistem telah dibangunkan menggunakan rangka kerja Django, bersama-sama dengan HTML, CSS dan JavaScript untuk pengayaan dan elemen interaktif. Prestasi sistem telah dinilai menggunakan set data artikel dan ringkasan yang sepadan. Kualiti ringkasan dinilai menggunakan metrik seperti ROUGE dan penilaian pakar, manakala keutamaan dinilai melalui tinjauan pengguna dan kecekapan masa diperhatikan daripada sistem. Hasil kajian menunjukkan sistem ini mampu menghasilkan ringkasan yang berkualiti, digemari pengguna, dan cekap dari segi masa. Secara keseluruhannya, sistem ringkasan artikel berasaskan web dengan teknik pembelajaran mesin menunjukkan potensi untuk menjadi alat yang berguna untuk memekatkan dan meringkaskan teks dengan cara yang lebih tepat dan koheren.

Chapter 1: Introduction

1.1 Preliminaries

A research article presents the results of original research, evaluates its contribution to the corpus of knowledge in a specific discipline, and is published in a peer-reviewed scholarly journal. The publication of university professors' research plays a crucial part in deciding whether they are given tenure (Hall, 2017). Research articles were primarily read by other researchers and students. Research articles are crucial for researchers and students nowadays. Project preparation is the most common reason for students to read research papers (Akmal, Dhivah, & Mulia, 2020). Students and researcher working on a project must read lengthy research articles in order to choose their topics and conduct their research.

In the age of the technology, a vast volume of research articles has produced by researchers and students. Hence, article summarization is important for research and student while doing their research study. One kind of information management is summarization (Jones, 1993). A summary is defined as a text that is derived from one or more texts, provides essential information from the original texts, and is no longer than half the length of the original texts, and frequently much shorter. (Allahyari, Pouriyeh, Assefi, Safaei, Trippe, Gutierrez, & Kochut, 2017). Summarization is the process of creating a shortened rendition of a lengthy literary work. It extracts the most essential information from the original text and eliminates irrelevant details and data. A summarization helps you to understand the article better (Ansari, 2022). People may quickly and readily understand a text without having to read it in its entirety if a brief summary is provided (Chuang, & Yang, 2000). It will be simpler to understand what the article is about and to understand the author's main points after reading through the article's summary. However, manual article summary might be inefficient and inconvenient due to its time-

consuming and would need more work force to complete it. Therefore, this Web-based Article Summarization system is introduced in this project.

By using this Web-based Article Summarization system, any lengthy article may be summarized into a short and understandable format. This approach employs machine learning techniques to improve the precision and speed of the summarization process. Through this system, researchers, students, and even professors may save time and obtain a deeper grasp of the article, allowing them to do their study more efficiently.

1.2 Problem Statement

Students engaged in research are required to read research articles in order to choose topics and perform experiments (Subramanyam, 2013). Knowing what has been discovered and what questions remain may help when planning a research project. Therefore, article summary may help in the management of the article's content. When manually summarizing an article, the process usually starts with reading many times and paying close attention to focus on the main argument of the article. The length of the summary article must then be determined by extracting the main concept and any relevant supporting details. However, the process of manually article summarization will cause the time consuming and gain the high workload for students and researchers. The average researcher spends 49–61 hours per year reading articles, with an average of 20 minutes each reading and an estimated 145–184 reads per year (Tenopir, King, Clarke, Na, & Zhou, 2007). A typical researcher probably spends hundreds of hours per year reading articles (Nüst, Boettiger, & Marwick, 2018). Therefore, a web-based system is required to automatically summarise articles so that students and researchers can spend less time on the task.

Furthermore, numerous researches have shown strong evidence that distracted reading has a negative impact on the learning process (Schmidt, 2020). When reading and summarizing the article, some students and researchers are not focusing on the most important aspect of the information being discussed in the article. This will actually occur since they were not paying attention, and they were also distracted by other details in the article that were not important. The difficulty of reading papers from 2015 is greater than that of those from the 1900s, and the issue is not related to words (Ball, 2017). Because of the difficulty of the article, it's possible that students and researchers may have difficulty understanding or will misunderstand the article's main idea when they read it or attempt to summarize it. Therefore, the web-based article summarization system will include machine learning techniques to provide a more accurate and precise summary of the research article.

In addition, the main concept presented in a single research article is the most challenging part to understand, and it is challenging to summarize it manually. When reading a research article, students and researchers may get discouraged to read it and attempt to summarize it since it contains unfamiliar terminology, aspects, and even methodologies. Therefore, different machine learning techniques will be included into the web-based article summarization system in order to provide users with the option of selecting their preferred article summarize result after processing by the different machine learning techniques.

1.3 Project Objectives

The Project aims to design and develop a web-based article summarization system.

The objectives of the project are:

- i. To design a web-based system which integrates machine learning algorithm for article summarization.
- ii. To implement the proposed web-based system integrating with program-based Python machine learning techniques to produce article summaries.
- iii. To analyse the effectiveness and efficiency of proposed machine learning techniques based on summarization quality, preferredness and time efficiency.

1.4 Scope

The system is available to all users. However, the focus of this project's development will be on assisting final year students who need to do research for their final-year projects. By using this system, users will speed up their research and gain a deep understanding of the article they research.

1.5 Significant of project

The proposed web-based system for summarizing articles has the potential to greatly benefit a variety of users. Students, researchers, and professors alike may find it useful when conducting research and reviewing articles. By providing a short summary of the research, the

system aims to reduce the time spent on manual summarization and improve the user's understanding of the article. This can increase efficiency during the research process.

In addition to its potential benefits for individual users, this system may also have a broader impact on the field of research as a whole. By streamlining the review process and allowing for more efficient analysis of articles, the system has the potential to facilitate faster and more comprehensive understanding of current developments in a given field.

Overall, the development of this web-based system for summarizing articles has the potential to greatly benefit both individual researchers and the research community as a whole. By providing a quick and easy way to review and understand research articles, the system has the potential to improve the efficiency and effectiveness of the research process.

1.6 Project Outcome

The functional web-based system that can summarize articles. By using this system, the uploaded article will be shortened and simplified so that it is short and easy to understand. Other from that, the system will use 3 different summarization machine learning algorithms to allow users to choose the quality, preferredness, and efficiency of the system's summarization.

1.7 Thesis Organization

Six chapters make up the report: Introduction, Literature Review, Methodology, Implementation, Evaluation, Testing, and Results, and Conclusion and Future Work.

1.7.1 Chapter 1: Introduction

This chapter presents an overview of the undertaking. This chapter discusses the problem statement, objectives, scope, significance of the project, project schedule, and anticipated outcome. The problem statement analyses the current problem at hand. The objectives define the requirements of the project, whereas the scope specifies the limitations or range to be covered during the two semesters of Final Year Project.

1.7.2 Chapter 2: Literature Review

Chapter 2 provides a literature review of the existing and similar systems. This chapter will analyse the existing system's constraints. The comparative analysis of the existing systems is followed by a review of the proposed system's advantages. The entire review of existing systems is conducted using reliable sources such as articles, journals, etc. This chapter will also provide a brief description of the project's implementation, including the algorithm, techniques, and technology used.

1.7.3 Chapter 3: Methodology

Chapter 3 focuses on the methodology used in the project's development. This chapter also includes the requirement analysis, in which the user and system needs are described in detail. In addition, the system design phase will be included at the end of this chapter, which acts as the blueprint for the whole project.

1.7.4 Chapter 4: Implementation

Chapter 4 describes the proposed system's implementation in detail. The proposed system's structure and prototype are discussed in detail.

1.7.5 Chapter 5: Evaluation, Testing and Result

Chapter 5 begins with an introduction to the evaluation and testing phase, providing an overview of the testing plan and strategy. The chapter concludes with a summary of the testing phase and its impact on the system's functionality.

1.7.6 Chapter 6: Conclusion and Future Works

Chapter 6 is the last chapter of the project and provides a summary of the whole project. This chapter provides a summary of the project's achievements, limitations, and suggestions for future enhancements.

1.8 Project Schedule

The development of a successful project mainly depends on the use of a precise project schedule. The identifying of the project milestone serves as a guideline for determining whether or not the project can be completed on time. The completion of this project requires a total of two semesters. The Gantt Chart was chosen as the tool to be used for arranging the various stages of the project's progression.

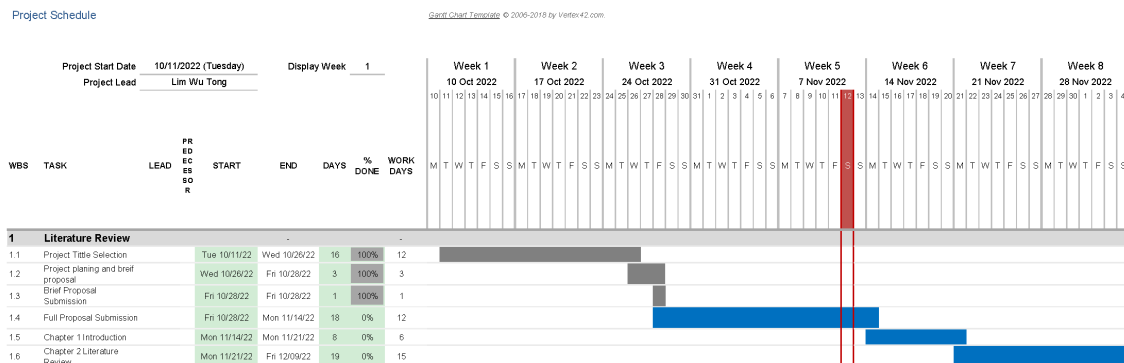


Figure 1. 1 Gantt Chart Week 1-8

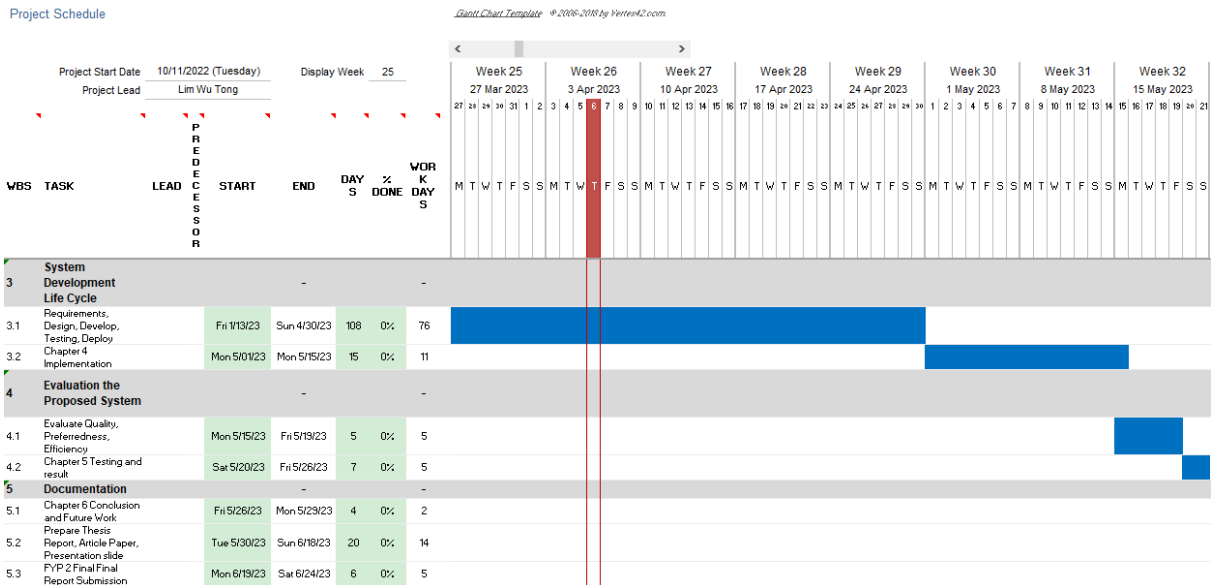


Figure 1. 4 Gantt Chart Week 25-32

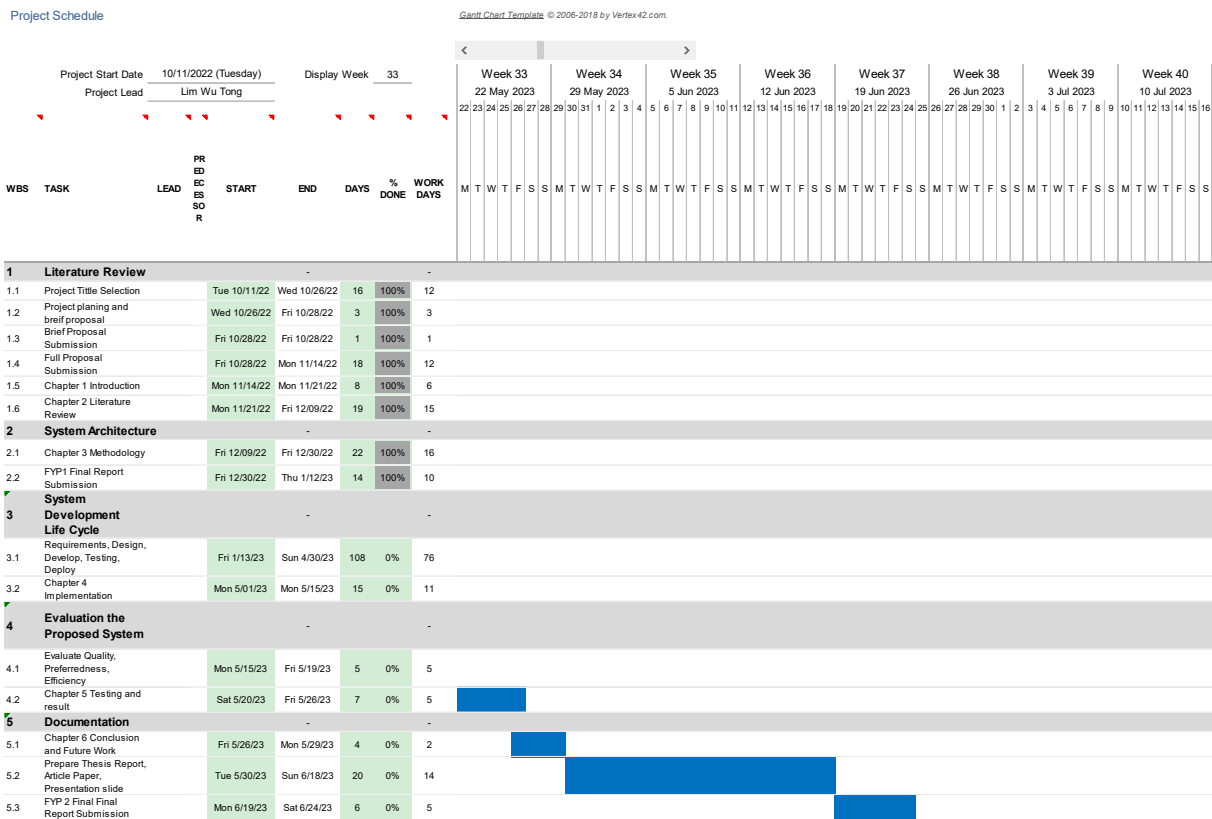


Figure 1. 5 Gantt Chart Week 33-40

Chapter 2: Literature Review

2.1 Introduction

This chapter discusses the approach that was utilised for text summarization, as well as the techniques used for text summarization that used machine learning. Next, the reviews on various other web-based text summarising systems that are already in operation is discussed. These reviews will aid in understanding the approaches that the system utilised and what features a text summarization system should consist of in order to be effective. The next step is a comparison of the proposed system with the already available web-based text summarization system. Through making this comparison, we are able to learn both the advantages and the disadvantages of the current system.

2.2 Text Summarization Approach

With the exponential availability of technology knowledge and Internet accessibility, text summarization has become crucial. Techniques developed for the processing and comprehension of text documents, such as machine learning (ML) and natural language processing (NLP), help to organise the syntactical relations and structural components of language in preparation for automatic processing and learning in the field of artificial intelligence (AI). In text summarization, machine learning techniques are used to determine the optimal feature to extract, whereas NLP techniques enable the use of natural language components such as text structure, document concepts, and lexical chains. (Iboi, Chua, Ranaivo-Malançon, & Kulathuramaiyer, 2017). Therefore, machine learning and natural language processing are consequently the most significant technologies employed in the process of text summarization.

2.2.1 Machine Learning

Machine learning is a subfield of computation algorithms designed to simulate human intelligence by acquiring knowledge from their surroundings as humans do. (El Naqa & Murphy, 2015). The field of artificial intelligence (AI) and computer science known as machine learning focuses on the use of data and algorithms to imitate how humans learn, with the aim of continuously improving the model's accuracy. An algorithm for machine learning is a type of computing method that uses input data to perform a desired task without being explicitly programmed (sometimes referred to as "hard coded") to produce a specific outcome.

Before generating the appropriate summarised texts, machine learning models are often trained to understand documents and extract relevant information. The basic concept of the system is machine learning's capacity to identify technical key terminologies (words, phrases, and sentences) in the context of the semantic relationships between training patents and corresponding summaries. (Trappey, Trappey, Wu & Wang, 2020). These terminologies are then used to generate corresponding summaries. After the age of early AI development, neural network modelling as a kind of machine learning became a common artificial intelligence (AI) technique. These are models of consciousness that imitate the neuronal and synaptic activity of the human brain for idea interpretation and learning. Particularly since the 1980s, several neural network model types and modelling methods have been successfully created for general and particular purposes and applications (Trappey et al., 2020).

For the purpose of text summarization, many various approaches of machine learning that were developed in the 1990s (Fatima & Cenek, 2015). These include Log-Linear Models, Neural Networks, Hidden Markov Models, Rich Features and Decision Trees, and Naive Bayes Methods. The Log-Linear Model incorporates an exponential technique, which consists of two labels indicating whether or not a phrase should be extracted for a summary (Osborne, 2002).

Neural networks are trained to comprehend the key elements of sentences that should be included in the article's summary. This is done by including the sentences in a training set. After that, the neural network is altered so that it can generalise and incorporate the relevant features that are visible in the summary phrases (Kaikhah, 2004). A sequential way to create local relationships between the phrases is called a Hidden Markov Model (HMM). It employs a set of characteristics, including the position of the sentence within the document (which is embedded into the state structure of the HMM), the number of terms contained within the sentence, and the likelihood of the sentence terms given the document terms (Conroy & O'leary, 2001). Rich features and decision trees utilise a position strategy, which is based on the concept that texts typically adhere to a predictable discourse structure and that sentences with a stronger subject centrality tend to appear at particular positions that can be specified (e.g., title, abstracts, etc.) (Saranyamol & Sindhu, 2014). The Naive Bayes Method uses a classifier called the Naive Bayes classifier to determine whether or not a sentence should be considered for extraction (Thu, 2014).

2.2.2 Natural Language Processing

The goal of natural language processing (NLP) is to convert human language into a formal, machine-readable format (Trappey et al., 2020). The objective of natural language processing is to construct machines that can comprehend and respond to input in the form of text or speech, as well as respond with text or speech in a manner analogous to how humans do so. Computer programmes that employ natural language processing are capable of translating text from one language to another, responding to spoken requests, and rapidly summarising enormous amounts of data in real time. Natural language processing (NLP) applications include information extraction, machine translation, text summarization, keyword search, and human-computer interfaces that use natural languages.

Natural Language Processing gives you the ability to carry out a wide range of activities, such as categorising text and removing material that is not related to the text, as well as translating text from one language to another and summarising lengthy sections of information. In the field of natural language processing, the term "text summarization" refers to the process of condensing vast amounts of text into their most essential components via the use of text extraction models guided. Text extraction is the process of removing certain bits of data that are already present in a text by searching for and removing them. It is an ideal method for automatically summarising material or locating the most important information. The following are some of the most typical examples of extraction models. A manuscript may have its most significant words and phrases mechanically extracted using a process known as keyword extraction. Without having to read each individual item, this might provide you somewhat of an overview of the material and the primary issues that are covered in it.

