



Faculty of Computer Science and Information Technology

**TOPIC MODELLING FOR IDENTIFYING TOPICS FROM
MALAYSIAN BUSINESS NEWS WEBSITES.**

VIANNEY CAMELIE ANAK KERBUN

Bachelor of Computer Science with Honours

(Network Computing)

2025

**TOPIC MODELLING FOR IDENTIFYING TOPICS FROM
MALAYSIAN BUSINESS NEWS WEBSITES.**

VIANNEY CAMELIE ANAK KERBUN

**This project is submitted in partial fulfilment of the
requirements for the degree of Bachelor of Computer Science with Honours
(Network Computing)**

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

2025

**PEMODELAN TOPIK UNTUK MENGENAL PASTI TOPIK
DARIPADA LAMAN WEB BERITA PERNIAGAAN MALAYSIA.**

VIANNEY CAMELIE ANAK KERBUN

**Projek ini merupakan salah satu keperluan untuk
Ijazah Sarjana Muda Sains Komputer dengan Kepujian
(Pengkomputeraan Rangkaian)**

Fakulti Sains Komputer dan Teknologi Maklumat

UNIVERSITI MALAYSIA SARAWAK

2025

DECLARATION

I, Vianney Camelie Anak Kerbun, matric number 81441, from Faculty of Computer Science and Information Technology hereby declare that the work entitled, **Topic Modelling for Identifying Topics from Malaysian Business News Websites** is my original work. I have not copied from any other students' work or from any sources except where due reference or acknowledgement is made explicitly in the text, nor has any part been written for me by another person.



VIANNEY CAMELIE ANAK KERBUN (81441)

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

Acknowledgement

I would like to express my heartfelt gratitude to those who have supported and contributed to the completion of my Final Year Project. I am deeply grateful to Ts. Dr. Sarah Flora Anak Samson Juan, my supervisor, whose expert guidance, valuable insights, and unwavering support have been instrumental in navigating the complexities in my research. Her mentorship has been crucial in shaping the direction and outcomes of my work, and I truly appreciate her time and dedication. I also wish to express my gratitude to Miss Vadrianey Anak Asas, a master's student, whose dedicated assistance and collaborative spirit greatly enhanced my research experience. Her contributions, from technical support to thoughtful feedback, have been essential in achieving my goals. Next, I would like to thank my family for their continuous support and encouragement, which motivated me to give my very best in completing this Final Year Project. Lastly, I am grateful to all those who have provided encouragement and constructive feedback throughout my research. Their support has been invaluable in refining my Final Year Project.

Table of Contents

Abstract	x
Abstrak	xi
Chapter 1: Introduction	1
1.1 Introduction	1
1.2 Problem Statement	2
1.3 Objective	3
1.4 Methodology	4
1.5 Project Scope	5
1.6 Significant of the projects	5
1.7 Project schedule	6
1.8 Expected outcomes	7
1.9 Project outline	7
1.10 Summary	8
Chapter 2: Literature Review	9
2.1 Introduction	9
2.2 Natural languages Processing (NLP)	9
2.3 Web scraping	10
2.3.1 Advantages and disadvantages of web scraping	10
2.3.2 Procedure of web scraping.	12
2.4 Data preprocessing	13
2.4.1 Data pre-processing techniques	14
2.5 Topic modelling	20
2.6 Brief background of Topic Modelling Techniques	21
2.6.1 Latent Dirichlet Allocation (LDA)	21
2.6.2 Non-Negative Matrix Factorization (NMF)	22
2.6.3 Top2vec	23
2.6.4 Bidirectional Encoder Representations from Transformers (BERT)	24
2.7 Evaluation of topic modelling performance	26
2.8 Web development and NLP tools	27
2.8.1 Web scraping tools and libraries	27
2.8.2 Data pre-processing tools and libraries	28
2.8.3 Topic modelling libraries	29

2.8.4	Web development tools	30
2.9	Motivation of research.....	32
2.10	Summary.....	33
Chapter 3:	Methodology	34
3.1	Introduction.....	34
3.2	Web application development	34
3.3	Web application development	37
3.3.1	Use case diagram.....	37
3.3.2	Sequence Diagram	38
3.3.3	Proposed GUI.....	39
3.4	Research Methodology.....	41
3.5	Data collection.....	42
3.6	Data cleaning or pre-processing.....	45
3.7	Topic modelling.....	46
3.8	Evaluation and result.....	47
3.9	Tool development and training.....	47
3.9.1	Web development tools	47
3.9.2	Web scraping tools and libraries	48
3.9.3	Data pre-processing tools and libraries	48
3.9.4	Topic modelling libraries	48
3.10	Summary.....	49
Chapter 4:	Implementation	50
4.1	Introduction.....	50
4.2	Data collection.....	50
4.2.1	Web Scraping	51
4.3	Data preprocessing.....	60
4.3.1	Data Cleaning	61
4.4	Topic modelling.....	65
4.4.1	Implement topic modelling.....	66
4.4.2	Visualisation	77
4.5	Web development implementation.....	78
4.6	Summary.....	86
Chapter 5:	Experimental and results.	87

5.1	Introduction.....	87
5.2	Analysis dataset	87
5.2.1	Analysis of the news data	87
5.2.2	Analysis of wordcount.....	91
5.3	Topic model results	91
5.3.1	Coherence score	91
5.3.2	Coherence score graph	93
5.3.3	Top words	94
5.4	Analysis of topic model on web applications	95
5.4.1	Analyse text.....	95
5.4.2	Analyse document.....	97
5.5	Functional Testing.....	104
5.6	User Acceptance Test (UAT)	107
5.7	Summary.....	115
Chapter 6: Conclusion and future work		116
6.1	Introduction.....	116
6.2	Objectives achievement.....	116
6.3	Limitations and Constraint	116
6.4	Future works	117
6.5	Summary.....	117
References		118
APPENDIX A: SOURCE CODE.....		123
APPENDIX B: TOPIC MODELLING		177
APPENDIX C: USER ACCEPTANCE TEST		186

Lists of figures

Figure 1.1: Project Methodology.....	4
Figure 1.2: Project schedule of FYP1.....	6
Figure 2.1: The procedure of web scraping (Matta Priya et al., 2020).	12
Figure 2.2: Example of web scraping based on (Kmgs & Rahm, 2022).	13
Figure 2.3: Example of tokenization based on (Kmgs & Rahm, 2022).	15
Figure 2.4: Example of TF-IDF. (Source: https://www.kaggle.com/code/amar09/text-pre-processing-and-feature-extraction).	18
Figure 2.5: Figure 2. 5: The explanation of TF-IDF. (Source: https://www.kaggle.com/code/amar09/text-pre-processing-and-feature-extraction).	18
Figure 2.6: Example of text cleaning and normalization based on (Kim & Gil, 2019).....	19
Figure 2.7: Svelte website (Source: https://medium.com/coderhack-com/comparing-javascript-frameworks-svelte-vue-react-and-angular-70474666e69d).....	31
Figure 2.8: Figure 2. 8: Visual Studio code application (Source: https://code.visualstudio.com/docs/editor/whyvscode).	31
Figure 2.9: Figma website (Source: https://www.figma.com/collaboration/).	32
Figure 3. 1: Waterfall methodology.	35
Figure 3. 2: Use case diagram of News Analysis Hub.	37
Figure 3. 3: Sequence diagram of News Analysis Hub.....	38
Figure 3. 4: The proposed web application GUI.	39
Figure 3. 5: The proposed web application GUI.	40
Figure 3. 6: Research methodology.	41
Figure 3. 7: Flowchart of the methodology.....	42
Figure 3. 8: The step for data collection.	42
Figure 3. 9: The Business Today news website.....	43
Figure 3. 10: Daily Express news website.....	44
Figure 3. 11: The Malaysia Reserve news website.....	44
Figure 3. 12: New Straits Times news website.....	44
Figure 3. 13: The step for data cleaning or pre-processing.	45
Figure 3. 14: Example of data after the cleaning process.....	45
Figure 3. 15: The step for train and test topic modelling.	46
Figure 4. 1: The step for data collection.	50
Figure 4. 2: Code to import libraries.	52
Figure 4. 3: Code to scrape content in Business today.....	53

Figure 4. 4: Code to scrape content in Daily express.....	54
Figure 4. 5: Code to scrape content in Malaysia Reserve.	55
Figure 4. 6: Code to scrape content in New Straits Times.	56
Figure 4. 7: Code to save data for Business today.....	58
Figure 4. 8: Code to save data for Daily Express.....	59
Figure 4. 9: Code to save data for Malaysia reserve.....	59
Figure 4. 10: Code to save data for NST.....	59
Figure 4. 11: Extracted data.....	59
Figure 4. 12: Sample of the content scraped.....	60
Figure 4. 13: The step for data cleaning or pre-processing.	60
Figure 4. 14: Master List dataset.....	61
Figure 4. 15: Code for preprocessing.	62
Figure 4. 16: Code to save data file.....	62
Figure 4. 17: The sample in the CSV file.....	65
Figure 4. 18: The step for train and test topic modelling.	65
Figure 4. 19: Code to split data.....	67
Figure 4. 20: Code to train LDA topic modelling.....	68
Figure 4. 21: Code to train NMF topic modelling.	69
Figure 4. 22: Code to train Top2vec topic modelling.	70
Figure 4. 23: Code to test LDA topic modelling.	71
Figure 4. 24: Code to test NMF topic modelling.	71
Figure 4. 25: Code to test Top2vec topic modelling.....	72
Figure 4. 26: Code to test BERTopic topic modelling.	73
Figure 4. 27: Code to calculate coherence score LDA topic modelling.....	74
Figure 4. 28: Code to calculate coherence score NMF topic modelling.....	75
Figure 4. 29: Code to calculate coherence score Top2vec topic modelling.	76
Figure 4. 30: Code to calculate coherence score BERTopic topic modelling.....	76
Figure 4. 31: Code for visualisation.	77
Figure 4. 32: Welcome page of News Analysis Hub.	79
Figure 4. 33: Home page of News Analysis News Hub.	80
Figure 4. 34: CSV file Analysis interface.	80
Figure 4. 35: Copy and paste news content.	81
Figure 4. 36: CSV template.	81
Figure 4. 37: CSV template with data.....	81
Figure 4. 38: Interface for upload document.	82
Figure 4. 39 : Upload file.....	82
Figure 4. 40: CSV topic result interface.	83
Figure 4. 41: CSV topic result interface.	83
Figure 4. 42: Text analysis interface.	84
Figure 4. 43: Copy and paste news content.	84
Figure 4. 44: Text Analysis Result interface.	85
Figure 4. 45: Text Analysis Result interface.	85
Figure 4. 46: Text Analysis Result interface.	86

Figure 5. 1: The bar graph number of articles by (2019-2023).....	87
Figure 5. 2: The bar graph number of articles published by month.	88
Figure 5. 3: The pie chart Percentage of Articles Published by Year.	89
Figure 5. 4: The doughnut chart of Percentage of Articles Published by Source..	89
Figure 5. 5: The heatmap of Number of Articles Based on Year and Month.	90
Figure 5. 6: Coherence scores comparison of topic model.	93
Figure 5. 7: Coherence score of NMF topic modelling for the year 2021.	93
Figure 5. 8: Bar chart of topic.	94
Figure 5. 9: Respondents faculty.	108
Figure 5. 10: Respondents' year of study.....	108
Figure 5. 11: Respondents gender.	109
Figure 5. 12: Analyse Document usability scale.	110
Figure 5. 13: Analyse Document usability scale.	110
Figure 5. 14: Analyse Document usability scale.	111
Figure 5. 15: Analyse Document usability scale.	111
Figure 5. 16: Analyse Document usability scale.	112
Figure 5. 17: Analyse Text usability scale.	112
Figure 5. 18: Analyse Text usability scale.	113
Figure 5. 19: Analyse Text usability scale.	113
Figure 5. 20: Analyse Text usability scale.	114
Figure 5. 21: Analyse Text usability scale.	114

List of tables

Table 1.1: Project schedule.	6
Table 2.1: Advantages and disadvantages of Latent Dirichlet Allocation (LDA). 22	
Table 2.2: Advantages and disadvantages of Non-Negative Matrix Factorization (NMF).	23
Table 2.3: Advantages and disadvantages of Top2vec.	24
Table 2.4: Advantages and disadvantages of BERTopic.	26
Table 2.5: Web scraping libraries.	28
Table 2.6: Web scraping Tools.	28
Table 2.7: Data pre-processing tools and libraries.	29
Table 2.8: Topic modelling libraries.	30
Table 3. 1: Web development tools.	47
Table 3. 2: Web scraping tools and libraries.	48
Table 3. 3: Data pre-processing tools and libraries.	48
Table 3. 4: Topic modelling libraries.	49
Table 4. 1: Web scraping tools and libraries.	51
Table 4. 2: Output of the scraping process.	57
Table 4. 3: The process of scraping content and URLs from multiple pages of a website.	58
Table 4. 4: Data pre-processing tools and libraries.	61
Table 4. 5: The sample of raw text and the cleaned text.	64
Table 4. 6: Topic modelling libraries.	67
Table 4. 7: The number of articles in training and testing.	68
Table 5. 1: Wordcount analysis.	91
Table 5. 2: Coherence score of test data.	92
Table 5. 3: Word clouds and Bar chart of analyse text.	96
Table 5. 4: Result topic overview of analyse text.	97
Table 5. 5: Word clouds and Bar chart of analyse document.	98
Table 5. 6: Result topic overview of analyse document.	103
Table 5. 7: Functional Testing.	107
Table 5. 8: Background information.	107

List of symbols and abbreviations

Bidirectional Encoder Representations from Transformers

BERT, 28

coherence score

CV, 38

comma-separated values

CSV, 4

Extensible Hypertext Markup Language

XHTML, 13

Extensible Markup Language

XML, 12

graphical user interface

(GUI), 42

Hierarchical Density-Based Spatial Clustering of Applications with Noise

(HDBSCAN), 27

Hypertext Markup Language

HTML, 12

integrated development environment

(IDE), 42

Latent Dirichlet Allocation

(LDA), vii

Natural Language Processing

(NLP), vii

Natural Language Toolkit

(NLTK), 15

News Straits Times

(NST), 2

Non-negative Matrix Factorization

(NMF), vii

part of speech

(PoS), 17

term frequency-inverse document frequency

TF-IDF, 15

Term Frequency-Inverse Document Frequency

TF-IDF, 19

Topic Coherence

(TC), 38

Uniform Manifold Approximation and Projection

(UMAP), 27

Uniform Resource Locator

URL, 21

Universal Sentence Encoder

(USE), 26

user experiences

(UX), 37

user interfaces

(UI), 37

Abstract

This research focuses on the topic modelling for identifying topics from Malaysian business news websites. In the digital age, businesses and organizations face challenges due to the large amount of information, particularly within the business news sector. The vast volume of data complicates the process of identifying meaningful insights, making it difficult for professionals to stay informed. This research investigates the Natural Language Processing (NLP) techniques, specifically topic modelling, to address this challenge. Topic modelling is a statistical method that helps identify key themes within large datasets, making it easier to summarize and extract valuable information. The study focuses on Malaysian business news websites, applying techniques such as Non-negative Matrix Factorization (NMF), Latent Dirichlet Allocation (LDA), Top2Vec, and BERTopic. Furthermore, a web-based application is developed to offer an interactive platform for users to explore the identified topics. The application illustrates the potential of topic modelling in enhancing decision-making and business intelligence by summarizing key topics within extensive datasets.

Keyword: Topic Modelling, Natural Language Processing (NLP), Malaysian Business News

Abstrak

Penyelidikan ini memberi tumpuan kepada pemodelan topik untuk mengenal pasti topik daripada laman web berita perniagaan Malaysia. Dalam era digital, perniagaan dan organisasi menghadapi cabaran disebabkan oleh jumlah maklumat yang besar, terutamanya dalam sektor berita perniagaan. Jumlah data yang besar menyukarkan proses mengenal pasti pandangan yang bermakna, menjadikannya sukar bagi para profesional untuk kekal maklum. Penyelidikan ini menyiasat teknik Pemprosesan Bahasa Semula Jadi (NLP), khususnya pemodelan topik, untuk menangani cabaran ini. Pemodelan topik adalah kaedah statistik yang membantu mengenal pasti tema utama dalam set data besar, menjadikannya lebih mudah untuk meringkaskan dan mengekstrak maklumat berharga. Kajian ini memfokuskan kepada laman web berita perniagaan Malaysia, menggunakan teknik seperti Pencernaan Matriks Tidak Negatif (NMF), Peruntukan Dirichlet Tersembunyi (LDA), Top2Vec, dan BERTopic. Tambahan pula, sebuah aplikasi berasaskan web dibangunkan untuk menawarkan platform interaktif bagi pengguna meneroka topik-topik yang telah dikenal pasti. Aplikasi ini menggambarkan potensi pemodelan topik dalam meningkatkan pembuatan keputusan dan kecerdasan perniagaan dengan meringkaskan topik-topik utama dalam set data yang luas.

Kata Kunci: Pemodelan Topik, Pemprosesan Bahasa Semula Jadi (NLP), Berita Perniagaan Malaysia

Chapter 1: Introduction

1.1 Introduction

In the digital age, the increasing number of information has made it difficult for individuals and organizations to effectively manage large volumes of data, particularly in the business news sector, where thousands of articles are published daily. This overload information makes it difficult for individuals and organizations to quickly identify important information.

A Natural Language Processing (NLP) enable computers to understand and interpret text. It is important in the massive quantity of data available on the web, over 20 billion pages which can be exploited as a valuable resource, as highlighted by (Amirifar et al., 2023). NLP technique called topic modelling where it can help to identify keyword or important topic in the large dataset. It can help user to identify or summarize the topic that has been discovered from the large dataset. Topic Modelling is a statistical technique used for managing unstructured data and extracting information from large collection of documents. This technique identifies underlying themes within a set of documents, facilitating their representation based on the occurrence of the words compose (Grisales et al., 2023). By identifying the topics, users can quickly understand the main topic within a large dataset and improving decision-making. In business news, knowing these key topics can assist professionals, researchers, and organizations in tracking market trends, assessing risks, and staying informed about political or economic changes.

This project explores topic modelling technique that can used to extract data particularly business news website from Malaysia. Handling huge datasets, such as news articles, comes with challenges such as noise and redundancy, topic modelling methods like NMF, LDA, Top2Vec, and BERTopic can address these by extracting and

summarizing the most relevant topics. By identifying the main topics in this news, the project aims to help user understand business activities, risks, and political issues affecting the Malaysian economy.

1.2 Problem Statement

The number of news articles published daily across various websites presents difficulties for individuals trying to stay informed. Reading through numerous articles is time-consuming and challenging due to variations in quality, length, and complexity, leading to information overload. This affects the speed and quality of decision-making. The internet is filled with information and sources of knowledge that may confuse readers, requiring additional time and effort to find relevant information on specific topics of interest (Albalawi et al., 2020). Business news website such as Business Today, Daily Express, Malaysia Reserve, and News Straits Times (NST) provide platform where readers learn about the trends related to economics, risk and other critical area. Readers need quick understanding of the topic covered in each article and to identify key topics from various news articles.

To address this issue, Natural Language Processing (NLP) seeks to facilitate effective communication between humans and computers by comprehending and processing natural language (Supriyono et al., 2024). NLP techniques can be utilized to automatically summarize news articles from multiple sources by extracting key information and generating concise summaries. This process involves web scraping to collect news articles, data cleaning to improve data quality, and the implementation of topic modelling to identify key topics within the articles. Topic modelling is a fundamental technique in text mining, that identifies groups of words, termed topics, within a body of text, as highlighted by (Albalawi et al., 2020). Techniques such as Latent Dirichlet Allocation (LDA), Non-Negative Matrix Factorization (NMF), Top2Vec, and BERTopic are

used for this project. A comparison of these four techniques is needed to identify which of this technique will produce better topic model when applied on the Malaysia business news website. This approach aims to provide users with the most relevant and concise news topic.

In this project, the research will focus on the Topic Modelling for identifying topics from Malaysian business news websites. The research question is as follow:

1. What techniques can be used effectively to collect, clean and preprocess data from Malaysian business news website?
2. Which topic modelling techniques are most effective for generating topics from Malaysian business news?
3. How can a user-friendly interface be designed to present topics from business news in an accessible way?

1.3 Objective

1. To implement data collection using web scraping techniques, followed by data cleaning and the application of topic modelling methods to identify and analyze topics in news articles.
2. To evaluate the performance of the topic modelling techniques for obtaining the best topic model on the Malaysia business news.
3. To integrate the best topic model into a web-based system that presents topic news articles from different news article websites.

1.4 Methodology

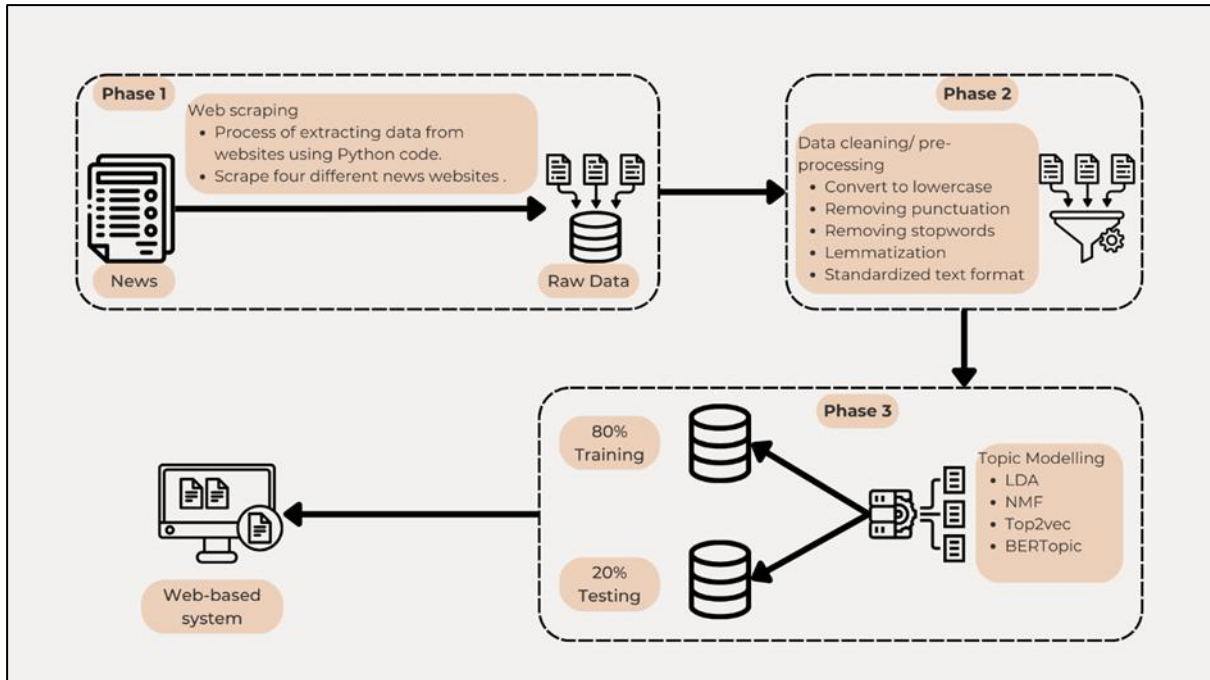


Figure 1.1: Project Methodology.

Figure 1.1 above shows the methodology of this research project divided into three phases. The phase involves a detailed process of automating web scraping, data preprocessing, and topic modelling using Python. In phase 1, web scraping is implemented to collect news articles from four different news websites, extracting content and saving it into structured CSV files for further analysis. This phase also involves generating insightful visualizations to explore the distribution of articles over time and by source, utilizing bar graphs, pie charts, doughnut charts, and heatmaps.

In phase 2, the preprocessing stage cleans and tokenizes the text data, stopwords, and removing punctuation, and then saving the cleaned data into a new CSV file. A word count analysis is performed on the cleaned articles, further enhancing the dataset.

Next, Phase 3 focuses on the implementation of topic modelling, using techniques such as Latent Dirichlet Allocation (LDA), Non-Negative Matrix Factorization (NMF),

Top2Vec, and BERTopic to identify topics within the dataset. The performance of the model is evaluated using coherence scores. The dataset is split into training and testing sets based on date ranges, and the topics are visualized using word clouds. Lastly, to integrate the best topic model into a web-based system that presents topic news articles from different news article websites.

1.5 Project Scope

The scope of this project focuses on designing, developing, and evaluating a Natural Language Processing (NLP) system that automates the extraction and summarization of news articles from selected sources such as Business Today, Daily Express, Malaysia Reserve, and News Straits Times (NST). Data collection will involve identifying and selecting specific news websites and applying web scraping techniques to collect news articles into structured CSV files. Following data collection, the project will focus on basic data cleaning, including tokenization and removing irrelevant content, to prepare the text for analysis. A topic modelling phase will identify key topics of the articles using techniques such as Latent Dirichlet Allocation (LDA), Non-Negative Matrix Factorization (NMF), Top2Vec, and BERTopic. The project will develop a web-based system that can automate the extraction process and provide users with the identified news topics.

1.6 Significant of the projects

The significant of the project is to address the problem of information overload, especially with the large number of news articles published daily. By developing a system that automatically collects, processes, and summarizes news articles, the project will help users quickly find relevant news topics without having to read through everything. This saves time and makes it easier for decision-makers to stay informed and make better choices. Additionally, the project employs Natural Language Processing (NLP) techniques

such as topic modelling, which could be useful for future research and practical applications in areas like business and media. Therefore, develop a web-based system that can automate the extraction and provides relevant news topics from different news article websites.

1.7 Project schedule

Description	Start	End	Days
Brief Proposal	1/10/2024	20/10/2024	20
Meet up with supervisor	1/10/2024	2/10/2024	2
Decide project title	3/10/2024	4/10/2024	2
Brief proposal	5/10/2024	20/10/2024	15
Brief Project proposal submission	20/10/2024	20/10/2024	1
Feedback and comment from Reviewer/Examiner	21/10/2024	28/10/2024	8
Final Full Proposal	29/10/2024	14/11/2024	17
Final Full Proposal	28/10/2024	13/11/2024	16
Full Proposal Submission	14/11/2024	14/11/2024	1

Table 1.1: Project schedule.

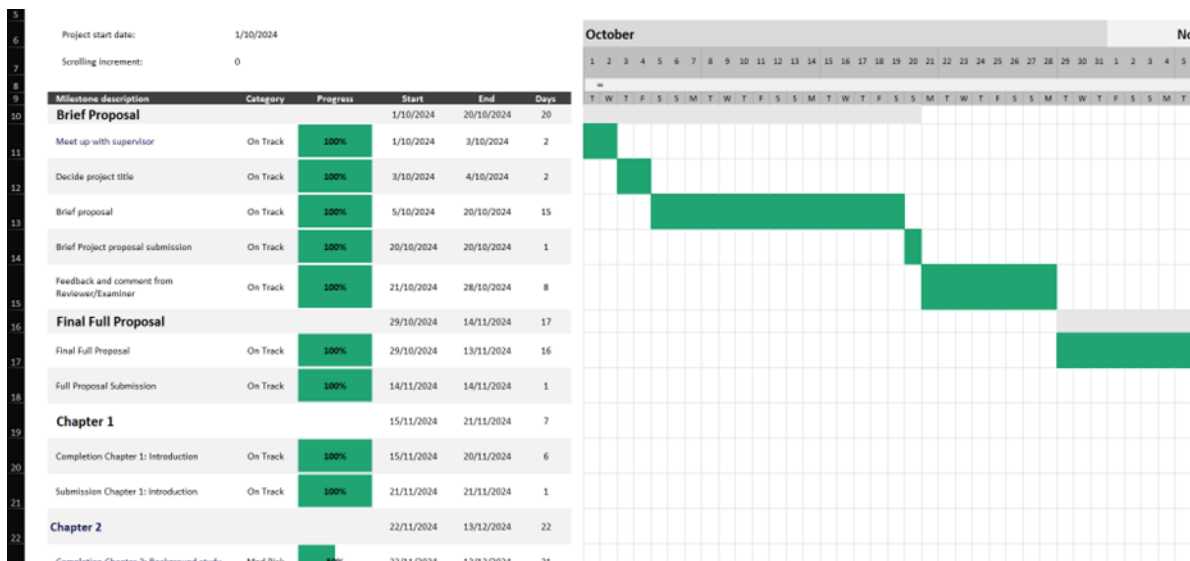


Figure 1.2: Project schedule of FYP1.

1.8 Expected outcomes

This project is expected to develop topic extraction system that automates the extraction and summarization of business news articles from selected Malaysian websites. The key outcomes include an automated web scraping procedure that collects and organizes news articles into structured CSV files, as well as a data cleaning process that ensures the text is clean for analysis. Topic modelling techniques such as Latent Dirichlet Allocation (LDA), Non-Negative Matrix Factorization (NMF), Top2Vec, and BERTopic, used to identify key topics effectively. Additionally, the output is to deliver a web-based system to present these news topics clearly for users.

1.9 Project outline

These are the remaining of the chapter will cover in this project.

Chapter 2: Background study

Chapter 2 will focus on the background study of NLP techniques, web scraping, data preprocessing, and topic modelling used in this project, with reference to relevant research articles.

Chapter 3: Methodology

This chapter will describe all methods and procedures used in this project. The methodology chapter outlines the implementation of the methods necessary to fulfil the objectives of this project.

Chapter 4: Implementation

In this chapter will focus on the data of coding, result, and plot for this project using software such as Python programming languages.

Chapter 5: Experiment and Results

Chapter 5 will cover the experiment conducted to evaluate the performance of the topic modelling techniques in obtaining the best topic model for Malaysia business news.

Chapter 6: Conclusion and future works

In this chapter will summarize the data collection and finding throughout the project. The objective of this project, which focused on the Topic Modelling for identifying topics from Malaysian business news websites has been successfully achieved.

1.10 Summary

This chapter outline the introduction, problem statement, objectives, methodology and expected outcomes of the project, which focused on applying NLP techniques for topic modelling to manage a large dataset in Malaysian business news. The main challenges users face is staying informed due to the large number of news articles published daily, making it time-consuming and difficult to make decision making.

Chapter 2: Literature Review

2.1 Introduction

This chapter describes selected Natural Language Processing techniques for topic modelling, web scraping and data preprocessing. It will also cover selected research on topic modelling techniques, tools, and libraries.

2.2 Natural languages Processing (NLP)

Natural Language Processing (NLP) is a combination of language studies, computer science, and AI to help computers understand and interact with human languages. The key goal of NLP is to create tools that can process large amount of language data, enabling computers to understand document content. NLP has made important progress in analysing, organizing, and grouping text using machine learning, allowing tasks like topic identification, sentiment analysis, recognizing entities, and sorting documents by category.

Today, NLP helps extract useful information from documents in many languages, making data more accessible across fields (Gonzalez-Gomez et al., 2024); (Amirifar et al., 2023); (Mridha et al., 2021). NLP become important due to the large amount of data available online, with countless web pages and social media content serving as valuable sources for analysis. NLP techniques, such as automated content analysis and topic modelling, are now widely used in the social sciences to study large collections of text, such as news articles and forums. Topic modelling is one of the NLP techniques, that collect a group of words and documents with similar meanings to identify hidden topics. First used in the 1950s, NLP is now widely used across industries to manage and interpret the rapid growth of text data, making it a powerful tool for collective and understanding large amount of information (Umamaheswaran et al., 2023); (Omar et al., 2022).