2.3 Text Summarization method

2.3.1 Naive Bayes method

The Naive Bayes method is a supervised learning algorithm that is employed to solve classification problems. The foundation of this algorithm is Bayes' theorem. Its primary function is text classification, which typically requires a high-dimensional training set. The Bayes' theorem, also known as Bayes' Rule or Bayes' law, is a method for calculating the probability of a hypothesis based on already-known information. It is depending on the probability under the given conditions (condition probability). The probability that something will happen in the future, given that something else has already taken place, is referred to as conditional probability (Dwivedi, 2020).. The equation of probability model that was developed

by Thomas Bayes (1701-1761) is relatively straightforward, despite the fact that it is highly effective. The equation model can be written as follows:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

$P(A|B)$ also known as the posterior probability, is the conditional probability that event A will occur if event B occurs. The likelihood that the hypothesis is accurate based on the evidence presented. (Dwivedi, 2020). While $P(B|A)$ known as conditional probability that event B will occur given that event A has occurred. $P(A)$ known as prior probability is the probability of the hypothesis prior to observing the evidence. and $P(B)$ represent the probability of the evidence.

The right side of the equation is based on training data with predefined classes. Using this information, it also attempting to determine the class of the test data on the left side of the equation. In machine learning, the model learns from qualities that are presented as independent random variables. Therefore, B represents the attribute set and A represents the class variable. If the class variable has a nondeterministic relationship with the attributes, it may use A and B as random variables and probabilistically represent their relationship using $P(A|B)$.

A machine learning model that divides various objects into separate categories based on particular characteristics of variables is called a classifier (Dwivedi, 2020). According to Kupiec, Pedersen, and Chen (1995), they designed an algorithm that makes use of a Naive Bayes method that learns from data and includes a Naive Bayes classifier in order to decide if a sentence should to be included in the summary or not. The Naive Bayes Classifier is one of the classification algorithms that is both straightforward and highly effective. The Naive Bayes classifiers that are based on the widely used Bayes' probability theorem. It contributes to the construction of rapid machine learning models that are capable of making predictions in an efficient way. It is a probabilistic classifier, which means that it makes its predictions based on

the probability that an object will be found. These classifiers are known for their ability to create models that are both simple and effective, particularly in the areas of document classification and document summarization.

A Naive Bayes Classifier could be used to select sentences for inclusion in an automatically text summarization to generate summary (Kupiec, Pedersen & Chen, 1995). The sentences are restricted as a non-summary and summary sentence based on the feature possessed by the sentence. The probability of classification is learnt from training data using the Bayes rule, where s represents the set of sentences in the document, f represents the features utilised during the classification step, and S represents the set of sentences in the summary. $P(s \in S|F_1, F_2, F_3, \dots, F_n)$ represents the probability that a particular sentence will be included in the summary, depending on the qualities it has.

$$P(s \in S|F_1, F_2, F_3, \dots, F_n) = \frac{\prod_{i=1}^n P(F_i|s \in S) \cdot P(s \in S)}{\prod_{i=1}^n P(F_i)}$$

The equation for the naive Bayesian classifier is shown above. In this equation, F_1, F_2, \dots, F_n are the sentence characteristics that will be used for classification (it is assumed that the features are independent of one another), and S is the summary that will be created. After that, each sentence is given a score based on the Equation, and those scores are sorted for selection in the summary (Yogan, Goh, Halizah, Ngo & Puspallata, 2016).

The Naive Bayes method is widely implemented in various machine learning libraries, including scikit-learn (Pedregosa *et al.*, 2011), a popular Python library. Scikit-learn provides an extensive collection of machine learning algorithms, including Naive Bayes classifiers, making it accessible for researchers and practitioners. In the literature review, several studies have recognized the effectiveness of Naive Bayes classifiers in different applications. For example, Kupiec, Pedersen, and Chen (1995) utilized a Naive Bayes method as part of their

algorithm for sentence selection in text summarization. The availability and mention of Naive Bayes in scikit-learn, along with its recognition in the literature, further emphasizes its significance and relevance in the field of machine learning and natural language processing.

2.3.2 Artificial Neural Network method

Artificial Neural Networks, a subset of machine learning, are the foundation of deep learning algorithms. These networks are also referred to as artificial neural networks (ANNs) and simulated neural networks (SNNs). The human brain was the source of both their name and their structure, which resembles the manner in which actual neurons communicate. The building blocks of artificial neural networks, also known as ANNs, are called node layers. A neural network has an input layer of neurons (or nodes, units), one to two (or potentially three) covert layers of neurons, and an output layer of neurons (Wang, 2003). Each node, also known as an artificial neuron, connects to another node and has a weight and threshold associated with it. If the output of any one individual node is greater than the value that has been determined to be the threshold for activation, then that node will become active and will begin sending data to the subsequent layer of the network. If this condition is not met, no data will be sent up to the subsequent layer of the network.

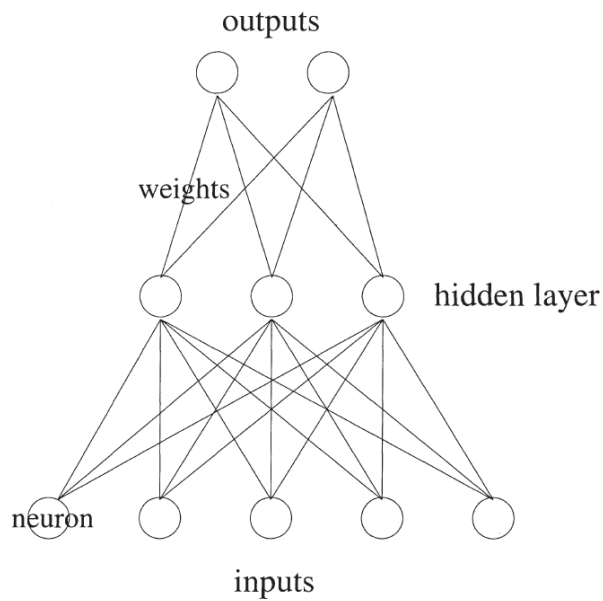


Figure 2. 1 Architecture of neural network (Wang, 2003)

In order to understand the characteristics of summary sentences, a few researchers have taken use of the learning capabilities offered by neural networks. Kaikhah (2004) used a three-layered Feed-forward neural network model to discover summary sentence patterns. During the training phase, the three-layered feed-forward neural network model will learn the characteristics of summary sentences and sentences that do not suit the summary patterns.

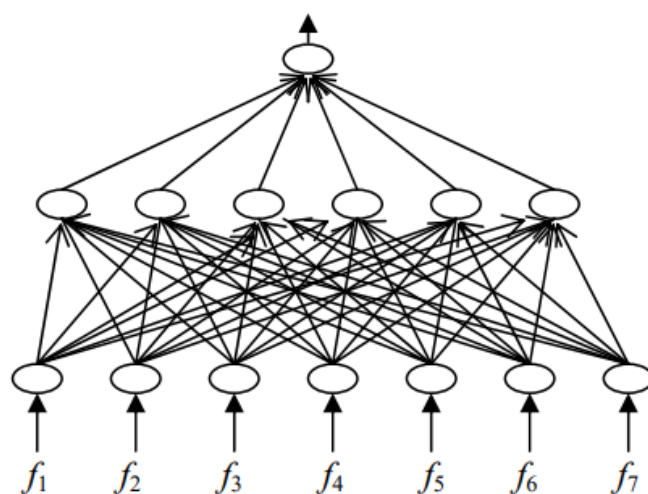


Figure 2. 2 Feed-forward neural network model after training (Kaikhah, 2004)

Signals are only permitted to move in one direction via feed-forward neural networks, which is from the input to the output. A network that feeds data from the input layer to the output layer is known as a feedforward neural network (Dematos, Boyd, Kermanshahi, Kohzadi, & Kaastra, 1996). There is no feedback, hence the fact that the output of one layer does not have an effect on that same layer is not a problem. In most cases, feed-forward networks are straightforward configurations that link inputs and outputs directly. From the approach above, seven characteristics were derived from the sentences that were provided as input. After the network has learned the characteristics that most accurately describe the summary phrase, a process known as feature fusing is carried out by eliminating and merging certain features. The most important step is called the feature fusion phase, and it's during this phase that the relationships between the features are figured out in two phases. 1) removing elements that are used seldom 2) reducing common characteristics into fewer groups before evaluating the sentences to determine which ones are the most significant summary sentences. After that, the pruned network model is used to identify the summary sentences.

Furthermore, Svore et al. (2007) at the Microsoft Research Department created a system for the summarization of a single document that is known as NetSum. A neural network model was used in the development of the NetSum system, which was designed to generate summaries automatically (Yogan et al., 2016). The NetSum system is a single-document summarizer that makes use of neural networks to improve sentence features. The network model is trained with the help of the training set, which consists of articles obtained from CNN.com suggest by the author. Using a neural network that contains a gradient descent approach for the training, the trained model was able to infer the correct ranking of sentences inside a test document. A sentence containing the keyword search entered by the user has a better probability of being included in the generated output summary (Svore et al., 2007). It employs machine learning

methods by labelling a train set such that the labels determine the most effective sentences. It then takes up to three phrases from the text that are the most relevant to its highlights. After that, the trained model is applied to the task of ranking newly created sentences. The RankNet algorithm (Burgess, Shaked, Renshaw, Lazier, Deeds, Hamilton & Hullender, 2005) is used in the process of sentence ranking carried out by the NetSum system.

Pretrained neural network models, such as the Bidirectional and Auto-Regressive Transformers (BART) model, have garnered significant attention in the field of text summarization (Lewis *et al.*, 2019). BART combines the power of the Transformer architecture with bidirectional training, enabling it to effectively capture semantic nuances and contextual dependencies in the input text. Through pretraining on a large corpus of data and subsequent fine-tuning, BART has demonstrated impressive performance in various natural language processing tasks, including text summarization. Leveraging its deep neural network architecture, BART generates abstractive summaries that capture the essence and key information of the input text, resulting in more coherent and human-like summaries.

BART has shown great promise in text summarization tasks by generating high-quality abstractive summaries (Lewis *et al.*, 2019). It utilizes a neural network architecture that incorporates both bidirectional training and autoregressive generation. During pretraining, BART learns to predict missing parts of the input text, enhancing its understanding of semantic relationships between words and sentences. The pretrained model can be fine-tuned on specific summarization datasets to adapt it to the task at hand (Dong *et al.*, 2019). Fine-tuning enables the model to generate concise and coherent summaries conditioned on the input document. BART's flexibility and power in capturing context and salient information result in more accurate and comprehensive summaries. By leveraging pretrained neural networks like BART, text summarization systems can benefit from state-of-the-art natural language understanding

and generation capabilities, ultimately improving the quality and efficiency of the summarization process (Dong *et al.*, 2019).

2.3.3 Decision Tree method

Decision tree learning is one of the most used and useful techniques for inductive approach (Mitchell, 1997). It is a technique for approximating discrete-valued functions that is able to learn disjunctive formulations and is robust to noisy input. In this technique, the learnt function is represented by a decision tree, and the technique is called decision tree learning. In order to make learned trees more comprehensible to humans, they may also be represented as sets of if-then rules. Breaking a data set down into smaller subsets while concurrently creating the corresponding decision tree is the process that decision trees use to construct a classification and regression tree model in the form of a tree structure. The decision tree is a hierarchical structure with one root node, and it is breaking its parent-child-related branches. The root node defining a testing condition which outcome corresponds to a branch leading to an internal node. Classifications are assigned to the terminal nodes of the tree in figure 2.3, also known as the leaf nodes. The Decision Node and the Leaf Node are the two different types of nodes that may be found in a decision tree. It is a classifier in the form of a tree, with internal nodes representing the characteristics of a dataset, branches representing the decision rules, and each leaf node representing the conclusion of the classification. Decision nodes are used to make decision and have numerous branches, whereas Leaf nodes represent the results of these decisions and do not contain any more branches.

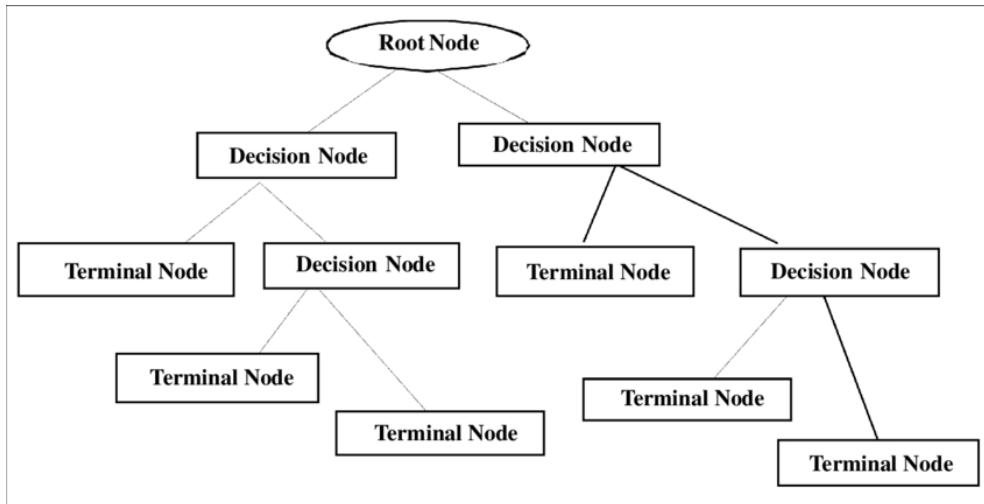


Figure 2. 3 Sample decision tree (Mirza, Mittal & Zaman, 2018)

One of the most common methods of inductive learning is carried out via the use of decision tree algorithms (Chuang & Yang, 2000). The C4.5 algorithm (Quinlan, 1993) was proposed by Chuang and Yang (2000) as the trainer for the text summarizer. Ross Quinlan is the creator of the algorithm known as C4.5, which is used to generate a decision tree. C4.5 is an expansion of an earlier algorithm developed by Quinlan called ID3 (Quinlan, 1986). Because C4.5's decision trees can be used for classification, the algorithm is generally referred to as a statistical classifier. This is one of the reasons why C4.5 is so popular. The C4.5 algorithm uses the amount of information gained as the criterion for splitting. It is able to take in data with either numerical or categorical values. To handle continuous values, a threshold is created, and then characteristics with values above the threshold are separated from those with values equal to or below the threshold. The C4.5 algorithm can readily deal with missing data, as missing attribute values are not used in gain calculations (Sharma & Kumar, 2016). Finding a characteristic that has the most potential for information gain is the first step towards developing a decision tree. The next step is the generation of a node that has a set of rules that relate to the

feature. This approach is repeated for other features in sequential order until there is no longer any opportunity to obtain more information gain.

The decision tree algorithm implemented in scikit-learn has been widely used in various domains, including text mining and natural language processing. Researchers have explored its effectiveness in tasks such as sentiment analysis, topic classification, and document summarization. Garg and Joshi (2018) employed decision trees from scikit-learn for text classification, achieving high accuracy in categorizing news articles into different topics. In another study, Ahmed et al. (2020) utilized decision trees to build a document summarization system, where features such as term frequency and sentence position were used to construct the decision tree model. Their approach demonstrated promising results in generating concise and informative summaries. These studies highlight the versatility of the decision tree algorithm in scikit-learn and its potential for various text-based applications.

2.4 Existing Web-based Text Summarization Software

2.4.1 QuillBot Summarizer Tool

Quillbot is an AI-powered writing application with a number of essential features (QuillBot, 2017). Its major function is that of a rephrasing tool that, through the application of artificial intelligence, rewrites what you have written using a variety of alternative formats. The summarizer tool is one of the features that Quillbot has to provide. The QuillBot summarizer tool compresses the content of lengthy pieces of writing, such as research papers, lengthy emails, news articles, etc, into a form that is simpler to understand and highlights important points. The artificial intelligence behind QuillBot employs natural language processing to search for important information while preserving the material's original context.

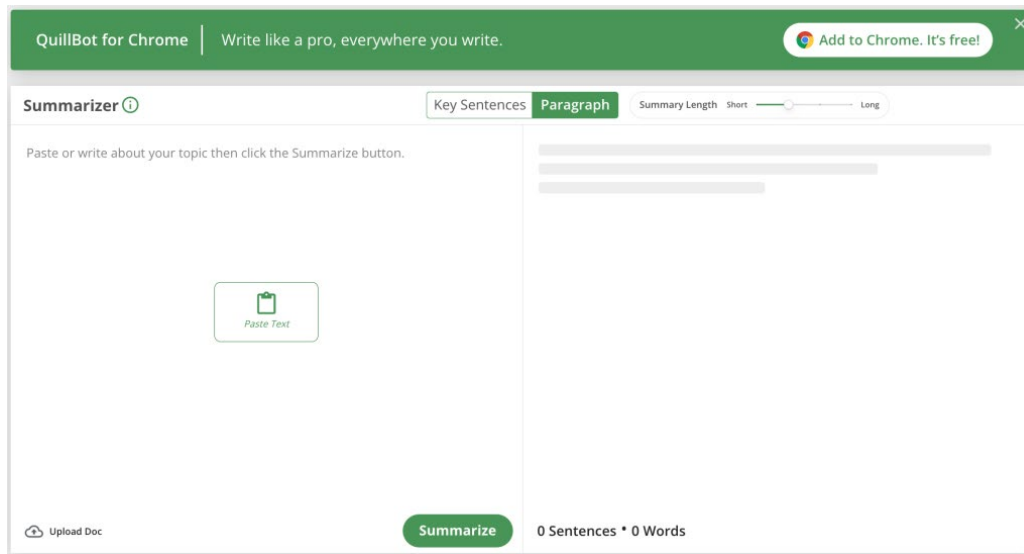


Figure 2. 4 QuillBot Summarizer

Within its summarizer tool, the QuillBot Summarizer provided users with two different types of summarizations based on the context material's key sentence and paragraphs. When the user switches to the Key Sentence mode, a summary of the user's lengthy material is generated in the form of bullet points or key sentences. The user will be able with the assistance of a summary length slider, to raise or reduce the number of bullet points as well as the length of those points.

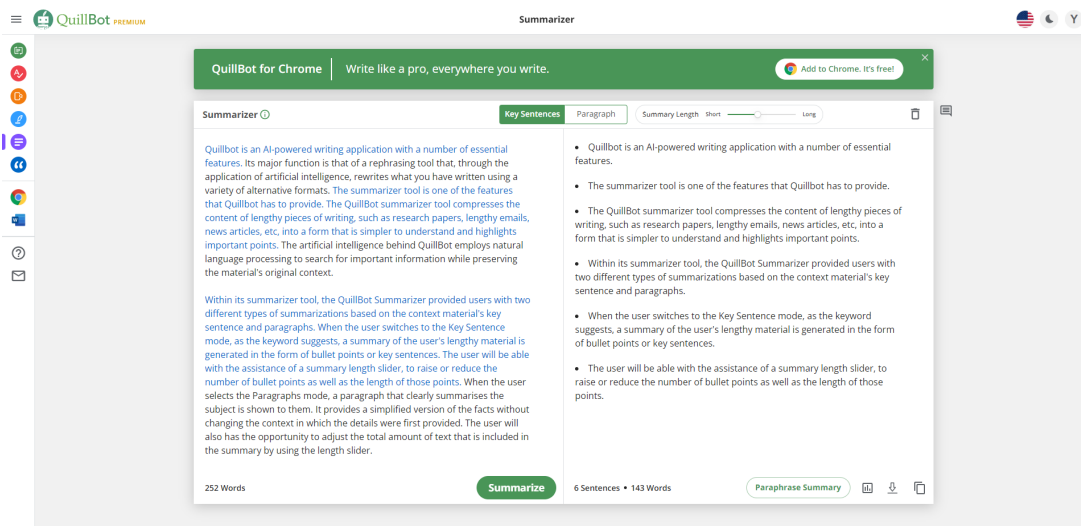


Figure 2. 5 QuillBot Summarizer in Key Sentences mode

When the user selects the Paragraphs mode, a paragraph that clearly summarises the details will be shown. It provides a simplified version of the facts without changing the context in which the details were first provided. The user will also have the opportunity to adjust the total amount of text that is included in the summary by using the length slider.

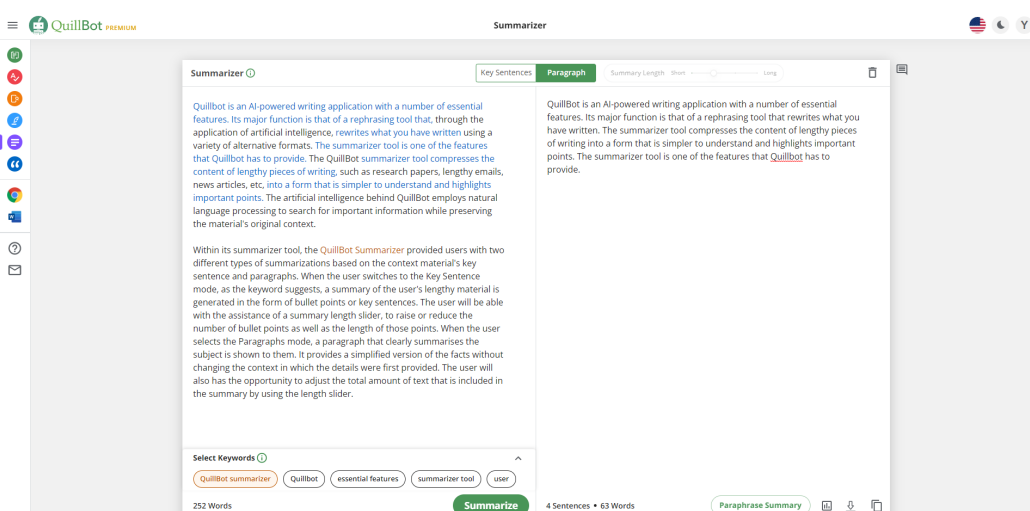


Figure 2. 6 QuillBot Summarizer in Paragraphs mode

The major disadvantages of Quillbot is its tiered price structure. The free plan has a relatively low maximum of 1,200 words and does not permit the development of long-form writing. In addition, Quillbot's pricing levels are very costly, given the availability of accessible and inexpensive summarization tools.

2.4.2 Scholarcy

Scholarcy (Scholarcy, 2019) is an online article summarizer that scans the content of the article in a short amount of time and then breaks it down into snackable parts so that the document may be accessed and evaluated more easily. Students, researchers, and other professionals use the application to instantly get the most important points of any report or paper they are working on, or to write helpful summaries. The user may also construct a summary flashcard of any report, document, or article that is in Word or PDF format within seconds, extracting essential data and references from the source material. Then, these flashcards may be used to rapidly scan a large number of documents, examine a handful, or give a summary to collaborators.

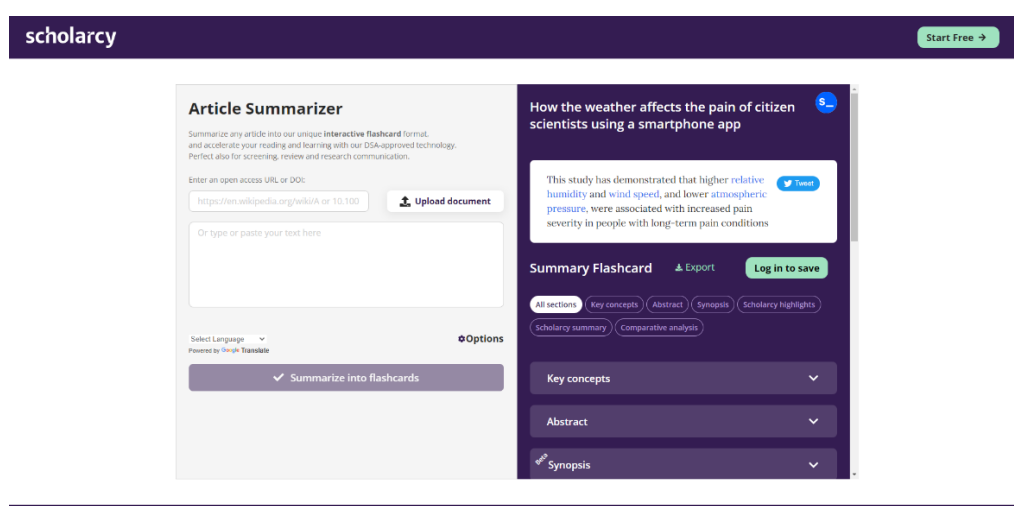


Figure 2. 7 Scholarcy Article Summarizer

The user may set the it to extract tables, figures, and photos from referenced sources, and it connects to open access copies of the sources directly. In addition, Scholarcy makes it easier for you to follow arguments, read content quickly, and extract the most important ideas in a short amount of time. One of the flashcards known as ‘Scholarcy Highlights’, as shown in Figure2.8. Scholarcy is a browser plugin for open access repositories that may be added to the Edge and Chrome web browsers.

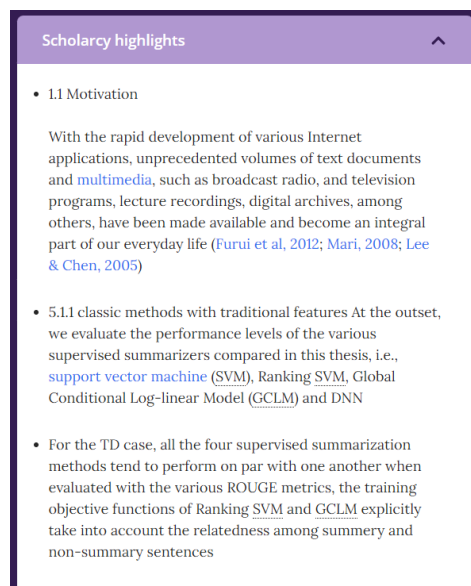


Figure 2. 8 Scholarcy Highlight

Additionally, it links with Scholarcy Library, a monthly subscription service that enables users to look through summary cards on any device. The application also recommends background reading, generates and locates summaries for the user, and retrieves tables and figures. The user may also download any tables discovered by the application in a Word or PDF document and execute calculations on the data. Scholarcy also searches the web for the articles cited in your report or assignment's reference. Using the UnPaywall API, it locates open-access PDFs from arXiv, Google Scholar, and other sites.

2.4.3 Comparison between the existing web-based text summarization software

Table 2. 1 Comparison between the existing web-based text summarization software

Characteristics	Web-based Text Summarization Software	
	QuillBot Summarizer	Scholarcy
Descriptions	A web-based summarising tool that extracts the most significant information from any text with a click.	An online article summarizer that scans your material in seconds and breaks it into snackable parts for easy access and evaluation.
Free	Yes, but character limitation.	Yes, but only in 14 days trial and restricted in some additional feature.
Advantages	It has a user-friendly interface. It can summarise content into a summary paragraph or even bullet points. It also generates original content effectively.	It provides a background reading list and an overview for those new to a field. Their unique Robo-Highlighter automatically highlights the paper's most significant contributions.
Disadvantages	The free plan restricts the number of words available for the summary. The subscription plan is expensive.	The free plan trial is only for 14days. The size of the flashcard UI is smaller.

2.5 Summary

In this chapter, background studies and review of text summarization approaches and techniques were conducted. In addition, we review the existing web-based text summarising software and compare the software. All the features of the existing software are described in detail, and each software has its own set of advantages and disadvantages.

Chapter 3: Methodology

3.1 Introduction

This chapter discusses the project methodology used for this project. Literature Review, System Architecture, System Development Life Cycle, Evaluation of the proposed system, and Documentation are the steps of the project methodology. System Development Life Cycle will use the Waterfall model to guide system development. System Requirement, System Design, System Implementation, System Testing, and System Deployment are the five phases of the Waterfall model.

3.2 Project Methodology

A project methodology is a "must" to prevent failure and decrease risks since it is one of the most important success elements (Ungureanu & Ungureanu, 2014). The overall finding from Chapter 2: Literature review and the proposed solution for this project will be discussed during the literature review step. During the system architecture step, the system architecture diagram will be designed and will describe the system structure for this project. In this step of the system development life cycle, the Waterfall model will be used. In evaluating the proposed system step, the three characteristics of the scenario that must be reviewed inside this system are the quality, preferredness, and time efficiency of the summarization. A thesis report, a research paper, and presentation slides will be prepared during the documentation step.

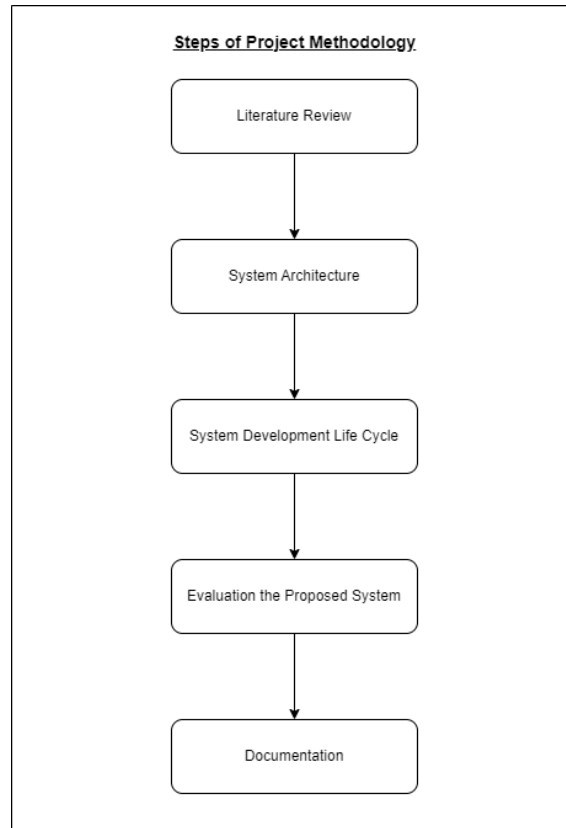


Figure 3. 1 Steps of Project Methodology

3.2.1 Step 1: Literature Review

In this step, it has conducted a review of the existing literature and research on text summarization approaches and method. This included studying the background and context of text summarization, as well as identifying and evaluating different methods and techniques that have been developed to summarize text.

Furthermore, a number of web-based text summarization software tools that are available also reviewed, and provided a detailed description of the features and capabilities of each software. Comparison has conducted by compared the different software tools in terms of their strengths and weaknesses, and discussed the advantages and disadvantages of each tool.

The purpose of this review was to provide a comprehensive overview of the current state of the field and help more understand the range of approaches and method that are available for text summarization.

3.2.2 Step 2: System Architecture

The system architectural diagram shown in Figure 3.2 illustrates the fundamental operation of the Web-based article summarization system. Two of the system's primary inputs are file input and text input. The input file is converted to text format before the summarization process. After that, every input will be sent to the process of summarization. There are three different machine learning methods that may be used in the process of summarization. The three different machine learning methods are Naive Bayes, Artificial neural network, and decision tree. Naive Bayes is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. It is a common choice for text classification tasks and is often used in text summarization systems. Neural networks are a set of algorithms inspired by the structure and function of the human brain, which are commonly used for machine learning tasks such as text summarization. Decision trees are a type of machine learning algorithm that can be used for classification and regression tasks, and are often used in text summarization systems due to their simplicity and interpretability. Both Naïve Bayes and Decision Tree algorithms can be implemented using the scikit-learn library in Python (2020), which is a powerful tool for machine learning tasks. While Neural Network algorithm can include the pretrained Bidirectional and Auto-Regressive Transformers (BART) model (Lewis *et al.*, 2019) for summarization article. After the process of summarization, each machine learning method may provide different results.

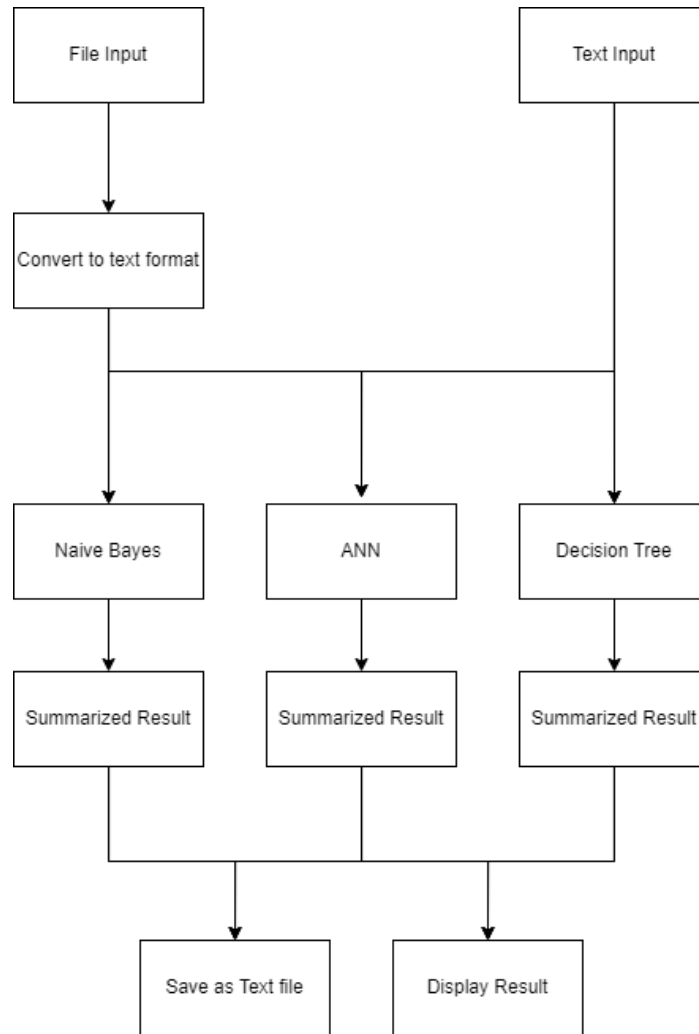


Figure 3. 2 System Architecture of Web-based Article Summarization system with Machine Learning Techniques

3.2.3 Step 3: System Development Life Cycle

Waterfall model's methodology is chosen for used in this System Development Life Cycle phase. Waterfall model is a Software Development Life Cycle (SDLC) in which progress is viewed as descending through a number of phases that must be completed to effectively construct a software system. Waterfall model consists of five phases which are system requirement phase, system design phase, system implementation phase, system testing phase and system deployment phase which shown in Figure 3.3. These five phases are executed

sequentially, which means that any phase of the development process cannot begin until the phase before it is complete.

In system requirement phase, the requirements to develop the system are identified and analysed. In system design phase, the UML diagrams and mock-up user interface design are provided in this phase. In system implementation phase, a Web-based article summarization system is implemented. In system testing phase, a set of tests are conduct to validate the functionally of the system. In system deployment phase, the system is upload to the web domain.

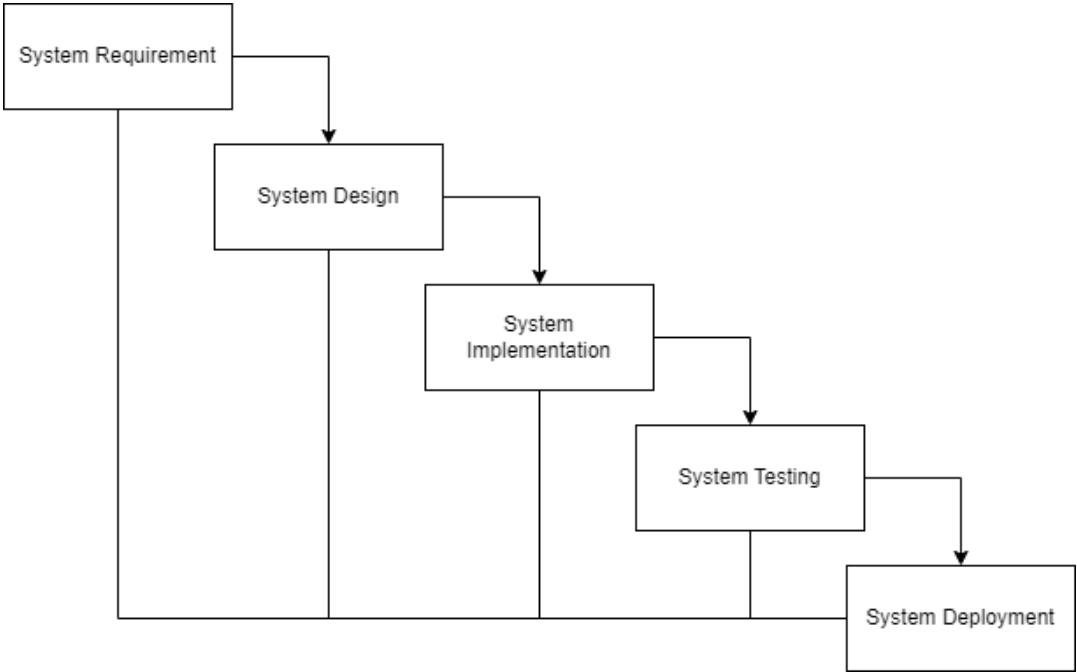


Figure 3. 3 Waterfall model

3.2.3.1 Phase 1: System requirement

The system requirement focuses on the proposed system's requirements. The requirement is determined through an analysis of all existing systems shown in Chapter 2. The system requirement includes both hardware and software requirements which are shown in Table 3.1 and 3.2. These are required to ensure that all processes are effectively completed before continuing to the next phase.

Web-based article summarization system designed for browser compatibility. Therefore, any device that can run a web browser, such as a laptop or desktop computer, is capable of utilising the Web-based article summarization system. Input and output devices, such as a monitor, keyboard, and mouse, are required for Web-based article summarization systems. Some interactions may also eliminate the need for a keyboard and mouse if the display (touch screen) can be used for input and output (display).

The Web-based article summarization system are developed in two parts: front-end and the summarization engine. Front-end development focuses on the design of the user interface (UI) and the functions that the interface can execute. There are some development scripts and languages will be used for front-end development. Programming languages and the scripting tools are selected to enable the system to run smoothly and minimize the complexity. Summarization engine development is concerned with the selected machine learning techniques, which is used to summarized the article submitted by the user. Therefore, a programming language will be chosen for the summarization engine development.

3.2.3.1.1 Hardware Requirement

The hardware requirements for the proposed system:

Table 3. 1 Hardware Requirement for the proposed system

No.	Hardware	Specifications
1.	CPU	Intel core i5
2.	RAM	8GB
3.	Operating System	Windows 10 (64-bits)

3.2.3.1.2 Software Requirement

The software requirements for the proposed system:

Table 3. 2 Software Requirement for the proposed system

No.	Software	Requirement
1	Integrated development environment (IDE)	Microsoft Visual Studio Code
2.	Front-end	CSS, JavaScript, Django framework
3.	Summarization Engine Programming Language	Python
3.	Web Browser	Google Chrome, Microsoft Edge, Firefox, Safari, Opera
4.	Web Domain	Google Cloud Platform

3.2.3.2 Phase 2: System Design

A web-based graph drawing software called Diagrams.net is used for system designing in this section. UML diagrams such as use case diagram, class diagram, activity diagram and sequence diagram will be presented for the system designing of the proposed system. A UML diagram is a partial graphical representation of a system's model, whether it is in the design, implementation, or planning stage. UML diagrams include graphical elements that represent elements in the UML model of the designed system. These elements are UML nodes linked by edges. The system's UML model may also include additional documentation, such as use cases specification.

3.2.3.2.1 Use Case Diagram

Use case diagram shown in Figure 3.4 is presented the interaction between an actor and a system. The use case diagram will describe the functionality of the proposed system and the interactions between the user and the system for this project.