2.3 Web scraping

Web scraping is one of the NLP techniques used for collecting data such as news articles from websites. The ability to extract relevant data from websites has grown increasingly important in the age of digital information for various kinds of applications in industries, from market research to academic studies. According to (Matta Priya et al., 2020), web scraping, also known as web harvesting, is considered the optimal method for extracting various types of data from websites. It involves the process of gathering information from web pages to obtain valuable insights, which are then typically exported into formats such as spreadsheets.

Web scraping can be performed manually or automated using software that is perceived as a human web surfing behaviour to extract specific data from websites. Web scraping is widely used in various applications across industries to compile large datasets automatically. In e-commerce, (Ullah et al., 2018) propose a novel approach to web scraping focused on revealing trends in target products and extracting new insights from online shopping websites. The author emphasizes the increasing significance of data-driven decision-making in e-commerce and underscores the need for efficient methods to analyze product trends and customer behaviour. On the other hand, (Sukmandhani et al., 2023) address the challenges faced by online marketplaces, particularly the unauthorized sale of branded products without proper licenses. Their research aims to identify illegally sold products and facilitate the reporting of violations by unauthorized sellers.

2.3.1 Advantages and disadvantages of web scraping

Web scraping, as described by (Awangga et al., 2019), involves programmatically extracting data from web pages, making it useful for gathering large amounts of data from websites like GitHub. This method allows data to be saved locally or in a structured

format, such as a database. Python, along with libraries like BeautifulSoup, is commonly used for web scraping tasks due to its effectiveness in parsing HTML and XML documents. (Adila, 2022) emphasizes Python's versatility and highlights BeautifulSoup as a handy tool for extracting data from websites. The simplicity and flexibility of web scraping with BeautifulSoup make it a valuable tool for various research and analysis tasks, as highlighted by (Grybauskas et al., 2021) and (Rita et al., 2023). Web scraping remains a valuable technique for accessing and utilizing data from the web, as demonstrated by (Domala et al., 2020), and (Ahmed et al., 2020), who leverage web scraping for various applications ranging from research to real estate investment analysis.

The disadvantages of web scraping, (Adila, 2022) highlights the limitation of incomplete data extraction from journal websites. Despite collecting data using web scraping techniques, the resulting dataset may still lack vital information due to inconsistencies in the availability of publication dates on journal websites. Furthermore, (Wani et al., 2021) points out a technical limitation of BeautifulSoup, wherein it cannot extract dynamically loaded content from web pages. This restriction restricts the scope of data extraction to only the content visible on the initial source page, potentially missing out on valuable information that may be loaded dynamically after the initial page load.

2.3.2 Procedure of web scraping.

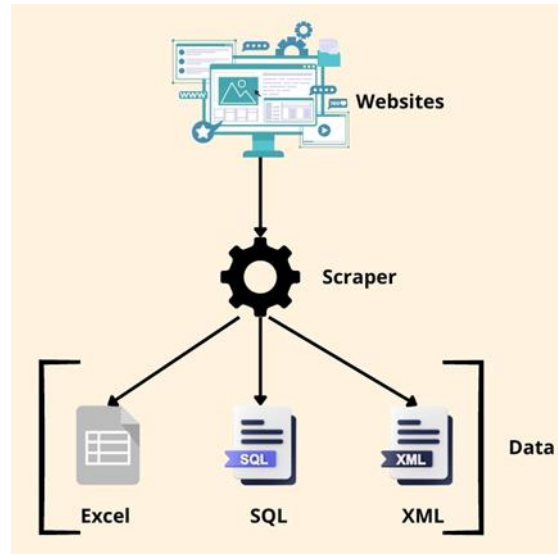


Figure 2.1: The procedure of web scraping (Matta Priya et al., 2020).

According to (Matta Priya et al., 2020), web scraping can be done manually or through software designed to simulate human web browsing behaviour, collecting specific information from websites. It involves extracting information from web pages, typically written in HTML or XHTML, and representing it in a hierarchical document model, as shown in Figure 2.1. The primary aim of web scraping tools is to automate the collection of required data and organize it. By facilitating the conversion of human-readable content into machine-readable data, web scraping enables researchers to extract structured datasets from web pages. Essentially, a web scraper is a tool that accesses web pages, identifies and extracts specified data elements, and saves them as a structured dataset after potential transformations.

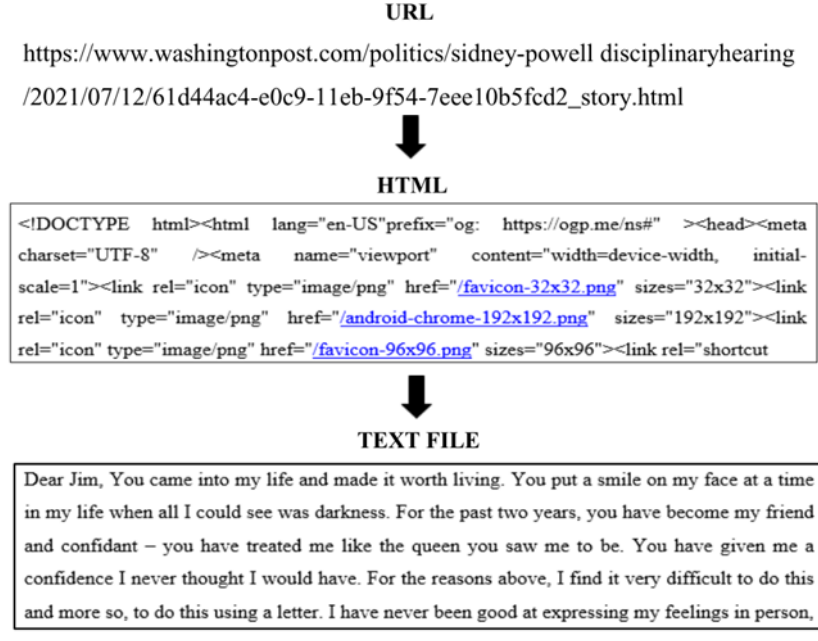


Figure 2.2: Example of web scraping based on (Kmgs & Rahm, 2022).

2.4 Data preprocessing

In the context of data mining in Natural Language Processing (NLP) refers to a technique that can be applied to raw data before utilized for further analysis or modelling tasks. The purpose of pre-processing is to clean, transform and prepare data in format that suitable for computational processing. The technique includes lemmatization, a more sophisticated approach, transforms words into their base forms based on part-of-speech tags, ensuring semantic accuracy. Tokenization, a foundational step in NLP, divides text into meaningful units such as words or sentences, serving as the basis for various analytical tasks. The removal of stop words eliminates frequently occurring but semantically insignificant words, streamlining the data for more focused analysis.

Similarly, handling rare words involves either removing or replacing infrequently occurring terms to reduce noise and enhance model performance. The TF-IDF (term frequency-inverse document frequency) weighting scheme is pivotal in highlighting the relevance of words within a document relative to a corpus, thus improving feature

extraction for machine learning models. Also highlights the key tools for data preprocessing in NLP. Python libraries such as the Natural Language Toolkit (NLTK) and TextBlob offer extensive functionalities for text manipulation, while CountVectorizer, part of the Scikit-learn library, is essential for constructing numerical word representations.

2.4.1 Data pre-processing techniques

During data pre-processing, cleaning and normalizing data by converting phrases and words into a standard form. This standardization ensures consistency across documents, helpful in the development of critical features and reduces noise caused by unwanted objects (Srivastava et al., 2022). The following are techniques used for data pre-processing.

2.4.1.1 Lemmatization

Lemmatization is a technique in natural language processing (NLP) that transforms a word into its base form, or "lemma," based on its part of speech (PoS) tags. For instance, the word "saw" can be lemmatized to "saw" if it is a noun (NN) or to "see" if it is a past tense verb (VBD) (Mariam Khader et al., 2018). This process ensures that each word is reduced to a meaningful base form, unlike stemming, which may produce non-meaningful roots. Lemmatization involves removing or replacing the suffix of a word to obtain the lemma. For example, "caring" is lemmatized to "care," which retains semantic meaning. This method is an essential text preprocessing step in NLP, often yielding excellent results (Tabassum & Patil, 2020). The process requires the correct order of implementation; for instance, removing stop words like "is," "the," and "in" before PoS tagging can lead to incorrect tagging and erroneous results. Furthermore, lemmatization determines the base form of words such as "am," "are," and "is," transforming them into the lemma "be," which helps in standardizing different word forms to their common base (Frank et al., 2024).

2.4.1.2 Tokenization

Tokenization is a fundamental step in natural language processing (NLP) that involves dividing text into smaller units called tokens. This process can be applied at various levels, including words, sentences, or sub word units, and serves as the foundation for numerous NLP tasks such as text classification, named entity recognition, and machine translation (Frank et al., 2024). This process typically begins with converting documents into plain text by removing HTML tags and other markup, considered part of text preprocessing, to ensure the preservation of essential information while maintaining security (Kadhim, 2018). Tokenization can be conducted at various levels, including word-level, sentence-level, and sub word-level tokenization, each serving different purposes in NLP tasks (Frank et al., 2024). Overall, effective tokenization ensures the segmentation of text data into meaningful units, enabling accurate analysis and modelling tasks by filtering out unwanted words and maintaining consistency in documents (Tabassum & Patil, 2020).

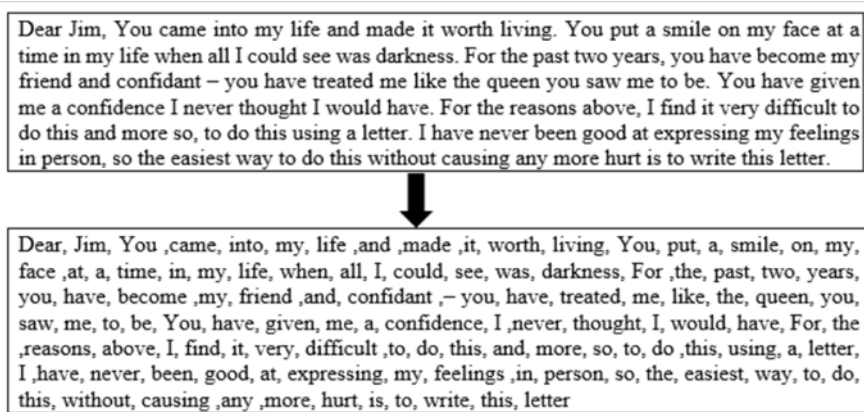


Figure 2.3: Example of tokenization based on (Kings & Rahm, 2022).

2.4.1.3 Stop-words removal

Stop-words removal is a technique used in text classification to eliminate frequently used words that are meaningless and do not contribute to the classification process. This reduces the corpus size without losing important information. These techniques aimed to enhancing text analysis and classification tasks (Kadhim, 2018). Stop words, such as articles, prepositions, and pronouns, add little semantic value to text documents and can increase noise, making analysis less efficient (Mothe et al., n.d.). Despite the challenges in developing comprehensive stop word lists due to inconsistencies across textual sources (Gurusamy & Professor, n.d.), removing them is important for improving data quality and enabling more focused analysis on content-bearing words (Frank et al., 2024). Reducing noise and improving the efficiency of subsequent analysis. By eliminating words like "the," "are," and "is," which do not significantly contribute to the context or content of textual documents, the dimensionality of the term space is controlled, leading to improved system performance (Kathiravan & Haridoss, 2018). While removing these stop words enhances the results of classification algorithms, it's important to consider use cases where certain negation words, like "no" or "not," are essential for understanding the context and intent of a sentence.

2.4.1.4 TF-IDF

Term Frequency (TF) referred to the number of occurrences of a given term within a specific document (Kalra et al., 2022). Words that appear more frequently are generally considered to be more representative of the document's content. Consequently, the TF metric assigns higher values to terms with greater frequency within the document (Uwiragiye & Rhinehardt, 2022). Inverse Document Frequency (IDF) measures how important a word is across a set of documents. It reduces the weight of words that appear frequently in many documents and increases the weight of words that are uncommon

throughout the collection (Uwiragiye & Rhinehardt, 2022). This approach is based on the idea that rare words provide more useful information for distinguishing between documents. The TF-IDF calculation is computed by multiplying the TF score by the IDF score, resulting in a composite metric that highlights terms that are both frequent in a particular document and infrequent across the corpus. This approach effectively identifies terms that are most characteristic and informative for individual documents within a larger dataset (Chawla et al., 2023) (Kalra et al., 2022). To handle the classification problems in terms of time, effort, and resources, it is important to handle sequences of same lengths, which makes direct similarity computation challenging. The TF-IDF method overcomes this by representing each sequence as a vector of uniform length, based on the complete set of terms across the dataset. Term Frequency (TF) values range from 0, when a term is absent in a sequence, when the sequence consists entirely of that term it became 1. The importance of each TF value is calculated for every attribute, enabling the ranking and comparison of attributes. Attributes with higher TF values that influence key decisions more frequently are considered more important (Uwiragiye & Rhinehardt, 2022).

```
In [14]: from sklearn.feature_extraction.text import TfidfVectorizer
# initialize sample document
sample_documents = ['Must have a subject and a verb', 'Must express a complete thought', 'Must only
have one clause']
# instantiate
vectorizer = TfidfVectorizer()
vectorizer.fit(sample_documents)
# summarize
print(f':: vector vocabulary - {vectorizer.vocabulary_}\n')
# encode document
vector = vectorizer.transform(sample_documents)
# summarize encoded vector
print(f':: vector shape - {vector.shape}\n')
print(f':: vector list - {vector.toarray()}\n')
```

```
:: vector vocabulary - {'must': 5, 'have': 4, 'subject': 8, 'and': 0, 'verb': 10, 'express':
3, 'complete': 2, 'thought': 9, 'only': 7, 'one': 6, 'clause': 1}

:: vector shape - (3, 11)

:: vector list - [[0.50461134 0.          0.          0.38376993 0.29803159
0.          0.          0.50461134 0.          0.50461134]
[0.          0.          0.54645401 0.54645401 0.          0.32274454
0.          0.          0.54645401 0.          ]
[0.          0.50461134 0.          0.          0.38376993 0.29803159
0.50461134 0.50461134 0.          0.          0.          ]]
```

Figure 2.4: Example of TF-IDF. (Source: <https://www.kaggle.com/code/amar09/text-pre-processing-and-feature-extraction>).

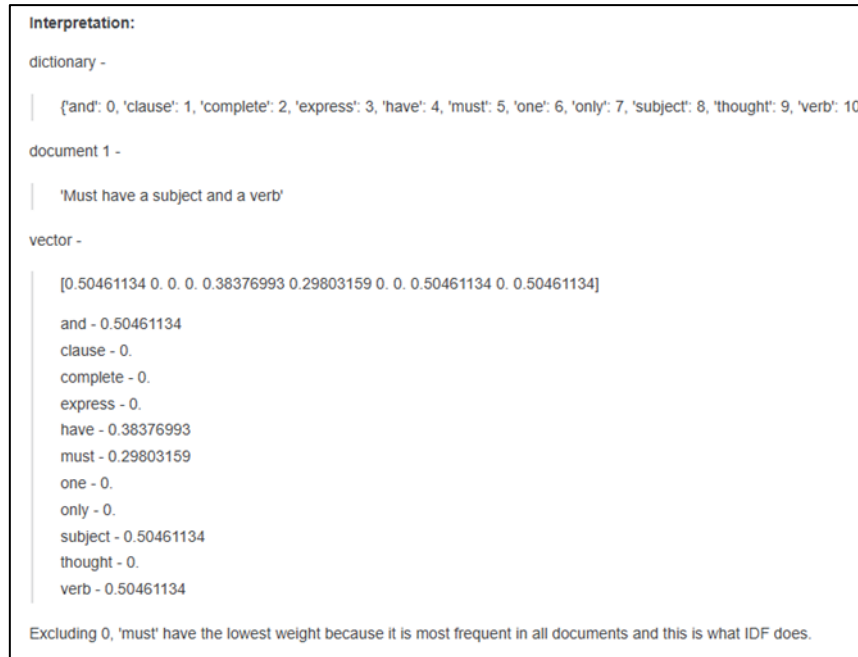


Figure 2.5: Figure 2. 5: The explanation of TF-IDF. (Source: <https://www.kaggle.com/code/amar09/text-pre-processing-and-feature-extraction>).

2.4.1.5 Text Cleaning and Normalization

These techniques include removing special characters and punctuation, which eliminates symbols and punctuation marks that carry little semantic value. Text is converted to lowercase to ensure case-insensitive analysis and reduce vocabulary size. Common words, known as stop words, such as "the" and "is," are removed to decrease noise and data dimensionality. Numerical data and symbols are treated or replaced based on the needs of the analysis. HTML tags and URLs are removed from web-sourced text using regular expressions (Frank et al., 2024). Additionally, irrelevant information such as headers, footers, and metadata is eliminated to focus on the core content of the text. (Tabassum & Patil, 2020) describe additional preprocessing methods that enhance the quality and consistency of data for subsequent NLP tasks. These methods include sentence segmentation, which splits text into individual sentences at punctuation marks to

facilitate further processing. Converting all text to lowercase prevents the same words in different cases from being treated differently, while parts-of-speech (POS) tagging assigns grammatical classes (e.g., noun, verb) to words, improving context understanding. Removing punctuation using regular expressions cleans up the text, making it more suitable for analysis. Overall, these techniques ensure cleaner and more consistent data, optimizing the preprocessing stage for NLP applications.



Figure 2.6: Example of text cleaning and normalization based on (Kim & Gil, 2019).

2.5 Topic modelling

Topic modelling serves as a fundamental technique within the realm of text mining, aimed at identifying groups of words, termed topics, within a body of text, as highlighted by (Albalawi et al., 2020). Topic modelling utilizes data mining algorithms, natural language processing, and machine learning to extract valuable insights from unstructured data sources like social media. By analysing user-generated content, text mining can provide invaluable insights that inform business strategies and trends across various domains, including health, agriculture, education, e-commerce, and transportation. The applications of topic modelling are diverse, encompassing fields such as natural language processing, information retrieval, text classification, clustering, machine learning, and recommendation systems.

As discussed by (Chauhan & Shah, 2022), topic modelling techniques reduce text collections to a compact set of topics, each consisting of related words, thereby uncovering the underlying structure of textual data, and facilitating exploration of large volumes of text. These statistical methods yield valuable insights into the distribution of topics within documents, the coverage of topics across documents, and the overall distribution of themes within the collection. Additionally, (Kwon et al., 2021) emphasize that topic modelling analysis involves sifting through large volumes of documents to propose relevant topics using probabilistic distribution models, while (Kherwa & Bansal, 2020) highlight the role of algorithms in uncovering thematic patterns within document collections and organizing them into clusters of topics based on the vector space model. Overall, topic modelling serves as a powerful tool for summarizing lengthy documents, analysing various text sources, and uncovering hidden semantic aspects within textual data.

2.6 Brief background of Topic Modelling Techniques

2.6.1 Latent Dirichlet Allocation (LDA)

LDA, or Latent Dirichlet Allocation, is widely recognized as a powerful probabilistic model for extracting topics from document collections due to its accuracy and online training capability (Albalawi et al., 2020). In the LDA framework mentioned by (Kherwa & Bansal, 2020), documents are treated as a blend of latent topics, each represented by a distribution of words.

This Bayesian statistical model operates in an unsupervised manner, extracting thematic information from documents without requiring prior training data. LDA excels in providing semantically interpretable data and can handle long documents, displaying both adjectives and nouns in topics. It also accommodates mixed-length documents and enhances transitive relations between topics, capturing high-order co-occurrence in small documents like paragraphs and sentences. However, (Chauhan & Shah, 2022) caution that LDA requires aggregation of short messages to mitigate data sparsity and is unable to model relations among topics to understand deep structures of documents. The algorithm's speed may be a limiting factor, and it necessitates a predefined number of topics, leading to issues if the number is too small or too large. Despite its strengths, LDA may sometimes produce semantically incorrect results.

The advantages and disadvantage of LDA as mentioned by (Albalawi et al., 2020).

Advantage	Disadvantage
• It does not require any previous training data. • Provides more semantically interpretable data and performs well if there is no time constraint.	• Needs aggregation of short messages to avoid data sparsity in short documents. • Unable to model relations among topics that help to understand deep structures of documents. A slow process algorithm.

• Handles long documents and can show adjectives and nouns in topics. • Handles mixed-length documents. • Able to enhance transitive relations between topics and obtain high-order co-occurrence in small documents like in paragraphs and sentences text.	• Sometimes provides semantically incorrect results.
---	--

Table 2.1: Advantages and disadvantages of Latent Dirichlet Allocation (LDA).

2.6.2 Non-Negative Matrix Factorization (NMF)

NMF, or Non-Negative Matrix Factorization, stands out as a versatile unsupervised method rooted in linear algebra, capable of simultaneously performing dimension reduction and clustering tasks, as emphasized by (Albalawi et al., 2020). While commonly employed in various topic modelling endeavours, its utilization in determining topics for short texts remains relatively limited. The essence of the NMF model lies in decomposing matrices, such as the term-document matrix, or bag-of-words representation of a text corpus, to derive meaningful topics without prior knowledge of the data's structure. Despite occasional semantic inaccuracies, NMF yields favourable outcomes in diverse domains, including image processing, text analysis, and transcription, as it adeptly handles non-interpretable data like videos.

Moreover, NMF excels in swiftly processing large volumes of real-time data and is particularly suited for tasks involving word and vocabulary recognition. According to (Egger & Yu, 2022), NMF operates on TF-IDF transformed data, decomposing it into terms-topics and topics-documents matrices iteratively. The algorithm's effectiveness hinges on the non-negativity of entries in these matrices, facilitating easier topic interpretation. Preprocessing steps like lowercasing, stop word removal, and lemmatizing are essential before applying NMF to meet its data requirements. In the domain of chemometrics, NMF, also known as "self-modelling curve resolution," pioneered by

Paatero and Tapper, serves to estimate basis vectors and weight coefficients to explain observation vectors. Its core assumptions, including additivity and non-negativity, underpin its efficacy in various applications, as highlighted by (Hamamoto et al., 2022).

The advantages and disadvantages of NMF as discussed by (Albalawi et al., 2020).

Advantages	Disadvantage
• Fast process for large amount of real-time data. • Able to extract meaningful topics without prior information or knowledge of the underlying meaning in the original data. • Appropriate for word and vocabulary recognition tasks.	• Sometimes provides semantically incorrect results.

Table 2.2: Advantages and disadvantages of Non-Negative Matrix Factorization (NMF).

2.6.3 Top2vec

Top2Vec represents a novel approach to topic modelling by utilizing jointly embedded document and word vectors to identify prominent topics within a corpus. This algorithm leverages semantic embeddings created using models like Doc2Vec or pre-trained transformers such as BERT and the Universal Sentence Encoder (USE) to capture the semantic meaning of words and documents. The process begins by embedding the documents and words into a high-dimensional semantic space. To mitigate the curse of dimensionality and enhance the clustering process, dimensionality reduction is performed using Uniform Manifold Approximation and Projection (UMAP). Following this, Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) is employed to identify dense clusters of document vectors, which correspond to distinct topics. These clusters are then represented by their centroids, known as topic vectors (Angelov, 2020). Additionally, the jointly embedded vectors allow for vector algebra

operations like those used in word2vec, facilitating the exploration of relationships between topics, documents, and words. This capability can be particularly useful for tasks such as semantic search and document clustering (Hendry et al., 2021).

The advantages and disadvantages of Top2vec as mentioned by (Egger & Yu, 2022).

Advantages	Disadvantages
• Supports hierarchical topic reduction • Allows for multilingual analysis • Automatically finds the number of topics • Creates jointly embedded word, document, and topic vectors • Contains built-in search functions (easy to go from topic to documents, search topics, etc.) • Can work on very large dataset sizes. • It uses embeddings, so no preprocessing of the original data is needed	• The embedding approach might result in too many topics, requiring • labor-intensive inspection of each topic • Generates many outliers • Not very suitable for small datasets (<1,000) • Each document is assigned to one topic • Objective evaluation metrics are missing

Table 2.3: Advantages and disadvantages of Top2vec.

2.6.4 Bidirectional Encoder Representations from Transformers (BERT)

According to (Alammary, 2022), BERT short for Bidirectional Encoder Representations from Transformers, developed by Jacob Devlin and colleagues at Google in 2018, has emerged as a foundation in natural language processing research. BERT adopts a bidirectional approach, allowing it to consider both left and right context when predicting words. This bidirectionality, combined with its unsupervised design, enables BERT to achieve robust performance across various NLP tasks, particularly text classification. As detailed by (Mohammadi & Chapon, 2020), BERT operates by extracting word and sentence embeddings, leveraging an attention mechanism to comprehend the relationships between words in input texts.

BERTopic is a combination of several modules that representing text as Embeddings using a pre-trained language model-specifically the Bidirectional Encoder Representations from Transformers (BERT). Generates a dense matrix where text either in sentences or paragraphs is presented as numeric vectors that capture the semantic similarity. (Umamaheswaran et al., 2023). BERTopic extracts topic representations using a class-based variation of TF-IDF. It performs dynamic topic modelling by tracking how topics grow over time. Each document is converted into its embedding representation using a pretrained language model. The BERTopic algorithm reduces the dimensionality of the embeddings to optimize the clustering process (Sibitenda et al., 2024). BERTopic relies on word embeddings to generate topic representations. It involves three steps including document embeddings, dimension reduction and clustering, and topic representations. First step, each document is converted to a dense vector through pre-trained language models. Second step, reducing the dimensionality of vector representations using the Uniform Manifold Approximation and Projection - UMAP algorithm. Third step, the model calculates the importance of words within a cluster through a modified TF-IDF representation called a class-based term frequency inverse document frequency (c-TF-IDF) (Kastrati et al., 2023).

BERTopic is a neural topic modelling with a class-based TF-IDF (Term Frequency-Inverse Document Frequency) procedure (Eren & Sert, 2023). BERTopic is recognized for its modularity, enabling the incorporation and utilization of diverse algorithms within its framework. (Jung et al., 2023). uses HDBSCAN clustering approach to group similar documents into topics based on their BERT embeddings. Post-preprocessing step, including remove noisy documents, similar topics were consolidated based on coherence and exclusivity measures (Banerjee & Pan, 2024). The modular nature of BERTopic allows

the exploration of different sentence transformers, clustering, and dimensionality reduction algorithms to design the topic model (Kumi et al., 2024).

The strengths and weakness as discussed by (Grootendorst, 2022).

Strengths	Weakness
• BERTopic is highly adaptable, delivering competitive performance with both classical and state-of-the-art language models. Its scalability with advanced models and stability across various setups, such as using Doc2Vec without a GPU, make it versatile for diverse scenarios. • Separating document embedding from topic representation adds flexibility, allowing different preprocessing for embeddings and topic generation. For instance, stopwords can be removed for topic representation but retained for embeddings. Clusters can also be fine-tuned (e.g., adjusting n-grams) without re-clustering. • By using a class-based TF-IDF approach, BERTopic represents topics as word distributions, enabling it to model dynamic topic evolution and variations across classes with minimal algorithm changes.	• BERTopic assumes each document contains only one topic, which may not reflect reality. While documents can be split into smaller segments or topic probabilities inferred using HDBSCAN's probability matrix, this approach is not ideal and does not fully address the issue during training. • BERTopic uses transformer-based models for contextual document representation, its topic representations are generated from bags of words, which lack contextual information. This can lead to redundancy and reduced interpretability, as related words in a topic may appear repetitive.

Table 2.4: Advantages and disadvantages of BERTopic.

2.7 Evaluation of topic modelling performance

Topic coherence measures how semantically related and consistent the top words in a topic are, indicating how well the topic represents the documents it is derived from (Umamaheswaran et al., 2023). For evaluation, Topic Coherence (TC) is commonly used, with higher values (closer to 1) indicating stronger relationships between topic words. This study employed the CV coherence measure, which combines techniques like sliding

windows, word segmentation, and normalized pointwise mutual information (NPMI) to assess semantic similarity (Mihajlov et al., 2024).

Coherence metrics evaluate the interpretability of topics by analysing the related of the top-k words. These metrics involve word segmentation and probability estimation, providing scores between 0 (low coherence) and 1 (high coherence) (Papadia et al., 2023). Traditional measures like perplexity and held-out likelihood are useful for comparing models with different topic numbers, but they focus on predictive accuracy rather than topic interpretability (Goyal & Kashyap, 2023).

2.8 Web development and NLP tools

Natural Language Processing (NLP) and web development are commonly used for reliable data processing and collection. To enable effective data collection and analysis, this section introduces some important web scraping libraries and tools.

2.8.1 Web scraping tools and libraries

Web scraping is the process of extracting data from websites to obtain valuable information. The extracted data is typically exported into a usable format, such as a spreadsheet (Matta Priya et al., 2020). The following tools and libraries are commonly used for this process, each designed for specific use cases.

No.	Libraries	Description
1.	BeautifulSoup	Python library designed for parsing HTML and XML documents, enabling users to navigate and manipulate Document Object Model (DOM) structures. Its simplicity and effectiveness make it a popular choice for extracting targeted data, such as restaurant details or user reviews, for analytical purposes (Agudelo-Santos & Zablah, 2023; Rita et al., 2023).
2.	Selenium	Selenium is an open-source web automation tool that supports multiple programming languages, including Python, Java, and JavaScript. Its cross-platform compatibility ensures seamless execution across operating systems without modifying scripts. Selenium integrates with major web browsers such as Google

		Chrome, Firefox, and Safari, enabling dynamic interaction with web elements using IDs, XPaths, and CSS Selectors (Kolte, 2023; Agrawal, 2023).
--	--	--

Table 2.5: Web scraping libraries.

No.	Tools	Description
1.	ChromeDriver	ChromeDriver acts as a standalone server facilitating browser automation via Selenium. It enables tasks such as navigating between pages, filling web forms, and executing JavaScript, making it indispensable for comprehensive web scraping workflows.
2.	WebDriverManager	WebDriverManager is a Java library that simplifies browser driver management for Selenium WebDriver. It automates the downloading and configuration of browser drivers, such as ChromeDriver and GeckoDriver, based on the user's operating system and browser version, thus reducing manual overhead and ensuring compatibility across environments.

Table 2.6: Web scraping Tools.

2.8.2 Data pre-processing tools and libraries

Data preprocessing is an important step in preparing raw data for natural language processing (NLP) tasks. The following libraries are used for text preprocessing.

No.	Libraries	Description
1.	Natural Language Toolkit (NLTK)	NLTK is a comprehensive Python library for natural language processing, offering tools for text cleaning, tokenization, and manipulation. It is widely used for preparing text data for machine learning tasks (Sakthi Vel, 2021). Key features of NLTK include: • <code>nltk.corpus.stopwords</code>: A module for removing common English stopwords. • <code>nltk.stem.WordNetLemmatizer</code>: A tool for lemmatization, which reduces words to their base or root form. • <code>nltk.download</code>: A function to download essential resources like stopword lists and lemmatization datasets.

2.	CountVectorizer	CountVectorizer is a feature extraction tool that converts textual data into numerical representations, essential for machine learning models. It tokenizes text and builds a vocabulary of unique words, representing each document as a sparse vector. Using the Scikit-learn library, CountVectorizer operates in three steps: instantiation, fitting to the dataset, and transforming text into vectors. These vectors capture word frequencies and positional data in lexicographic order, facilitating efficient text analysis (Sakthi Vel, 2021).
----	-----------------	--

Table 2.7: Data pre-processing tools and libraries.

2.8.3 Topic modelling libraries

The libraries listed were chosen based on frequent use in topic modelling tasks. These selected tools are widely recognized for their efficiency, versatility, and support within the Python, making them important for projects involving natural language processing and data-driven insights.