In the proposed web-based article summarization system with Machine Learning Techniques, the primary actor is the user. The user has the ability to input text directly into the system, as well as the ability to upload a file for summarization. When a file is uploaded, it is first converted to text format in order to be processed by the system. Once the text is in the appropriate format, the user is able to select from different machine learning techniques for summarization, such as Naive Bayes, Artificial Neural Network, and Decision Tree. After the summarization process is complete, the user is presented with the summary, and has the option to copy or export it for further use. The use case diagram for the web-based article summarization system is shown as below.

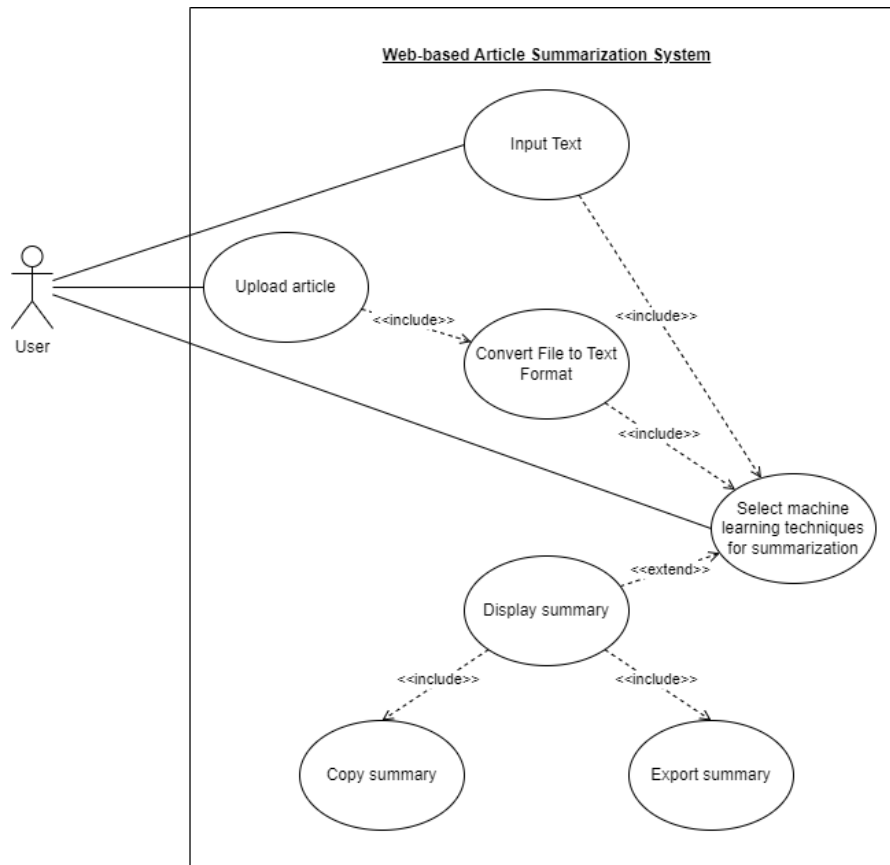


Figure 3. 4 Use Case diagram of Web-based Article Summarization system with Machine Learning Techniques

3.2.3.2.2 Class Diagram

A class diagram is a static diagram in the Unified Modelling Language (UML) that shows the classes in a system, their attributes and operations, and the relationships between them. Class diagrams are used to model the structure of a system and the relationships between different components of the system.

In the context of the web-based article summarization system with machine learning techniques, the class diagram shown in Figure 3.5 include the classes such as User, Article and machineLearning. The User class have the operations such as *uploadFile()* and *summarizes()*. The Article class have attributes such as content and fileName. The machineLearning class have

attributes such as model and tokenizer. It also an abstract class with subclasses such as NaiveBayes, NeuralNetwork, and DecisionTree, each with their own attributes and operations.

The class diagram for web-based article summarization system with machine learning techniques show relationships between these classes, such as inheritance relationships between the machineLearning subclasses and the machineLearning abstract class, and association relationships between the User class and the Article and machineLearning classes.

Class diagrams are useful for designing and documenting the structure of a text summarization system, and can help to visualize the relationships between different components of the system.

Class Diagram For Web-based Article Summarization system with Machine Learning Techniques

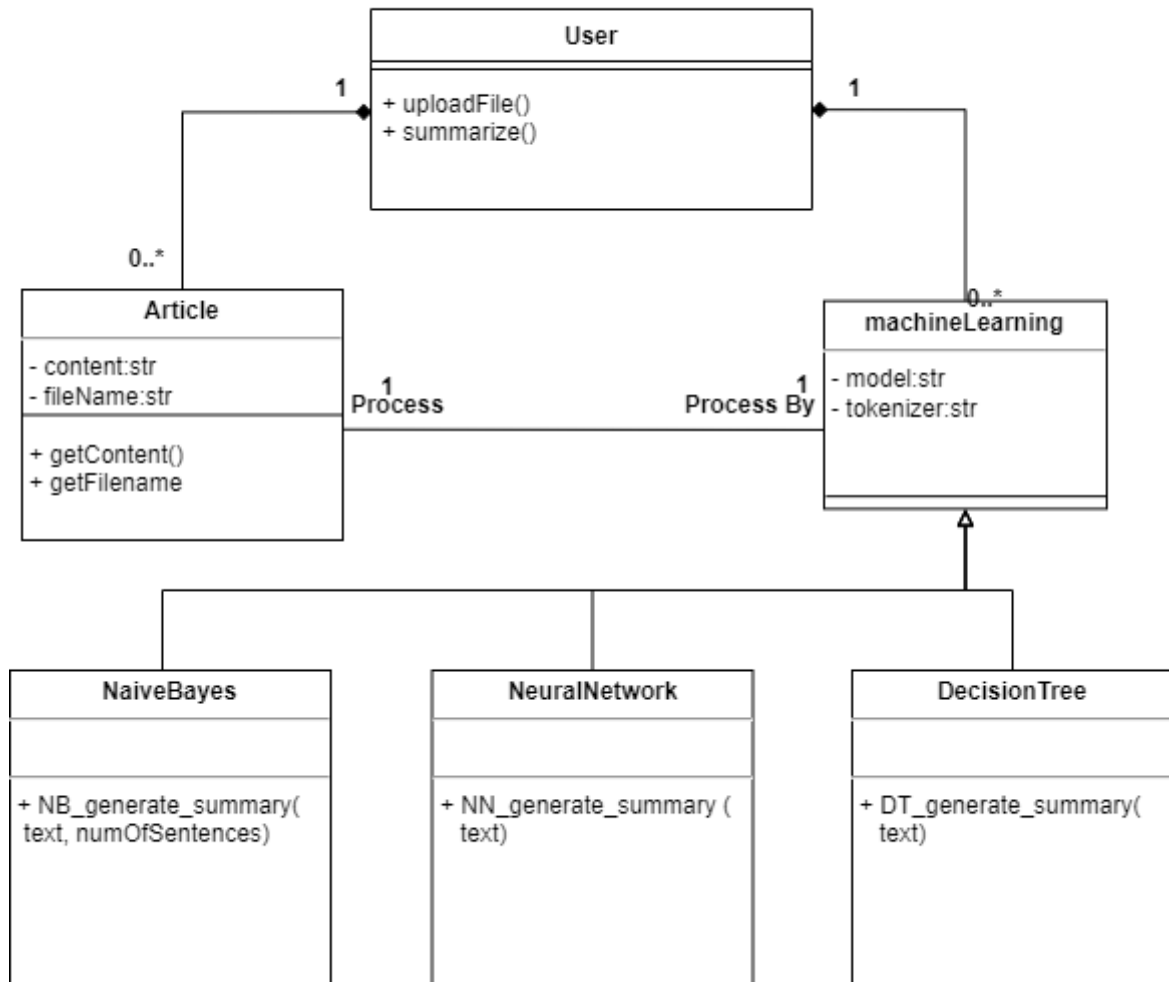


Figure 3. 5 Class Diagram of Web-based Article Summarization system with Machine Learning Techniques

3.2.3.2.3 Sequence Diagram

A sequence diagram is a dynamic diagram in the Unified Modelling Language (UML) that shows the interactions between objects or components in a system, and the sequence of messages exchanged between them. Sequence diagrams are used to model the dynamic behaviour of a system and the interactions between objects or components in the system.

In the context of the web-based article summarization system with machine learning techniques, a sequence diagram shown in Figure 3.6 shows the interactions between objects such as User, Web-based article summarization system and Machine learning techniques. The User object represents the user interacting with the system and the Web-based article summarization system object represents the text summarization system. The Machine learning techniques objects represent components of the system that are responsible for preparing the text for summarization and generating the summary, respectively.

The sequence diagram shows messages being exchanged between these objects, such as the User object sending article by input text or file to the Web-based article summarization system object, and the Web-based article summarization system object sending the prepared summarize article to the Machine learning techniques object. The diagram also shows the User object sending a Select machine learning techniques message to the Web-based article summarization system object, and the Web-based article summarization system object sending a selected technique to the Machine learning techniques object. After the summarized result generated, the result will send back to the Web-based article summarization system object and display to the User object. User object can also export the result to file by sending the message to Web-based article summarization system object and the exported result in file format will reply to User object.

Sequence diagrams are useful for modelling the dynamic behaviour of web-based article summarization system with machine learning techniques and the interactions between different objects or components in the system.

Sequence Diagram for Web-based Article Summarization System with Machine Learning Techniques

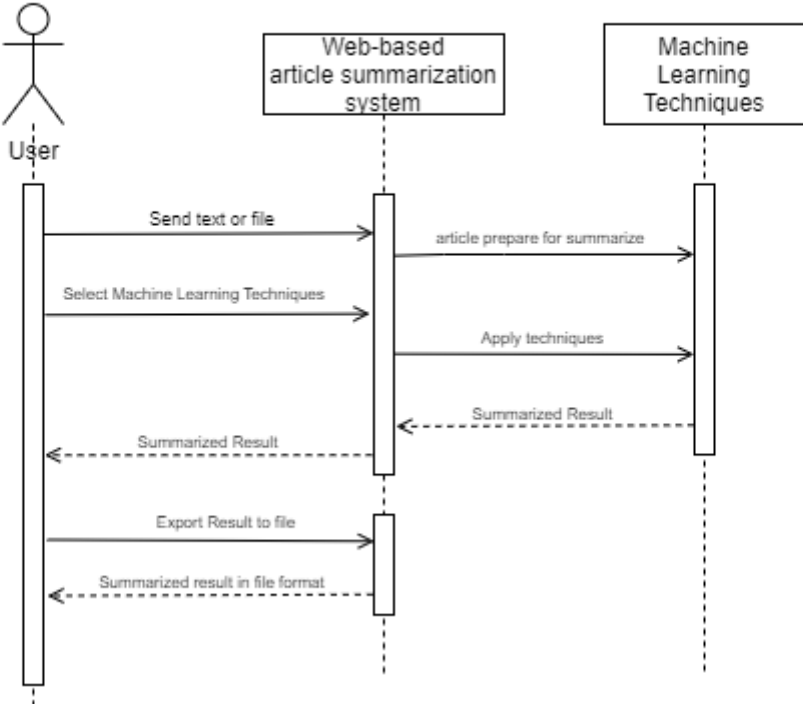


Figure 3. 6 Sequence Diagram of Web-based Article Summarization system with Machine Learning Techniques

3.2.3.2.4 State Diagram

A state diagram is a dynamic diagram in the Unified Modelling Language (UML) that shows the different states of an object or component in a system, and the transitions between those states. State diagrams are used to model the dynamic behaviour of a system and the events that trigger state transitions.

In the context of the web-based article summarization system with machine learning techniques, a state diagram shown in Figure 3.7 shows the different states that the system can be in such as Input text or file, Convert File to text format, Select machine learning techniques for summarization, Naïve Bayes summarizer, Neural Network summarizer, Decision tree summarizer, summarized result, Display result and export result to file. The diagram also shows the transitions between these states, triggered by events such as the user inputting a file or text, or selecting a machine learning method for summarization.

The system begins in the "Input text or file" state, waiting for the user to input a file or text that they would like to summarize. When the user inputs a file or text, the system checks the format of the input. If the input is a file, the system transitions to the "Convert file to text" state, where it converts the file to a text format that can be processed by the summarization engine. If the input is already in text format, the system moves directly to the "Select machine learning techniques" state. In this state, the system waits for the user to select their preferred machine learning technique for summarization. Once the user has selected, the system transitions to the "Summarizer" state, where it uses the selected technique to generate a summary of the text. Finally, the system moves to the "Summarized result" state, where the summary generated by the selected machine learning technique is displayed to the user. From this state, the user has the option to export the summary to a file if they wish.

State Diagram for Web-based Article Summarization system with Machine Learning Techniques

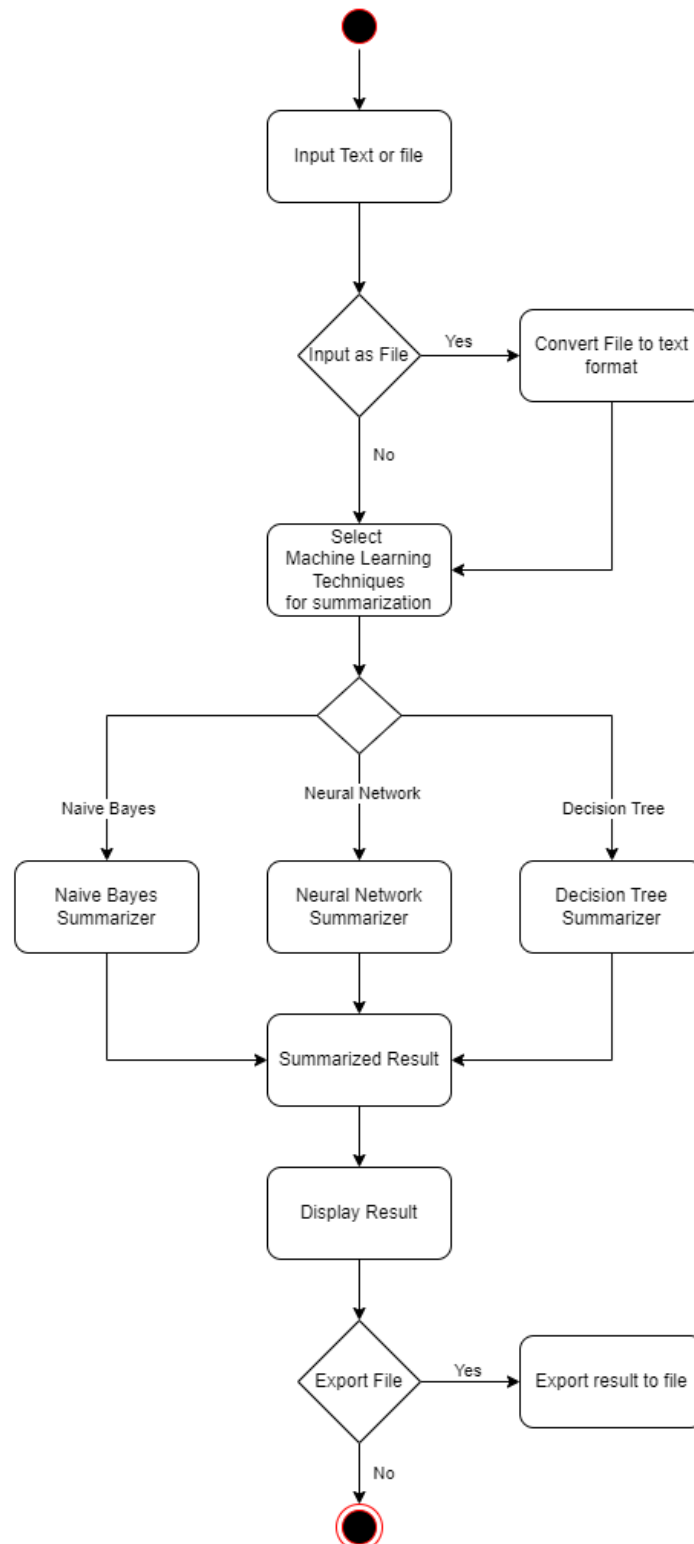


Figure 3. 7 State Diagram of Web-based Article Summarization system with Machine Learning Techniques

3.2.3.2.5 User Interface

Before the implementation phase of the project, it is important to carefully design the user interface for the proposed system in order to ensure that it is user-friendly and intuitive to use. To do this, Figma is selected, a design tool that allows us to create wireframes and mock-ups of the user interface. These pre-designs shown in Figure 3.8 will provide a visual representation of the structure and functionality of the system, and will give both the users and developers an idea of what the system will look like when it is complete. This will help to increase the chances of success for the system and ensure that it is widely adopted by students and researchers.

The user interface of the proposed Web-based article summarization system with machine learning techniques has been designed with ease of use in mind. One of the key features of the interface is the use of the left-right concept. This means that the original article is input on the left side of the screen, while the summarized result is displayed on the right side. This allows the user to easily compare the two versions and see how the system has summarized the original article. This feature is particularly useful for students and researchers who are working on final year projects or academic papers, as it allows them to quickly identify the key points in a article without having to read through the entire document. Additionally, the interface has been designed to be responsive, which means that it can be used on a variety of different devices, including laptops, tablets, and smartphones. This makes it easy for users to access the system and use it on the go. Overall, the user interface is intuitive and easy to use, making it a valuable tool for anyone looking to quickly and effectively summarize article.

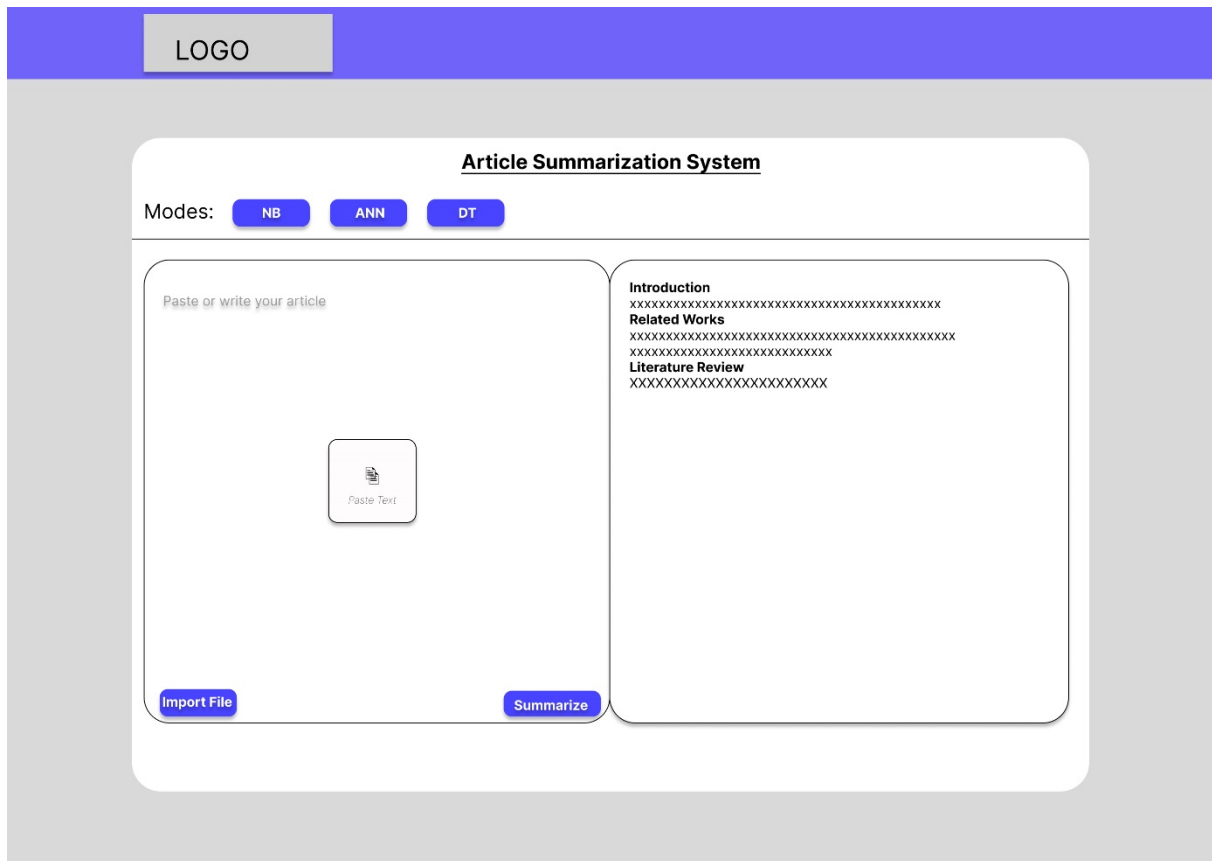


Figure 3. 8 Wireframe of User Interface of Web-based Article Summarization system with Machine Learning Techniques

3.2.3.3 Phase 3: System Implementation

The system implementation phase of the project will likely take longer to complete than the previous stages, as it involves building the proposed system from scratch and testing it to ensure that it works as intended. The front-end version of the Web-based Article Summarization System will be developed using the Django framework, along with HTML, CSS and JavaScript for styling and interactive elements. The machine learning summarization engine will be implemented using Python, a popular programming language known for its ease of use and powerful data processing capabilities. All scripting for the project will be performed using the Microsoft Visual Code integrated development environment (IDE), which provides a range of

tools and features to facilitate efficient coding and debugging. By utilizing these tools and technologies, it is hoped that a user-friendly, efficient, and reliable system will be created.