No	Libraries	Description
1.	Pandas	Used for performing DataFrame operations such as analysing data in tables and organizing or filtering data. (Source: https://www.shiksha.com/online-courses/articles/top-10-powerful-python-libraries-for-data-science/)
2.	NumPy	A library for numerical operations, arrays, and mathematical computations. (Source: https://www.shiksha.com/online-courses/articles/top-10-powerful-python-libraries-for-data-science/)
3.	Matplotlib	A data visualization library for creating charts and plots. (Source: https://www.shiksha.com/online-courses/articles/top-10-powerful-python-libraries-for-data-science/)
4.	Scikit-learn	A machine learning library used for data mining, training, and deploying models. (Source: https://www.shiksha.com/online-courses/articles/top-10-powerful-python-libraries-for-data-science/)

5.	Gensim	An open-source Python library designed for efficient topic modelling and vector space modelling (Kherwa & Bansal, 2020).
6.	WordCloud	A visualization tool for creating tag clouds or weighted lists from text data. (Source: https://www.datacamp.com/tutorial/wordcloud-python)
7.	Pickle	A library for saving and loading Python objects like data or models for reuse. (Source: https://docs.python.org/3/library/pickle.html)
8.	UMAP (umap-learn)	A dimensionality reduction technique for visualizing or simplifying large datasets. (Source: https://umap-learn.readthedocs.io/en/latest/)
9.	HDBSCAN	A library for unsupervised learning to find clusters or dense regions in datasets. (Source: https://hdbscan.readthedocs.io/en/latest/)
10	Seaborn	An extension of Matplotlib that provides a high-level interface for statistical and graphical analysis. (Source: https://www.shiksha.com/online-courses/articles/top-10-powerful-python-libraries-for-data-science/)
11.	AdjustText	A library for adjusting text positions to make labels on plots easier to read. (Source: https://adjusttext.readthedocs.io/en/latest/)

Table 2.8: Topic modelling libraries.

2.8.4 Web development tools

2.8.4.1 Svelte

A framework that acts as a compiler, allowing developers to write minimal code in the browser. It enables direct updates to variables without additional code or hooks, simplifying the handling of updates and changes in projects. Developers can create concise components using familiar languages like HTML, CSS, and JavaScript.

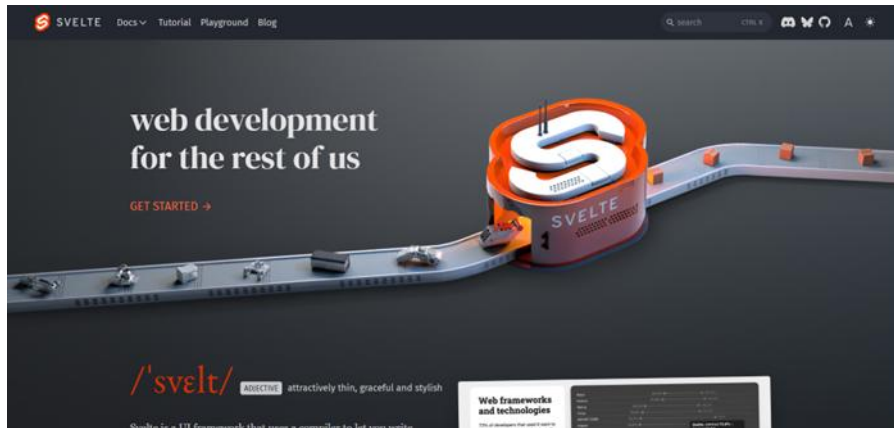


Figure 2.7: Svelte website (Source: <https://medium.com/coderhack-com/comparing-javascript-frameworks-svelte-vue-react-and-angular-70474666e69d>).

2.8.4.2 Visual Studio code

A source code editor with powerful developer tools like IntelliSense code completion and debugging. It supports macOS, Linux, and Windows and has built-in support for JavaScript, TypeScript, and Node.js. It also allows extensions for various programming languages (e.g., C++, Python, Java) and environments (e.g., Docker, Kubernetes) and integrates with cloud platforms like AWS, Azure, and Google Cloud.

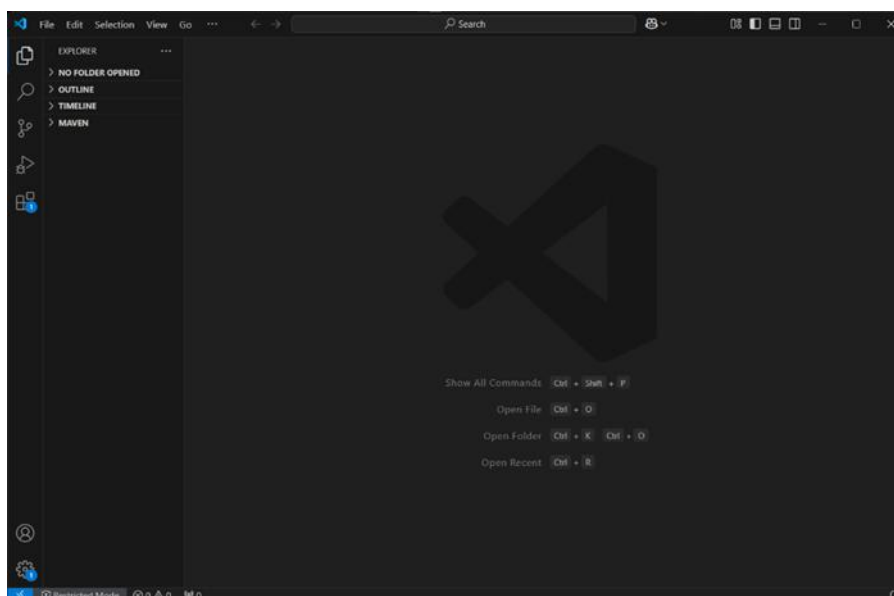


Figure 2.8: Visual Studio code application (Source: <https://code.visualstudio.com/docs/editor/whyvscode>).

2.8.4.3 Figma

Figma is a cloud-based design tool widely used for the creation of user interfaces (UI) and user experiences (UX). Its primary advantage lies in its ability to facilitate real-time collaboration among designers, making it particularly beneficial for remote teams. Figma supports vector graphics editing, prototyping, and wireframing, while also enabling the development of interactive designs through features such as components, styles, and design systems. Additionally, Figma provides developers with the capability to inspect and export design assets directly from the platform, streamlining the design-to-development process.

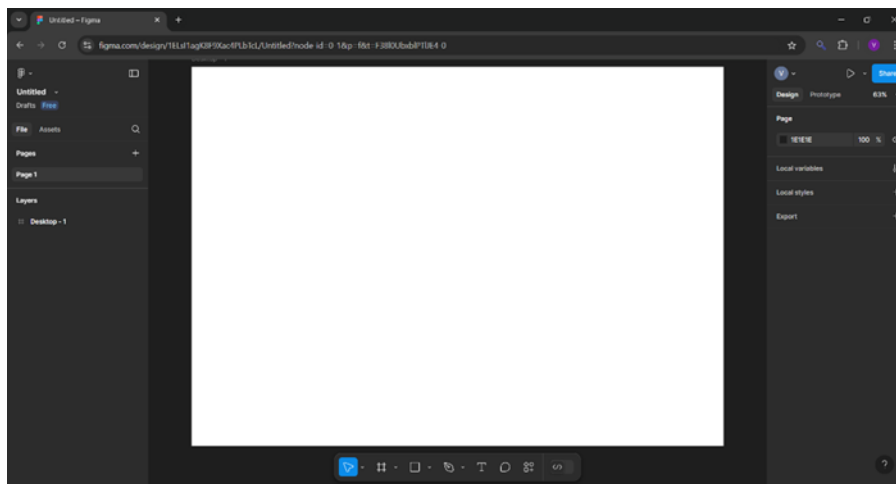


Figure 2.9: Figma website (Source: <https://www.figma.com/collaboration/>).

2.9 Motivation of research

In today's digital world, businesses and individuals face challenges related to the large amount of information, particularly in business news. This information overload complicates decision-making and makes it difficult to extract meaningful information. Extracting and processing data from business news involves several key steps, such as web scraping, data cleaning, and text pre-processing, which are essential for transforming

raw data into a structured format suitable for analysis. This gap presents an opportunity to investigate how such techniques can be utilized to support decision-making in this specific context, especially through the effective extraction and processing of relevant data.

This research aligns with the growing need for automation in information processing, especially as businesses increasingly rely on data-driven strategies to navigate economic and political risks. Motivated by the challenges faced during my industrial training in automating news collection and analysis, this research seeks to refine topic modelling approaches for practical applications in understanding business activities and risks. The research will explore various topic modelling methods, such as Latent Dirichlet Allocation (LDA), Non-negative Matrix Factorization (NMF), Top2Vec and BERTopic, to determine the most effective approach for extracting relevant topics from Malaysian business news. By comparing these methods, the study will identify the technique that best captures the themes within the data. Once the topics are identified, they can be used to generate actionable insights for decision-making. These insights could guide businesses in understanding emerging trends, identifying risks, and making informed decisions based on the patterns detected in the news data. Through this approach, the research aims to demonstrate how topic modelling can enhance business intelligence and support strategic decision-making in the context of Malaysian business news.

2.10 Summary

In this chapter, the Natural Language Processing techniques, web scraping, data preprocessing, and topic modelling are explained. It also covers the topic modelling techniques, tools, and libraries.

Chapter 3: Methodology

3.1 Introduction

This chapter focuses on the development and implementation of a web-based application designed to demonstrate the application of topic modelling techniques. The application provides an interactive platform for users to explore topics derived from Malaysian business news articles. The chapter begins by outlining the development methodology, including the tools and technologies for creating back end and front-end interfaces. It further details the research methodology, which outline the data collection through web scraping, preprocessing raw text data, and the application of topic modelling. Integrating these techniques into the web application highlights their utility in analysing datasets and extracting topics. Through a structured approach, this chapter demonstrates the development and methodology, ensuring the research objectives are effectively achieved.

3.2 Web application development

This section outlines the development of a simple web-based application designed to display the results of the topic models generated in this research. The web-based application integrates the topic model into a system accessible to users, demonstrating its practical applicability as a proof of concept. To demonstrate the usage of the topic model, a simple web-based system has been developed for the study. The proposed application aims to provide users with an interactive interface to explore the identified topics. By enabling dynamic interaction with the data, the application demonstrates the practical utility of the topic models trained in this study. The following subsections describe the development methodology employed for the application and the graphical user interface

(GUI) design, highlighting the key features and functionalities integrated into the application.

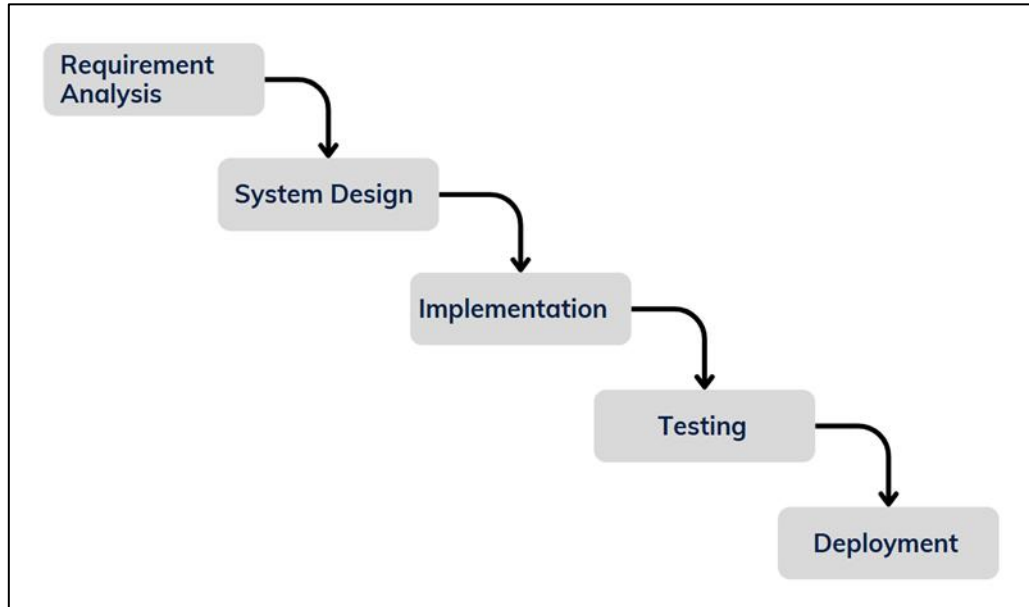


Figure 3. 1: Waterfall methodology.

The **Requirement analysis** phase involved selecting tools that support the development of the system with back end and front-end interface. Python has extensive libraries for natural language processing (NLP) and topic modelling. Figma is selected for designing a user-friendly graphical user interface (GUI). Visual Studio Code used as the integrated development environment (IDE) for coding and debugging, while Svelte is use as the front-end framework for its lightweight and reactive capabilities. During the **System design** phase, Figma was used to create wireframes and prototypes of the graphical user interface (GUI). The design focused on user experience, ensuring that detected topics were displayed clearly and interactively. Figma allowed for iterative design improvements based on feedback. The finalized designs were translated into components using Svelte, applying its effective state management and rendering to provide a smooth user experience.

The **Implementation** phase involved developing both the back end and front-end components of the system. For the back end, Python was used in combination with libraries to handle server-side logic, integrate topic modelling algorithms, and process data efficiently. On the front end, Svelte was employed to build a responsive and dynamic user interface based on the designs created in Figma. Its component-based architecture allowed for modular development and smooth integration with the back end. Visual Studio Code was used as the primary development environment for both back-end and front-end development. Its features, including syntax highlighting, debugging tools, and extensions for Python and Svelte, streamlined the coding and debugging process. Next, the **Testing** phase was conducted to ensure the system's functionality, usability, and integration. Back-end testing utilized Python's framework to validate the functionality of topic modelling algorithms. Front-end testing relied on Svelte debugging tools and browser-based testing to confirm that the user interface performed as expected. Integration testing was carried out using Visual Studio Code's debugging tools, ensuring smooth communication between the front-end and back-end components.

Finally, the **Deployment** phase involved optimizing the Svelte-based front-end for performance and deploying the Python back-end on a local or cloud server. Visual Studio Code was used for managing version control and finalizing the system for usability testing. The GUI, based on Figma designs, was tested for user-friendliness and accessibility.

3.3 Web application development

3.3.1 Use case diagram

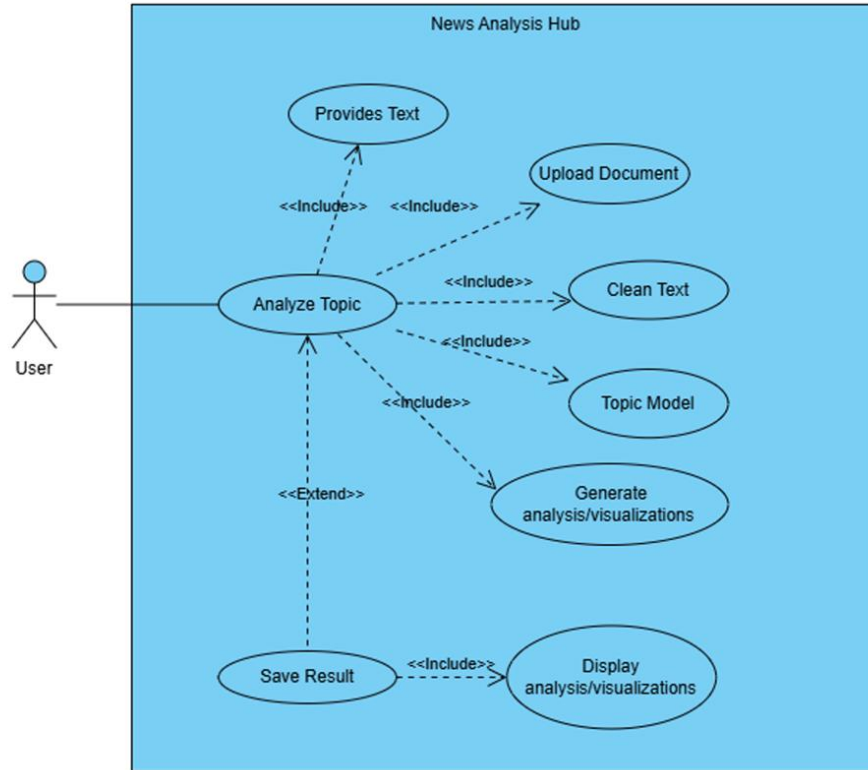


Figure 3. 2: Use case diagram of News Analysis Hub.

The use case provides illustrations of the News Analysis Hub is a simple, web application designed to analyse text and document for underlying topics and provide visualizations of the results. The system does not have a database, instead it processes user-provided text and document dynamically. The primary actor in the system is the User, who interacts directly with the application. The main use case is to Analyse Topic, which consists of several steps. First, the User inserts text by entering news content that needs analysis or upload document of data of news articles. The system performs data preprocessing to cleans the text, automatically removing noise such as stopwords and punctuation. Next, the system implements a topic modelling algorithm (e.g., NMF) to extract key topics from the cleaned data. After processing, the system generates analysis

and visualizations, presenting outputs such as topics related, word clouds, and graphical representations of the discovered topics. Additionally, there is an optional use case, Save Result, which allows the user to save the analysis result. This action extends the main use case since saving is optional and occurs only after the results have been generated. Before saving, the system includes the step of displaying analysis and visualizations, enabling the user to review the findings on-screen before save the result.

3.3.2 Sequence Diagram

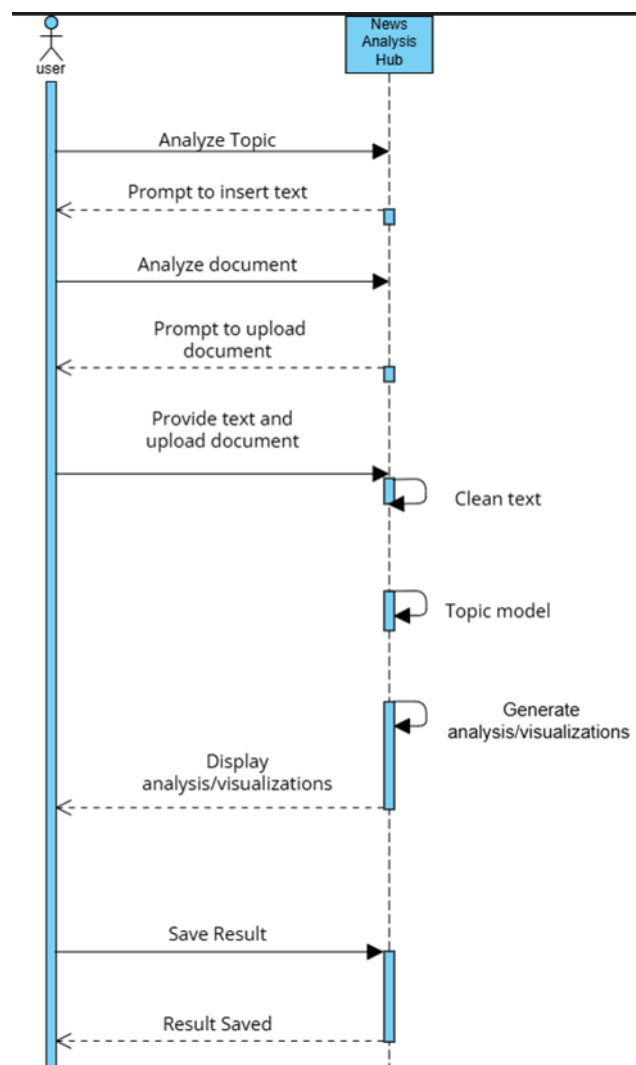


Figure 3. 3: Sequence diagram of News Analysis Hub.

This sequence diagram illustrates the News Analysis Hub is a web application that enables a single user to enter text or upload document and quickly obtain topic analysis along with visualizations. The system follows a structured sequence of interactions. First, the user initiates the Analyse Topic process, by inserting text or upload document in the input box. Once the User insert or upload, the system receives it and begins internal processing. The next step, Clean Text and Topic Modelling, where the system automatically removes unwanted text before implementing a topic modelling algorithm such as NMF. Then, the system will Generate Analysis/Visualizations, producing outputs such as the top topics, word clouds, and graphical representations. In the Display Analysis/Visualizations step, allowing the user to review the discovered topics along with visual elements. The user can save the result using Save Result option, which enables saving or downloading the analysis. This system is one-time-use web application and does not utilize a database.

3.3.3 Proposed GUI



Figure 3. 4: The proposed web application GUI.

Figure 3.4 illustrates the proposed graphical user interface (GUI) for the news topics analysis system. This web application integrates functionalities for web scraping and data cleaning directly within its interface. Users can input text from news websites by copying and pasting it into the designated text box. Upon clicking the "Start" button, the application initiates the web scraping and data cleaning processes, ensuring the input data is prepared for subsequent topic detection. This design outlines the workflow, allowing users to perform essential preprocessing tasks conveniently within the web application.

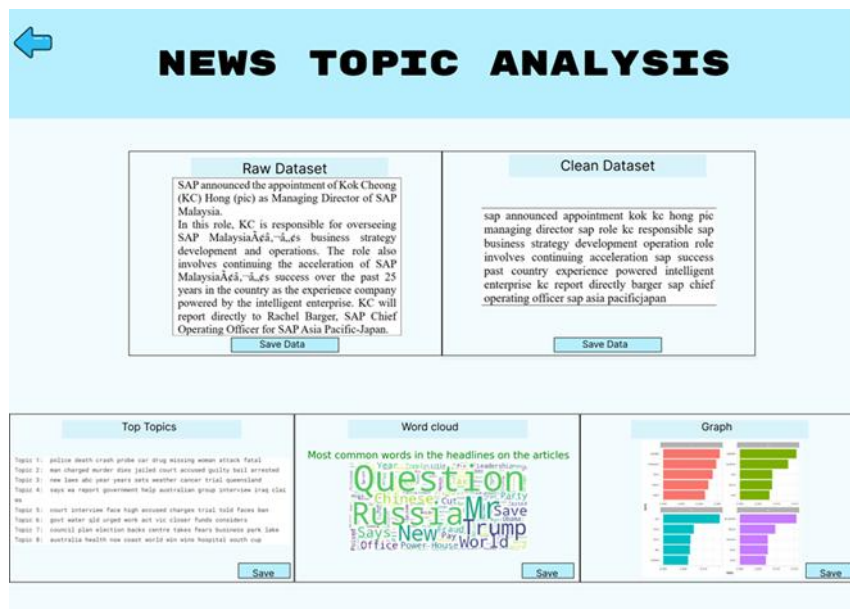


Figure 3. 5: The proposed web application GUI.

Figure 3.5 show the proposed graphical user interface (GUI) for the web application, highlighting how users can interact with and view the results. This page presents the outcomes of the web scraping and data cleaning processes, offering users a clear and structured view of the processed data. Additionally, the GUI integrates the topic modelling phase, displaying visualizations such as the top topics, word clouds, and graphical representations of the extracted data. The web application is developed using Svelte UI

for the front-end interface and Visual Studio Code as the development environment, ensuring a user-friendly and efficient platform for topic detection and visualization.

3.4 Research Methodology

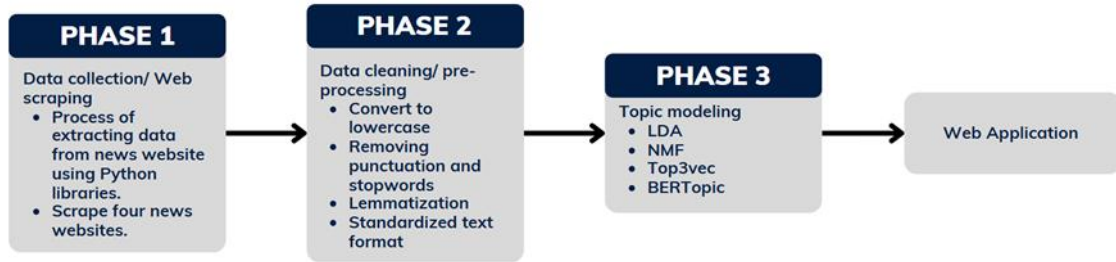


Figure 3. 6: Research methodology.

Figure 3.6 outlines the research methodology, divided into three primary phases: web scraping, data preprocessing, and topic modelling. In the first phase, web scraping techniques are employed to collect business news articles from four selected Malaysian news websites. The extracted data, including headlines, publication dates, and article bodies, is organized and stored in structured formats to facilitate subsequent analysis. The second phase involves preprocessing the raw text data to improve its quality and suitability for topic modelling. This includes cleaning and tokenizing the text, removing stopwords and punctuation, and saving the processed data in a newly structured file. The final phase implements topic modelling techniques such as Latent Dirichlet Allocation (LDA), Non-Negative Matrix Factorization (NMF), Top2Vec, and BERTopic to identify latent topics within the dataset. The performance of the models is evaluated using coherence scores to ensure the extracted topics are interpretable and relevant. Furthermore, the best-performing topic model is integrated into a web-based system designed to present topic-specific news articles collected from news websites, providing a practical demonstration of the methodology.

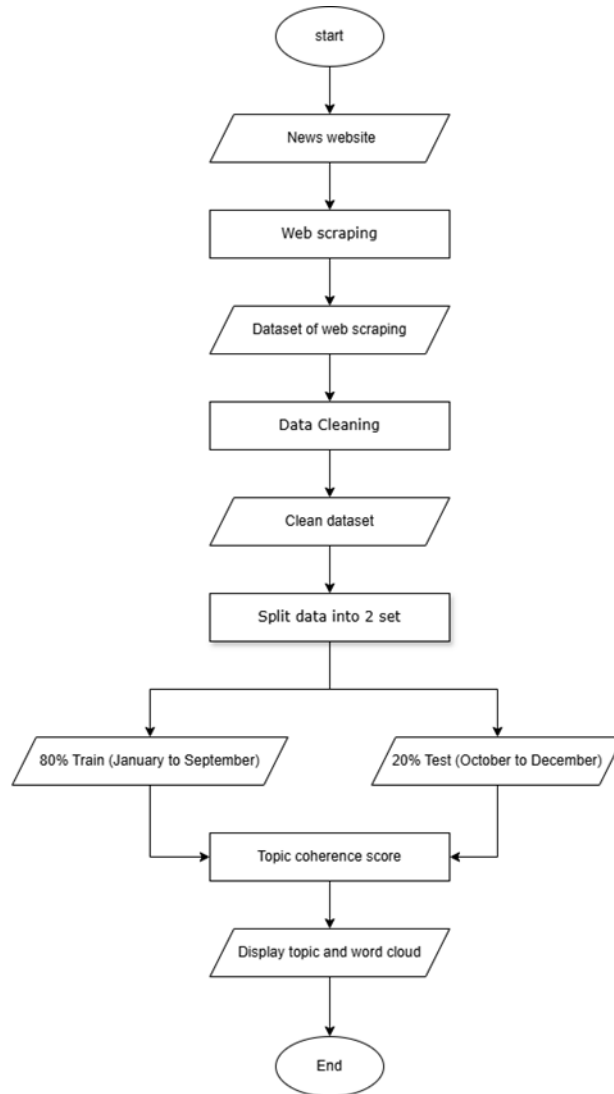


Figure 3. 7: Flowchart of the methodology.

Figure 3.7 illustrates the flowchart of the methodology. The figure provided a detailed view of the workflow, illustrating the step involved in this research.

3.5 Data collection

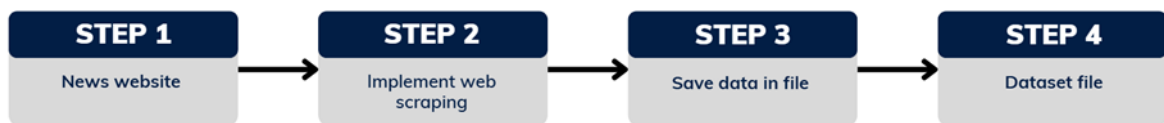


Figure 3. 8: The step for data collection.

The first phase focuses on data collection a comprehensive dataset of business news articles from selected Malaysian news websites. To achieve this, web scraping techniques are utilized, potential Python libraries such as BeautifulSoup and Selenium to automate the extraction of article content, headlines, publication dates, and other metadata. Articles spanning a specific timeframe, such as the past year, are collected to ensure the dataset reflects current trends in Malaysian business activities, political influences, and risks. The news articles are collected from selected Malaysian websites, including Business Today, Daily Express, Malaysian Reserve, and NST. To maintain data integrity, measures are taken to handle potential issues like duplicate articles, missing metadata, or broken links. The scraped data is stored in structured formats such as CSV files, ensuring compatibility with subsequent preprocessing steps. This phase ensures that the dataset is both diverse and representative of the research context, laying the foundation for meaningful topic modelling.

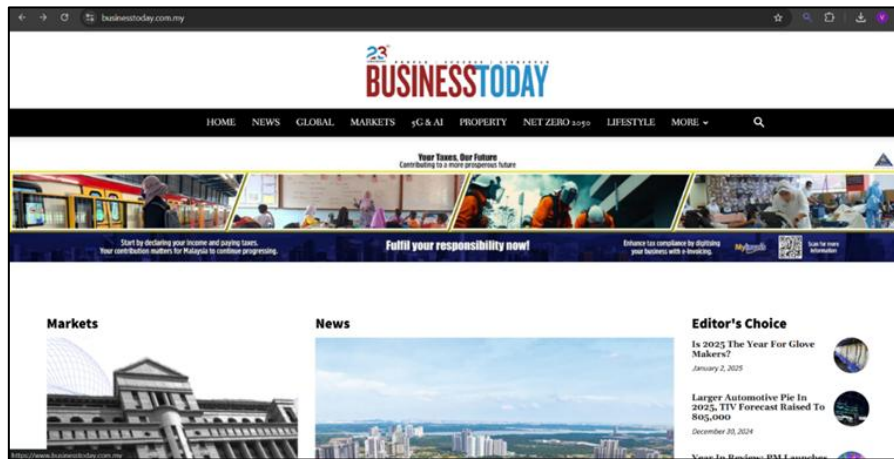


Figure 3. 9: The Business Today news website.

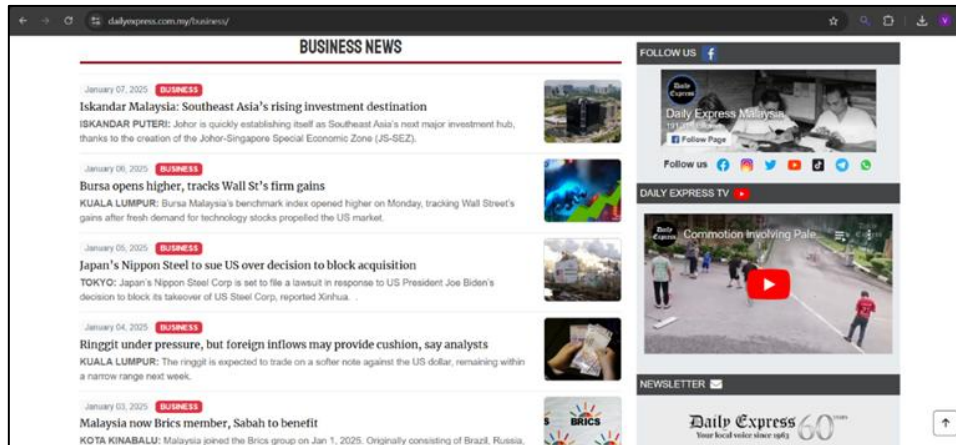


Figure 3. 10: Daily Express news website.

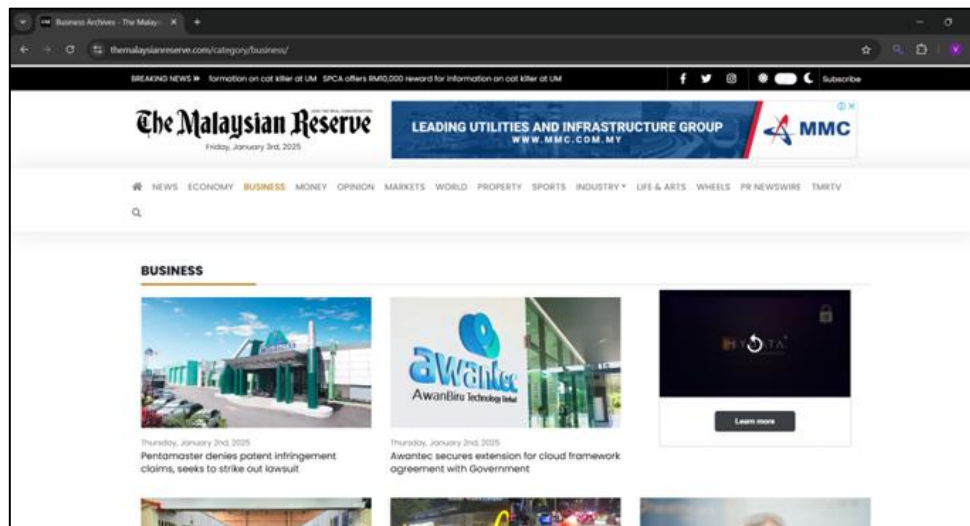


Figure 3. 11: The Malaysia Reserve news website.



Figure 3. 12: New Straits Times news website.

3.6 Data cleaning or pre-processing

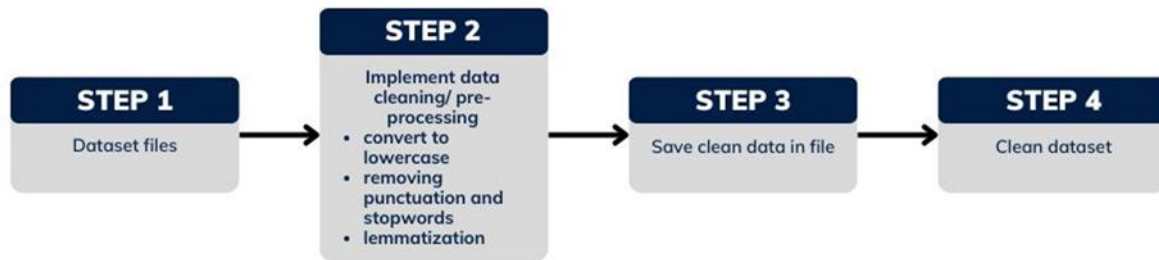


Figure 3. 13: The step for data cleaning or pre-processing.

The second phase involves preparing the raw text data for analysis by applying various preprocessing techniques. This step is crucial to improve the quality of the data and ensure that it is suitable for input into topic modelling algorithms. Key preprocessing tasks include text cleaning, which involves the removal of irrelevant elements such as HTML tags, special characters, numbers, and URLs. Lowercasing is applied to convert all text to lowercase, ensuring uniformity and eliminating case sensitivity. Stopword removal eliminates common stopwords (e.g., “and” “the,” “of”) that do not contribute to topic extraction, while lemmatization reduces words to their root forms (e.g., “running” to “run”) to standardize the vocabulary. The pre-processed dataset is saved in a clean and structured format, ready for modelling.

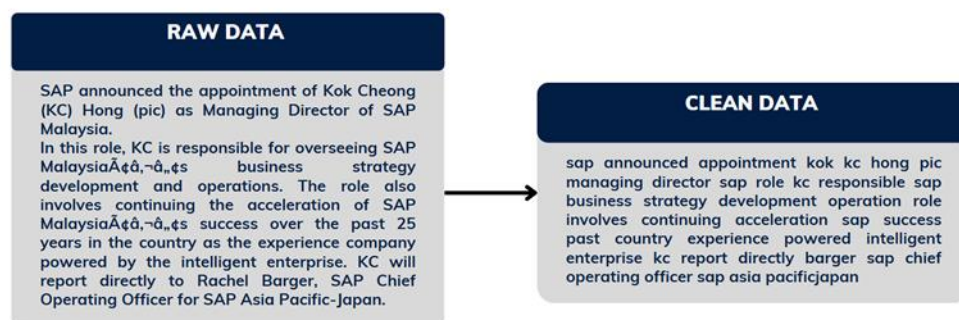


Figure 3. 14: Example of data after the cleaning process.

3.7 Topic modelling

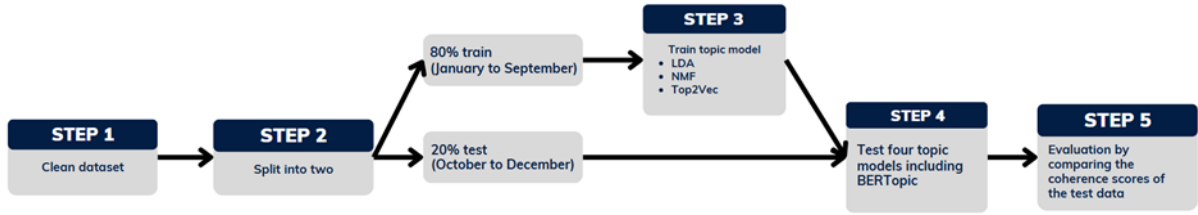


Figure 3. 15: The step for train and test topic modelling.

The third phase applies multiple topic modelling techniques to the cleaned dataset. The data is divided into two sets: a training set, comprising 80% of the articles (from January to September), and a testing set, comprising 20% of the articles (from October to December). This split ensures that the models are trained on one set of data and evaluated on a separate test set, enhancing their generalizability and reliability. These techniques aim to uncover hidden patterns and topics within the text data. The techniques include Latent Dirichlet Allocation (LDA), a probabilistic model that identifies topics as distributions over words. Non-Negative Matrix Factorization (NMF), a matrix decomposition technique that discovers latent topics by factorizing the term-document matrix. Top2Vec, a novel approach that uses semantic embeddings to identify topics, offering an advantage in capturing the relationships between words, and BERTopic, a pre-trained transformer-based model that generates interpretable topics using dynamic topic reduction and clustering. For training topic modelling, three techniques LDA, NMF, and Top2Vec will be used, as BERTopic is a pre-trained model and will not be included in the training phase. In the testing phase, all four techniques will be used. Each model is configured with hyperparameters, such as the number of topics and the number of words per topic, to ensure optimal performance. The generated topics are analysed for interpretability and coherence, with the top words for each topic being extracted for

further examination. Visualizations such as word clouds and bar plots are also created to enhance the understanding of the topics.

3.8 Evaluation and result

The final phase focuses on evaluating the performance of the topic modelling techniques based on their ability to generate coherent and meaningful topics. Evaluation metrics, such as coherence scores (e.g., CV), are calculated to assess the semantic consistency and interpretability of the topics, with higher scores indicating better performance. The four techniques LDA, NMF, Top2Vec, and BERTopic are compared to determine the most effective method for extracting topic from Malaysian business news. The results of this evaluation guide the selection of the most suitable topic modelling approach for the study. Finally, the findings are integrated into a simple web application to demonstrate the potential and usability of the topic models.

3.9 Tool development and training

This section provides a summary of the tools selected for web-based application development and model training, detailing their roles and functionalities within the research.

3.9.1 Web development tools

No.	Tools	Description
1.	Svelte UI	A framework that compiles code for efficient updates, enabling concise components using HTML, CSS, and JavaScript.
2.	Visual Studio code	A versatile code editor with features like IntelliSense, debugging, and support for multiple languages and extensions.
3	Figma	A web-based design tool for collaborative UI/UX design and prototyping.

Table 3. 1: Web development tools.

3.9.2 Web scraping tools and libraries

No.	Libraries	Description
1.	BeautifulSoup	Parses HTML/XML to navigate and manipulate DOM structures for targeted data extraction.
2.	Selenium	Automates web interactions across browsers using IDs, XPaths, and CSS Selectors.
3.	ChromeDriver	Facilitates Selenium automation by enabling browser interactions like navigation and form-filling.
4.	WebDriverManager	Automates browser driver downloads and configuration, ensuring compatibility and ease of use.

Table 3. 2: Web scraping tools and libraries.

3.9.3 Data pre-processing tools and libraries

No.	Libraries	Description
1.	Natural Language Toolkit (NLTK)	Offers tools for text cleaning, tokenization, and lemmatization to prepare data for NLP tasks.
2.	CountVectorizer	Converts text into numerical vectors by tokenizing and capturing word frequencies.

Table 3. 3: Data pre-processing tools and libraries.

3.9.4 Topic modelling libraries

No	Libraries	Description
1.	Pandas	Used for performing DataFrame operations such as analysing data in tables and organizing or filtering data.
2.	NumPy	A library for numerical operations, arrays, and mathematical computations.
3.	Matplotlib	A data visualization library for creating charts and plots.
4.	Scikit-learn	A machine learning library used for data mining, training, and deploying models.
5.	Gensim	An open-source Python library designed for efficient topic modelling and vector space modelling.
6.	WordCloud	A visualization tool for creating tag clouds or weighted lists from text data.

7.	Pickle	A library for saving and loading Python objects like data or models for reuse.
8.	UMAP (umap-learn)	A dimensionality reduction technique for visualizing or simplifying large datasets.
9.	HDBSCAN	A library for unsupervised learning to find clusters or dense regions in datasets.
10	Seaborn	An extension of Matplotlib that provides a high-level interface for statistical and graphical analysis.
11.	AdjustText	A library for adjusting text positions to make labels on plots easier to read.

Table 3. 4: Topic modelling libraries.

3.10 Summary

This chapter described the development of a web-based application, and the research methodology. The application serves as a platform for exploring topics identified from Malaysian business news articles, integrating both front-end and back-end components for functionality. The research methodology was divided into three phases data collection through web scraping, preprocessing to clean and structure the data, and the application of topic modelling techniques. LDA, NMF, Top2Vec, and BERTopic to identify topics, with coherence scores used to evaluate their effectiveness. The integration of these techniques into the web application demonstrated their practical utility in processing and visualizing large datasets, providing users with actionable insights.

Chapter 4: Implementation

4.1 Introduction

This chapter presents the implementation and analysis phase of the project, focusing on the coding processes, results, and visualizations generated using Python programming. It highlights the methods and tools used to process the data, apply topic modelling techniques, and generate meaningful insights from the textual datasets. The chapter includes detailed explanations of the code used, the output produced, and the interpretation of plots such as top words and coherence score. Python was use due to its libraries and frameworks suitable for natural language processing (NLP), data analysis, and visualization, making it an ideal tool for this research.

4.2 Data collection

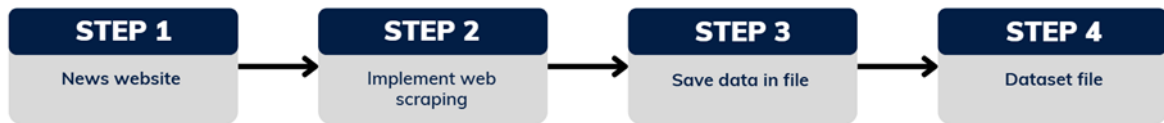


Figure 4. 1: The step for data collection.

Phase 1 focuses on data collection a comprehensive dataset of business news articles from selected Malaysian news websites. Web scraping techniques are utilised, with potential Python libraries such as BeautifulSoup and Selenium to automate the extraction of article content, headlines, publication dates, and other metadata. Articles spanning a specific timeframe, such as the past year, are collected to ensure the dataset reflects current trends in Malaysian business activities. The news articles are collected from selected Malaysian websites, including Business Today, Daily Express, Malaysian Reserve, and NST. The scraped data is stored in structured formats such as CSV files, ensuring compatibility with subsequent preprocessing steps.

4.2.1 Web Scraping

Step 1: News Website

To collect the news articles, implemented a web scraping procedure that targets four selected Malaysian websites: Business Today, Daily Express, Malaysian Reserve, and New Straits Times (NST). These websites were chosen based on their reputability and extensive coverage of business-related news. Using Python libraries such as BeautifulSoup for parsing static HTML content and Selenium for handling dynamic web pages, the scraping scripts automatically extracted key information including article headlines, publication dates, content, and metadata from each site.

Step 2: Implement web scraping

No.	Libraries	Description
1.	BeautifulSoup	Parses HTML/XML to navigate and manipulate DOM structures for targeted data extraction.
2.	Selenium	Automates web interactions across browsers using IDs, XPaths, and CSS Selectors.
3.	ChromeDriver	Facilitates Selenium automation by enabling browser interactions like navigation and form-filling.
4.	WebDriverManager	Automates browser driver downloads and configuration, ensuring compatibility and ease of use.

Table 4. 1: Web scraping tools and libraries.

The tools and libraries listed in Table 4.1 are the components for developing effective web scraping pipelines. Each plays a specific role in the process of automating data extraction from websites.

```
1  import requests
2  from bs4 import BeautifulSoup
3  from selenium import webdriver
4  from selenium.webdriver.common.keys import Keys
5  from webdriver_manager.chrome import ChromeDriverManager
6  import time
7  import pandas as pd
```

Figure 4. 2: Code to import libraries.

This code imports essential libraries for web scraping and data handling requests enable sending HTTP requests to retrieve webpage data, BeautifulSoup from the bs4 module parses and extracts information from HTML content, and selenium with its webdriver and Keys modules automates browser actions for dynamically loaded pages. The ChromeDriverManager from webdriver_manager.chrome automatically installs and manages the appropriate ChromeDriver, while time provides functions like sleep to pause execution and ensure pages load completely. Lastly, pandas are used to organize, manipulate, and eventually export the scraped data into formats like CSV.

1. Business today

```

# Extract content based on your desired structure (replace placeholders with actual selectors)
page_title = soup.select_one('h1.entry-title') # Assuming the title is within an h2 tag
content_divs = soup.select("div.dable-content-wrapper") # Assuming content is in divs with this class

for page_num in range(96, 102): # Adjust the range as needed
    print("Processing page", page_num)
    url = f"https://www.businesstoday.com.my/category/news/business/page/{page_num}"
    driver.get(url)
    time.sleep(5) # Adjust sleep time as needed

    html = driver.page_source
    soup = BeautifulSoup(html, "html.parser")

    # Extract links from the current page
    category_links = [a['href'] for a in soup.select("div.td-image-container a")]

    # Filter links for the year
    year_links = [link for link in category_links if f'/{2019}/' in link]

    # Check if any links were found after filtering
    if not year_links:
        print(f"No relevant links found on page {page_num}. Stopping scraping.")
        break # Stop scraping if no relevant links found

    # Scrape content from each link
    for link in year_links:
        scraped_content = scrape_content(link)
        if scraped_content:
            content.append(scraped_content)
            all_links.append(link)

df = pd.DataFrame(data={"Content": content, "URL": all_links})

```

Figure 4. 3: Code to scrape content in Business today.

Figure 4.3 shows the code snippet part of a web scraping script designed to extract business news articles from the website "Business Today Malaysia" for the year 2019. The code uses BeautifulSoup to extract specific HTML elements, it finds the first <h1> element with the class "entry-title" intended to hold the article's title and gathers all <div> elements with the class "dable-content-wrapper", which are assumed to contain the article's main content. It iterates through specific pages (from page 96 to 101) of the website's business news category, using a web driver to load each page and retrieve its HTML content. It then uses BeautifulSoup to parse the HTML and extract article links from image containers. These links are filtered to include only those containing "/ {2019}/"

in the URL, ensuring only articles from 2019 are collected. If no such links are found on a page, the scraping process stops early. For each relevant link, the script calls a `scrape_content` function to extract the article's content, storing it along with the URL in separate lists. Finally, the collected data is organized into a panda DataFrame for further use or analysis. Refer to Appendix A for the full source code.

2. Daily express

```
# Extract content based on your desired structure (replace placeholders with actual selectors)
page_title = soup.select_one('div.title.titlemobile') # Assuming the title is within a div with class 'title.titlemobile'
article_date = soup.select_one('div[class*="col-xs-12 col-sm-12 col-md-6 col-lg-6 col-xl-6"]') # Adjust the selector accordingly
content_divs = soup.select("span.newsContent") # Assuming content is in spans with class 'newsContent'

# Scrape content, date, and URL from each link
for link in year_links:
    full_url = f"https://www.dailyexpress.com.my{link}" # Add base URL to the relative link
    title, date, content = scrape_content(full_url)
    if title and date and content:
        all_links.append(full_url)
        all_dates.append(date)
        all_content.append(content)
```

Figure 4. 4: Code to scrape content in Daily express.

This code snippet is part of a web scraping process that collects article data from the “Daily Express” website. The code uses BeautifulSoup to extract key elements from a webpage: it retrieves the page title from a `<div>` with the compound class "title titlemobile," captures the article's date from a `<div>` whose class includes various Bootstrap grid identifiers and collects the article's content from all `<span>` elements with the class "newsContent.". It iterates over a list of filtered article links (`year_links`), appending the base URL (`https://www.dailyexpress.com.my`) to each relative link to form the full URL of the article. For each full URL, the script calls a `scrape_content` function to extract the article's title, publication date, and main content. If all three elements (title, date, and content) are successfully retrieved, the script appends the full URL to `all_links`, the date to `all_dates`, and the article content to `all_content`, thereby building separate lists

of cleaned and complete article information for further processing or analysis. Refer to Appendix A for the full source code.

3. Malaysia reserve

```
# Extract content based on your desired structure (replace placeholders with actual selectors)
page_title = soup.select_one('h2') # Assuming the title is within an h2 tag
content_divs = soup.select("div.dable-content-wrapper") # Assuming content is in divs with this class

for page_num in range(740,750): #change according to the page
    print("Processing page", page_num)
    # page_num += 1
    url = f"https://themalaysianreserve.com/category/business/page/{page_num}"
    driver.get(url)
    time.sleep(5) # Adjust sleep time as needed

    html = driver.page_source
    soup = BeautifulSoup(html, "html.parser")

    # Extract links from the current page
    category_links = [a['href'] for a in soup.select("div.col-xl-4.col-lg-6.col-md-12.col-sm-6 a")]

    # Filter links for the year
    year_links = [link for link in category_links if f'/{2019}/' in link]

    # Check if any links were found after filtering
    if not year_links:
        print(f"No relevant links found on page {page_num}. Skipping scraping.")
        continue # Skip to the next page

    # Scrape content from each link
    for link in year_links:
        scraped_content = scrape_content(link)
        if scraped_content:
            content.append(scraped_content)
            all_links.append(link)
```

Figure 4. 5: Code to scrape content in Malaysia Reserve.

This snippet performs web scraping on “The Malaysian Reserve” website to collect business news articles from the year 2019. The code uses BeautifulSoup to extract specific parts of a webpage, it selects the first <h2> element (assumed to contain the title) and gathers all <div> elements with the class "dable-content-wrapper" (assumed to hold the article's content). It loops through a specified range of pages (from 740 to 749), constructs each page's URL, and uses a web driver to load and render the page content. After a short delay to ensure the page is fully loaded, it parses the HTML using BeautifulSoup and

extracts article links from specific <div> elements. It then filters these links to include only those containing "/ {2019} /", targeting articles published in that year. If no such links are found on a given page, the script logs a message and moves on to the next page. For each valid 2019 article link, it calls a `scrape_content` function to extract the article's content, which is then appended to the content list along with the article's URL saved in `all_links`, facilitating the collection of year-specific business news data. Refer to Appendix A for the full source code.

4. New Straits Times (NST)

```
# to get content and title
content = []
for page in pages:
    link = 'https://www.nst.com.my/business' + page
    driver.get(link)
    time.sleep(5)
    html = driver.page_source
    soup = BeautifulSoup(html, "html.parser")
    all_divs = soup.select("div.dable-content-wrapper")
    page_title = soup.select('h1.page-title.mb-2')
    if page_title and all_divs: # Ensure both title and content are present before appending
        content.append(page_title[0].text + "\n" + all_divs[0].text)
        print(page_title[0].text + "\n" + all_divs[0].text)
    else:
        print("Skipping:", link)
```

Figure 4. 6: Code to scrape content in New Straits Times.

This code shows to scrape the title and main content of business articles from the “New Straits Times” or (NST) website. It loops through a list of page paths stored in `pages`, appending each to the base URL (`https://www.nst.com.my/business`) to create the full article link. The script uses a web driver to load each page and waits for 5 seconds to ensure the content is fully loaded. It then parses the page using `BeautifulSoup`, extracting the article title from the <h1> tag with class `page-title mb-2` and the article content from a <div> with class `dable-content-wrapper`. If both the title and content are successfully found, they are combined with a newline separator and appended to the content list;

otherwise, the script prints a message and skips that page. Refer to Appendix A for the full source code.

```
PS C:\Users\user\Desktop\Test web scraping> &
C:/Users/user/AppData/Local/Programs/Python/Python312/python.exe
"c:/Users/user/Desktop/Test web scraping/themalaysianreserve"

DevTools listening on ws://127.0.0.1:50503/devtools/browser/63b97dcc-7332-4695-a0af-
da7bf214594c
Processing page 670
[21944:22280:0328/000854.900:ERROR:ssl_client_socket_impl.cc(970)] handshake
failed; returned -1, SSL error code 1, net_error -101
Processing page 671
Processing page 672
Processing page 673
Processing page 674
Processing page 675
Processing page 676
Processing page 677
Processing page 678
Processing page 679
Scraping complete.
```

Table 4. 2: Output of the scraping process.

Content,URL
"Pos Malaysia offers luggage shipping service to umrah pilgrims by BERNAMA / pic by BERNAMA Pos Malaysia Bhd is offering a luggage shipping service to Malaysians performing the umrah pilgrimage from Jan 1, 2020 to facilitate their return to Malaysia. Group chief executive officer Syed Md Najid Syed Md Noor said the service provides an easy, worry-free option for Malaysians who have performed their umrah. “With the extension of our luggage services to Madinah, Malaysian pilgrims can use this service to facilitate the shipping of their luggage home. “It could help reduce the burden on them, enabling them to avoid costly excess luggage fees,” he said in a statement today. Pos Malaysia charges SR/RM20 for each kilogramme of luggage for shipping to Peninsular Malaysia and SR/RM25 for each kilogramme to Sabah and Sarawak. He said pilgrims may pay cash in Riyal Saudi for shipping from Makkah and Madinah, or pay when their luggage is received in Malaysia.

A packaging service is also provided, with charges ranging from SR8 for a small box to SR10 for a medium-sized box and SR12 for a large box.

Pos Malaysia will pick up the pilgrims' luggage by appointment until a permanent counter is set up, the statement added.",<https://themalaysianreserve.com/2019/12/31/pos-malaysia-offers-luggage-shipping-service-to-umrah-pilgrims/>

Table 4. 3: The process of scraping content and URLs from multiple pages of a website.

The table 4.2 and 4.3 show the output of the process of web scraping content and URLs from multiple pages of The Malaysian Reserve website. The script processes pages sequentially, starting from page 670, and encounters an SSL handshake error at the beginning but continues scraping without interruption. After reaching page 679, the process completes successfully. The extracted data includes article content and URLs, such as a news piece about Pos Malaysia offering a luggage shipping service for umrah pilgrims, detailing the service's pricing, payment methods, and benefits. The data is structured in a CSV format, with each article's content paired with its respective URL.

Step 3: Save data in CSV file

1. Business Today

```
df.to_csv('businesstoday_2019(1).csv', index=False)
```

Figure 4. 7: Code to save data for Business today.

This code is used to save the returned DataFrame to a CSV file named businesstoday_2019(1).csv and print a success message; if an error occurs during scraping, it prints an error message, and regardless of the outcome, the WebDriver is closed in the final code.

2. Daily Express

```
df.to_csv('dailyexpress_2024(test).csv', index=False)
```

Figure 4. 8: Code to save data for Daily Express.

In this code, the scraping process is executed the script in the main program it tries to scrape data using the same function as business today code and it saves the results to dailyexpress_2019(1).csv.

3. Malaysia reserve

```
df.to_csv('malaysiareserve_2019(14).csv', index=False)
```

Figure 4. 9: Code to save data for Malaysia reserve.

This code wraps the scraping function and saves the scraped data to malaysiareserve_2019(1).csv for the Malaysian reserve.

4. New Straits Times (NST)

```
df.to_csv('nst(49).csv')
```

Figure 4. 10: Code to save data for NST.

This code shows save and then writes the DataFrame to a CSV file named nst(49).csv.

Step 4: Dataset file

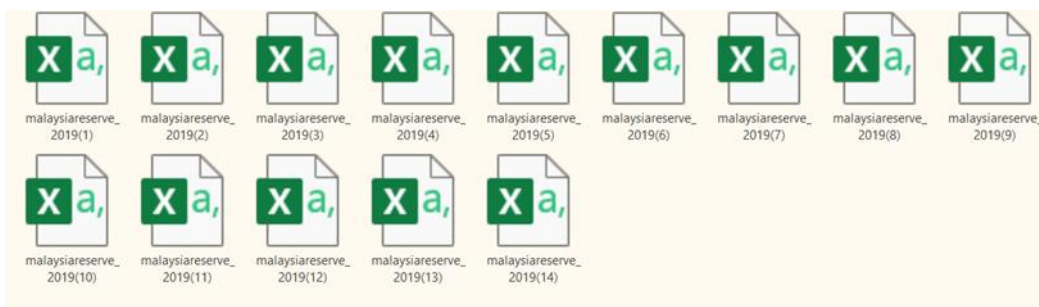


Figure 4. 11: Extracted data.

Figure 4.11 shows the extracted data was saved into a structured CSV file for further analysis.

Content	URL
by BERNAMA / pic by BERNAMA Pos Malaysia Bhd is offering a luggage shipping service to Malaysians performing the umrah pilgrimage from Jan 1, 2020 to facilitate their return to Malaysia. Group chief executive officer Syed Md Najid Syed Md Noor said the service provides an easy, worry-free option for Malaysians who have performed their umrah.	
With the extension of our luggage services to Madinah, Malaysian pilgrims can use this service to facilitate the shipping of their luggage home. It could help reduce the burden on them, enabling them to avoid costly excess luggage fees," he said in a statement today. Pos Malaysia charges SR/RM20 for each kilogramme of luggage for shipping to Peninsular Malaysia and SR/RM25 for each kilogramme to Sabah and Sarawak. He said pilgrims may pay cash in Riyal Saudi for shipping from Makkah and Madinah, or pay when their luggage is received in Malaysia. A packaging service is also provided, with charges ranging from SR8 for a small box to SR10 for a medium-sized box and SR12 for a large box. Pos Malaysia will pick up the pilgrims' luggage by appointment until a permanent counter is set up, the statement added.	https://themalaysianreserve.com/2019/12/31/pos-malaysia-offers-luggage-shipping-service
U Mobile partners Singapore's Starhub for maiden 5G trials	https://themalaysianreserve.com/2019/12/31/u-mobile-partners-singapores-starhub-for-r
East Malaysia-based planters benefit from CPO rally, lower production	https://themalaysianreserve.com/2019/12/31/east-malaysia-based-planters-benefit-from
TNB's Jimah East Power plant now fully operational	https://themalaysianreserve.com/2019/12/31/tnbs-jimah-east-power-plant-now-fully-ope
Rapid KL extends selected services on New Year's Eve	https://themalaysianreserve.com/2019/12/31/rapid-kl-extends-selected-services-on-new
Proton X70 delivered to MoF	https://themalaysianreserve.com/2019/12/31/proton-x70-delivered-to-mof/

Figure 4. 12: Sample of the content scraped.

Figure 4.12 shows the sample of the dataset. Content scraped from websites, along with their corresponding URLs, is stored in a CSV dataset.

4.3 Data preprocessing

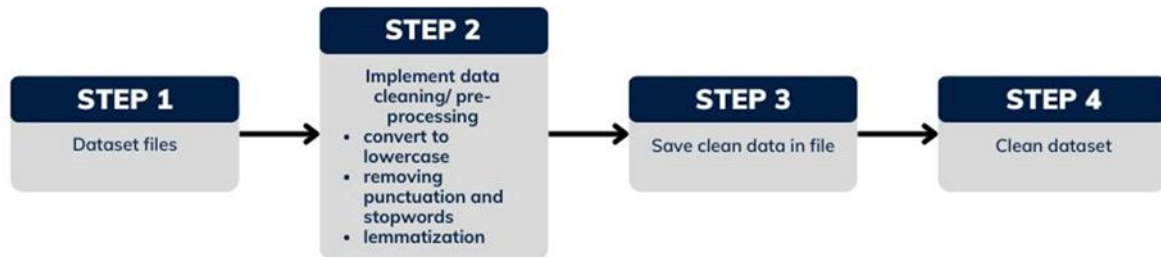


Figure 4. 13: The step for data cleaning or pre-processing.

In Phase 2, raw text data is prepared for analysis by applying preprocessing techniques. This step aims to improve the quality of the data and ensure that it is suitable for input into topic modelling algorithms. The process begins with text cleaning, which removes irrelevant components such as HTML tags, special characters, numbers, and URLs. Lowercasing is applied to convert all text to lowercase, ensuring uniformity and

eliminating case sensitivity. Common stopwords like “and” “the,” and “of,” which do not contribute to identifying topics, are then removed, while lemmatization is applied to reduce words to their root forms. For example, changing “running” to “run” to standardize the vocabulary and enhance the effectiveness of topic extraction.

4.3.1 Data Cleaning

No.	Libraries	Description
1.	Natural Language Toolkit (NLTK)	Offers tools for text cleaning, tokenization, and lemmatization to prepare data for NLP tasks.

Table 4. 4: Data pre-processing tools and libraries.

The table 4.4 show the Natural Language Toolkit (NLTK) is one of the widely used Python libraries for this purpose. It provides a comprehensive suite of tools for various pre-processing tasks.

Step 1: Dataset files

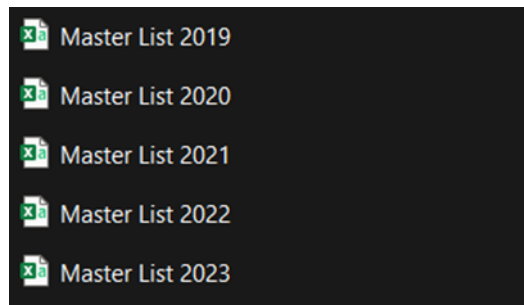


Figure 4. 14: Master List dataset.

Figure 4.14 shows the master list dataset that has been collected and saved in the CVS file format from the previous data collection phase.

Step 2: Implement data cleaning

```

1  import os
2  import pandas as pd
3  import nltk
4  from nltk.corpus import stopwords
5  from nltk.stem import WordNetLemmatizer
6  import re
7  from collections import Counter
8  import time

10 def preprocess_text(text, lemmatizer):
11     text = text.lower()
12     text = re.sub(r'^\x00-\x7F+', ' ', text) # Remove non-ASCII characters
13     text = re.sub(r'^\w\s]', '', text) # Remove punctuation
14     text = re.sub(r'\d+', '', text) # Remove digits
15     tokens = text.split()
16     stop_words = set(stopwords.words('english'))
17     tokens = [word for word in tokens if word not in stop_words]
18     tokens = [lemmatizer.lemmatize(word) for word in tokens]
19     return tokens

```

Figure 4. 15: Code for preprocessing.