3.2.3.4 Phase 4: System Testing

After the implementation of the Web-based Article Summarization system is complete, it will be necessary to conduct a series of tests to confirm its operational effectiveness. These tests will be carried out using a web browser, and will involve using the system to summarize a variety of different texts and comparing the results to human-generated summaries. The purpose of testing is to identify any defects or failures in the system and fix them before the system is deployed for use by students and researchers. It is important to conduct multiple tests, using a range of different inputs and settings, in order to ensure that the system is robust and meets the requirements specified in the design. If a bug or flaw is identified during testing, it will be necessary to implement a fix as soon as possible in order to ensure that the system is reliable and user-friendly. Thorough testing of the system can help to minimize the risk of defects or failures and increase the chances of its success in its intended use.

3.2.3.5 Phase 5: System Deployment

Before the system can be deployed for use by students, it is important to ensure that it meets the required standards for quality. Therefore, the first step in the deployment process will be to thoroughly test the system and check its quality. This may involve conducting additional evaluations using metrics like Recall-Oriented Understudy for Gisting Evaluation (ROUGE) (Lin, 2004) and expert evaluation, as well as testing the system under a variety of different conditions to ensure that it is reliable and performs well. To assess the usability of the system from the perspective of students, a group of these students will be asked to complete the System Usability Scale (SUS) (Brooke, 1996) questionnaire shown in Appendix A by using Google Form. This questionnaire will provide valuable insights into the usability of the system from

the perspective of the intended user group and help identify any areas for improvement. Once the quality of the system has been verified and the questionnaire has been done, it will be ready to be deployed to the web domain. This will make it possible for students who are working on their final year projects to access the system directly from their web browsers on their portable digital devices. This will allow them to use the system anytime, anywhere, as long as they have an internet connection. The deployment of the system to the web domain will also make it more widely available to other users who may be interested in using it for their own purposes.

3.2.4 Step 4: Evaluation The proposed system

The proposed system will be evaluated in terms of four aspects: quality, preferredness, effectiveness, and time efficiency. The Recall-Oriented Understudy for Gisting Evaluation (ROUGE) (Lin, 2004) metric will be used to evaluate the quality of the summarization produced by the system. ROUGE is a widely used set of metrics for evaluating the quality of automatic summarization and machine translation systems. It compares the summary produced by the system to a human-generated reference summary, providing a score that reflects the overlap between the two. In addition to evaluating the quality of the summary, it is also important to determine whether users prefer the summary produced by the system to other alternatives. The system's preferredness will be evaluated using the SUS (Brooke, 1996) questionnaire, which will be completed by a group of final year students working on their own projects. The time efficiency of the machine learning techniques used to generate the summary will also be measured to ensure that the system is able to produce high-quality summaries efficiently.

3.2.5 Step 5: Documentation

After completing all the previous tasks, the final step in the process will be to prepare the documentation for the project. This documentation will include a detailed thesis report, a research paper that summarizes the key findings and contributions of the project, and presentation slides that can be used to present the work to a wider audience. The thesis report will provide a comprehensive overview of the project, including the background and motivation, the research questions and objectives, the methods and techniques used, the results and findings, and the conclusion and future work. The research paper will be a shorter and more concise version of the thesis, highlighting the main contributions and implications of the project in a format suitable for publication in a research journal. The presentation slides will provide an overview of the project, including the key results and implications.

3.8 Summary

The project methodology discussed in this chapter consists of several different phases that are important for the development of the proposed web-based article summarization system. The literature review phase involves reviewing the existing research and literature on text summarization approaches and methods, as well as studying the features and capabilities of different web-based text summarization software tools. The system architecture phase involves designing a diagram that illustrates the fundamental operation of the system, including the input and process of summarization using different machine learning methods. The system development life cycle phase involves using the Waterfall model, which consists of five phases: system requirements, system design, system implementation, system testing, and system deployment. In the system design phase, various UML diagrams such as the use case, class, sequence, state, and user interface diagrams are created to provide a detailed overview of the

system's structure and functionality. The system implementation phase involves building the front-end user interface and the summarization engine, while the system testing phase involves conducting a series of tests to validate the system's functionality. Finally, the system deployment phase involves uploading the system to a web domain for use by students and researchers. The project will also involve evaluating the proposed system for its quality, preferredness, effectiveness, and time efficiency, as well as preparing a thesis report, research paper, and presentation slides for documentation purposes.

Chapter 4: Implementation

4.1 Introduction

This chapter discusses in the implementation process of the Web-based Article Summarization system using Machine Learning Techniques is discussed in detail. The chapter covers the tools and methods utilized to develop the system, including the front-end graphical user interface designed with HTML/CSS. Furthermore, the chapter explains how Python programming language is use to code and integrate three machine learning techniques for article summarization into the system.

4.2 Hardware and Software Environment

The hardware and software environment used for the implementation of the Web-based Article Summarization system is outlined in Table 4.1 below:

Table 4. 1: Hardware and Software Environment

No.	Component	Specification
1	CPU	Intel Core i5 2.4GHz
2	RAM	12GB
3	Operating System	Windows 10 (64-bit)
4	Integrated Development Environment (IDE)	Microsoft Visual Studio Code
5	Front-end	CSS, JavaScript, Django framework
6	Summarization Engine Programming Language	Python
7	Web Browsers	Google Chrome
8	Web Domain	Google Cloud Platform

The system was developed on a personal laptop with an Intel Core i5 CPU, clocked at 2.4 GHz, and 12 GB of RAM, running the Windows 10 (64-bit) operating system. Table 1 summarizes the hardware and software components used for the implementation, including the integrated development environment (IDE), front-end web development tools, summarization engine programming language, web browsers for testing, and the web domain where the system was deployed. The hardware and software requirements for the implementation were moderate, and the personal laptop used for the implementation provided sufficient processing power and memory to handle the development environment and the computational requirements of the summarization engine.

4.3 Environment Setup

The development of any software project requires an environment setup that provides the necessary tools and frameworks to develop and deploy the system. In this section, we will discuss the environment setup for the Django framework, which was used to develop the web-based article summarization system. Furthermore, an exploration of Visual Studio Code will be conducted, as it serves as the integrated development environment (IDE) utilized in this project. The following subsections will detail the installation and setup process of these essential tools.

4.3.1 Django Framework

The environment setup for the Django framework (Django Software Foundation, 2019) was accomplished by following the installation guidelines provided by the official Django documentation. The setup process involved the installation of Python (Python, 2020) and pip (Python Package Installer) as prerequisites, followed by the installation of Django itself via pip. The required packages were installed using the command prompt interface of Windows 10.



Figure 4. 1 The webpage of Python website

```
Command Prompt
C:\Users\wuton\OneDrive\Desktop\UNIMAS\FYP\FYP2\Implementation\FYP>pipenv install django
Creating a virtualenv for this project...
Pipfile: C:\Users\wuton\OneDrive\Desktop\UNIMAS\FYP\FYP2\Implementation\FYP\Pipfile
Using C:\Users\wuton\AppData\Local\Programs\Python\Python311\python.exe (3.11.2) to create virtualenv...
[*] Creating virtual environment...created virtual environment (CPython3.11.2.final.0-64 in 1062ms)
creator CPythonWindows(dest=C:\Users\wuton\.virtualenvs\FYP-lzdv50s9, clean=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip-bundle, setuptools-bundle, wheel-bundle, via=copy, app_data_dir=C:\Users\wuton\AppData\Local\pip\python\virtualenv)
added seed packages: pip==23.0.1, setuptools==67.6.0, wheel==0.40.0
activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

Successfully created virtual environment!
Virtualenv location: C:\Users\wuton\.virtualenvs\FYP-lzdv50s9
Creating a Pipfile for this project...
Installing django...
Resolving django...
Installing...
Adding django to Pipfile's [packages] ...
Installation Succeeded
Pipfile.lock not found, creating...
Locking [packages] dependencies...
Building requirements...
Resolving dependencies...
Success!
Locking [dev-packages] dependencies...
Updated Pipfile.lock (af42abefb766e975f7680f10368735353569fb4fe0114e59496a7202658362fc)!
Installing dependencies from Pipfile.lock (8362fc)...
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
```

Figure 4. 2 Screenshot of Command Prompt install Django

A screenshot of the environment setup process is included in Figure 1 below. This screenshot demonstrates the successful installation of Django and its necessary packages on the system. The setup was performed using the latest version of Django at the time, which was version 4.1.7.

```
Command Prompt
C:\Users\wuton\OneDrive\Desktop\UNIMAS\FYP\FYP2\Implementation\FYP>django-admin startproject WebbasedArticleSummarization
C:\Users\wuton\OneDrive\Desktop\UNIMAS\FYP\FYP2\Implementation\FYP>
```

Figure 4. 3 Screenshot of Command Prompt Create Django project

```
Command Prompt - python manage.py runserver
C:\Users\wuton\OneDrive\Desktop\UNIMAS\FYP\FYP2\Implementation\FYP\WebbasedArticleSummarization>python manage.py runserver
Matching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, au
th, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
May 11, 2023 - 18:45:37
Django version 4.1.7, using settings 'WebbasedArticleSummarization.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figure 4. 4 Screenshot of Command Prompt Run Django Project

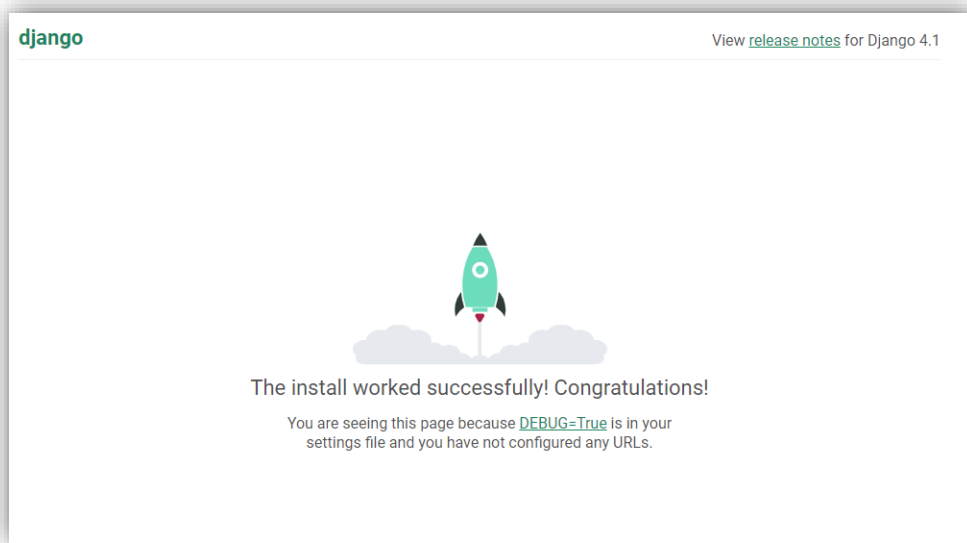


Figure 4. 5 Screenshot of Web Browser for Django Project Install and Run Successfully

This setup was critical for the development and implementation of the web-based article summarization system as it provided the necessary tools and framework to develop and deploy the system. With the Django framework, it was possible to create a web application with a robust back-end that could handle user input, process the input using machine learning techniques, and return the summary to the user.

4.3.2 Visual Studio Code

The integrated development environment (IDE) used in this project is Microsoft Visual Studio Code (VS Code) (Microsoft, 2022), which is a lightweight yet powerful source code editor that supports a wide range of programming languages. To set up Visual Studio Code for

this project, the latest version was downloaded from the official website and installed on the Windows 10 64-bit operating system running on an Intel Core i5 processor with 12GB of RAM. After installation, the necessary extensions for the Python programming language, such as Python and Django, were added to the IDE to enable efficient development. Additionally, Visual Studio Code was configured with the appropriate settings and preferences, such as the code formatting and debugging options, to optimize the development process. A screenshot of the Visual Studio Code interface with the project files and extensions is provided in Figure 4.6 and 4.7.

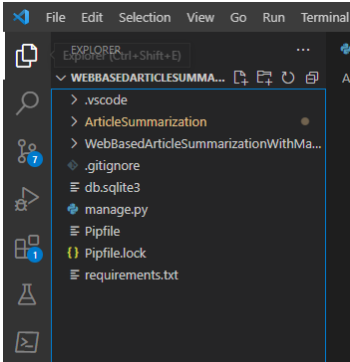


Figure 4. 6 Screenshot of VS Code Interface with Project file

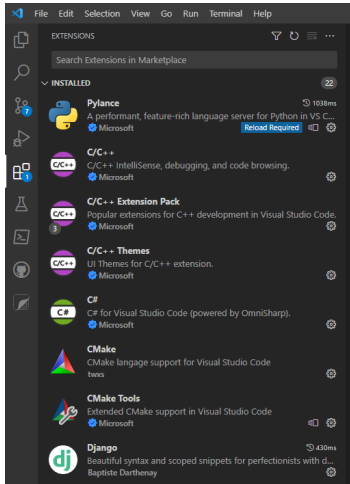


Figure 4. 7 Screenshot of VS Code Interface with Installed Extension

4.3 System Modules

The system for Web-Based Article Summarization with Machine Learning Techniques has been implemented based on the design presented in Chapter 3 of this report. The system is developed as a web application using the Django framework. This section will discuss the implementation of the graphical user interface system, Pre-processing text and the Programmed-based Machine Learning Techniques in the system.

4.3.1 Graphical User Interface (GUI) System

ArtSum is a cutting-edge Web-based article summarization system that uses machine learning techniques to generate precise and fast article summaries. ArtSum, which was created using the Django framework, a high-level Python web framework, features a simple yet elegant interface that enables users to input the original article on the left side of the screen and view the result on the right. To generate summaries, the system provides multiple algorithms, including Naive Bayes, Neural Network, and Decision Tree. Users can simply select the desired algorithm from the menu bar, input text into the left-hand text field, or upload a PDF file and click "Summarize" to receive a concise summary in the right-hand text field. ArtSum can be accessed on a variety of devices due to its responsive design, making it a valuable resource for anyone seeking to save time and obtain precise article summaries.

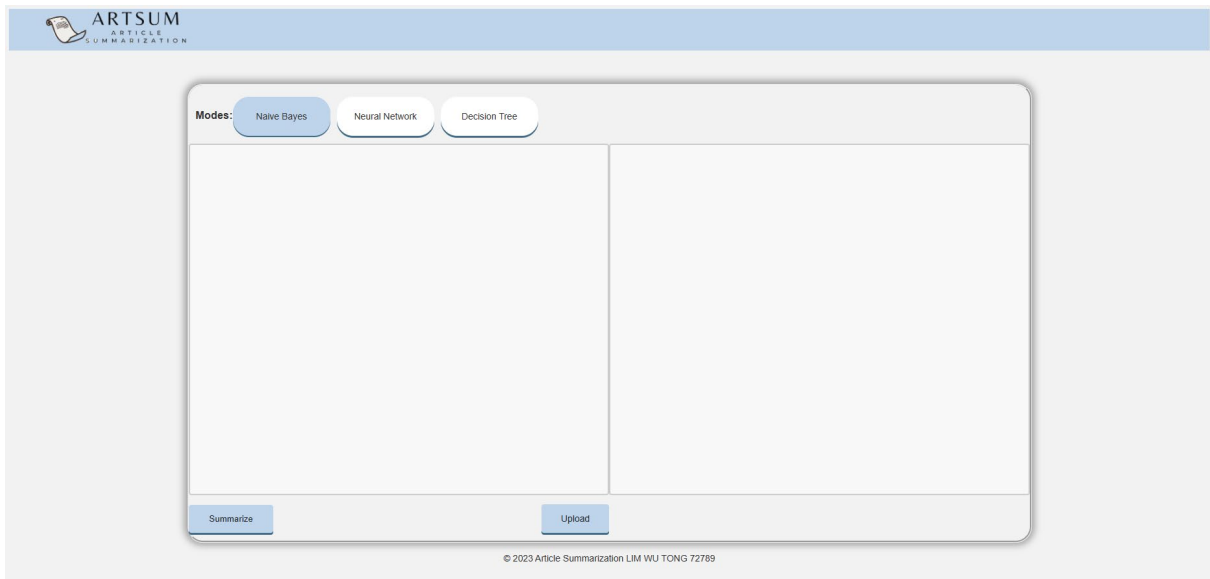


Figure 4. 8 The main page of ArtSum Web-based System

4.3.2 User class

The User class is a central component of the ArtSum article summarization system, serving as a bridge between the system's interface and the underlying machine learning algorithms. The User class contains two methods which are *upload_file* and *summarize*. The *upload_file* method takes a PDF file and extracts the text from it using the fitz library. It returns the extracted text as a string. The *summarize* method takes the extracted text as input and generates summaries for each section of the document. The method first defines a dictionary of section keywords and uses these to split the text into sections based on the headers. It then uses a machine learning algorithm to generate a summary for each section of the text. The algorithm used is determined by the mode parameter passed to the method, with the default being *naive_bayes*. The supported algorithms are *naive_bayes*, *neural_network*, and *decision_tree*, each with a corresponding method call. The resulting summaries are joined together and returned as a single string.

```

9     def upload_file(self, pdf_file):
10         text = ""
11         if pdf_file is not None:
12             with io.BytesIO(pdf_file.read()) as pdf_buffer:
13                 pdf_doc = fitz.open(stream=pdf_buffer.read(), filetype="pdf")
14                 for page_num in range(pdf_doc.page_count):
15                     page = pdf_doc.load_page(page_num)
16                     text += page.get_text()
17
18         return text

```

Figure 4. 9 Upload_file() in User class

```

20     # Generate summaries
21     def summarize(self, text, mode):
22         summary = ""
23         if mode is None:
24             # Define section keywords
25             section_keywords = {...}
26             # Split text into sections based on section headers
27             sections = {}
28             current_section = None
29             for line in text.splitlines():
30
31                 section_summaries = {}
32                 for text in sections:
33
34                     # Join section summaries with line breaks
35                     summary = "\n".join([f"{section.upper()}\n{section_summaries[section]}\n" for section in section_summaries])
36                     print(summary)
37                 return summary

```

Figure 4. 10 summarize() in User class

4.3.3 Pre-processing Text

In ArtSum, the pre-processing of text is conducted differently for the Naive Bayes and Decision Tree algorithms and the Neural Network algorithm. For Naive Bayes and Decision Tree, the text is tokenized into sentences using the *nltk.sent_tokenize* method, followed by removing all punctuation and converting all characters to lowercase using regular expressions. The *PorterStemmer* method is then applied to stem the words and remove any stop words using the *stopwords.words('english')* method. Finally, the pre-processed sentence is appended to a list of processed sentences. For the Neural Network algorithm, the text is tokenized using the *BartTokenizer* class from the transformers (Wolf, T. *et al.*, 2020) library. This class is specifically designed for tokenizing text for the Bart model, which is a state-of-the-art neural network model for sequence-to-sequence tasks. The *BartTokenizer* tokenizes the input text into a sequence of tokens that can be fed into the Bart model for further processing.