This code snippet sets up the environment for data cleaning by importing necessary libraries such as pandas for data handling and nltk for natural language processing and defines a `preprocess_text` function that cleans text data by converting it to lowercase, removing non-ASCII characters, punctuation, and digits, tokenizing the text, filtering out English stopwords, and finally applying lemmatization to reduce words to their base forms.

Step 3: Save the clean data.

```

training_processed_file_path = 'Processed_Training_Set_2022.csv'
testing_processed_file_path = 'Processed_Testing_Set_2022.csv'

training_data.to_csv(training_processed_file_path, index=False)
testing_data.to_csv(testing_processed_file_path, index=False)

```

Figure 4. 16: Code to save data file.

Figure 4.16 shows the code takes the preprocessed text from both training and testing datasets and saves the updated DataFrames into separate CSV files.

Step 4: Clean dataset

article_content	processed_text_str
CIMB Bank Berhad and CIMB Islamic Bank Berhad have announced their refreshed SME business proposition to provide customers with a complete suite of solutions that go beyond banking. CIMB Bank and CIMB Islamic are also on track to disburse RM15 billion to SMEs in 2019 – 2020, having provided RM3.9 billion to Malaysian SMEs for the six-month period ending June 2019. The refreshed SME business proposition is anchored on three pillars i.e., SME Transact (e.g., cash management services; reward opportunities of up to two percent per annum with CIMB Islamic business current account; and fee waivers on online business banking); SME Financing (e.g., no-collateral and business property financing); and SME Partners (various services from logistics, workspace and connectivity, to digital, e-commerce, as well as accounting and HR solutions in the pipeline, offered by CIMB partners). Tengku Dato Sri Zafrul Aziz, Group CEO, CIMB Group said, “With SMEs currently contributing more than one third of Malaysia’s GDP, banks have a responsibility to grow this sector, and to support the National Entrepreneurship Policy’s objective to have SMEs contribute 50 percent to the GDP by 2030. We have deepened our understanding of our customers, their pain points and how best to transform their journey with CIMB. Hence, the three pillars in our new SME proposition represent all the support that our SME customers need, not just in financing, but also in terms of business solutions that support their growth. Beyond financing, our support comprises knowledge-sharing and strategic partnerships, so that SMEs, start-ups and entrepreneurs can develop and scale up systematically. In facilitating SMEs to scale up regionally, CIMB’s partnership with Axiata Digital Capital provide 700,000 SMEs in	cimb bank berhad cimb islamic bank berhad announced refreshed sme business proposition provide customer complete suite solution go beyond banking cimb bank cimb islamic track smes provided malaysian smes sixmonth period ending june refreshed sme business proposition three pillar sme cash management service reward opportunity two percent per annum cimb islamic business current account fee online business banking sme financing business property financing sme partner various service logistics workspace connectivity digital ecommerce well accounting solution pipeline offered cimb partner tengku dato sri aziz group ceo cimb group smes currently contributing one third gdp bank responsibility grow sector support national entrepreneurship policy objective smes contribute percent gdp understanding customer pain point best transform journey cimb hence three pillar sme proposition represent support sme customer need financing business solution support growth beyond financing support comprises strategic partnership smes startup entrepreneur develop scale facilitating smes scale regionally cimb partnership axiata digital capital provide smes indonesia access financing solution cimb collaboration pte ltd facilitate smes capital raising cimb world demand halal product service far supply cimb strong global network halal corridor provide smes perfect platform access halal business need collaboration credit guarantee corporation cimb islamic allocated unsecured shariah

Malaysia and Indonesia with access to financing solutions. CIMB is also in collaboration with CapBridge Pte Ltd to facilitate SMEs' capital raising by CIMB. Further, with the world's demand for halal products and services far outstripping supply, CIMB's strong global network and well-established ASEAN-China Halal Corridor provide SMEs with the perfect platform to access the halal market. For this and other business needs, in collaboration with Credit Guarantee Corporation (CGC), CIMB Islamic has allocated RM2 billion unsecured Shariah financing; as well as micro financing to small business owners via the imSME, Malaysia's first SME financing/loan referral platform, managed by CGC. Customers now have access to micro financing via imSME at 10 selected branches from September onwards with the aim of making this available nationwide by year-end. "As an ASEAN multinational with a strong regional footprint, we are well-positioned to support SMEs that wish to take advantage of this market with its 648-million population," added Tengku Zafrul. In support of Malaysia's national aspiration to nurture entrepreneurs, CIMB has committed to hold at least 1,000 talks in 2019, with 550 sessions held year-to-date, and with around 60 of these talks held in September alone, in conjunction with Bank Negara Malaysia's SME Month. CIMB's complimentary talks feature key tips and advice on running a successful SME, covering topics such as credit score, forex hedging, as well as growth opportunities afforded by digital, big data, e-commerce and communications. To date, over 4,000 SME business owners have benefitted from such talks. CIMB's refreshed SME proposition forms part of CIMB's current mid-term growth plan, Forward23. For more details, customers can visit https://cimbbank.com.my/smebanking	financing well financing small business owner via first sme platform managed customer access financing via selected branch september onwards aim making available nationwide yearend asean multinational strong regional footprint support smes wish take advantage population added tengku support national aspiration entrepreneur cimb committed hold least talk session held yeartodate around talk held september alone conjunction bank negara sme month cimb talk feature key advice running successful sme covering topic credit score forex hedging well growth opportunity digital big data ecommerce communication date sme business owner benefitted talk cimb refreshed sme proposition form part cimb current growth plan forward detail customer visit
--	--

Table 4. 5: The sample of raw text and the cleaned text.

j6	thought leadership forum conjunction technology conference hosted engineering sdn call malaysian business quickly embrace digital transformation rate digital disruption accelerating comprising industry expert attended																
1	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
2	title	article_content	processed_text_str														
3		CIMB Bank Berhad and CIMB Islamic Bank Berhad, have announced their refreshed SME business proposition to provide customers with a complete suite of solutions that go beyond banking. CIMB Bank and CIMB Islamic are also on track to disburse RM15 billion to SMEs in 2019, having provided RM3.9 billion to Malaysian SMEs for the six-month period ending June 2019. The refreshed SME business proposition is anchored on three pillars i.e., SME Transact (e.g., cash management services; reward opportunities of up to two percent per annum with CIMB Islamic business current account; and fee waivers on online business banking); SME Financing (e.g., no-collateral and business property financing); and SME Partners (e.g., various services from logistics, workspace and connectivity, to digital, e-commerce, as well as accounting and HR solutions in the pipeline, offered by CIMB partners). Tengku Dato, Group CEO, CIMB Group said, "Weh SMEs currently contributing more than one third of Malaysia's GDP, banks	cimb bank berhad cimb islamic bank berhad announced refreshed sme business proposition provide customer complete suite solution go beyond banking cimb bank														
4		On Track T have a responsibility to grow this sector, and to support the National	solution berhad group belief growth potential sap division solution sdn growing professional workforce investing research development managing director dato bin abd														
5		TFP Group. The TFP Solutions Berhad Group believes in the growth potential of its SAP division, Boost Chai Boost, the homegrown leading e-wallet brings Malaysians together in a campaign	boost homegrown leading ewallet brings malaysian together campaign day campaign focus five core value healthier progressive key building block better nation boost														
6		WeWork C WeWork announced plans last Friday to reduce the power of the co-founder and Business: During the thought leadership forum in conjunction with the Altair Technology	weework announced plan friday reduce power cofounder chief executive giant ready wall street share offering security filing parent ceo adam neumann would give profi														
7		Cycle & Ca For the second time, Cycle & Carriage Bintang Berhad (Cycle & Carriage) achieved a	thought leadership forum conjunction technology conference hosted engineering sdn call malaysian business quickly embrace digital transformation rate digital disru														
8		KPMG Wei KPMG in Malaysia has developed a Digital Board Effectiveness Evaluation	second time cycle bintang berhad cycle achieved perfect score daimler ag stringent centre certification batu body paint by facility johor bukit facility percent percent r														
9		Maybank 2 Despite a 1.2 percent rise in net operating income and a 19.6 percent decline in net	developed digital board evaluation bee tool featuring automated reporting check customised improve board evaluation process elevate good corporate governance bo														
10			despite percent rise net operating income percent decline net impairment loss maybank profit tax second quarter ended june qly rose percent net profit quarter howev														

Figure 4. 17: The sample in the CSV file.

Table 4.5 and figure 4.17 shows the sample output shows a cleaned dataset of article_content, which contains the original full raw text of an article, and processed_text_str, which presents the corresponding cleaned version of the text. The cleaned text has had unnecessary elements like punctuation, digits, and stopwords removed, providing a streamlined version suitable for further analysis or model training. Refer to Appendix A for the full source code.

4.4 Topic modelling

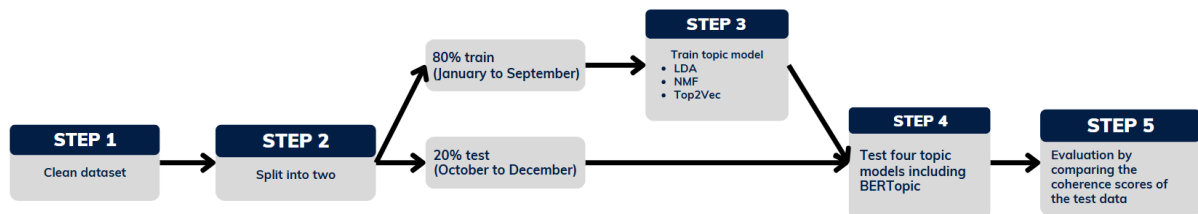


Figure 4. 18: The step for train and test topic modelling.

Phase 3 focuses on the topic-modelling techniques. First, using the cleaned dataset, it is divided into two sets: a training set, 80% of the articles (from January to September), and a testing set, 20% of the articles (from October to December). This split ensures that the models are trained on one set of data and evaluated on a separate test set. These techniques aim to uncover hidden patterns and topics within the text data. The techniques include Latent Dirichlet Allocation (LDA), a probabilistic model that identifies topics as

distributions over words. Non-Negative Matrix Factorization (NMF), a matrix decomposition technique that discovers latent topics by factorizing the term-document matrix. Top2Vec, an approach that uses semantic embeddings to identify topics, offers an advantage in capturing the relationships between words, and BERTopic, a pre-trained transformer-based model that generates interpretable topics using dynamic topic reduction and clustering. During the training stage, three topic modelling techniques, LDA, NMF, and Top2Vec, were applied to 80 % of the data, as BERTopic is a pre-trained model and will not be included in the training phase. In the testing stage, all four techniques, including BERTopic, were used on 20 % of the data. Each model is configured with hyperparameters, such as the number of topics and the number of words per topic, to ensure optimal performance.

4.4.1 Implement topic modelling.

No	Libraries	Description
1.	Pandas	Used for performing DataFrame operations such as analysing data in tables and organizing or filtering data.
2.	NumPy	A library for numerical operations, arrays, and mathematical computations.
3.	Matplotlib	A data visualization library for creating charts and plots.
4.	Scikit-learn	A machine learning library used for data mining, training, and deploying models.
5.	Gensim	An open-source Python library designed for efficient topic modelling and vector space modelling.
6.	WordCloud	A visualization tool for creating tag clouds or weighted lists from text data.
7.	Pickle	A library for saving and loading Python objects like data or models for reuse.
8.	UMAP (umap-learn)	A dimensionality reduction technique for visualizing or simplifying large datasets.
9.	HDBSCAN	A library for unsupervised learning to find clusters or dense regions in datasets.

10	Seaborn	An extension of Matplotlib that provides a high-level interface for statistical and graphical analysis.
11.	AdjustText	A library for adjusting text positions to make labels on plots easier to read.
12.	CountVectorizer	Converts text into numerical vectors by tokenizing and capturing word frequencies.

Table 4. 6: Topic modelling libraries.

Table 4.6 shows the list of Python libraries utilized throughout the topic modelling process, highlighting each library’s role in data preprocessing, modelling, analysis, and visualization. These libraries collectively support tasks such as handling and transforming textual data, training topic models, evaluating their quality, and visualizing the results for interpretation. Refer to Appendix A for the full source code.

Step 1: Clean dataset

In this step the process has been done in previous preprocessing phase.

Step 2: Split data into two

```

45 # Separate data into training (January to September) and testing (October to December) sets
46 training_data = data[(data['date'].dt.month >= 1) & (data['date'].dt.month <= 9)]
47 testing_data = data[(data['date'].dt.month >= 10) & (data['date'].dt.month <= 12)]

```

Figure 4. 19: Code to split data.

Figure 4.19 shows code uses pandas filtering to split the dataset based on the month extracted from a datetime column. It assigns data from January to September (months 1–9) to a training set and data from October to December (months 10–12) to a testing set, enabling a temporal division of the dataset.

Year	Number of article (training)	Number of article (testing)
2019	4654	1325
2020	4995	1997
2021	6444	2052

2022	6137	1687
2023	5672	704

Table 4. 7: The number of articles in training and testing.

This table 4.7 shows the distribution of articles designated for training and testing over five years. In 2019, 4,654 articles were used for training and 1,325 for testing. The following year, 2020, saw an increase to 4,995 training articles and a notable rise in testing articles to 1,997. In 2021, the training dataset peaked at 6,444 articles, while testing slightly increased to 2,052. In 2022, the numbers dipped slightly to 6,137 training and 1,687 testing articles, and in 2023, the training set decreased further to 5,672 with testing articles dropping to 704. This distribution illustrates how the volume of data for training and testing has varied across the years.

Step 3: Train topic modelling

1. LDA

```

1  import os
2  import pandas as pd
3  import numpy as np
4  import matplotlib.pyplot as plt
5  from sklearn.feature_extraction.text import CountVectorizer
6  from sklearn.decomposition import LatentDirichletAllocation
7  from gensim.models.coherencemodel import CoherenceModel
8  from gensim.corpora.dictionary import Dictionary
9  from wordcloud import WordCloud
10 import time
11 import pickle

150 file_path = 'Processed_Training_Data_2023.csv' # Update with your processed training data file path

22  lda_model = LatentDirichletAllocation(n_components=n_topics, random_state=42)
23  lda_topic_matrix = lda_model.fit_transform(count)

```

Figure 4. 20: Code to train LDA topic modelling.

Figure 4.20 shows the LDA code first imports libraries needed for text processing, vectorization, and topic modelling. The LatentDirichletAllocation class is used to create an instance of the LDA model, where `n_components=optimal_topics` specifies the number of topics to extract and `random_state=42` ensures reproducibility of the results. The model

is then trained using the `fit_transform()` method on `count_train`, which is the document-term matrix generated from the training dataset. This method returns `lda_topic_matrix_train`, a matrix that represents the topic distribution for each document in the training set, showing how much each topic contributes to each document.

2. NMF

```
1 import os
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import re
6 from sklearn.feature_extraction.text import TfidfVectorizer
7 from sklearn.decomposition import NMF
8 from gensim.models.coherencemodel import CoherenceModel
9 from gensim.corpora.dictionary import Dictionary
10 from wordcloud import WordCloud
11 import time
12 import pickle

173 file_path = 'Processed_Training_Data_2023.csv' # Update with your processed training data file path

19 nmf_model = NMF(n_components=n_topics, random_state=42)
20 nmf_topic_matrix = nmf_model.fit_transform(tfidf)
```

Figure 4. 21: Code to train NMF topic modelling.

Figure 4.21 shows the NMF code imports libraries for text vectorization and topic modelling, then loads the processed training dataset. It initializes a Non-negative Matrix Factorization model with the specified number of topics and fits it to a TF-IDF matrix of the training texts, a topic distribution matrix. This approach factorizes the document-term matrix while maintaining non-negative values, which often results in more interpretable, parts-based topic representations.

3. Top2vec

```

1  import os
2  import pandas as pd
3  import time
4  from top2vec import Top2Vec
5  import nltk
6  import gensim
7  from gensim.models.coherencemodel import CoherenceModel
8  import matplotlib.pyplot as plt
9  import umap
10 import hdbscan

13  # Load the data
14  df = pd.read_csv('Master List 2023 Jan to Sep.csv')

41  # Train the doc2vec model
42  topic_model = Top2Vec(
43      documents=docs_filtered,
44      embedding_model='doc2vec', # Generate document embeddings using doc2vec
45      speed="deep-learn",
46      tokenizer=tok,
47      ngram_vocab=True,
48      ngram_vocab_args={"connector_words": "phrases.ENGLISH_CONNECTOR_WORDS"}
49  )

```

Figure 4. 22: Code to train Top2vec topic modelling.

Figure 4.22 shows the code begins by importing several necessary libraries, including those for file operations (os), data manipulation (pandas), time tracking, topic modelling (Top2Vec), natural language processing (nltk and 70enism), visualization (matplotlib), and dimensionality reduction/clustering (umap and hdbscan). It then loads a CSV file named 'Master_List_2023_Jan_to_Sep.csv' into a DataFrame using pandas. Following the data loading, the code proceeds to train a Top2Vec model using the doc2vec embedding model this is done by passing a filtered list of documents (docs_filtered) along with a custom tokenizer function (tok). The model is configured for deep learning speed, and it is set to use an n-gram vocabulary with additional arguments specifying the use of English connector words for phrases, ensuring that the model captures multi-word expressions effectively.

Step 4: Test topic modelling

1. LDA

```

113 # Function to evaluate LDA model on test data
114 def evaluate_lda_model(test_file_path):
115     if not os.path.exists(test_file_path):
116         raise FileNotFoundError(f"Test data file {test_file_path} not found.")
117
118     # Load test data
119     test_data = pd.read_csv(test_file_path)
120     print(test_data.head())
121
122     # Handle missing or non-string values in the 'processed_text_str' column
123     test_data['processed_text_str'] = test_data['processed_text_str'].fillna('')
124
125     # Extract processed text from the dataframe
126     test_texts = [str(text).split() for text in test_data['processed_text_str']]
127
128     # Load the trained LDA model and CountVectorizer
129     lda_model = load_lda_model('lda_model_2019.pkl')
130     count_vectorizer = load_count_vectorizer('count_vectorizer_2019.pkl')

```

Figure 4. 23: Code to test LDA topic modelling.

Figure 4.23 shows the `evaluate_lda_model` function first checks if the specified test file exists if not, it raises an error. It then loads the test data from a CSV file and prints the first few rows. The function handles missing values in the 'processed_text_str' column by filling them with an empty string and splits each processed text into tokens. It loads a pre-trained LDA model and its associated CountVectorizer from pickle files.

2. NMF

```

97 def test_nmf_model(test_texts, model_file_path, tfidf_vectorizer_file_path, output_dir='test_output'):
98     """
99     Apply the pre-trained NMF model to the new dataset and extract new topics.
100     """
101     # Load the pre-trained model and TF-IDF vectorizer
102     with open(model_file_path, 'rb') as file:
103         nmf_model = pickle.load(file)
104
105     with open(tfidf_vectorizer_file_path, 'rb') as file:
106         tfidf_vectorizer = pickle.load(file)
107
108     # Transform the new dataset using the pre-trained TF-IDF vectorizer
109     tfidf_test = tfidf_vectorizer.transform(test_texts)
110
111     # Apply the NMF model to the new data to extract topics
112     nmf_topic_matrix_test = nmf_model.transform(tfidf_test) # The document-topic distribution for the test data
113     # print(nmf_topic_matrix_test)

```

Figure 4. 24: Code to test NMF topic modelling.

Figure 4.24 shows the `evaluate_nmf_model_on_test` function, testing texts are first vectorized by joining tokens into a single string for each document and transforming them

using a pre-trained TF-IDF vectorizer. The pre-trained NMF model is then applied to this TF-IDF representation to generate a topic distribution (topic matrix) for the test data.

3. Top2Vec

```
42     # Tokenize the test documents
43     test_docs_tokenized = [tok(doc) for doc in test_docs]
44
45     # Infer topics for each document in the testing data
46     inferred_topics = []
47     inferred_scores = []
48     for doc in test_docs:
49         # Infer the most similar topic for each document
50         results = topic_model.query_topics(doc, num_topics=1) # Query for the top topic
51         topic_num = results[0][0] # Extract topic number from results
52         similarity_score = results[1][0] # Extract similarity score from results
53         inferred_topics.append(topic_num)
54         inferred_scores.append(similarity_score)
```

Figure 4. 25: Code to test Top2vec topic modelling.

Figure 4.25 shows over each document in the test set (stored in `test_docs`) and uses the `query_topics` method of a pre-trained Top2Vec model to infer topics. For each document, it retrieves the most similar topic along with a similarity score. These results are then appended to two lists `inferred_topics` (content the topic numbers) and `inferred_scores` (content the similarity scores) which can be later used for analysis or further evaluation.

4. BERTopic

```

1  import pandas as pd
2  import numpy as np
3  import time
4  from sentence_transformers import SentenceTransformer
5  from umap import UMAP
6  from hdbscan import HDBSCAN
7  from bertopic import BERTopic
8  from gensim.models.coherencemodel import CoherenceModel
9  from gensim.corpora.dictionary import Dictionary
10 from sklearn.feature_extraction.text import CountVectorizer
11 from wordcloud import WordCloud
12 import seaborn as sns
13 import matplotlib.pyplot as plt

19 # Load CSV file
20 file_path = "Processed_Testing_Data_2020.csv"
21 df = pd.read_csv(file_path)
22 docs = df['article_content'].astype(str).tolist()

37 # Train BERTopic model
38 topic_model_start = time.time()
39 topic_model = BERTopic(
40     embedding_model=embedding_model,
41     umap_model=umap_model,
42     hdbscan_model=hdbscan_model,
43     vectorizer_model=CountVectorizer(stop_words="english")
44 ).fit(docs, embeddings=embeddings)
45 print(f"BERTopic model trained in {time.time() - topic_model_start:.2f} seconds")

```

Figure 4. 26: Code to test BERTopic topic modelling.

Figure 4.26 shows the BERTopic testing code starts by loading test data from a CSV file, extracting the document content into a list of strings. It then trains a BERTopic model using a SentenceTransformer for generating embeddings, along with UMAP for dimensionality reduction and HDBSCAN for clustering. A CountVectorizer with English stopwords is also specified to convert the text into a document-term matrix. The model is fitted on the documents and their corresponding embeddings, and the training time is printed. This process results in a fully trained BERTopic model that has discovered topics from the test data, and it can be used for further evaluation or visualization.

Step 5: Evaluation of topic modelling.

The evaluation of topic modelling involves comparing models based on the coherence score. A higher coherence score indicates that a model's topics are more semantically cohesive and, consequently, more interpretable to human analysts.

1. LDA

```

31 # Function to calculate and plot coherence scores for test data
32 def calculate_coherence_for_test(test_texts, lda_model, count_vectorizer, output_dir):
33     # Transform the test data using the trained CountVectorizer
34     count_test = count_vectorizer.transform([' '.join(text) for text in test_texts])
35
36     # Generate topic distribution for test data
37     lda_topic_matrix_test = lda_model.transform(count_test)
38
39     # Extract topics
40     topics = []
41     feature_names = count_vectorizer.get_feature_names_out()
42     for topic_idx, topic in enumerate(lda_model.components_):
43         topics.append([feature_names[i] for i in topic.argsort()[::-10:-1]])
44
45     # Calculate coherence score for the test set
46     dictionary = Dictionary(test_texts)
47     corpus = [dictionary.doc2bow(text) for text in test_texts]
48     cm = CoherenceModel(topics=topics, texts=test_texts, dictionary=dictionary, coherence='c_v')
49     coherence_score = cm.get_coherence()
50
51     print(f"Coherence score for test set: {coherence_score:.4f}")
52
53     # Save the test topic matrix
54     np.save(os.path.join(output_dir, 'lda_topic_matrix_test_2019.npy'), lda_topic_matrix_test)
55     print("LDA topic matrix for test data saved as lda_topic_matrix_test_2019.npy")
56
57     return coherence_score, lda_topic_matrix_test

```

Figure 4. 27: Code to calculate coherence score LDA topic modelling.

Figure 4.27 shows the function `calculate_coherence_for_test()` is designed to calculate the coherence score. It begins by transforming the `test_texts` into a document-term matrix using the already fitted `CountVectorizer`. The function then applies the trained `lda_model` to this matrix to generate a topic distribution for each document in the test set. It extracts the top 10 words from each topic based on their importance and uses these to form a list of topics. A Gensim Dictionary and corresponding corpus are created from the tokenized test texts, which are then used to compute the coherence score using the `CoherenceModel` with the 'c_v' metric a common measure of topic interpretability. The coherence score gives an indication of how semantically consistent the topics are. The function also saves the resulting topic matrix for the test data as a .npy file for later use. Finally, it returns both the coherence score and the topic matrix

2. NMF

```

14 def calculate_coherence_scores(texts, tfidf, tfidf_vectorizer, max_topics, step=1):
15     coherence_scores = []
16     for n_topics in range(2, max_topics + 1, step):
17         start_time = time.time()
18
19         nmf_model = NMF(n_components=n_topics, random_state=42)
20         nmf_topic_matrix = nmf_model.fit_transform(tfidf)
21
22         topics = []
23         feature_names = tfidf_vectorizer.get_feature_names_out()
24         for topic_idx, topic in enumerate(nmf_model.components_):
25             topics.append([feature_names[i] for i in topic.argsort()[::-10:-1]])
26
27         dictionary = Dictionary(texts)
28         corpus = [dictionary.doc2bow(text) for text in texts]
29         cm = CoherenceModel(topics=topics, texts=texts, dictionary=dictionary, coherence='c_v')
30         coherence_score = cm.get_coherence()
31         coherence_scores.append((n_topics, coherence_score))
32
33         end_time = time.time()
34         print(f'Calculating coherence for {n_topics} topics took {end_time - start_time:.2f} seconds.')
35         print(f'Coherence score for {n_topics} topics: {coherence_score:.4f}')
36
37     return coherence_scores

```

Figure 4. 28: Code to calculate coherence score NMF topic modelling.

Figure 4.28 shows the function `calculate_coherence_scores` follows a similar process to LDA but evaluates NMF models. It iterates over topic counts, records execution time, initializes an NMF model with the specified number of topics, and fits it to a TF-IDF matrix. The top 10 words per topic are extracted from the model's components and stored. A Gensim Dictionary is then created from tokenized texts, and a bag-of-words corpus is generated. A `CoherenceModel` is instantiated with the extracted topics, dictionary, and 'c_v' coherence metric to evaluate topic quality. The function logs and returns the coherence scores for each topic count, helping assess how well topics are formed at different levels. Unlike LDA, NMF is a deterministic model, making its extracted topics more stable.

3. Top2vec

```

65     # Calculate coherence score for the testing data
66     topic_words, _, _ = topic_model.get_topics()
67     coherence_model_test = CoherenceModel(
68         topics=topic_words,
69         texts=test_docs_tokenized,
70         dictionary=gensim.corpora.Dictionary(test_docs_tokenized),
71         coherence='c_v'
72     )
73     coherence_score_test = coherence_model_test.get_coherence()
74     print("Coherence score for testing data:", coherence_score_test)

```

Figure 4. 29: Code to calculate coherence score Top2vec topic modelling.

Figure 4.29 shows the code to calculate the coherence score. It retrieves the top words for each topic from the trained topic_model using the get_topics() method. Then, it creates a Gensim CoherenceModel by providing the extracted topic words (topics), the tokenized test documents (test_docs_tokenized), and a dictionary built from these tokenized documents. The coherence='c_v' parameter specifies the use of the 'c_v' coherence measure, which evaluates how consistently related the words within each topic. The coherence score is computed using get_coherence(), and the result is printed, indicating the overall interpretability and quality of the topics generated by the model on the test data.

4. BERTopic

```

90     # Get topic info and coherence score
91     coherence_start = time.time()
92     tokenized_docs = [doc.split() for doc in docs]
93     dictionary = Dictionary(tokenized_docs)
94     corpus = [dictionary.doc2bow(doc) for doc in tokenized_docs]
95
96     topic_words = [[word for word, _ in topic_model.get_topic(topic_num)[:20]] for topic_num in topics.keys()] # Adjust coherence for top 20 words
97     coherence_model = CoherenceModel(topics=topic_words, texts=tokenized_docs, dictionary=dictionary, coherence='c_v')
98     coherence_score = coherence_model.get_coherence()
99     print(f"Coherence Score: {coherence_score:.4f}")
100    print(f"Coherence model calculated in {time.time() - coherence_start:.2f} seconds")

```

Figure 4. 30: Code to calculate coherence score BERTopic topic modelling.

Figure 4.30 show the BERTopic, coherence evaluation starts by tokenizing documents using simple text splitting, resulting in a structured list of words. A Gensim Dictionary and bag-of-words corpus are then created from the tokenized texts. The function extracts the top 20 words per topic from the BERTopic model using topic_model.get_topic(topic_num), focusing on the most relevant terms. A

CoherenceModel is then initialized with these topics, tokenized texts, and dictionary, using the 'c_v' metric to assess topic quality. The function logs execution time and the coherence score, ensuring that the model produces interpretable and well-separated topics.

4.4.2 Visualisation

```
def plot_word_clouds(nmf_matrix, feature_names, save_dir=None):
    """
    Generate and save word clouds for each topic.
    """
    os.makedirs(save_dir, exist_ok=True)

    for idx, topic in enumerate(nmf_matrix):
        word_cloud = WordCloud(background_color='white', max_words=50).generate_from_frequencies(
            dict(zip(feature_names, topic))
        )

        plt.figure(figsize=(10, 10))
        plt.imshow(word_cloud, interpolation='bilinear')
        plt.axis("off")
        plt.title(f"Topic {idx + 1}", fontsize=16)

        # Save the word cloud if save directory is specified
        if save_dir:
            plt.savefig(os.path.join(save_dir, f'topic_{idx + 1}_wordcloud.png'))
        # plt.show()

def plot_top_words(nmf_model, feature_names, num_top_words, title, save_dir=None):
    """
    Plot the top words for each topic.
    """
    os.makedirs(save_dir, exist_ok=True)

    fig, axes = plt.subplots(1, nmf_model.components_.shape[0], figsize=(30, 15), sharex=True)
    axes = axes.flatten()

    for topic_idx, topic in enumerate(nmf_model.components_):
        top_feature_indices = topic.argsort()[::-num_top_words - 1:-1]
        top_features = [feature_names[i] for i in top_feature_indices]
        weights = topic[top_feature_indices]

        ax = axes[topic_idx]
        ax.barh(top_features, weights, height=0.7)
        ax.set_title(f'Topic {topic_idx + 1}', fontdict={'fontsize': 16})
        ax.invert_yaxis()
        ax.tick_params(axis='both', which='major', labelsize=12)
        for i in 'top right left'.split():
            ax.spines[i].set_visible(False)

    fig.suptitle(title, fontsize=22)
    fig.tight_layout()
    fig.subplots_adjust(top=0.90)
```

Figure 4. 31: Code for visualisation.

Figure 4.31 shows the example code to visualize the results of NMF (Non-negative Matrix Factorization) topic modelling. The `plot_word_clouds` function generates word clouds for each topic using the topic-term matrix (`nmf_matrix`) and corresponding feature names, saving the images in a specified directory. Each word cloud visually represents the most significant words in a topic based on their frequency or weight. The `plot_top_words` function creates horizontal bar charts showing the top words for each topic based on their weights in the NMF model's components. It organizes the subplots side-by-side and customizes the appearance for better readability, also saving the plots if a directory is provided. Together, these functions provide both visual and quantitative insights into the most relevant words for each discovered topic.