```

19 | sentences = nltk.sent_tokenize(text)
20 |
21 | # Preprocess the text
22 | processed_sentences = []
23 | for sentence in sentences:
24 |     words = re.sub('[^a-zA-Z]', ' ', sentence)
25 |     words = words.lower()
26 |     words = words.split()
27 |     ps = PorterStemmer()
28 |     words = [ps.stem(word) for word in words if not word in set(stopwords.words('english'))]
29 |     words = ' '.join(words)
30 |     processed_sentences.append(words)

```

Figure 4. 11 Pre-processing Text for Naïve Bayes and Decision Tree

```

class NeuralNetwork(MachineLearning):
    def __init__(self):
        super().__init__()
        self.model = AutoModelForSeq2SeqLM.from_pretrained('facebook/bart-large-cnn')
        self.tokenizer = BartTokenizer.from_pretrained('facebook/bart-large-cnn')

    def NN_generate_summary(self, text):
        tokenizer = self.tokenizer
        model = self.model
        if text is not None:
            input_ids = tokenizer(text, truncation=True, max_length=1024, padding='max_length', return_tensors='pt')

```

Figure 4. 12 Pre-processing Text for Neural Network

4.3.4 Naïve Bayes Summarizer

The NaiveBayes class is a machine learning model that generates a summary of a given text by classifying each sentence as a summary or non-summary sentence. The model uses the Multinomial Naive Bayes algorithm for classification and the *CountVectorizer* from the Scikit-learn (Pedregosa *et al.*, 2011) library for feature extraction.

To classify each sentence, the text is first tokenized by sentence and pre-processed. The resulting processed sentences are then used for feature extraction. The feature extraction process involves vectorizing the sentences using the *CountVectorizer*, which creates a sparse matrix of token counts for each sentence. Labels are then assigned to each sentence based on whether it is a summary or non-summary sentence. This is done by selecting the top '*num_sentences*' sentences with the highest token counts and assigning them the 'summary' label, while the remaining sentences are assigned the 'non-summary' label.

The model is trained on the features and labels using the *MultinomialNB()* function from Scikit-learn. The classifier is then used to predict the probability of each sentence being a summary sentence, based on which the top '*num_sentences*' sentences with the highest probabilities are selected as the summary sentences. Finally, the summary is generated by combining the selected summary sentences into a single string using the *join()* function.

```

44     # feature extraction
45     cv = self.tokenizer
46     sentence_features = cv.fit_transform(processed_sentences)
47
48     # creating labels for training data
49     summary_sentences_idx = heapq.nlargest(num_sentences, range(len(sentences)), key=lambda i: sentence_features[i].sum())
50     labels = ['summary' if i in summary_sentences_idx else 'non-summary' for i in range(len(sentences))]
51
52     # training the classifier
53     clf = self.model
54     clf.fit(sentence_features, labels)
55
56     # selecting the top 'num_sentences' sentences based on the classifier scores
57     sentence_scores = clf.predict_proba(sentence_features)[: , 0]
58     summary_sentences_idx = heapq.nlargest(num_sentences, range(len(sentences)), key=lambda i: sentence_scores[i])
59     summary_sentences = [sentences[idx] for idx in sorted(summary_sentences_idx)]
60
61     # combining the selected sentences to generate the summary
62     summary = ' '.join(summary_sentences)

```

Figure 4. 13 Naïve Bayes Summarize Process

4.3.5 Neural Network Summarizer

NeuralNetwork Class that is a subclass of MachineLearning. It uses a pre-trained BART-large-cnn model (M., Liu *et al.*, 2020) to provide a summary of the input text using the Hugging Face's Transformers library. By importing a pre-trained BART-large-cnn model and tokenizer from the Transformers (Wolf, T. *et al.*, 2020) library of the Hugging Face, the *__init__()* method initialises the model and tokenizer. The BART-large-cnn model is used by the *NN_generate_summary()* function to create a summary from the input text after it has been tokenized using the tokenizer. The summary is then produced and returned as a string. In particular, the tokenizer object tokenizes the input text, and the model object uses the tokenized input to provide a summary. The tokenized input text is stored in the *input_ids* variable, while the resulting summary is stored in the *summary_ids* variable. The result of the procedure, which

is the decoded version of the *summary_ids* without any special tokens, is stored in the *summary* variable.

```
1 < from transformers import AutoModelForSeq2SeqLM, BartTokenizer
2 < from ArticleSummarization.classes.machineLearning import MachineLearning
3
4 < class NeuralNetwork(MachineLearning):
5 <     def __init__(self):
6 <         super().__init__()
7 <         self.model = AutoModelForSeq2SeqLM.from_pretrained('facebook/bart-large-cnn')
8 <         self.tokenizer = BartTokenizer.from_pretrained('facebook/bart-large-cnn')
9
10 <     def NN_generate_summary(self, text):
11 <         tokenizer = self.tokenizer
12 <         model = self.model
13 <         if text is not None:
14 <             input_ids = tokenizer(text, truncation=True, max_length=1024, padding='max_length', return_tensors='pt')
15 <             summary_ids = model.generate(input_ids['input_ids'], num_beams=10, max_length=512)
16 <             summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)
17
18 <         return summary
```

Figure 4. 14 NeuralNetwork Class

4.3.6 Decision tree Summarizer

The *DecisionTree* class in the provided code generates a summary of the input text by first pre-processing the text into individual sentences and then selecting a subset of those sentences as the summary based on their predicted importance scores. To predict sentence importance, the class uses a decision tree regression model that is trained on bag-of-words (BoW) feature vectors derived from the pre-processed sentences.

Next, the pre-processed sentences are transformed into BoW feature vectors using the *CountVectorizer* from the scikit-learn library. The feature vectors are used as input to train a decision tree regression model, where the target variable is the length of each sentence. This is because shorter sentences are often more important for summarization than longer ones.

After training, the decision tree model is used to predict importance scores for each sentence in the pre-processed text. The top-scoring sentences, as determined by the importance scores, are then selected as the summary. The number of summary sentences to select can be

controlled by setting the *num_sentences* parameter. The final summary is generated by concatenating the selected sentences.

```
32     # Create Bow feature vectors
33     vectorizer = self.tokenizer
34     X = vectorizer.fit_transform(processed_sentences)
35
36     # Train a decision tree to predict sentence importance
37     y = [len(sentence) for sentence in sentences] # Use sentence length as target variable
38     dt = self.model
39     dt.fit(X, y)
40
41     # Use the decision tree to predict sentence importance scores
42     scores = dt.predict(X)
43
44     # Select the top-scoring sentences as the summary
45     num_sentences = 3 # Set the desired number of summary sentences
46     summary_indices = sorted(range(len(scores)), key=lambda i: scores[i], reverse=True)[:num_sentences]
47     summary_sentences = [sentences[i] for i in summary_indices]
48     summary = " ".join(summary_sentences)
49
50     return summary
```

Figure 4. 15 Decision Tree Summarize Process

4.4 Deployment

This section discusses the deployment process of the implemented system. First, the Google Cloud Platform was chosen as the deployment environment for its scalability and reliability. To begin the deployment, a Django stack VM instance was created from the marketplace. This instance provides the necessary infrastructure and dependencies for running the Django application.

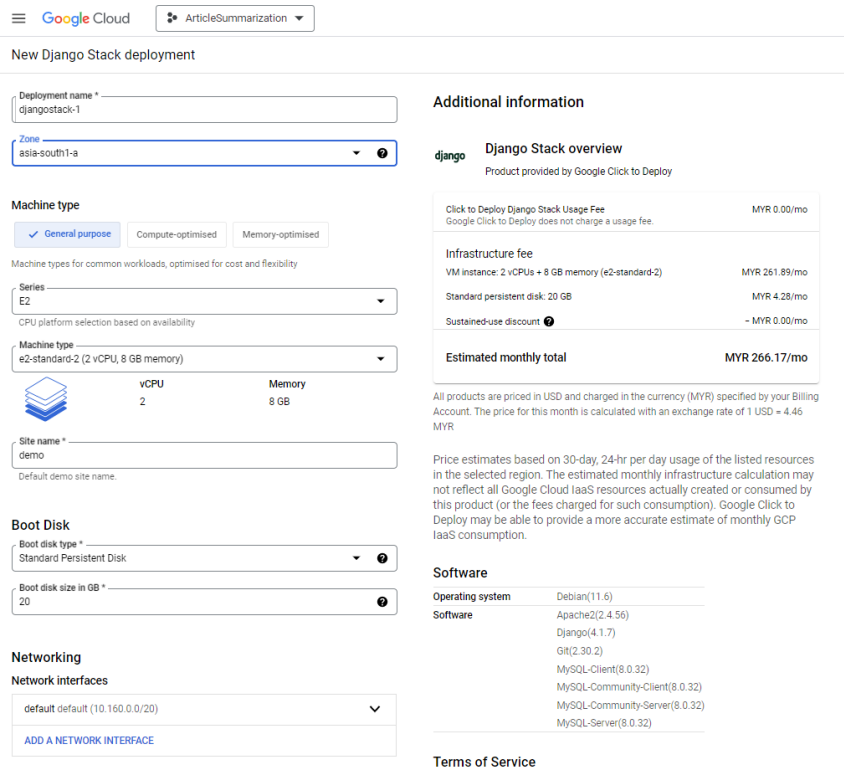


Figure 4. 16 Creating Django stack VM instance from the marketplace

Next, the project's source code was initialized with Git and pushed to a GitHub repository. This step allows for version control and easy access to the code from the deployment environment. The VM instance was then connected via SSH using the `gcloud` command, enabling remote access for configuration and management.

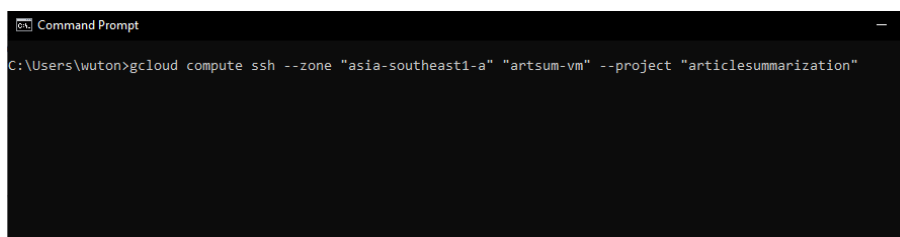


Figure 4. 17 Connecting VM instance via SSH

To ensure that the required dependencies are installed, several necessary libraries were installed on the VM instance. This included Python 3, Git, and Virtualenv. Python 3 was

installed to ensure compatibility with the project's dependencies, Git was needed for cloning the project repository, and Virtualenv provided an isolated Python environment for the project.

Once the necessary libraries were set up, the project repository was cloned onto the VM instance using the Git command. This retrieved the source code from the GitHub repository. Subsequently, a virtual environment was created to isolate the project's dependencies and ensure a clean and controlled environment for running the system.

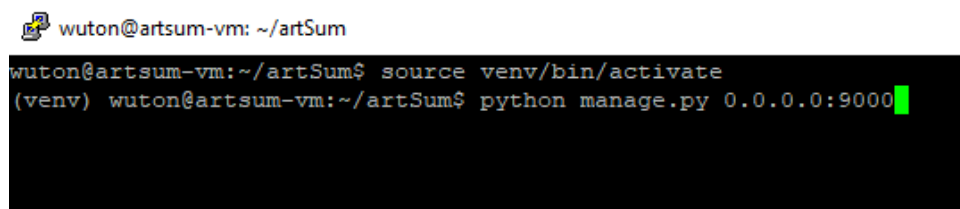
A terminal window with a black background and white text. The title bar shows 'wuton@artsum-vm: ~'. The command 'git clone https://github.com/wutong3027/artSum.git' is entered and executed, with a green cursor at the end of the line.

```
wuton@artsum-vm: ~  
wuton@artsum-vm:~$ git clone https://github.com/wutong3027/artSum.git
```

Figure 4. 18 Cloning the project repository

The project's required libraries were installed within the virtual environment using the command "pip install -r requirements.txt". This command installs all the dependencies specified in the requirements.txt file.

Finally, the web-based system was deployed by running the command "python manage.py runserver 0.0.0.0:9000". This started the Django server on port 9000, making the system accessible.

A terminal window with a black background and white text. The title bar shows 'wuton@artsum-vm: ~/artSum'. The commands 'source venv/bin/activate' and 'python manage.py 0.0.0.0:9000' are entered and executed, with a green cursor at the end of the second line.