4.5 Web development implementation

The News Analysis Hub is designed to provide users with a smooth and efficient experience for analysing textual content from news articles. This section highlights the platform's main interfaces, including the main page, home dashboard, and document upload features. Users can perform advanced NLP analysis on text snippets or bulk CSV files and explore insightful visualizations such as topic models and word clouds. Each interface component is designed to facilitate intuitive navigation and handle large volumes of news data with ease.

1. Main page or Welcome page

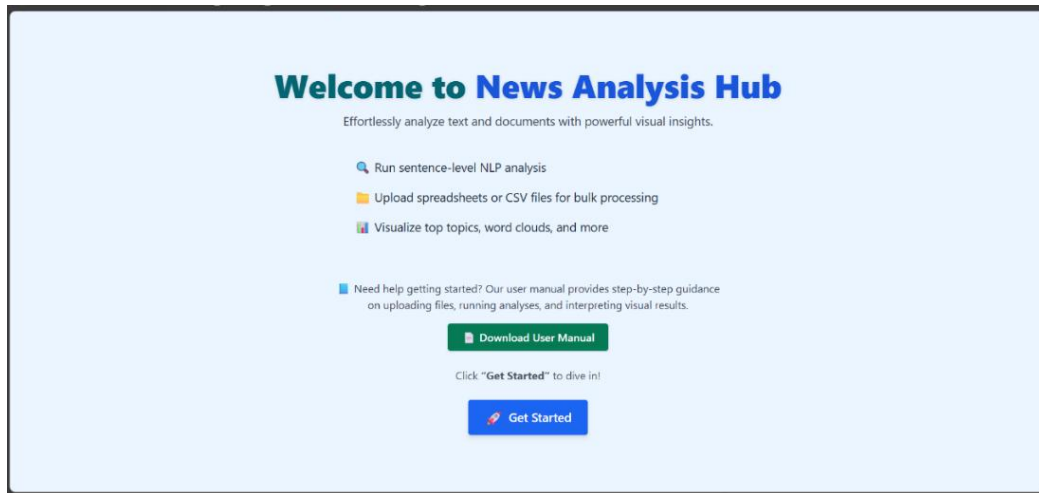


Figure 4. 32: Welcome page of News Analysis Hub.

Upon entering the web, users are welcomed by the News Analysis Hub’s main screen. A clear tagline communicates the platform’s ability to “effortlessly analyse text and documents with powerful visual insights.” Three iconized bullet points summarize key functions sentence-level NLP, bulk CSV processing, and topic or word-cloud visualization. A user manual is available for download to help users understand the web application. A centrally placed Get Started button encourages users to begin their analysis journey immediately, reducing cognitive load and guiding newcomers seamlessly into the workflow.

2. Home page

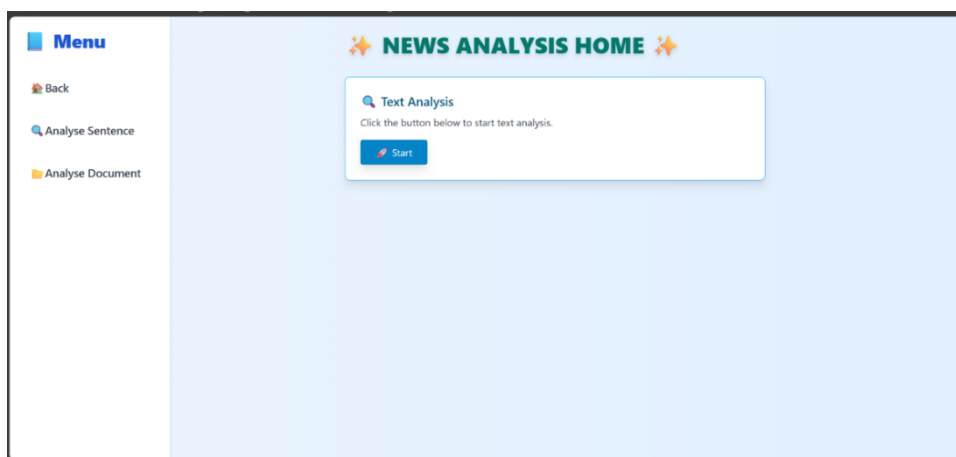


Figure 4. 33: Home page of News Analysis News Hub.

After proceeding, users will navigate to Home page dashboard, which features a persistent side menu containing links to “Back,” “Analyse Sentence,” and “Analyse Document.” The main panel displays a simple Text Analysis card with a prompt inviting users to start text analysis and a Start button. This layout ensures clarity and easy access to either quick text snippets or bulk document processing, with navigation always visible for convenience.

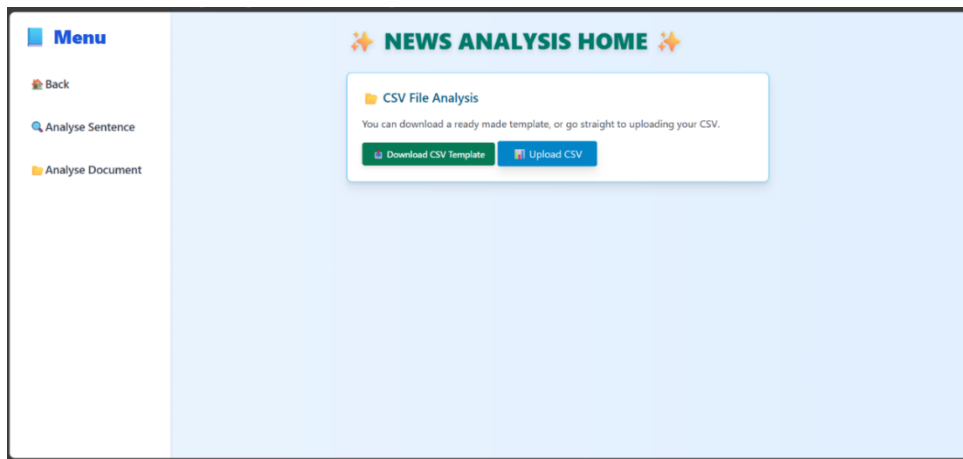


Figure 4. 34: CSV file Analysis interface.

Selecting “Analyse Document” will open the CSV File Analysis interface. At the top, users find buttons to download a CSV template preformatted with columns like Title, URL, Date, Source, and Article Content and to upload their own CSV files.

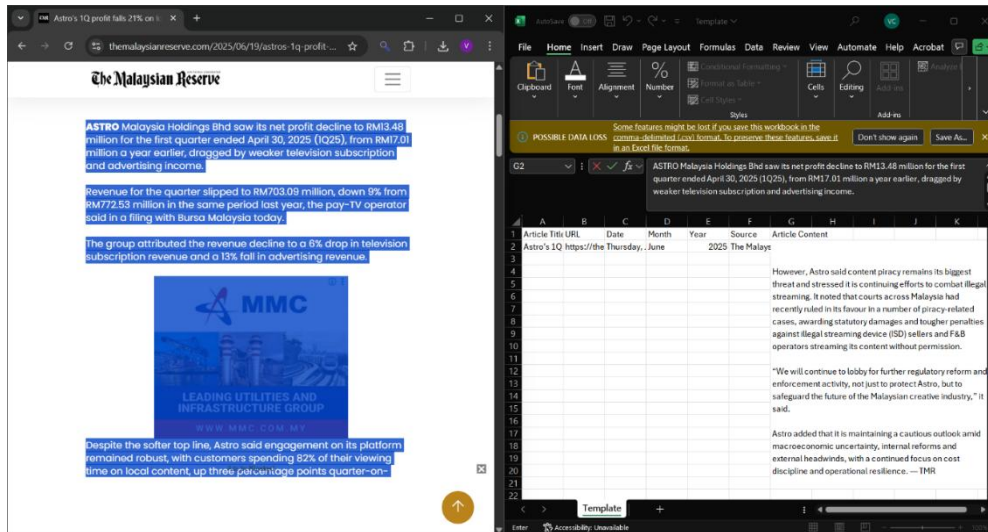


Figure 4. 35: Copy and paste news content.

Figure 4.35 displays the news content that has been copied and pasted into the CSV template.

	A	B	C	D	E	F	G	H
1	Article Title	URL	Date	Month	Year	Source	Article Content	
2								
3								
4								
5								

Figure 4. 36: CSV template.

	A	B	C	D	E	F	G
1	Article Title	URL	Date	Month	Year	Source	Article Content
2	Covid-19: A wake-up call for the hotel industry	https://themalaysianreser	Friday, Januar	January	2021	Malaysia Reserve	The pandemic has given all of us a window or opportunity to step up the game together as an industry
3	The boom of e-commerce in Covid-19 era	https://themalaysianreser	Friday, Januar	January	2021	Malaysia Reserve	The digital economy in Malaysia is expected to continue to be on a growth momentum
4	The longest year	https://themalaysianreser	Friday, Januar	January	2021	Malaysia Reserve	
5	Food delivery services: From odd job to the most	https://themalaysianreser	Friday, Januar	January	2021	Malaysia Reserve	
6	Matrix Concepts leverages tech to sail through C	https://themalaysianreser	Friday, Januar	January	2021	Malaysia Reserve	

Figure 4. 37: CSV template with data

Figures 4.36 and 4.37 show a spreadsheet preview shows the template's structure and example entries, helping users format their data correctly for smooth ingestion.

3. Upload document

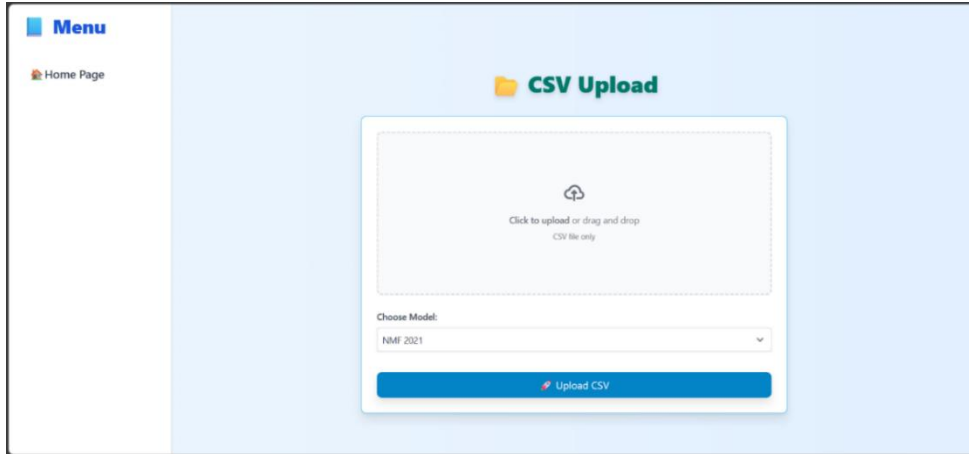


Figure 4. 38: Interface for upload document.

The CSV Upload page supports bulk processing of multiple news articles. Users can click or drag-and-drop a properly formatted CSV file into the upload area. After selecting a topic modelling option (e.g., “NMF 2021”), then upload the file. The system then parses each article’s details and queues them for NLP processing, enabling rapid handling of large datasets with minimal user effort.

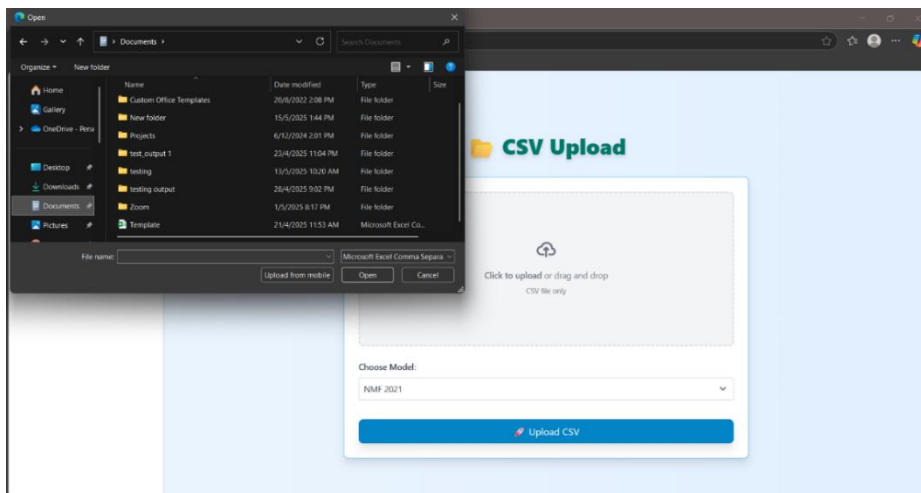


Figure 4. 39: Upload file.

The process of selecting a CSV file from the user’s computer Documents folder. The upload window allows users to either click to upload or drag and drop a CSV file into the designated area

4. Result page

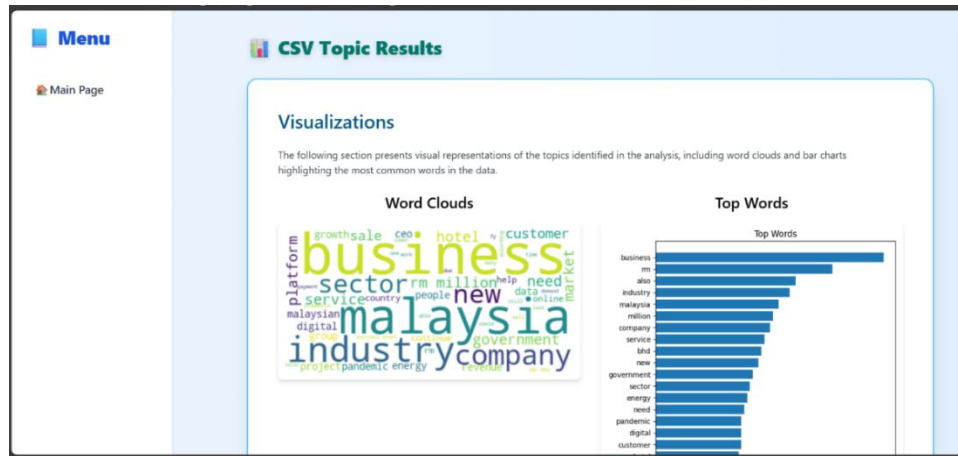


Figure 4. 40: CSV topic result interface.

The users are directed to the CSV Topic Results page, split into two main panels: visualizations and topic overview. On the left, a dynamic word cloud highlights key terms across the dataset, while the right panel shows a bar chart ranking the top words per topic.

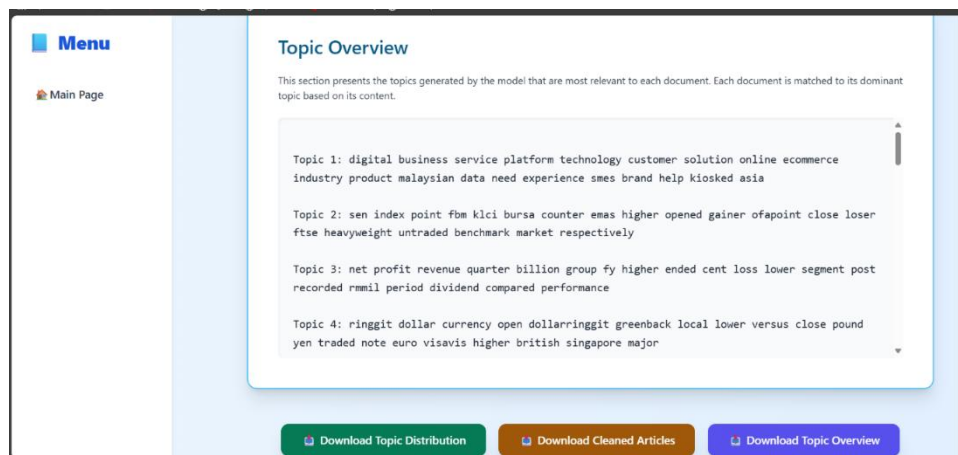


Figure 4. 41: CSV topic result interface.

Below, a scrollable log lists each article with its assigned topic number. Download options allow users to export the topic distribution matrix, cleaned article texts, or a simple topic summary for offline use or further integration.

5. Analyse text

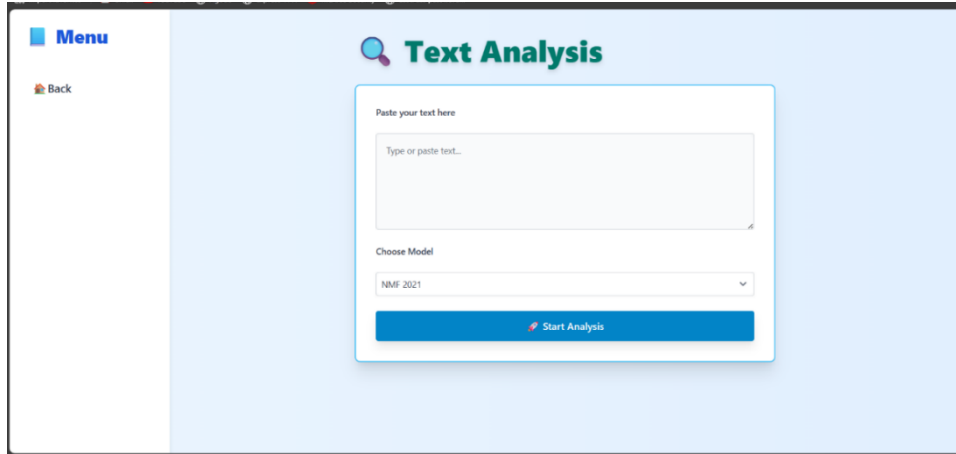


Figure 4. 42: Text analysis interface.

For quick analysis or one news article, the Text Analysis page lets users input freeform text into a large box, select a model, and start the analysis. This triggers the same backend pipeline used for CSV uploads but processes a single document, ideal for prototyping or exploring model behaviour on custom inputs.

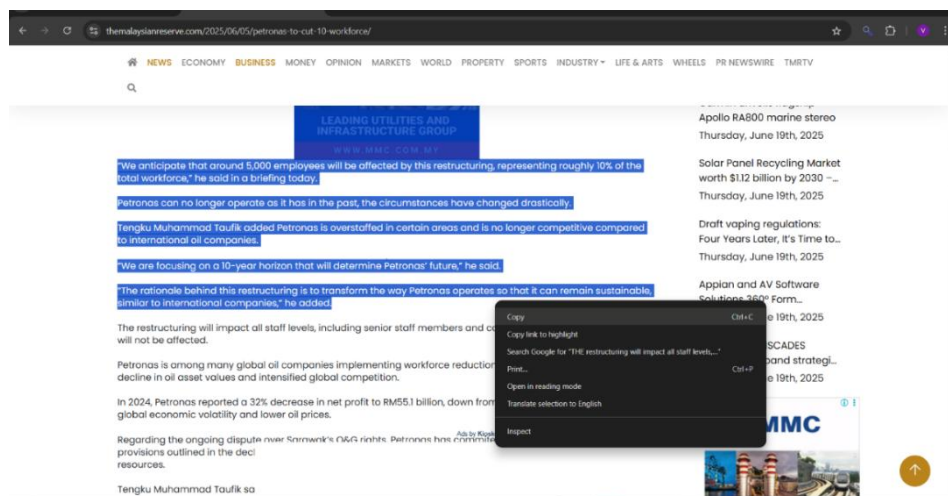


Figure 4. 43: Copy and paste news content.

Figure 4.43 below shows the process by which users can copy and paste news articles. Users highlight the desired content, copy it, and then paste the text into the input field labelled “Type or paste here.”

6. Result page

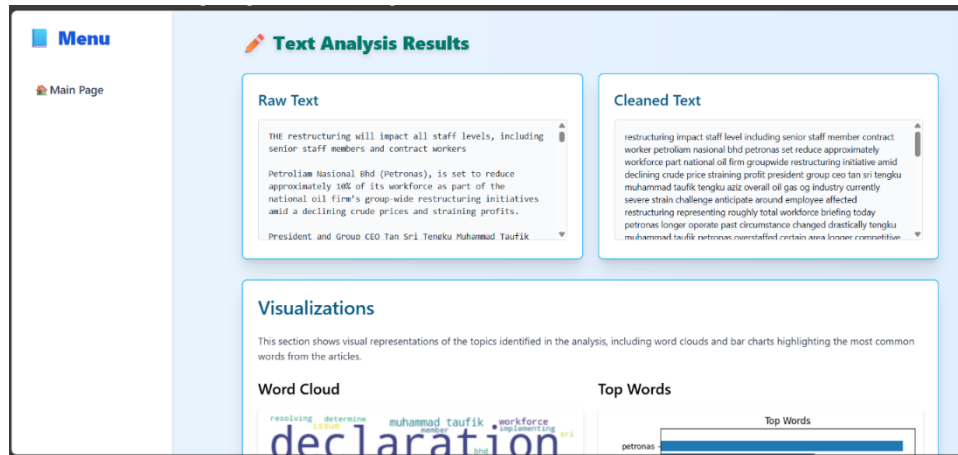


Figure 4.44: Text Analysis Result interface.

The Text Analysis Results screen show both the raw and cleaned versions of the input text side by side. This interface provides immediate insight into how the text is tokenized, cleaned, and transformed by the NLP pipeline, helping users verify the model's effectiveness even on a single paragraph of news content.

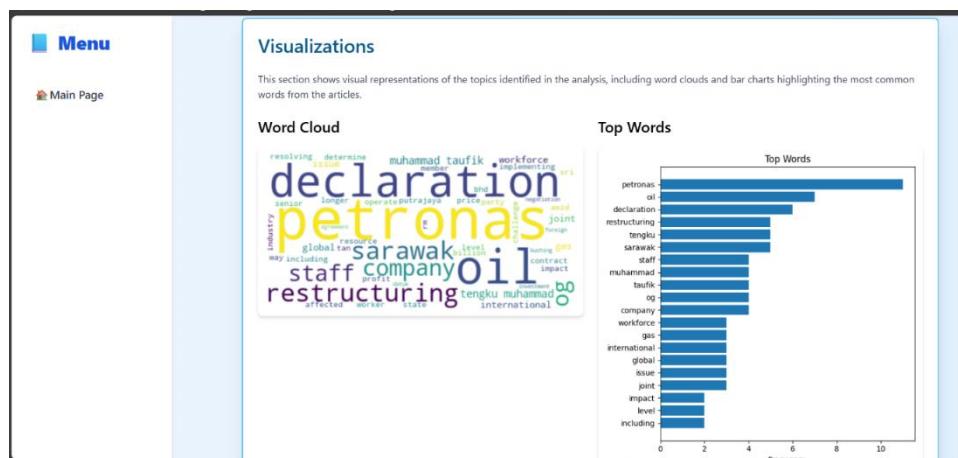


Figure 4.45: Text Analysis Result interface.

Figure 4.45 shows the visualizations appear a word cloud and a bar chart of top words.

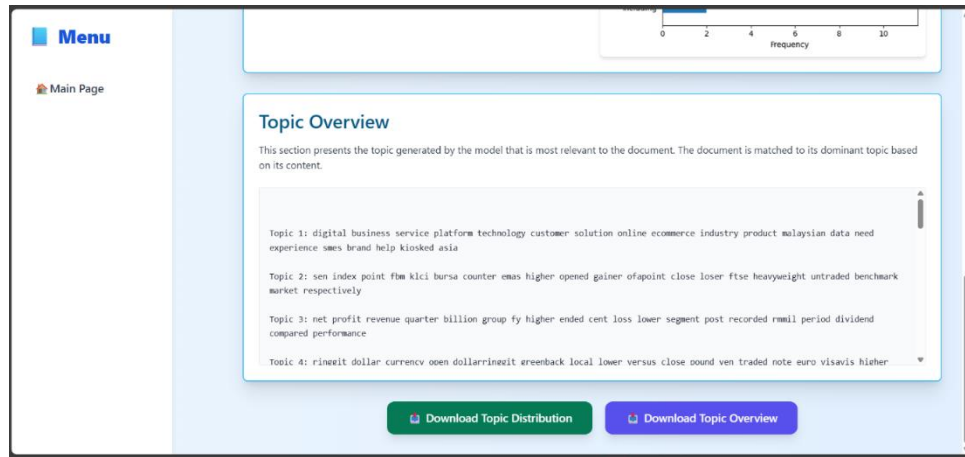


Figure 4. 46: Text Analysis Result interface.

The Topic Overview section provides a summary of the main topics extracted from the data, using natural language to describe each topic making it accessible to non-technical users. Additionally, the application includes download options that allow users to export both the topic distribution and overview summaries. These features support further analysis, reporting, and sharing, thereby enhancing the application's usefulness for research and business decision-making.

4.6 Summary

This chapter summarizes the data collection process, data preprocessing techniques, and the implementation of topic modelling. It also describes how the web application integrates the best-performing topic model into a web application system for presenting topics extracted from news articles.

Chapter 5: Experimental and results.

5.1 Introduction

This chapter discusses on the experimental and result on this research, the analysis of data collected from web-scraping news articles, preprocessing data and the evaluation of the topic-modelling results. This includes the analysis of the User Acceptance Test (UAT) conducted on the web applications.

5.2 Analysis dataset

This section discusses the analysis of web-scraping Malaysian business news from 2019 to 2023, the preprocessing workflow and the evaluation of four topic-modelling. Also, the outcomes of the User Acceptance Test (UAT), covering the usability and the user feedback.

5.2.1 Analysis of the news data

This analysis focuses on the news data collected from the years 2019 to 2023, covering a five-year period of business-related developments. The collected data includes articles from multiple Malaysian news sources.

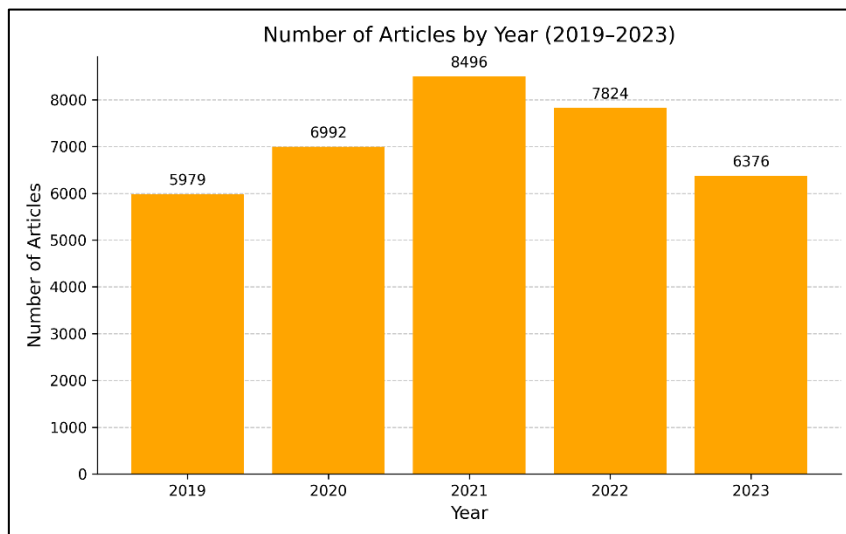


Figure 5. 1: The bar graph number of articles by (2019-2023).

The bar chart illustrates the yearly count of articles obtained through web scraping, showing an increase from 5,979 in 2019 to a peak of 8,496 in 2021, which represents the highest volume within the five-year span. Following this peak, the number of articles declined to 7,824 in 2022 and further to 6,376 in 2023, suggesting that data availability or collection activities increased up to 2021 before experiencing a subsequent reduction.

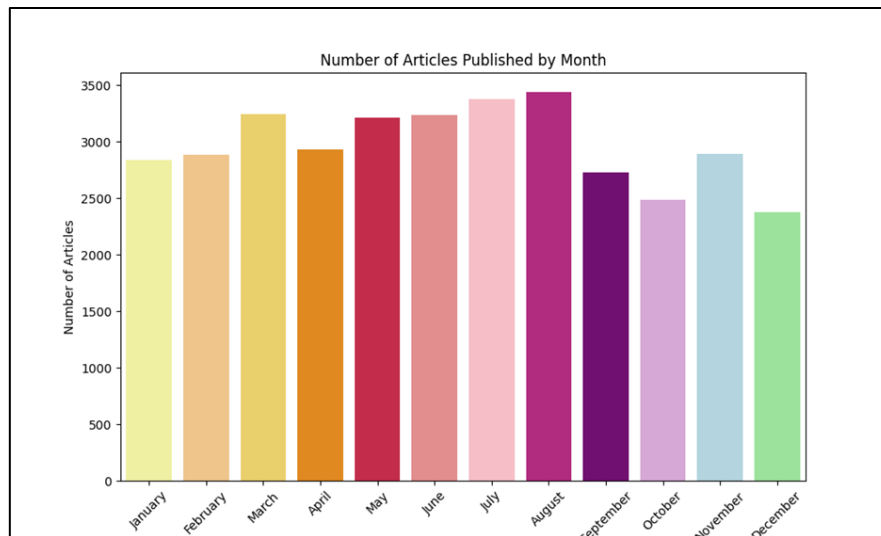


Figure 5. 2: The bar graph number of articles published by month.

The bar graph shows several insights about news article publication trends spanning the years 2019 to 2023, based on the graphs. First, examining the "Number of Articles Published by Month" graph, we observe that the monthly article count generally ranges between 2500 and 3500. August has the highest number of articles, while December has the lowest. The graph shows that August is the peak month for article publication across these years, closely followed by July. On the other hand, December consistently records the lowest publication numbers throughout the observed years. This graph provides insights into the number of articles published over the five-year period, emphasizing fluctuations in publishing activity across different months.

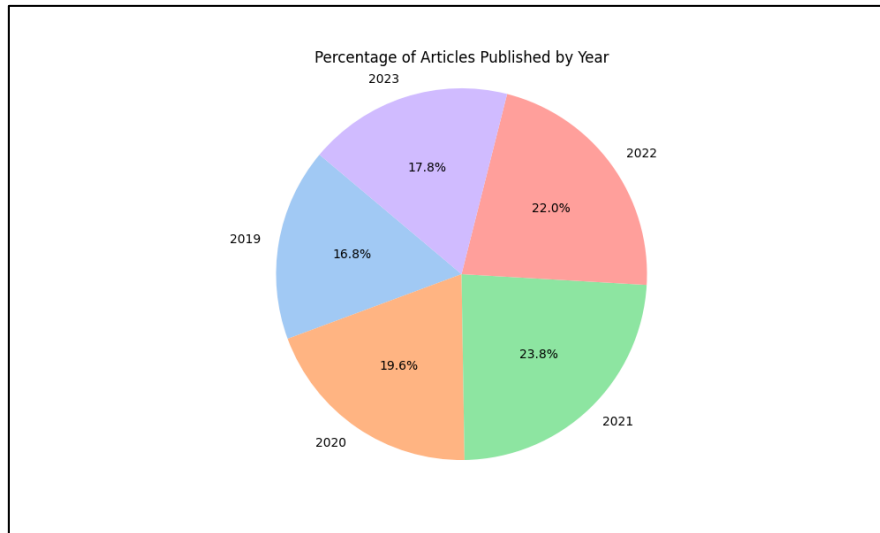


Figure 5. 3: The pie chart Percentage of Articles Published by Year.

The pie chart provides an overview of the "Percentage of Articles Published by Year" published between 2019 and 2023. The highest number of articles published was in 2021, accounting for 24.8% of the total. This suggests that significant global events, such as the continued impact of the COVID-19 pandemic, economic recovery efforts, or political changes, may have driven increased media coverage during this year. In contrast, 2019 has the lowest percentage at 16.8%, potentially indicating fewer major events or reduced media focus during that year.

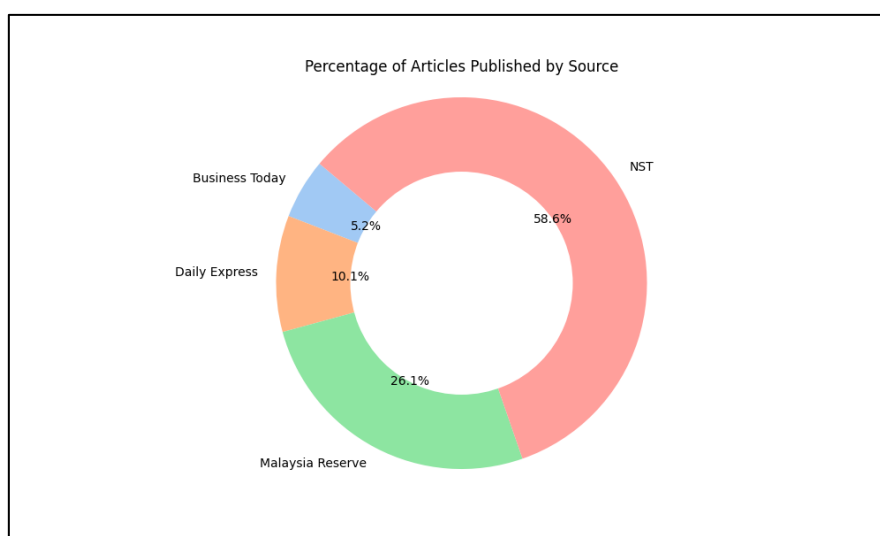


Figure 5. 4: The doughnut chart of Percentage of Articles Published by Source.

The doughnut chart provides the "Percentage of Articles Published by Source" and illustrates that the New Straits Times (NST) dominates with 58.6% of the articles. This dominance suggests a larger editorial team or broader coverage compared to other sources. Malaysia Reserve follows with 26.1%, Daily Express with 10.1%, and Business Today with 5.2%, indicating that these sources might have more niche coverage or limited resources.

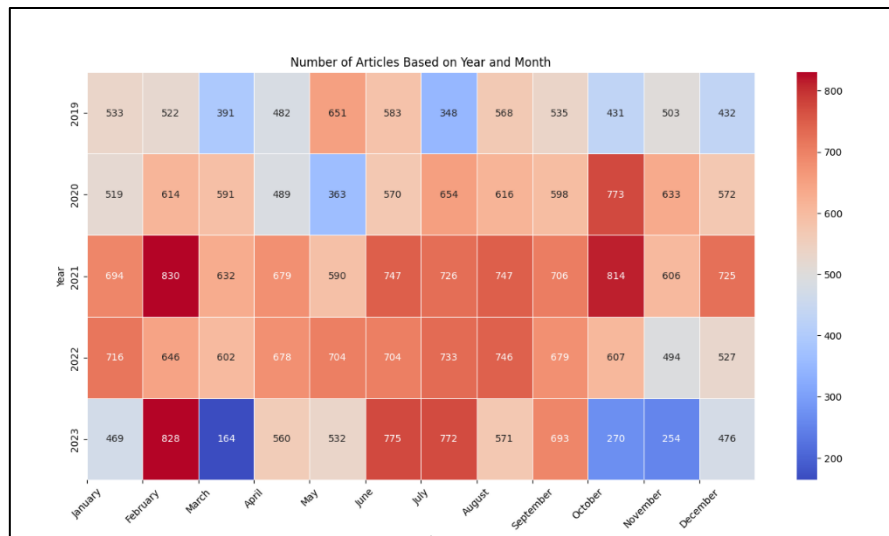


Figure 5. 5: The heatmap of Number of Articles Based on Year and Month.

The heatmap shows the "Number of Articles Based on Year and Month." There is notable variability in article counts. Certain months, such as February 2023 and November 2022, exhibit peaks in article counts, whereas March 2023 shows a significant drop. February 2023 has a high count, while the low in March 2023 could be due to fewer notable events during that period. The analysis shows that 2021 exhibited the peak volume of articles based on the overall number of articles from 2019 to 2023. These peaks could potentially correlate with recurring annual events or major global occurrences. Furthermore, the NST news article was a significant contributor to the overall article count, highlighting its prominent role in news dissemination compared to other sources.

5.2.2 Analysis of wordcount

Year	Wordcount
2019	2637629
2020	3020971
2021	3771605
2022	3210996
2023	2567718
TOTAL	15208919

Table 5. 1: Wordcount analysis.

Table 5.1 data show the annual word counts from 2019 to 2023, with a cumulative total of 15,208,919 words. Starting with 2,637,629 words in 2019, the count increases significantly each year, reaching a peak of 3,771,605 words in 2021. However, in 2022, there was a slight drop to 3,210,996 words, and in 2023, the word count decreased further to 2,567,718 words. This trend shows a general increase in word production up to 2021, followed by a decline over the next two years. Despite these variations, the overall word count for the five-year period remains substantial.

5.3 Topic model results

This section presents the coherence scores of four topic modelling techniques LDA, NMF, Top2Vec, and BERTopic evaluated over five years (2019–2023). Coherence scores were calculated using the Cv metric to assess the semantic consistency of topics generated by each model, offering insights into their performance and suitability for analysing Malaysian business news.

5.3.1 Coherence score

Model	LDA	NMF	Top2vec	BERTopic
2019	0.575	0.719	0.529	0.574

2020	0.615	0.711	0.547	0.608
2021	0.603	0.731	0.531	0.631
2022	0.562	0.708	0.508	0.546
2023	0.560	0.665	0.485	0.573

Table 5. 2: Coherence score of test data.

Table 5.2 summarizes the testing-set coherence scores for four topic modelling approaches LDA, NMF, Top2Vec, and BERTopic over the five-year period from 2019 through 2023. Coherence was computed using the Cv metric on held-out documents, ensuring comparability across models and years. LDA shows a moderate upward trend from 0.575 in 2019 to a peak of 0.615 in 2020, followed by a steady decrease, reaching 0.560 in 2023. This suggests that while LDA captured increasingly coherence topics at first, its performance decrease in recent years. NMF consistently performance with coherence scores maintaining levels near or above 0.71 between 2019 and 2022, before a minor decline to 0.665 in 2023. Its stability indicates robust topic extraction across the corpus. Top2Vec starts lowest at 0.529, shows a slight improvement in 2020 to 0.547, then decrease through 2023, reaching 0.485 possibly reflecting sensitivity to data. BERTopic increases from 0.574 in 2019 to a high of 0.631 in 2021, before dropping to 0.546 in 2022 and recovering somewhat to 0.573 in 2023, comparing to LDA’s pattern but with a stronger peak in the middle years. Overall, NMF achieves the highest average coherence with the highest 0.731 in 2021, indicating its ability to group semantically related words in Malaysian business-news corpus.

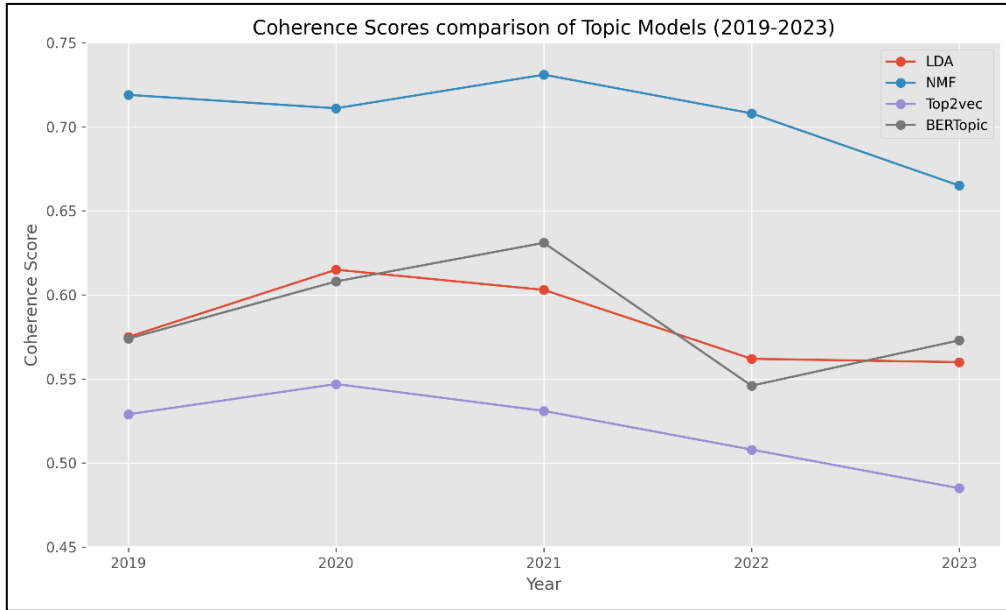


Figure 5. 6: Coherence scores comparison of topic model.

Figure 5.6 shows the graph of coherence scores comparison of topic model from 2019 to 2023. Based on the figure, NMF demonstrates more consistent performance compared to the other topic models.

5.3.2 Coherence score graph

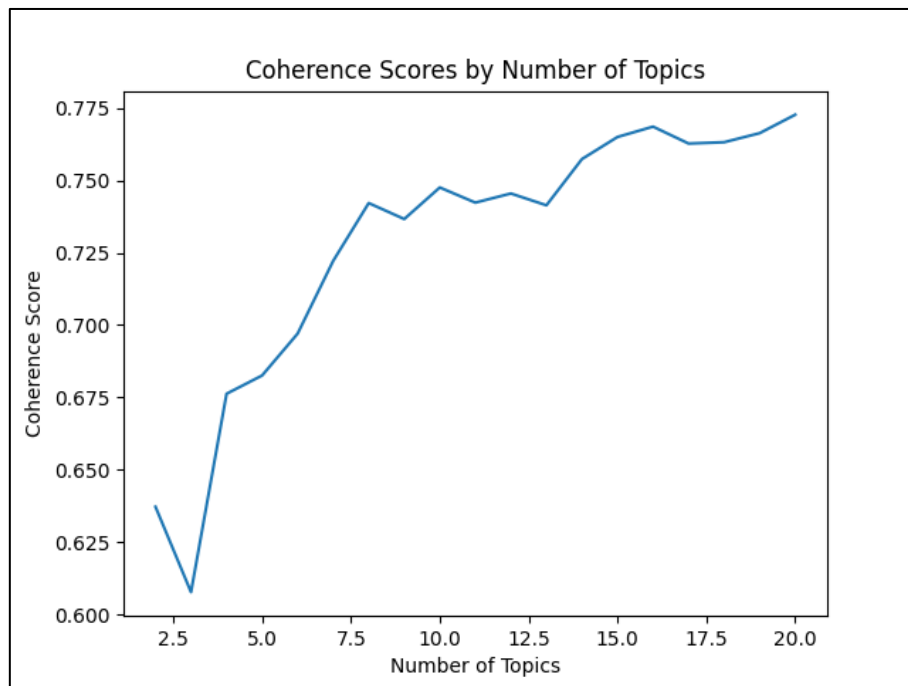


Figure 5. 7: Coherence score of NMF topic modelling for the year 2021.

The figure 5.7 shows an example of coherence score graph for NMF topic modelling for the year 2021. The x-axis represents the number of topics (ranging from 2 to 20), while the y-axis shows the corresponding coherence scores. The coherence score tends to be relatively low with fewer topics, indicating that the topics are too broad and less meaningful. As the number of topics increases from 4 to around 14, the coherence score improves steadily, suggesting that the model is better capturing distinct and interpretable. After 15 topics, the coherence improvement becomes minimal, suggesting a point of saturation where increasing the number of topics offers little added semantic value. The highest coherence score is observed at 20 topics, but the increase from 15 onwards is minimal. Therefore, an optimal range for topic selection appears to be between 15 and 17 topics, balancing interpretability and topic distinctiveness. Refer to Appendix B for the additional figure showing the coherence score.

5.3.3 Top words

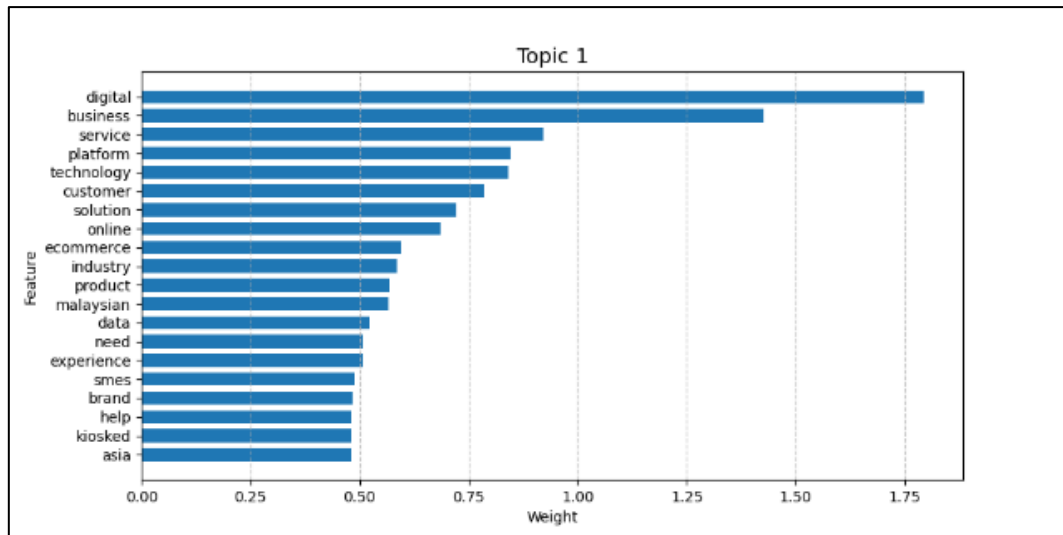


Figure 5. 8: Bar chart of topic.

Figure 5.8 shows an example of a bar chart representing the top words associated with Topic 1, from a topic modelling process of NMF. The bar graph is indicated the topics that are been discuss averagely. The x-axis indicates the weight or importance of each

word in defining the topic, while the y-axis lists the individual keywords (features). From the graph, we observe that words like “digital,” “business,” and “service” have the highest weights, indicating that they are the most representative terms within this topic. These high-weight keywords suggest that the topic likely centres around digital business services and technology platforms, as further emphasized by related terms such as “platform,” “technology,” “customer,” and “ecommerce.” This visualization is useful for interpreting the focus and scope of a topic discovered from a larger corpus. Refer to Appendix B for the additional figure of the topic bar chart.

5.4 Analysis of topic model on web applications

The analysis of topic models on web applications is based on documents related to topic words extracted from the NMF topic model. When users paste or upload data into the system to identify related topics, studying these topics year by year reveals shifts in focus and emerging patterns within the web application domain. Mapping documents to their main topics helps to understand how the content is distributed, and which topics are most important, giving a clear picture of patterns in web applications.

5.4.1 Analyse text

Using one example of a news article, identify the relationship between the topic model and the news data.

Visualization

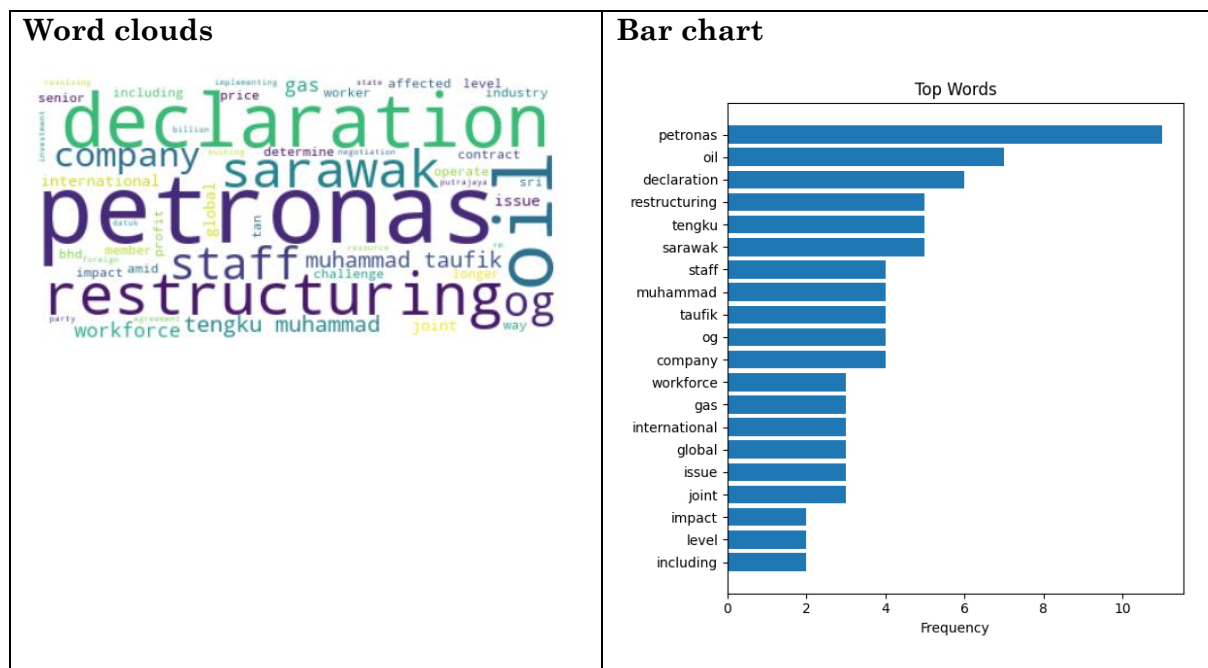


Table 5.3 shows the topics extracted from the news articles using a topic modelling approach. The word clouds display the most frequent and representative words for each topic, with larger words indicating higher importance or frequency within that topic. The bar charts show the importance of the top words by displaying their relative frequency.

NMF Model	The document related to topic word.	Topic words
2019	This document is related to topic 10	Topic 10: oil petronas gas price barrel crude production energy petroleum sapura og contract saudi opec offshore lng upstream drilling brent exploration
2020	This document is related to topic 8	Topic 8: oil price barrel crude petronas percent cut brent opec gas demand saudi petroleum ron market global arabia production russia energy
2021	This document is related to topic 16	Topic 16: petronas gas lng energy oil natural offshore vessel petroliam barrel carbon field exploration petroleum og block billion upstream production contract

2022	This document is related to topic 5	Topic 5: oil palm tonne cpo price plantation export production crude labour supply indonesia industry mpob worker zuraida output ffb product mspo
2023	This document is related to topic 13	Topic 13: oil palm tonne cpo price plantation production crude export indonesia gas ffb barrel eu output petronas fruit commodity boustead nino

Table 5. 4: Result topic overview of analyse text.

Table 5.4 shows how the same article was analysed using models trained on data from different years. Although each model assigns the article to a different topic, the associated topic words consistently relate to a similar topic. For example, terms like “oil” and “Petronas” appear across multiple years. This indicates that despite variations in topic numbers and word composition, the models are still capturing the same underlying subject matter or context.

5.4.2 Analyse document

Using the bulk dataset applied in the templates, the visualization provides a summary of the whole data collection. The topic words demonstrate the connection between each document and its related topic.

Visualization

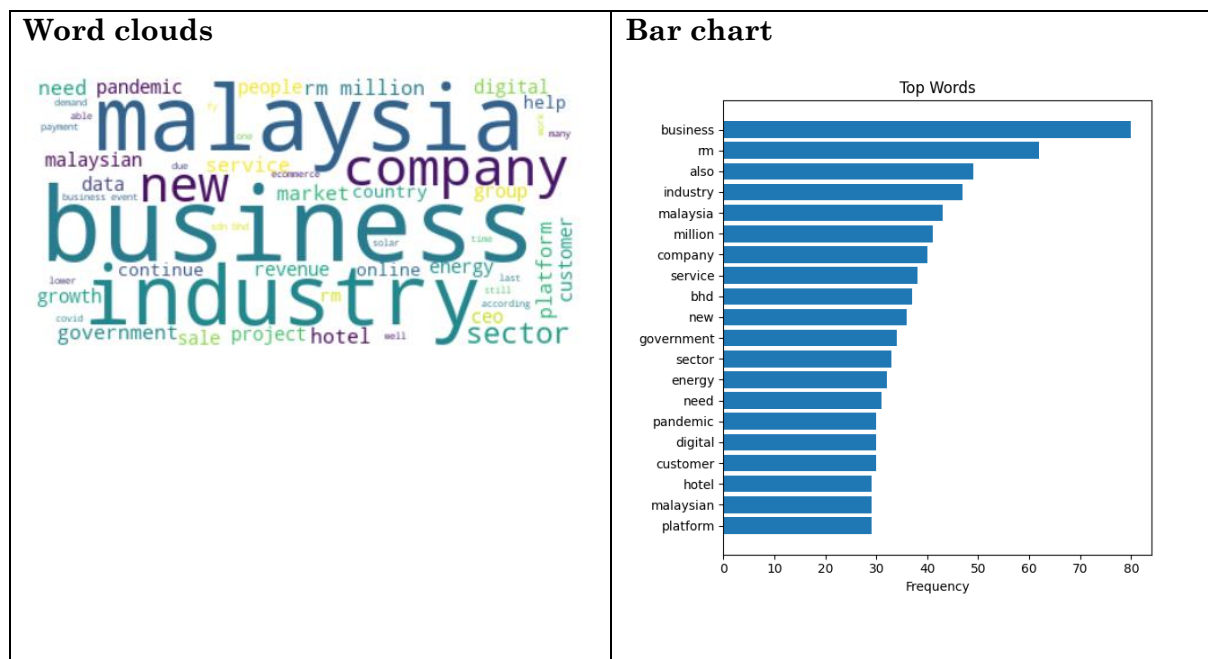


Table 5. 5: Word clouds and Bar chart of analyse document.

Table 5.5 presents visualizations of the entire dataset through word clouds and bar charts. The word clouds highlight the most frequent and significant topic words, with the size of each word indicating its importance within the dataset. The bar charts show the relative frequency of these top words, providing a clear indication of their importance.

NMF Model	The document related to topic word.	Topic words
2019	Document 1 is related to Topic 1 Document 2 is related to Topic 1 Document 3 is related to Topic 1 Document 4 is related to Topic 1 Document 5 is related to Topic 12 Document 6 is related to Topic 14 Document 7 is related to Topic 1 Document 8 is related to Topic 14 Document 9 is related to Topic 1 Document 10 is related to Topic 1 Document 11 is related to Topic 1	Topic 1: digital business service industry technology customer platform product solution online smes country brand asia malaysian experience consumer ecommerce group programme Topic 10: oil petronas gas price barrel crude production energy petroleum sapura og contract saudi opec offshore lng upstream drilling brent exploration Topic 12: project contract construction property

	Document 12 is related to Topic 12 Document 13 is related to Topic 15 Document 14 is related to Topic 15 Document 15 is related to Topic 1 Document 16 is related to Topic 10 Document 17 is related to Topic 1 Document 18 is related to Topic 12 Document 19 is related to Topic 10 Document 20 is related to Topic 1	development work sdn group land ecrl rail contractor phase engineering subsidiary expected secured residential filing hotel Topic 14: unit proton sale vehicle car perodua model automotive suv maa sold month volume drbhicom aruz carmaker honda tiv brand geely Topic 15: tn timer solar power electricity plant renewable mw generation project coal yeo malakoff tenaga nasional utility pv capacity green moggie
2020	Document 1 is related to Topic 16 Document 2 is related to Topic 1 Document 3 is related to Topic 10 Document 4 is related to Topic 1 Document 5 is related to Topic 19 Document 6 is related to Topic 15 Document 7 is related to Topic 16 Document 8 is related to Topic 15 Document 9 is related to Topic 16 Document 10 is related to Topic 1 Document 11 is related to Topic 1 Document 12 is related to Topic 19 Document 13 is related to Topic 12 Document 14 is related to Topic 12 Document 15 is related to Topic 1 Document 16 is related to Topic 8 Document 17 is related to Topic 20 Document 18 is related to Topic 20 Document 19 is related to Topic 8	Topic 1: business digital online customer platform smes service ecommerce company technology solution mco new delivery product malaysian need help food consumer Topic 8: oil price barrel crude petronas percent cut brent opec gas demand saudi petroleum ron market global arabia production russia energy Topic 10: glove rubber demand medical piece manufacturer supermax hartalega production capacity maker share kossan company price margma asp corp billion nitrile Topic 12: tn timer solar power electricity renewable plant petronas gas tenaga amir generation pv wind mw asset nasional company dividend acquisition

	Document 20 is related to Topic 1	Topic 15: sale vehicle proton car automotive unit perodua tax tiv maa volume model new nap cent month exemption industry sold auto Topic 16: hotel tourism travel guest tourist room industry association package tour mco property mall resort kl outbreak covid matta health agent Topic 19: project contract construction million property development group company sdn land work share agreement engineering proposed building infrastructure new subsidiary unit Topic 20: chairman appointed pnb board group appointment director ceo datuk abdul investment management khazanah executive chief tan company sri fund effective
2021	Document 1 is related to Topic 15 Document 2 is related to Topic 1 Document 3 is related to Topic 8 Document 4 is related to Topic 1 Document 5 is related to Topic 5 Document 6 is related to Topic 18 Document 7 is related to Topic 14 Document 8 is related to Topic 18 Document 9 is related to Topic 1 Document 10 is related to Topic 1 Document 11 is related to Topic 1	Topic 1: digital business service platform technology customer solution online ecommerce industry product malaysian data need experience smes brand help kiosk asia Topic 5: contract project construction property development work sdn land bag building secured engineering subsidiary group rmmil worth award infrastructure win expected

	Document 12 is related to Topic 20 Document 13 is related to Topic 20 Document 14 is related to Topic 20 Document 15 is related to Topic 1 Document 16 is related to Topic 6 Document 17 is related to Topic 1 Document 18 is related to Topic 5 Document 19 is related to Topic 16 Document 20 is related to Topic 1	Topic 6: oil palm cpo tonne price plantation production crude export fgv mpob labour barrel demand india mom shortage edible ffb industry Topic 8: glove rubber asp piece production hartalega labour cbp billion demand nitrile forced capacity line manufacturing price factory supermax latex mah Topic 14: vaccine pharmaniaga dos covid sinovac vaccination supply government medical healthcare zulkarnain pharmaceutical health distribution hospital programme approval private clinical national Topic 15: airline travel airasia air aircraft passenger aviation flight mag carrier mab international domestic recovery group demand border cargo fleet tourism Topic 16: petronas gas lng energy oil natural offshore vessel petroliaam barrel carbon field exploration petroleum og block billion upstream production contract Topic 18: sale unit vehicle automotive proton perodua umw car toyota exemption production tax motor model maa volume industry customer sold order
--	--	--

		Topic 20: solar energy power tn timer renewable mw plant project pv rooftop electricity solarvest capacity green malakoff photovoltaic carbon installation sustainable generation
2022	Document 1 is related to Topic 7 Document 2 is related to Topic 2 Document 3 is related to Topic 2 Document 4 is related to Topic 2 Document 5 is related to Topic 2 Document 6 is related to Topic 9 Document 7 is related to Topic 2 Document 8 is related to Topic 9 Document 9 is related to Topic 2 Document 10 is related to Topic 2 Document 11 is related to Topic 2 Document 12 is related to Topic 8 Document 13 is related to Topic 8 Document 14 is related to Topic 8 Document 15 is related to Topic 2 Document 16 is related to Topic 5 Document 17 is related to Topic 2 Document 18 is related to Topic 13 Document 19 is related to Topic 3 Document 20 is related to Topic 2	Topic 2: digital business technology service industry customer solution platform malaysian investment product experience development country group programme asia opportunity collaboration smes Topic 3: research fy earnings hlib price rhb firm target cost forecast cent buy margin glove kenanga maintained midf expectation higher affin Topic 5: oil palm tonne cpo price plantation export production crude labour supply indonesia industry mpob worker zuraida output ffb product mspo Topic 7: airline airport airasia passenger flight travel international mahb aircraft aviation air route traffic border airway movement destination carrier cargo traveller Topic 8: project energy contract solar construction sdn power engineering plant tn timer development property petronas gas renewable subsidiary work group sapura carbon Topic 9: unit sale vehicle proton toyota automotive umw perodua car model ev exemption sold

		month motor volume tiv production maa property Topic 13: share million ipo listing public bursa placement capital market ace shareholder sen issue proceeds investor proposed group director private acquisition
2023	Document 1 is related to Topic 4 Document 2 is related to Topic 4 Document 3 is related to Topic 4 Document 4 is related to Topic 4 Document 5 is related to Topic 5 Document 6 is related to Topic 14 Document 7 is related to Topic 4 Document 8 is related to Topic 14 Document 9 is related to Topic 4 Document 10 is related to Topic 4 Document 11 is related to Topic 4 Document 12 is related to Topic 5 Document 13 is related to Topic 15 Document 14 is related to Topic 15 Document 15 is related to Topic 4 Document 16 is related to Topic 13 Document 17 is related to Topic 4 Document 18 is related to Topic 5 Document 19 is related to Topic 13 Document 20 is related to Topic 4	Topic 4: digital business industry technology investment service malaysian government country solution smes new platform customer programme product need minister global officer Topic 5: property project development land construction developer johor residential lb sunway billion estate city gamuda housing park sale gdv unit home Topic 13: oil palm tonne cpo price plantation production crude export indonesia gas ffb barrel eu output petronas fruit commodity boustead nino Topic 14: ev vehicle unit sale automotive proton car model electric toyota tesla perodua umw sime tiv charging motor darby new industry Topic 15: energy solar power renewable tnb plant project transition petronas mw green gas carbon sustainable emission solution capacity clean electricity pv

Table 5. 6: Result topic overview of analyse document.

Table 5.6 presents the analysis of a bulk news article using topic models trained on data from different years. Although the specific topic numbers and word differ across years, many of the topics reflect consistent themes such as digital business, energy, automotive, and infrastructure. For example, documents related to “digital business platforms” appear across multiple years with similar keywords like “platform,” “technology,” and “ecommerce,” even though they are assigned to different topic numbers. This consistency across years indicates that the model successfully identifies recurring themes and adapts to changes in context, such as the emergence of pandemic-related topics in 2020 or renewable energy in later years. Overall, the results show that topic modelling effectively captures the evolving but thematically connected focus of news articles over time.