```
wuton@artsum-vm: ~/artSum  
wuton@artsum-vm:~/artSum$ source venv/bin/activate  
(venv) wuton@artsum-vm:~/artSum$ python manage.py 0.0.0.0:9000
```

Figure 4. 19 Running the project and deploying the system

By following these steps, the web-based system was successfully deployed on the Google Cloud Platform using a Django stack VM instance. The system can now be accessed and utilized through the specified IP address and port.

4.5 Summary

The implementation of a Web-based Article Summarization system using machine learning techniques was discussed in this report. The system was developed using Python programming language, Django framework, HTML/CSS, and machine learning algorithms such as Naive Bayes, Neural Network, and Decision Tree. The system's graphical user interface enables users to input text or upload a PDF file and receive a concise summary generated by the machine learning algorithms. The hardware and software environment used for the implementation was moderate, and the system was successfully deployed on a web domain for accessibility. The system's capabilities make it a valuable resource for anyone seeking to save time and obtain precise article summaries. Overall, the implementation process was successful, and the system's functionality and usability demonstrate the potential of machine learning techniques in text summarization tasks.

Chapter 5: Evaluation, Testing and Result

5.1 Introduction

This chapter provides an in-depth analysis of the evaluation, testing, and results of the implemented system. This chapter begins with an introduction to the importance of evaluation and testing in assessing the system's performance and functionality.

5.2 Evaluation

A comprehensive evaluation was conducted to assess the effectiveness and efficiency of proposed machine learning techniques for text summarization. The evaluation focused on summarization quality, preferredness, and time efficiency. The evaluation phase measured the effectiveness and performance of the web-based article summarization system using ROUGE (Lin, 2004) metrics, the System Usability Scale (SUS) (Brooke, 1996) questionnaire, and expert evaluation. This section discusses the metrics and methodologies employed, including the application of ROUGE metrics and the SUS questionnaire, as well as the involvement of an English expert from the Faculty of Language and Communication at UNIMAS in evaluating the summary quality. The time efficiency evaluation involved measuring the processing time required for the system to produce a summary for a given text.

5.2.1 Summarization Quality Evaluation

The evaluation of summarization quality aimed to assess the effectiveness of the proposed machine learning techniques in generating accurate and informative summaries. Two primary methods were employed for this evaluation: ROUGE metrics and expert evaluation. The study by Chua et al. (2018) entitled "A Comparative Study of Sentiment-Based Graphs of Text Summaries" served as our inspiration for evaluating the summary quality. While our

evaluation techniques differed in certain aspects, their work provided valuable insights into text summary evaluation and influenced the development of our evaluation criteria.

5.2.1.1 ROUGE Metrics Evaluation

The Recall-Oriented Understudy for Gisting Evaluation (ROUGE) metrics were utilized to measure the similarity between the system-generated summaries and the representative summaries generated by the Quillbot summarizer, which served as a proxy for human-generated summaries. The ROUGE metrics used in this evaluation included ROUGE-N, ROUGE-L, and ROUGE-S, which captured different aspects of summarization quality, such as lexical overlap, structural similarity, and content-related similarities (Lin, 2004).

By calculating precision, recall, and F1 scores based on these metrics, a comprehensive analysis was conducted to determine the extent of overlap and similarity between the system-generated summaries and the representative summaries generated by the Quillbot summarizer. The precision score measures the proportion of correctly identified summary content, recall captures the completeness of the summary, and the F1 score combines both precision and recall into a single metric (Lin, 2004).

This evaluation approach provided objective and quantitative assessment of the summarization quality, considering the representative summaries generated by the Quillbot summarizer as a reference for human-generated summaries. It provided valuable insights into the performance of each technique, including Naive Bayes, Neural Network, and Decision Tree, and served as a benchmark for comparing the effectiveness of these approaches in generating accurate and informative summaries.

Table 5. 1 ROUGE Evaluation Results

Technique	ROUGE-1 Recall	ROUGE-1 Precision	ROUGE-1 F1 Score	ROUGE-2 Recall	ROUGE-2 Precision	ROUGE-2 F1 Score	ROUGE-L Recall	ROUGE-L Precision	ROUGE-L F1 Score
Naive Bayes	0.385	0.427	0.405	0.190	0.214	0.201	0.378	0.419	0.398
Neural Network	0.393	0.593	0.473	0.176	0.302	0.222	0.375	0.566	0.451
Decision Tree	0.636	0.433	0.515	0.375	0.258	0.306	0.622	0.423	0.504

The ROUGE evaluation was conducted to assess the performance of the different machine learning techniques in generating summaries. The results of the ROUGE evaluation are presented in the table above.

The Naïve Bayes technique achieved moderate performance in generating summaries. It had a recall of 0.385, indicating that it captured 38.5% of the relevant information. The precision score of 0.427 suggests that 42.7% of the generated summary content was correct. The F1-score of 0.405 represents a balanced assessment of the summarization system's performance.

The Neural Network technique showed improvement compared to Naïve Bayes. It had a recall of 0.393, indicating a slightly higher ability to capture relevant information. The precision score of 0.593 suggests a higher accuracy of the generated summaries. The F1-score of 0.473 represents a balanced performance in terms of recall and precision.

The Decision Tree technique exhibited the highest scores among the three techniques. It achieved a recall of 0.636, indicating a better ability to capture relevant information compared to the other techniques. The precision score of 0.433 suggests a lower level of irrelevant or

incorrect information in the generated summaries. The F1-score of 0.515 represents a balanced assessment of the summarization system's performance.

5.2.1.2 Expert Evaluation

In addition to the evaluation using ROUGE metrics, an expert evaluation of the system-generated summaries was conducted by Madam Siti Marina binti Kamil, a language expert and Senior Lecturer from the Faculty of Language and Communication at Universiti Malaysia Sarawak (UNIMAS). Her expertise in English language and extensive experience in linguistic analysis made her a valuable evaluator for the system-generated summaries.

Madam Siti Marina binti Kamil carefully examined the summaries generated by the Naive Bayes, Neural Network, and Decision Tree techniques. Her evaluation focused on assessing the quality of the summaries from a linguistic perspective, with specific attention to grammatical correctness, coherence, and overall readability. Through her qualitative analysis, she provided feedback on the linguistic quality of each summary, ensuring that they met high standards in terms of language usage, coherence, and accuracy.

According to Madam Siti Marina binti Kamil's evaluation, the Naive Bayes technique summary and Decision Tree technique summary were considered quite accurate in providing sufficient information from the original article. However, she found that the Neural Network technique summary was too general. Her expert evaluation played a crucial role in determining the extent to which the generated summaries effectively captured the main ideas and key points of the original articles, considering grammatical correctness and overall readability. The evaluation form filled by Madam Siti Marina will be at APPENDIX C.

Madam Siti Marina binti Kamil's expertise and evaluation provided valuable insights into the strengths and weaknesses of each technique's summarization capabilities. Her

assessment contributed to the refinement and improvement of the system's ability to generate high-quality summaries.

5.2.2 Preferredness Evaluation

The preferredness evaluation aimed to assess the user satisfaction and usability of the system. For this purpose, the System Usability Scale (SUS) (Brooke, 1996) questionnaire was administered to a group of final-year students, representative of the intended user group.

Based on the ratings provided by the participants, an overall assessment of the system's preferredness can be drawn. The majority of participants (70%) expressed a positive inclination towards using the system frequently, indicating their interest in utilizing it regularly. They found the system to be easy to use (75%) and well-integrated (85%), contributing to a cohesive user experience. Additionally, a significant portion (70%) believed that most people would quickly learn how to use the system, highlighting its user-friendly nature. However, a small percentage found the system unnecessarily complex (10%) or encountered inconsistencies (15%). Some participants also expressed the need for additional knowledge or skills before fully utilizing the system (10%). These insights provide valuable feedback to assess the system's strengths and areas for improvement, ensuring enhanced user satisfaction and usability.

The SUS evaluation provided valuable insights into the users' preferredness of the system. It helped identify strengths and weaknesses in terms of user experience, highlighting areas that were well-received by users and areas that may require improvement. By analysing the participants' responses, patterns and trends were identified, contributing to a comprehensive understanding of the system's preferredness among its intended users.

The preferredness evaluation utilized the System Usability Scale (SUS) questionnaire to assess user satisfaction and usability.

5.2.3 Time Efficiency Evaluation

The time efficiency evaluation aimed to assess the speed and efficiency of the machine learning techniques employed for generating summaries, specifically for the Naive Bayes, Neural Network, and Decision Tree models. The evaluation focused on ensuring that the system could generate high-quality summaries within reasonable timeframes.

To measure time efficiency, the processing time required for each machine learning technique to generate summaries was recorded. This evaluation involved measuring the elapsed time before and after the summarization process. The difference in time provided insights into the efficiency of each technique in producing the desired summaries. By evaluating the time efficiency of the system, it was possible to analyse the performance and determine if any optimizations were necessary to enhance the overall efficiency of the system.

Table 5. 2 Processing Time for Each Summarizer Techniques

Technique	Processing Time
Naive Bayes	0m 1s
Neural Network	5m 6s
Decision Tree	0m 1s

The Naive Bayes and Decision Tree techniques demonstrated efficient processing times, completing the summarization task quickly, with both techniques requiring only 1 second. However, it is important to note that the processing time can be influenced by various factors, including the computational power of the system running the algorithms. More powerful hardware and resources can potentially reduce the processing time for computationally intensive techniques.

On the other hand, the Neural Network technique took significantly longer, with a processing time of 5 minutes and 6 seconds. This suggests that the Neural Network technique may be more computationally intensive and may require more computational resources or optimization techniques to improve its time efficiency.

By evaluating the time efficiency of the system, it was possible to analyse the performance of each technique and determine if any optimizations were necessary to enhance the overall efficiency of the system. Additionally, the processing time highlights the importance of considering the computational power and resources available when using machine learning techniques for text summarization tasks.

5.3 Testing

To ensure the reliability and functionality of the system, a comprehensive testing phase was conducted. The testing process involved the use of unit testing to verify the correctness of individual components and functionalities within the system.

Unit testing focused on testing each module, function, or method of the system in isolation, ensuring that they performed as intended and produced the expected results. This approach allowed for the identification and resolution of any defects or bugs within specific components, ensuring the overall stability and reliability of the system.

Various test cases were designed and executed to cover different scenarios and edge cases, thoroughly examining the system's behaviour and performance under various conditions. The unit testing process involved providing inputs, executing the corresponding functions, and comparing the output against expected results.

By conducting unit testing, potential issues and errors were identified early in the development process, enabling prompt fixes and improvements. This iterative testing approach contributed to the overall quality and robustness of the system.

Table 5. 3 Test Case

Test Case Description	Test Method	Expected Result	Actual Result	Pass/Fail
Test successful upload	test_successful_upload	JSON response contains uploaded text	JSON response matches expected result	Pass
Test failed upload	test_failed_upload	JSON response contains error message	JSON response matches expected result	Pass
Test successful summarization	test_successful_summarization	JSON response contains generated summary	JSON response matches expected result	Pass
Test failed summarization	test_failed_summarization	JSON response contains error message	JSON response matches expected result	Pass

5.4 Result

The evaluation phase aimed to assess the effectiveness, preferredness, and time efficiency of the implemented system. The evaluation process consisted of multiple components, including the evaluation of summarization quality, preferredness, and testing.

The evaluation of summarization quality utilized a combination of ROUGE (Lin, 2004) metrics analysis and expert evaluation. The results revealed that the Decision Tree technique achieved the highest recall, precision, and F1 scores among the three techniques, indicating its superior performance in generating accurate and informative summaries. The expert evaluation found the Naive Bayes and Decision Tree technique summaries to be accurate in providing sufficient information, while the Neural Network technique summary was deemed too general,

contributing to a comprehensive assessment of the summaries' quality and their adherence to grammatical correctness and overall readability, as detailed in Appendix C.

In terms of preferredness, user satisfaction and usability were assessed using the System Usability Scale (SUS) (Brooke, 1996) questionnaire. The participants' ratings provide an overall assessment of the system. Most participants expressed a positive inclination towards using the system frequently and found it easy to use. They also perceived the various functions to be well integrated. However, a minority found the system unnecessarily complex and identified inconsistencies. Overall, the system was regarded as user-friendly and instilled confidence, although some participants felt the need for additional learning. These insights highlight both strengths and areas for improvement.

The evaluation also included a time efficiency assessment, which focused on measuring the processing time required for each machine learning technique to generate summaries. The Naive Bayes and Decision Tree techniques demonstrated efficient processing times, completing the summarization task within 1 second. However, the Neural Network technique exhibited significantly longer processing time, indicating a need for further optimization in terms of time efficiency.

To ensure the reliability and functionality of the system, a comprehensive testing phase was conducted. Unit testing was employed to verify the correctness of individual components and functionalities in isolation. Various test cases were designed and executed to thoroughly examine the system's behaviour and performance under different conditions. Through this iterative testing approach, potential issues and errors were identified early in the development process, allowing for prompt fixes and improvements. The system successfully passed all test cases, demonstrating its ability to handle various scenarios and produce the expected outcomes.

In conclusion, the evaluation results highlight the strengths and areas for improvement of the proposed machine learning techniques for text summarization. The Decision Tree technique performed exceptionally well in terms of summarization quality, while the system showed positive user satisfaction and usability. The testing phase confirmed the reliability and functionality of the system. However, further optimization is recommended to improve the time efficiency of the Neural Network technique.

5.5 Summary

Chapter 5 discussed a comprehensive evaluation, testing, and result analysis of the implemented system. The Decision Tree technique shows superior performance in terms of summarization quality, and users generally find the system usable and informative. The testing phase confirms the reliability and functionality of the system. However, further optimization is recommended to improve the time efficiency of the Neural Network technique. Overall, the evaluation, testing, and results provide valuable insights into the system's strengths and areas for improvement.

Chapter 6: Conclusion and Future Work

6.1 Introduction

Chapter 6 serves as the conclusion of the project, summarizing the achieved objectives, limitations encountered, and future work. It provides an overview of the implemented web-based system for article summarization and the machine learning techniques used.

6.2 Achievements

The objectives of the project were successfully achieved, including the design and implementation of a web-based system that integrates machine learning algorithms for article summarization. The system utilizes Python-based machine learning techniques to generate article summaries. The effectiveness and efficiency of the proposed machine learning techniques were analysed based on summarization quality, preferredness, and time efficiency.

6.3 Limitations

Several limitations were identified in the system. Firstly, it requires user's effort in processing the input text. They need to manually remove watermarks or table information from the scientific paper article in the text area after uploading it to improve the quality of the generated summary. Additionally, the system cannot accurately recognize section headers in scientific paper articles, resulting in the categorization of unrecognized sections under the "overall" category.

6.4 Future Works

Future works can focus on addressing the identified limitations. Improvements can be made to automate the removal of watermarks or table information from scientific paper articles to enhance the pre-processing stage. Additionally, efforts can be made to develop techniques that accurately recognize and utilize section headers in article for more precise summarization.

6.6 Summary

Chapter 6 concludes the project by summarizing the achieved objectives, limitations, and future work. The web-based system successfully integrates machine learning algorithms for article summarization. However, limitations exist, such as the manual removal of watermarks and the inability to accurately recognize section headers in scientific papers. Future work can address these limitations and further enhance the system's performance and usability. The project contributes to the field of article summarization and lays the foundation for future research and development in this area.

References

- Ahmed, N., Naushad, U., & Israr, M. (2020). Automated document summarization using decision tree algorithm. *In 2020 7th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 663-668.
- Akmal, S., Dhivah, I., & Mulia, M. (2020). Investigating Students' Interest on Reading Journal Articles: Materials, Reasons and Strategies. *Studies in English Language and Education*, 7(1), 194-208.
- Allahyari, M., Pouriye, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., & Kochut, K. (2017). Text summarization techniques: a brief survey. *arXiv preprint arXiv:1707.02268*.
- Ansari, S. (2022, August 10). *What Are the Advantages of Summary Generator for Your Article?* Retrieved from ThePrint: <https://theprint.in/theprint-valuead-initiative/what-are-the-advantages-of-summary-generator-for-your-article/1076679/>
- Ball, P. (2017). It's not just you: science papers are getting harder to read. *Nature*.
- Brooke, J. (1996). SUS-A quick and dirty usability scale. *Usability evaluation in industry*, 189(194), 4-7.
- Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. *Proceedings of the 22nd international conference on Machine learning*, pp. 89-96.
- Chuang, W. T., & Yang, J. (2000). Extracting sentence segments for text summarization: a machine learning approach.

- Conroy, J. M., & O'leary, D. P. (2001). Text summarization via hidden markov models. *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 406-407.
- Dematos, G., Boyd, M. S., Kermanshahi, B., Kohzadi, N., & Kaastra, I. (1996). Feedforward versus recurrent neural networks for forecasting monthly japanese yen exchange rates. *Financial Engineering and the Japanese Markets*, 3(1),59-75.
- Django Software Foundation. (2019). *Django*. Retrieved from <https://djangoproject.com>
- Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., ... & Hon, H. W. (2019). Unified language model pre-training for natural language understanding and generation. *Advances in neural information processing systems*, 32.
- Dwivedi, R. (2020, April 30). *What Is Naive Bayes Algorithm In Machine Learning?* Retrieved from Analytics Steps: <https://www.analyticssteps.com/blogs/what-naive-bayes-algorithm-machine-learning>
- El Naqa, I., & Murphy, M. J. (2015). *What is machine learning?* Springer International Publishing Switzerland.
- Fatima, Q., & Cenek, M. (2015). New graph-based text summarization method. *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, (pp. 396-401). IEEE.
- Garg, A., & Joshi, A. (2018). Text classification using decision trees in scikit-learn. *In 2018 2nd International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 261-265.

- Hall, S. (2017). *Definition of a Research Article*. Retrieved from Pen & The Pad: <https://penandthepad.com/present-journal-article-5117494.html>
- Iboi, H., Chua, S., Ranaivo-Malançon, B., & Kulathuramaiyer, N. (2017). Performance of Opinion Summarization towards Extractive Summarization. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 9(2-10), 57-64.
- Jones, K. S. (1993). What might be in a summary? *Information retrieval*, 93(1), 9-26.
- Kaikhah, K. (2004). Automatic text summarization with neural networks. *2004 2nd International IEEE Conference on 'Intelligent Systems'. Proceedings (IEEE Cat. No.04EX791)*, pp. 40-44 Vol.1, doi: 10.1109/IS.2004.1344634.
- Kaikhah, K. (2004). Text summarization using neural. *Faculty Publications-Computer Science*.
- Kupiec, J., Pedersen, J., & Chen, F. (1995). A trainable document summarizer. *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 68-73.
- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint*, arXiv:1910.13461.
- Lin, C. Y. (2004). Rouge: A package for automatic evaluation of summaries. *Text summarization branches out*, (pp. 74-81).
- M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... Zettlemoyer, L. (2020). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation. *Translation, and Comprehension*, arXiv preprint arXiv:1910.13461.

- Microsoft. (2022). *Visual Studio Code [Computer Software]*. Retrieved from <https://code.visualstudio.com>
- Mirza, S., Mittal, S., & Zaman, M. (2018). Applying decision tree for prognosis of diabetes mellitus.
- Mitchell, T. M. (1997). *Machine learning*. New York: McGraw-hill.
- Nüst, D., Boettiger, C., & Marwick, B. (2018). How to read a research compendium. *arXiv:1806.09525*.
- Osborne, M. (2002). Using maximum entropy for sentence extraction. *Proceedings of the ACL-02 Workshop on Automatic Summarization*, pp. 1-8.
- Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, pp. 2825-2830.
- Python, S. F. (2020). *Python Programming Language*. Retrieved from <https://www.python.org/>
- QuillBot, I. (2017). *Quillbot Summarizer*. Retrieved from QuillBot: <https://quillbot.com/summarize>
- Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1), 81-106.
- Quinlan, R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.
- Saranyamol, C. S., & Sindhu, L. (2014). A survey on automatic text summarization. *International Journal of Computer Science and Information Technologies*, International Journal of Computer Science and Information Technologies.
- Schmidt, S. J. (2020). Distracted learning: Big problem and golden opportunity. *Journal of Food Science Education*, 19(4), 278-291.

- Scholarcy. (2019). *Article Summarizer Scholarcy*. Retrieved from Scholarcy: <https://article-summarizer.scholarcy.com/>
- Sharma, H., & Kumar, S. (2016). A survey on decision tree algorithms of classification in data mining. *International Journal of Science and Research (IJSR)*, 5(4), 2094-2097.
- Subramanyam, R. V. (2013). Art of reading a journal article: Methodically and effectively. *Journal of oral and maxillofacial pathology: JOMFP*, 17(1), 65.
- Svore, K., Vanderwende, L., & Burges, C. (2007). Enhancing single-document summarization by combining RankNet and third-party sources. *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pp. 448-457.
- Tenopir, C., King, D. W., Clarke, M. T., Na, K., & Zhou, X. (2007). Journal reading patterns and preferences of pediatricians. *Journal of the Medical Library Association*, 95(1), 56.
- Thu, H. N. (2014). An optimization text summarization method based on naive bayes and topic word for single syllable language. *Applied mathematical sciences*, 8(3), 99-115.
- Trappey, A. J., Trappey, C. V., Wu, J. L., & Wang, J. W. (2020). Intelligent compilation of patent summaries using machine learning and natural language processing techniques. *Advanced Engineering Informatics*, 43, 101027.
- Ungureanu, A., & Ungureanu, A. (2014). Methodologies Used in Project Management. *Annals of Spiru Haret University, Economic Series*, 5(2), 47-53.
- Wang, S. C. (2003). Artificial neural network. In S. C. Wang, *Interdisciplinary computing in java programming* (pp. pp. 81-100). Boston, MA: Springer.

Wolf, T., Sanh, V., Chaumond, J., & Delangue, C. (2020). *Transformers: State-of-the-Art Natural Language Processing*. Association for Computational Linguistics.

Yogan, J. K., Goh, O. S., Halizah, B., Ngo, H. C., & Puspalata, C. (2016). A review on automatic text summarization approaches. *Journal of Computer Science*, 12(4), 178-190.

APPENDIX A

System Usability Scale (SUS)

	Strongly Disagree				Strongly Agree
	1	2	3	4	5
1. I think that I would like to use this system frequently					
2. I found the system unnecessarily complex					
3. I thought the system was easy to use					
4. I think that I would need the support of a technical person to be able to use this system					
5. I found the various functions in this system were well integrated					
6. I thought there was too much inconsistency in this system					
7. I would imagine that most people would learn to use this system very quickly					
8. I found the system very cumbersome to use					
9. I felt very confident using the system					
10. I needed to learn a lot of things before I could get going with this system.					

APPENDIX B

Source Code of ArtSum

articleSummarization/manage.py

```
#!/usr/bin/env python

"""Django's command-line utility for administrative tasks."""

import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
        'articleSummarization.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()
```

requirements.txt

```
asgiref==3.6.0
certifi==2023.5.7
charset-normalizer==3.1.0
click==8.1.3
colorama==0.4.6
```

```
Django==4.2.1
filelock==3.12.0
fsspec==2023.5.0
huggingface-hub==0.14.1
idna==3.4
Jinja2==3.1.2
joblib==1.2.0
MarkupSafe==2.1.2
mpmath==1.3.0
networkx==3.1
nltk==3.8.1
numpy==1.24.3
packaging==23.1
PyMuPDF==1.22.3
PyYAML==6.0
regex==2023.5.5
requests==2.31.0
scikit-learn==1.2.2
scipy==1.10.1
sqlparse==0.4.4
sympy==1.12
threadpoolctl==3.1.0
tokenizers==0.13.3
torch==2.0.1
tqdm==4.65.0
transformers==4.28.1
typing_extensions==4.5.0
tzdata==2023.3
urllib3==2.0.2
```

articleSummarization/settings.py

```
"""
```

```
Django settings for articleSummarization project.
```

```
Generated by 'django-admin startproject' using Django 4.0.4.
```

```
For more information on this file, see
```

```
https://docs.djangoproject.com/en/4.0/topics/settings/
```

```
For the full list of settings and their values, see
```

```
https://docs.djangoproject.com/en/4.0/ref/settings/
```

```
"""
```

```
from pathlib import Path
```

```

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-
hrkox*ejkfndq#!^ibnqjj$r*0)&f#q1jkn(w8b=d8vw(8im!'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['*']

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'artSum'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'articleSummarization.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ['templates'],
        'APP_DIRS': True,
    }
]

```

```

        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    ],
]

WSGI_APPLICATION = 'articleSummarization.wsgi.application'

# Database
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

# Password validation
# https://docs.djangoproject.com/en/4.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
        'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
        'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
        'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
        'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

```

```

# Internationalization
# https://docs.djangoproject.com/en/4.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.0/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

```

articleSummarization/urls.py

```

"""articleSummarization URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.0/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('artSum.urls')),

```

```

    path('upload/', include('artSum.urls')),
    path('summarize/', include('artSum.urls')),
    path('upload/summarize/', include('artSum.urls')),
]

```

artSum/urls.py

```

from django.urls import path
from . import views

urlpatterns = [
    path('', views.home),
    path('upload/', views.upload),
    path('summarize/', views.summarize),
    path('upload/summarize/', views.summarize),
]

```

artSum/views.py

```

import nltk
from django.shortcuts import render
from django.http import JsonResponse
from artSum.classes.user import User
from artSum.classes.article import Article
from artSum.classes.naiveBayes import NaiveBayes
from artSum.classes.neuralNetwork import NeuralNetwork
from artSum.classes.decisionTree import DecisionTree
nltk.download('punkt')
nltk.download('stopwords')

user = User()
nb = NaiveBayes()
nn = NeuralNetwork()
dt = DecisionTree()
# Create your views here.

def home(request):
    return render(request, 'home.html', {'name': 'Django'})

def upload(request):
    if request.method == 'POST':
        try:
            text = request.POST.get('text', '')
            pdf_file = request.FILES.get('pdf_file', None)

```

```

        text = user.upload_file(pdf_file)
        article = Article(pdf_file.name, text)
        num_words = len(text.split())
        return JsonResponse({'text': text, 'num_words': num_words})
    except Exception as e:
        num_words = 0
        return JsonResponse({'text': 'File upload failed. Please try
again.', 'num_words': num_words})

    else:
        return render(request, 'home.html')

# Generate summaries
def summarize(request):
    try:
        # Get text from POST request
        text = request.POST.get('text', '')
        mode = request.POST.get('mode', 'naive_bayes') # Default to naive
        bayes if mode is not provided
        summary = user.summarize(text, mode)
        summary_count = len(summary.split())
        return JsonResponse({'summary': summary, 'summary_count':
summary_count})
    except Exception as e:
        summary_count = 0
        return JsonResponse({'summary': 'Summarization failed. Please try
again.', 'summary_count': summary_count})

```

artSum/templates/home.html

```

<!-- article summarization web page-->
{% load static %}
<html>
  <head>
    <title>ArtSum</title>
    <link rel="stylesheet" href="{% static 'css/styles.css'%}">
    <link rel="icon" type="image/png" href="{% static 'ArtSum/2.png'%}">
  </head>
  <body>
    <header>
      <nav>
        <div class="nav-wrapper">
          <a href="" class="brand-logo"></a>

        </div>

```

```

        </nav>
    </header>
    <!--form input article at left and textfield output result at right-->
    <div class="container">
        <div class="menuBar">
            <h3>Modes:</h3>
            <div>
                <input type="radio" id="naive_bayes" name="mode"
value="naive_bayes" checked>
                <label for="naive_bayes">Naive Bayes</label>
            </div>
            <div>
                <input type="radio" id="neural_network" name="mode"
value="neural_network">
                <label for="neural_network">Neural Network</label>
            </div>
            <div>
                <input type="radio" id="decision_tree" name="mode"
value="decision_tree">
                <label for="decision_tree">Decision Tree</label>
            </div>
        </div>
        <div class="leftRightContainer">
            <div class="left">
                <form>
                    {% csrf_token %}
                    <textarea name="text">{{ text }}</textarea>
                    <input class="submit" type="submit" value="Summarize"
onclick="summarize(); return false;" />
                </form>
                <form action="upload/" method="POST"
enctype="multipart/form-data" id="summarize-form">
                    {% csrf_token %}
                    <label class="uploadtag">
                        Upload
                        <input class="upload" type="file" name="pdf_file"
accept=".pdf" onchange="upload()"/>
                    </label>
                </form>
                <center><div class="word-count" id="word-count"
style="display:none;"><span id="word-count"></span> Words </div></center>
            </div>
            <div class="right">
                <div id="loading" class="loading"></div>
                <form>
                    {% csrf_token %}
                    <textarea readonly name="summary">

```



```

    })
    .catch(function (error) {
        console.log(error);
    });
}

function summarize() {
    var startTime = performance.now();
    // clear the summary textarea
    document.getElementsByName('summary')[0].value = '';
    // Show the loading animation
    document.getElementById('loading').style.display = 'block';
    // Get the selected algorithm
    var algorithm;
    if (document.getElementById('naive_bayes').checked) {
        algorithm = 'naive_bayes';
    } else if (document.getElementById('neural_network').checked) {
        algorithm = 'neural_network';
    } else if (document.getElementById('decision_tree').checked) {
        algorithm = 'decision_tree';
    }

    // Send a POST request to the summarize view using the fetch API
    var formData = new FormData();
    formData.append('text',
document.getElementsByName('text')[0].value);
    formData.append('mode', algorithm);
    fetch('summarize/', {
        method: 'POST',
        body: formData,
        headers: {
            'X-CSRFToken': "{{ csrf_token }}"
        }
    }).then(function(response) {
        return response.text();
    }).then(function(summary) {
        document.getElementById('loading').style.display = 'none';
        document.getElementById('summarydiv').style.display = 'block';
        var jsonString = summary;
        var jsonObject = JSON.parse(jsonString);
        var summaryResult = jsonObject.summary;
        console.log(summary)
        document.getElementsByName('summary')[0].value =
summaryResult;
        var summaryCount = jsonObject.summary_count;
        var summaryCountLabel =
document.querySelector('span[id="summary-count"]');

```

```

summaryCountLabel.innerHTML = summaryCount;
// Calculate the elapsed time
var elapsedTime = performance.now() - startTime;

// Convert the elapsed time to a human-readable format
var elapsedSeconds = Math.round(elapsedTime / 1000);
var elapsedMinutes = Math.floor(elapsedSeconds / 60);
var elapsedSecondsRemainder = elapsedSeconds % 60;
var elapsedFormatted = elapsedMinutes + "m " +
elapsedSecondsRemainder + "s";

// Update the time counter element in the HTML
var timeCounter = document.getElementById("time-counter");
timeCounter.textContent = elapsedFormatted;
});

}
</script>
</html>

```

artSum/static/styles.css

```

body {
    font-family: 'Montserrat', sans-serif;
    background-color: #f5f5f5;
}

body {
    background-color: #f2f2f2;
    font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
    font-size: 14px;
    line-height: 1.42857143;
    color: #333;
}

.logo {
    height: 56px;
}

h1 {
    font-size: 36px;
    font-weight: 300;
    line-height: 1.1;
    color: inherit;
}

```

```

        margin-top: 20px;
        margin-bottom: 10px;
        text-align: center;
    }
    nav .nav-wrapper {
        position: relative;
        height: 100%;
    }
    nav {
        color: #212529;
        background-color: #BED4EB;
        width: 100%;
        height: 56px;
        line-height: 56px;
    }
    nav, nav .nav-wrapper i, nav a.sidenav-trigger, nav a.sidenav-trigger i {
        height: 64px;
        line-height: 64px;
    }
    nav .brand-logo {
        position: absolute;
        color: #212529;
        display: inline-block;
        font-size: 2.1rem;
        padding: 0;
        padding-left: 50px;
    }
    .nav-wrapper a{
        text-decoration: none;
    }
    .container{
        margin: auto;
        width: 70%;
        height: 700px;
        display: flex;
        flex-wrap: wrap;
        flex-direction: row;
        justify-content: space-between;
        border-style: groove;
        border-radius: 25px;
        margin-top: 50px;
        box-shadow: 0 0 10px 5px rgba(0,0,0,0.3);
    }
    .menuBar{
        width: 100%;
        height: 80px;
        display: flex;

```

```

}
.menuBar h3{
    float: left;
    list-style: none;
    text-align: center;
    margin-left: 10px;
    line-height: 60px;
}

input[type="radio"] {
    display: none;
}

.menuBar label {
    float: left;
    list-style: none;
    text-align: center;
    background-color: #fff;
    margin-right: 10px;
    margin-top: 18px;
    width: 150px;
    line-height: 60px;
    border-radius: 25px;
    box-shadow: 0 3px #426b87;
    cursor: pointer;
    text-align: center;
    font-size: 18px;
}

label:hover {
    background-color: #6394b6;
}

input[type="radio"]:checked + label {
    background-color: #BED4EB;
    box-shadow: 0 2px #426b87;
}

.leftRightContainer{
    width: 100%;
    height: 600px;
}
.left{
    width: 50%;
    float: left;
    height: 100%;
}

```

```

.right{
    margin-left: 50%;
    height: 100%;
}
textarea {
    width: 100%;
    height: 90%;
    font-size: 16px;
    padding: 12px 20px;
    box-sizing: border-box;
    border: 2px solid #ccc;
    border-radius: 4px;
    background-color: #f8f8f8;
    resize: none;
    outline: none;
}
input[type=submit] {
    width: 20%;
    background-color: #BED4EB;
    color: #212529;
    box-shadow: 0 3px #426b87;
    padding: 14px 20px;
    margin: 15px 0;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    float:left;
    font-size: 16px;
}
input[type=submit]:hover {
    background-color: #6394b6;
}
.uploadtag{
    width: 10%;
    background-color: #BED4EB;
    color: #212529;
    box-shadow: 0 3px #426b87;
    padding: 12px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    text-align: center;
    float: right;
    font-size: 16px;
}
.uploadtag:hover {
    background-color: #6394b6;
}

```

```

}
.upload{
  display: none;
}
.loading {
  position: absolute;
  right: 0;
  top: 0%;
  transform: translateY(-50%);
  z-index: 9999;
  height: 2em;
  width: 2em;
  overflow: show;
  margin: auto;
  left: 35%;
  bottom: 0;
  display: none;
}

.loading:before {
  content: '';
  display: block;
  margin: auto;
  width: 2em;
  height: 2em;
  border-radius: 0.5em;
  border: 0.2em solid rgba(0, 0, 0, .3);
  border-top-color: #fff;
  animation: load8 1.1s infinite linear;
}

@keyframes load8 {
  0% {
    transform: rotate(0deg);
  }
  100% {
    transform: rotate(360deg);
  }
}

```

artSum/classes/user.py

```
import io
import fitz
from artSum.classes.naiveBayes import NaiveBayes
from artSum.classes.neuralNetwork import NeuralNetwork
from artSum.classes.decisionTree import DecisionTree

class User:

    def upload_file(self, pdf_file):
        text = ""
        if pdf_file is not None:
            with io.BytesIO(pdf_file.read()) as pdf_buffer:
                pdf_doc = fitz.open(stream=pdf_buffer.read(), filetype="pdf")
                for page_num in range(pdf_doc.page_count):
                    page = pdf_doc.load_page(page_num)
                    text += page.get_text()

        return text

    # Generate summaries
    def summarize(self, text, mode):
        summary = ""
        if mode is None:
            mode = 'naive_bayes'
        # Define section keywords
        section_keywords = {
            'introduction': ['introduction', 'background', 'motivation'],
            'related work': ['related work', 'previous work', 'literature
review', 'related works', 'review', 'reviews'],
            'methodology': ['methodology', 'method', 'approach'],
            'results': ['results', 'findings', 'experiment'],
            'conclusion': ['discussion', 'conclusion', 'implications', 'future
work', 'future works'],
        }
        # Split text into sections based on section headers
        sections = {}
        current_section = None
        for line in text.splitlines():
            # Check if line matches any of the section headers
            matched_header = None
            for header, keywords in section_keywords.items():
                if any(keyword.lower() in line.lower() for keyword in
keywords) and len(line.split()) <= 5:
                    matched_header = header
                    break
```

```

    if matched_header is not None:
        # If so, start a new section
        current_section = matched_header
        sections[current_section] = ""
    # Otherwise, add the line to the current section (if one exists)
    elif current_section is not None:
        sections[current_section] += line.strip() + " "
    # If the line doesn't match any section header and a section is
currently open,
    # add the line to the current section. Otherwise, create a new
"Miscellaneous" section.
    else:
        if "Overall" not in sections:
            sections["Overall"] = ""
            sections["Overall"] += line.strip() + " "

section_summaries = {}
for text in sections:
    # Select classifier based on mode and fit classifier to data
    if mode == 'naive_bayes':
        section_summaries[text] =
NaiveBayes().NB_generate_summary(sections[text])
    elif mode == 'neural_network':
        section_summaries[text] =
NeuralNetwork().NN_generate_summary(sections[text])
    elif mode == 'decision_tree':
        section_summaries[text] =
DecisionTree().DT_generate_summary(sections[text])
    else:
        section_summaries[text] =
NaiveBayes().NB_generate_summary(sections[text])

    # Join section summaries with line breaks
    summary =
'\n'.join([f"{section.upper()}\n{section_summaries[section]}\n" for section in
section_summaries])

return summary

```

artSum/classes/article.py

```
class Article:

    def __init__(self, filename, content):
        self.filename = filename
        self.content = content

    def get_filename(self):
        return self.filename

    def get_content(self):
        return self.content
```

artSum/classes/machineLearning.py

```
class MachineLearning:
    def __init__(self):
        self.model = None
        self.tokenizer = None
```

artSum/classes/naiveBayes.py

```
import re
import heapq
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import sent_tokenize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from artSum.classes.machineLearning import MachineLearning

nltk.download('punkt')
nltk.download('stopwords')

class NaiveBayes(MachineLearning):
    def __init__(self):
        super().__init__()
        self.model = MultinomialNB()
        self.tokenizer = CountVectorizer(stop_words='english')

    def NB_generate_summary(self, text, num_sentences=3):
```

```

# tokenizing the text by sentence
sentences = sent_tokenize(text)

num_sentences = (len(sentences) // 2)
if num_sentences <= 2:
    num_sentences = 3
if not isinstance(text, str):
    raise TypeError('Input must be a string')
if not isinstance(num_sentences, int) or num_sentences < 1:
    raise ValueError('Number of sentences must be a positive integer')

processed_sentences = []
for sentence in sentences:
    words = re.sub('[^a-zA-Z]', ' ', sentence)
    words = words.lower()
    words = words.split()
    ps = PorterStemmer()
    words = [ps.stem(word) for word in words if not word in
set(stopwords.words('english'))]
    words = ' '.join(words)
    processed_sentences.append(words)

# feature extraction
cv = self.tokenizer
sentence_features = cv.fit_transform(processed_sentences)

# creating labels for training data
summary_sentences_idx = heapq.nlargest(num_sentences,
range(len(sentences)), key=lambda i: sentence_features[i].sum())
labels = ['summary' if i in summary_sentences_idx else 'non-summary'
for i in range(len(sentences))]

# training the classifier
clf = self.model
clf.fit(sentence_features, labels)

# selecting the top 'num_sentences' sentences based on the classifier
scores
sentence_scores = clf.predict_proba(sentence_features)[: , 0]
summary_sentences_idx = heapq.nlargest(num_sentences,
range(len(sentences)), key=lambda i: sentence_scores[i])
summary_sentences = [sentences[idx] for idx in
sorted(summary_sentences_idx)]

# combining the selected sentences to generate the summary
summary = ' '.join(summary_sentences)
return summary

```

artSum/classes/neuralNetwork.py

```
from transformers import AutoModelForSeq2SeqLM, BartTokenizer
from artSum.classes.machineLearning import MachineLearning

class NeuralNetwork(MachineLearning):
    def __init__(self):
        super().__init__()
        self.model = AutoModelForSeq2SeqLM.from_pretrained('facebook/bart-large-cnn')
        self.tokenizer = BartTokenizer.from_pretrained('facebook/bart-large-cnn')

    def NN_generate_summary(self, text):
        tokenizer = self.tokenizer
        model = self.model
        if text is not None:
            input_ids = tokenizer(text, truncation=True, max_length=1024,
padding='max_length', return_tensors='pt')
            summary_ids = model.generate(input_ids['input_ids'], num_beams=10,
max_length=512)
            summary = tokenizer.decode(summary_ids[0],
skip_special_tokens=True)

        return summary
```

artSum/classes/decisionTree.py

```
import re
import nltk
from sklearn.tree import DecisionTreeRegressor
nltk.download('punkt')
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from artSum.classes.machineLearning import MachineLearning

class DecisionTree(MachineLearning):

    def __init__(self):
        super().__init__()
        self.model = DecisionTreeRegressor()
        self.tokenizer = CountVectorizer()
```

```

def DT_generate_summary(self, text):
    sentences = nltk.sent_tokenize(text)

    num_sentences = (len(sentences) // 2)
    if num_sentences <= 2:
        num_sentences = 3

    # Preprocess the text
    processed_sentences = []
    for sentence in sentences:
        words = re.sub('[^a-zA-Z]', ' ', sentence)
        words = words.lower()
        words = words.split()
        ps = PorterStemmer()
        words = [ps.stem(word) for word in words if not word in
set(stopwords.words('english'))]
        words = ' '.join(words)
        processed_sentences.append(words)

    # Create BoW feature vectors
    vectorizer = self.tokenizer
    X = vectorizer.fit_transform(processed_sentences)

    # Train a decision tree to predict sentence importance
    y = [len(sentence) for sentence in sentences] # Use sentence length
as target variable
    dt = self.model
    dt.fit(X, y)

    # Use the decision tree to predict sentence importance scores
    scores = dt.predict(X)

    # Select the top-scoring sentences as the summary
    summary_indices = sorted(range(len(scores)), key=lambda i: scores[i],
reverse=True)[:num_sentences]
    summary_sentences = [sentences[i] for i in summary_indices]
    summary = " ".join(summary_sentences)

    return summary

```

APPENDIX C

Evaluation Form of ArtSum (Web-based Article Summarization System)

Name:

Siti Marina Kamil
Senior Lecturer

Matric No.:

Faculty of Language and Communication
UNIVERSITI MALAYSIA SARAWAK

	Strongly Disagree /Poor /Dissatisfied				Strongly Agree /Poor /Satisfied
	1	2	3	4	5
1. How effectively do the generated summaries capture the main ideas and key points of the original articles?				✓	
2. Are the summaries grammatically correct and well-written?					✓
3. Do the summaries effectively convey the essential information without omitting crucial details?				✓	
4. How satisfied are you with the speed at which the system generates the summaries?				✓	
5. Overall, how satisfied are you with the quality of the summaries generated by the system?				✓	

*No. 1 and 3 is quite accurate in terms of provided sufficient info from the original work.
No 2 is too general.*

1. Naive Bayes Technique

2. Neural Network Technique

3. Decision Tree Technique

APPENDIX D

Timestamp	Name:	Matric No.:	Year of Study	I think that I would like	I found the system un	I thought the system v	I think that I would nee	I found the various func	I thought there was too	I would imagine that m	I found the system ver	I felt very confident usi	I needed to learn a lot
5/27/2023 23:15:52	Loh Weng Keong	70236	Year4	5	1	5	1	4	2	5	1	5	1
5/28/2023 0:49:43	NGIAM JOHN TZE	72815	Year4	5	5	5	5	5	5	5	5	5	5
5/28/2023 11:27:45	Lee Jo Sheng	70149	Year4	5	1	5	1	5	3	5	1	5	1
5/28/2023 11:53:58	Andy Chieng Ging We	69073	Year4	4	3	5	4	5	3	4	5	3	4
5/28/2023 14:28:53	LIONG KAH PONG	70220	Year4	4	1	5	1	5	1	5	1	4	1
5/28/2023 14:30:55	Wee Quo Lung	71935	Year4	4	4	4	3	3	2	4	2	4	2
5/28/2023 15:00:45	Freddy Wong	69754	Year4	5	4	5	3	5	4	3	5	4	5
5/28/2023 16:38:34	Bong Meng Lun	69229	Year4	4	2	5	4	4	2	4	3	4	2
5/28/2023 23:06:03	Hon Poh Hian	69863	Year4	5	4	4	2	3	2	4	4	4	2
5/30/2023 15:20:28	Wong Rou Yi	71970	Year4	4	3	5	1	4	3	4	2	4	1
5/30/2023 15:27:29	Afiq Imran Bin Roslan	72056	Year4	4	5	4	4	3	1	4	2	5	2
5/30/2023 15:30:30	Afffa Nabila Binti Azm	72748	Year4	5	5	5	1	5	1	5	1	5	1
5/30/2023 15:34:08	MUHAMMAD FIRDAU	70588	Year4	5	2	4	3	4	2	4	1	4	2
5/30/2023 15:42:06	NICHOLAS ANAK BO	80373	Year 2	5	4	5	1	5	1	5	1	5	2
5/30/2023 15:43:22	HAROLD BOAZS ANA	79521	Year 2	5	4	4	2	4	1	5	1	5	2
5/30/2023 15:44:33	Clementine anak Mail	82285	Year 2	5	5	5	2	5	2	5	2	5	2
5/30/2023 15:51:16	Tan Shu Han	76951	Year 3	5	4	5	3	5	5	5	2	4	3
5/30/2023 15:54:44	Christ Samuel Anak A	74471	Year 3	4	3	3	3	3	3	3	2	3	4
5/30/2023 16:01:22	UMMIE NAJWA NOR	85961	Year 1	5	5	5	2	5	2	5	1	5	1
5/30/2023 16:01:25	Thanish A/L Jayarama	82200	Year 2	5	1	5	2	4	1	5	1	4	2
5/30/2023 16:01:36	Mirza Muhammad Bin	82669	Year 1	5	5	5	5	5	5	5	5	5	5
5/30/2023 16:02:28	NUR HANISAH BINTI	85103	Year 1	3	2	3	2	5	1	5	3	3	2
5/30/2023 16:22:38	MUHAMMAD FARIS E	84681	Year 1	4	4	5	3	4	2	5	1	5	5
5/30/2023 16:31:44	Te Xiang Jie	76966	Year 3	5	5	5	3	5	2	5	2	3	1
5/30/2023 16:36:54	Ong Xiao Qing	80930	Year 2	5	5	5	3	5	1	5	1	5	3
5/30/2023 16:37:14	Chai Ee Ling	79028	Year 2	4	5	4	2	4	3	4	2	4	4
5/30/2023 19:10:58	Mohd Irfan Hazrain	20020102	Year 2	5	5	5	5	3	1	5	1	5	2
5/30/2023 20:10:33	Ben O'Ryan Anak Clar	73377	Year 3	4	5	5	4	5	2	4	1	4	2