5.5 Functional Testing

No.	Test Objective	Test Step	Expected Results	Pass/Fail
1.	Verify that the welcome page of “News Analysis Hub” is displayed.	1. Launch the website. 2. Observe the initial screen	The “News Analysis Hub” welcome page is displayed.	Pass
2.	Verify that the user can click and download the “User Manual”	1. Click on the “User Manual”. 2. Observe the pop-up message.	The User Manual PDF file downloads successfully, and a confirmation message appears.	Pass
3.	Verify that the user can click the “Get Started” button.	1. Click the “Get Started” button. 2. Observe the loaded screen.	The “News Analysis Home” page is displayed.	Pass
4.	Verify that the sidebar allows selection of “Back,” “Analyse Sentence,” and	1. Click each sidebar button.	Each button loads its corresponding page.	Pass

	"Analyse Document."	2. Observe the loaded screen.		
5.	Verify that clicking "Analyse Sentence" displays the Text Analysis prompt and a "Start" button.	1. Click "Analyse Sentence." 2. Observe the displayed screen.	The Text Analysis prompt appears with a "Start" button.	Pass
6.	Verify that clicking "Start" reveals the text-input box, model selector, and "Start Analysis" button.	1. Click "Start" button. 2. Observe the displayed screen.	The text-input box, "Choose Model" dropdown, and "Start Analysis" button are displayed.	Pass
7.	Verify that the user can paste text, select a model, and start analysis.	1. Paste text into the input box. 2. Select a model from the dropdown. 3. Click "Start Analysis."	The pasted text appears in the input box, the selected model is shown, and analysis begins.	Pass
8.	Verify that the "Text Analysis Results" screen shows Raw Text, Cleaned Text, Visualizations, and Topics.	1. Wait for analysis to complete. 2. Observe the results screen.	The results screen displays: • "Raw Text" section. • "Cleaned Text" section. • Visualizations (word cloud & bar chart). • "Topic Overview" section.	Pass
9.	Verify that the user can download the topic distribution and overview.	1. Click "Download Topic Distribution." 2. Click "Download Topic Overview."	CSV files download successfully, and a confirmation message appears.	Pass

		3. Observe pop-up messages.		
10.	Verify that the sidebar “Back” button returns to News Analysis Home from the Text Analysis page.	1. Click “Back” in the sidebar. 2. Observe the displayed screen.	The News Analysis Home page is displayed.	Pass
11.	Verify that clicking “Analyse Document” shows the CSV analysis options.	1. Click “Analyse Document.” 2. Observe the displayed screen.	The page displays a prompt with “Download CSV Template” and “Upload CSV” buttons.	Pass
12.	Verify that the “Download CSV Template” button downloads the template.	1. Click “Download CSV Template.” 2. Observe the browser download.	The CSV template downloads successfully, and a confirmation appears.	Pass
13.	Verify that clicking “Upload CSV” reveals the file upload, model selector, and “Upload CSV” button.	1. Click “Upload CSV.” 2. Observe the displayed screen.	A drag-and-drop upload box, “Choose Model” dropdown, and “Upload CSV” button are displayed.	Pass
14.	Verify that the user can upload a CSV, select a model, and start analysis.	1. Upload or drag-and-drop a CSV file. 2. Select a model. 3. Click “Upload CSV.”	The file uploads successfully, the selected model is shown, and analysis begins.	Pass
15.	Verify that the “CSV Topic Results” screen shows visualizations and topic overview.	1. Wait for analysis to complete. 2. Observe the results screen.	The “CSV Topic Results” page displays visualizations (word cloud & bar chart) and the “Topic Overview” section.	Pass

16.	Verify that downloads for CSV results work on the CSV results page.	1. Click “Download Topic Distribution,” “Download Cleaned Articles,” and “Download Topic Overview.” 2. Observe pop-up messages.	All three CSV files download successfully, and confirmation messages appear.	Pass
17.	Verify that the “Main Page” button in the sidebar returns to News Analysis Home.	1. Click “Main Page” in the sidebar. 2. Observe the displayed screen.	The News Analysis Home page is displayed.	Pass

Table 5. 7: Functional Testing.

5.6 User Acceptance Test (UAT)

In this section, a User Acceptance Test (UAT) was conducted to collect feedback on the web application. Respondents tested and evaluated the application by completing a questionnaire, using a five-point scale: “Strongly disagree,” “Disagree,” “Neutral,” “Agree,” and “Strongly agree,” based on their experience with the system. The questionnaire is divided into two sections: Demographic Information and Web Application Usability Scale.

Section 1: Demographic Information

Background questions:

Question	Respond
Faculty	Respondent’s faculty or course
Year	Respondent’s year of study
Gender	Respondent’s gender (Male/Female)

Table 5. 8: Background information.

Table 5.8 presents the background information of the respondents.

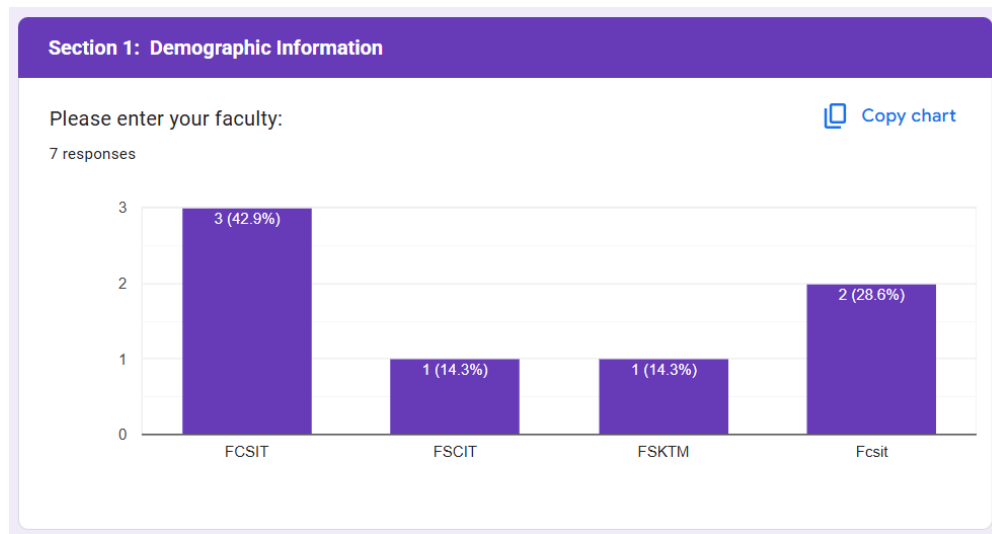


Figure 5. 9: Respondents faculty.

Figure 5.9 shows the distribution of respondents by faculty. Most of the respondents are from the Faculty of Computer Science and Information Technology.

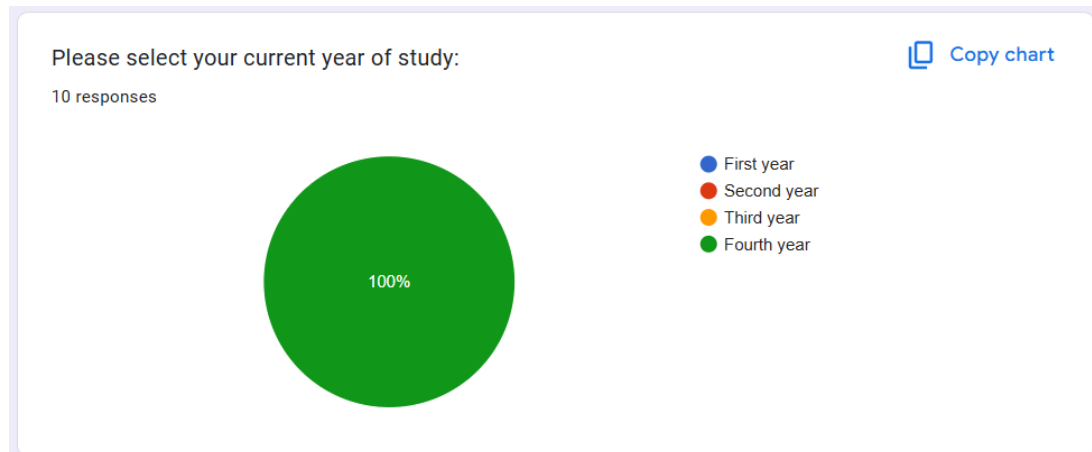


Figure 5. 10: Respondents' year of study.

Figure 5.10 shows the current year of study of the respondents. All respondents (100%) are in their fourth year of study.

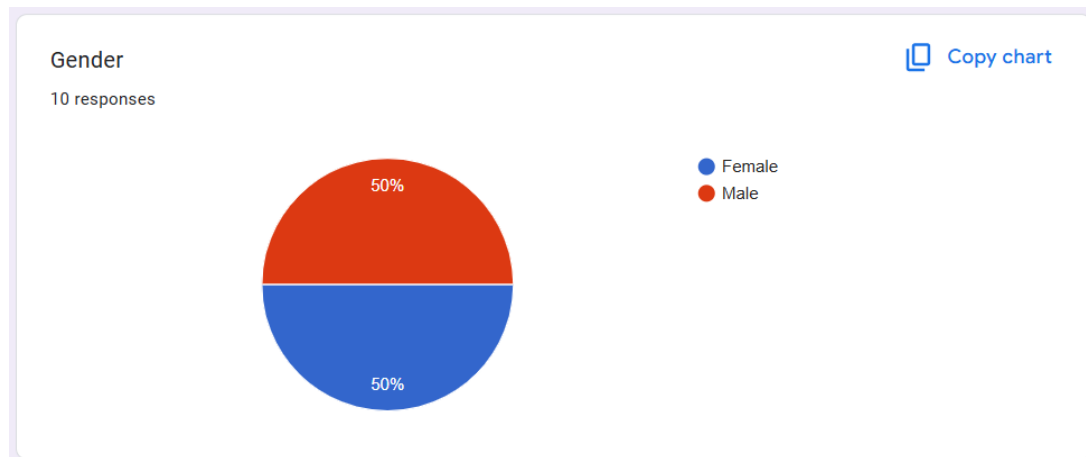


Figure 5. 11: Respondents gender.

Figure 5.11 displays the gender distribution of the respondents. An equal number of male and female respondents participated in this feedback session (5 males and 5 females).

Section 2: Web Application Usability Scale

Questions about the usability of the web application. This section is further split into two subsections:

A) Analyse Documents:

1. The interface for uploading and viewing documents is simple and easy to use.

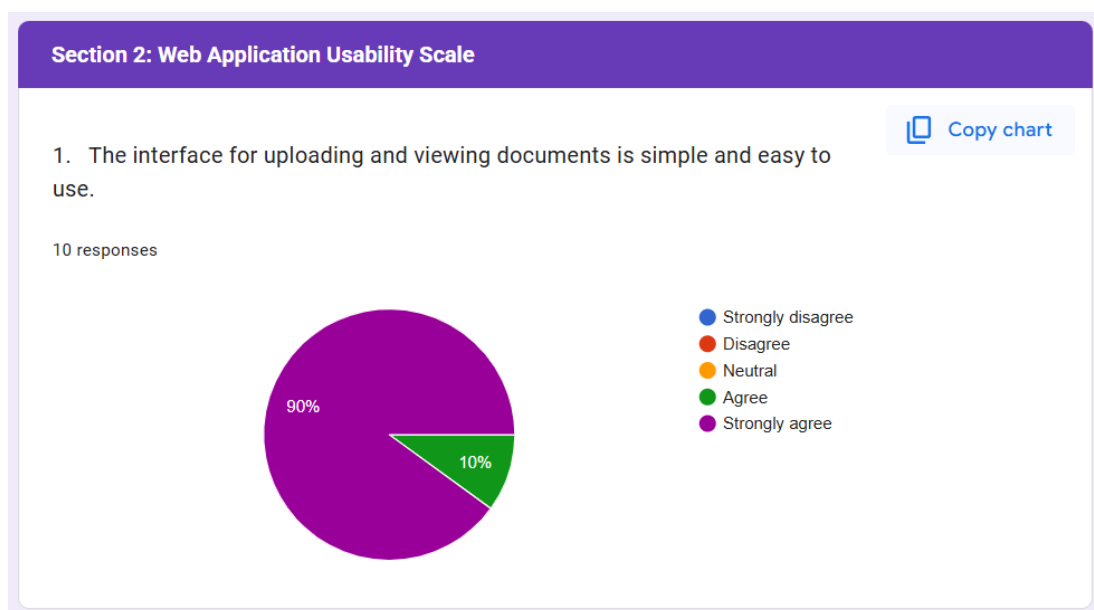


Figure 5. 12: Analyse Document usability scale.

Figure 5.12 shows the respondents' feedback. Based on the figure, 90% of respondents strongly agreed that the interface for uploading and viewing documents is simple and easy to use.

2. The provided file format or template was clear and easy to use.

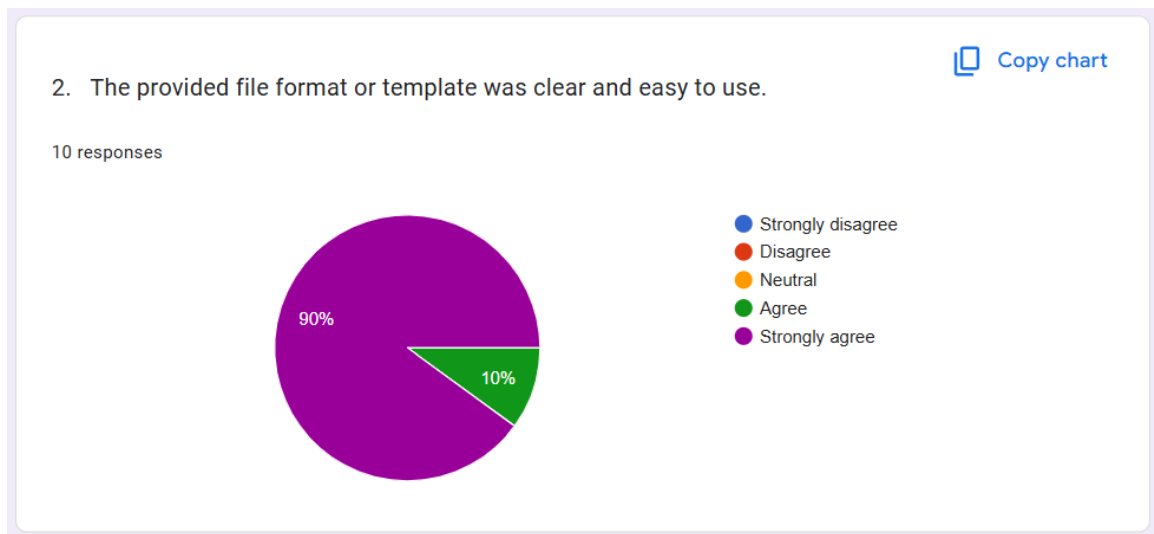


Figure 5. 13: Analyse Document usability scale.

Figure 5.13 shows the feedback on the provided file format or template. 90% of respondents strongly agreed, while 10% agreed.

3. Uploading documents or templates was smooth and without issues.

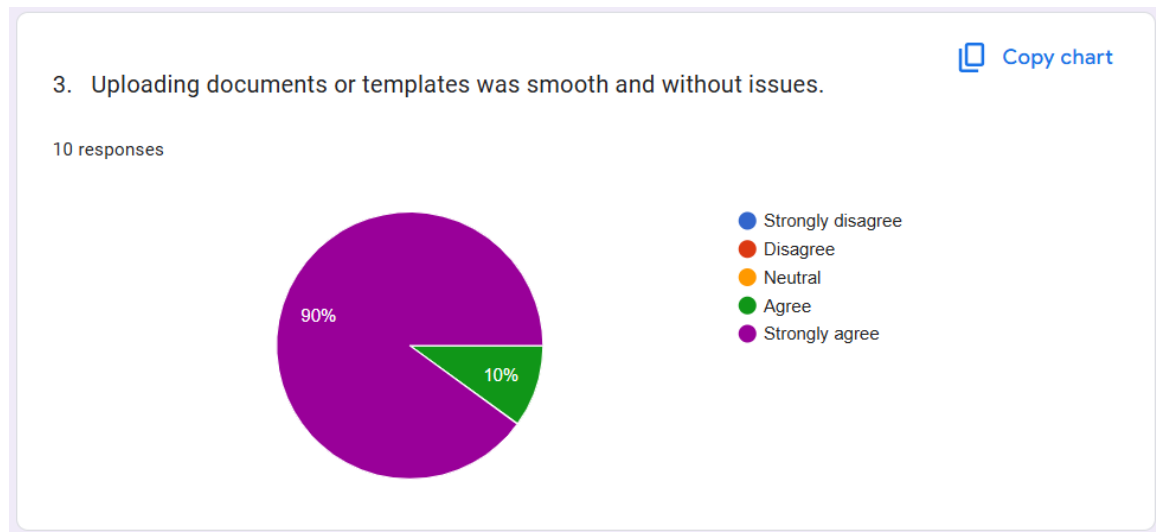


Figure 5. 14: Analyse Document usability scale.

Figure 5.14 presents the feedback showing that 90% of respondents strongly agreed that uploading documents or templates was smooth and without issues.

4. The analysis results were presented clearly and were easy to interpret.

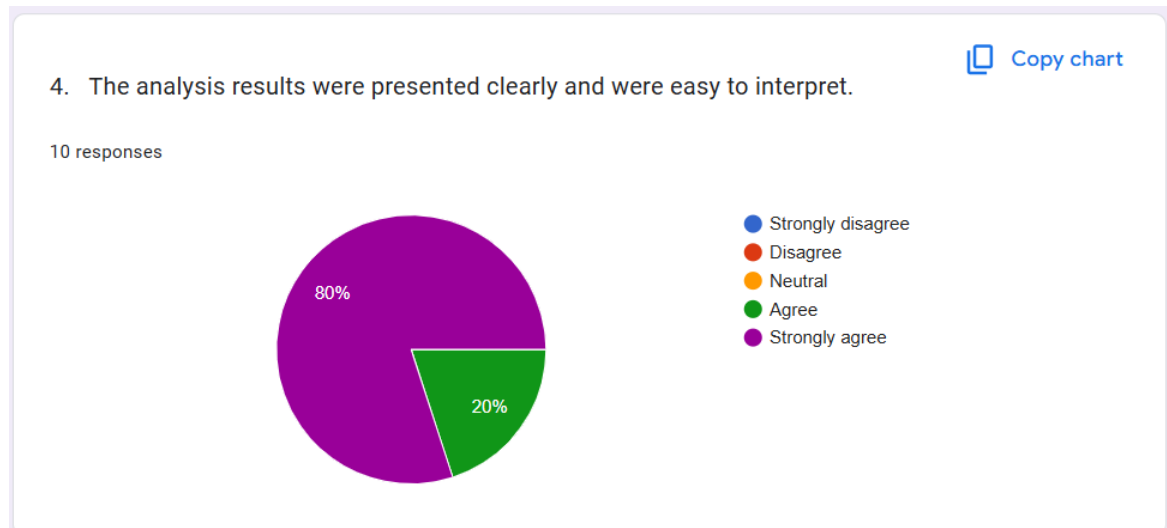


Figure 5. 15: Analyse Document usability scale.

Figure 5.15 shows the feedback indicating that 80% of respondents strongly agreed, and 20% agreed that the analysis results were clear and easy to interpret.

5. I was able to download the analysis results easily.

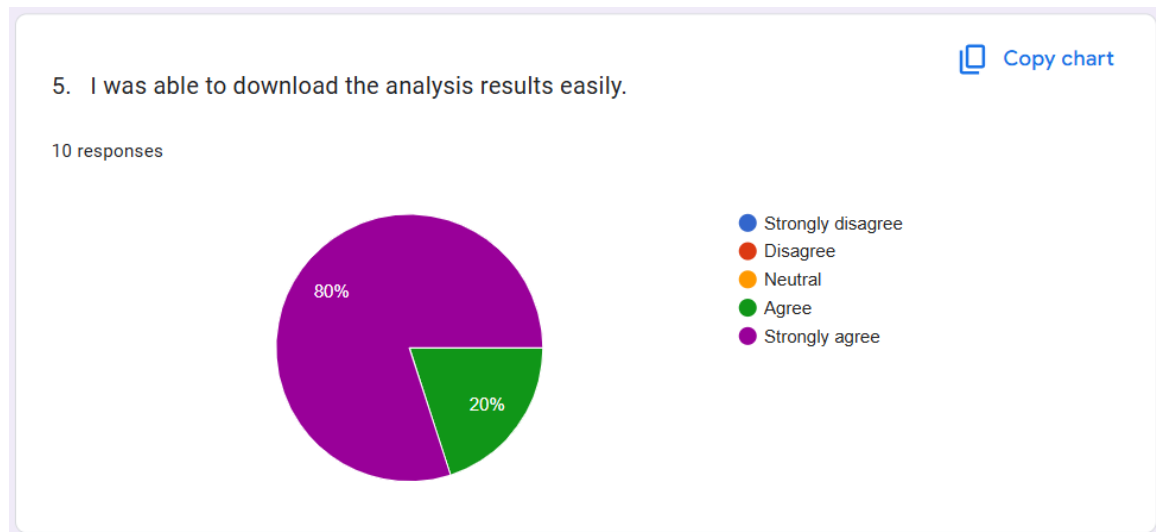


Figure 5. 16: Analyse Document usability scale.

Figure 5.16 shows that 80% of respondents strongly agreed, and 20% agreed that they could easily download the analysis results.

B) Analyse Sentences:

1. Entering or pasting the text for analysis was straightforward.

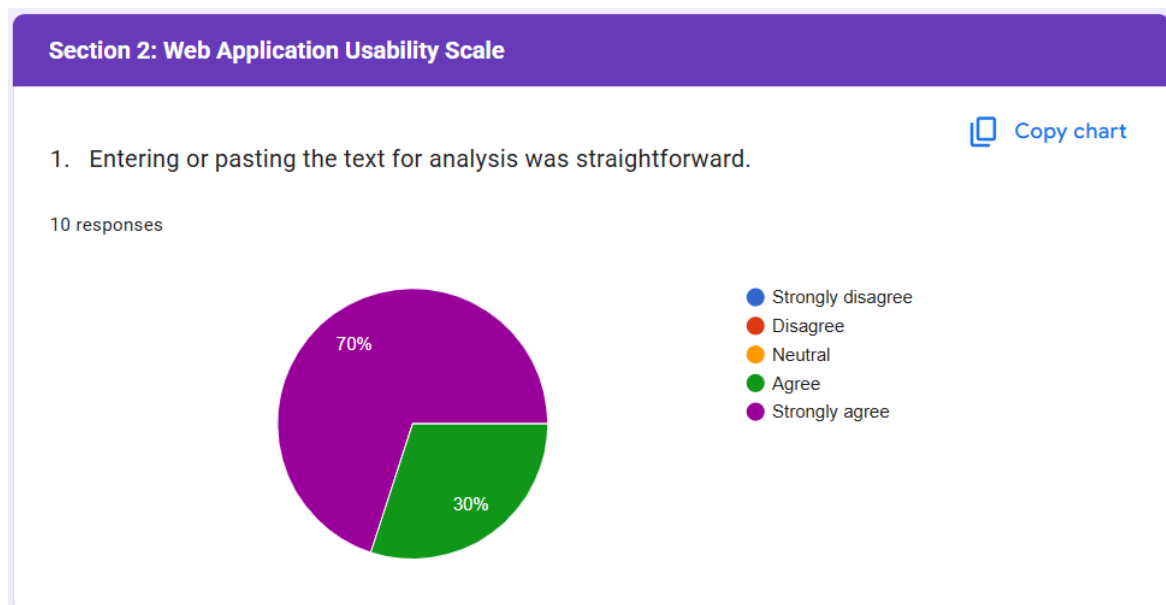


Figure 5. 17: Analyse Text usability scale.

Figure 5.17 shows that 70% of respondents strongly agreed, and 30% agreed that entering or pasting text for analysis was straightforward.

2. I understood how the input text was handled or processed before analysis.

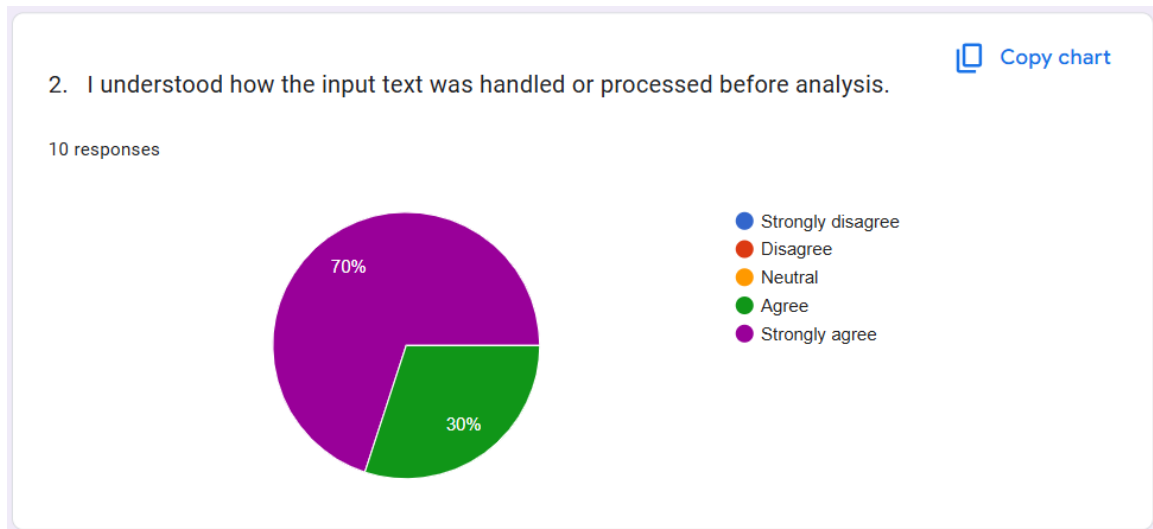


Figure 5. 18: Analyse Text usability scale.

Figure 5.18 indicates that 70% of respondents strongly agreed, and 30% agreed that they understood how the input text was handled or processed.

3. The analysis results were presented clearly and were easy to interpret.

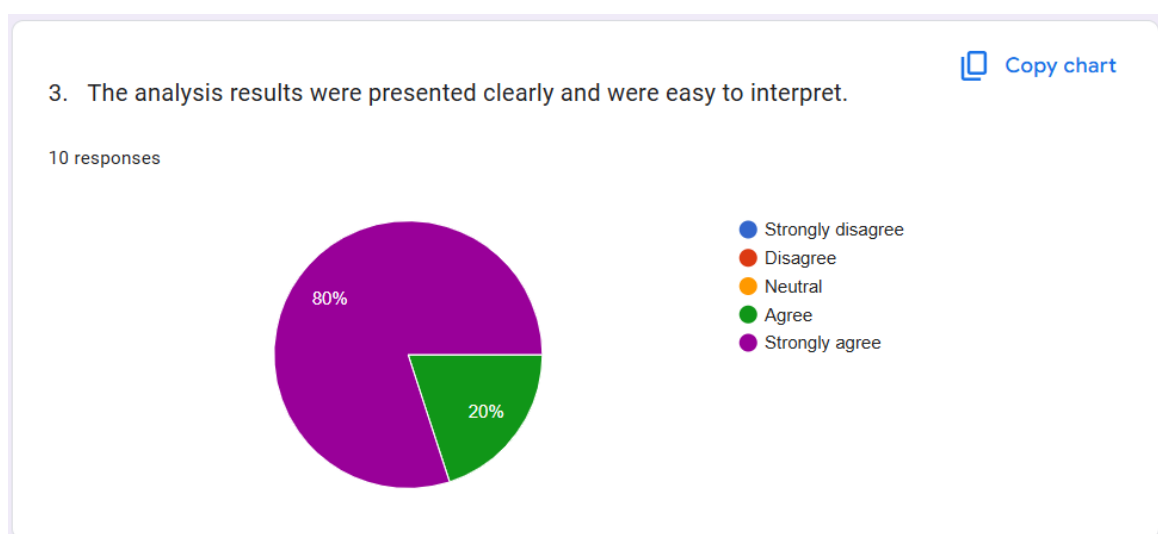


Figure 5. 19: Analyse Text usability scale.

Figure 5.19 displays that 80% of respondents strongly agreed, and 20% agreed that the analysis results were clear and easy to interpret.

4. The results were easy to download without any issues.

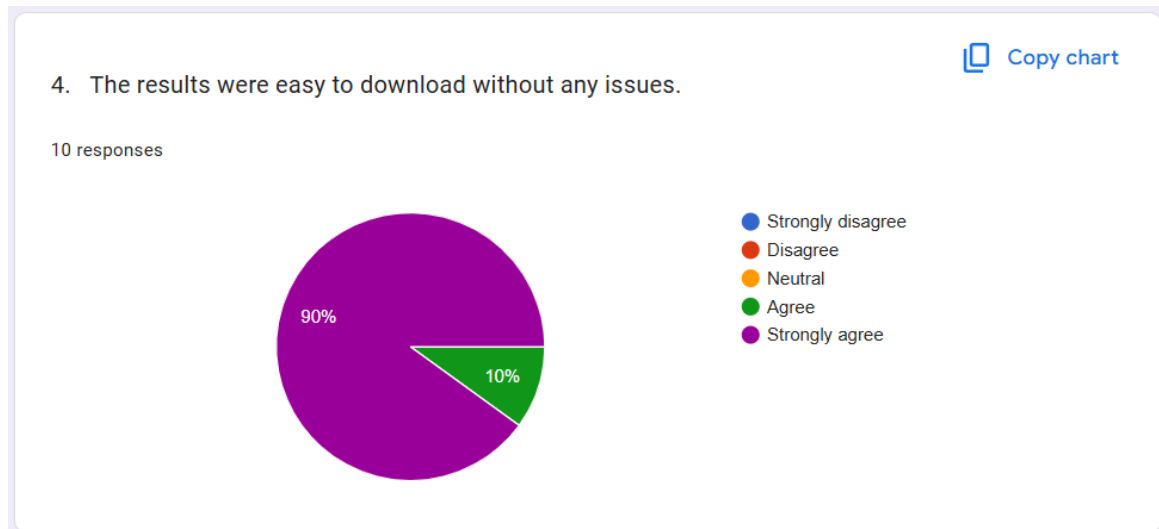


Figure 5. 20: Analyse Text usability scale.

Figure 5.20 presents feedback showing that 90% of respondents strongly agreed, and 10% agreed that downloading results was easy and without any issues.

5. I found the single-text analysis process to be effective and user-friendly.

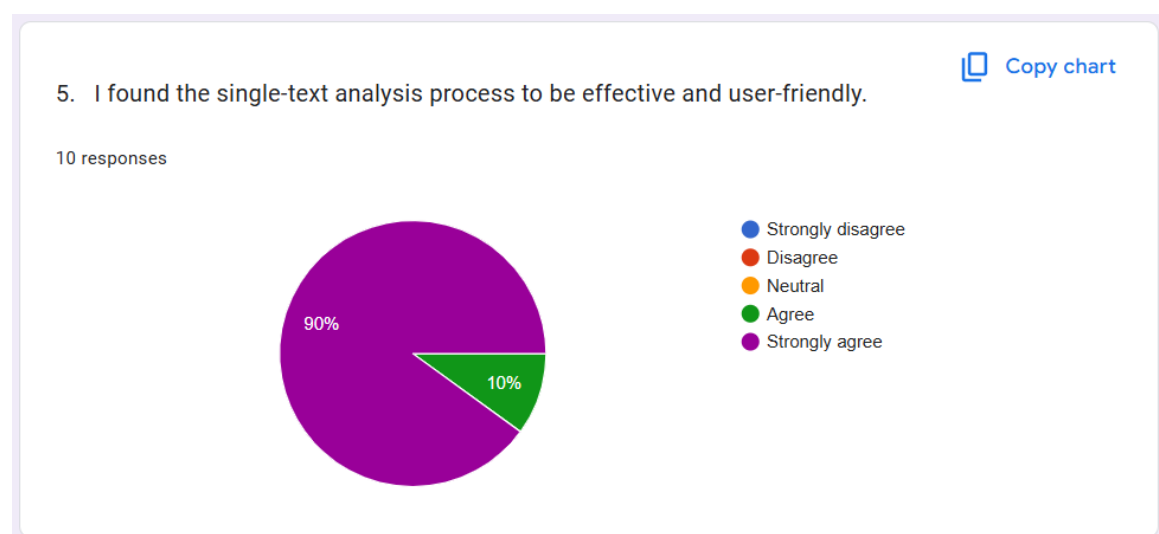


Figure 5. 21: Analyse Text usability scale.

Figure 5.21 shows that 90% of respondents strongly agreed, and 10% agreed that the single-text analysis process was effective and user-friendly.

5.7 Summary

This chapter describes how the testing phase for the web application is conducted. Functional testing ensures that the application operates correctly and produces the expected results. User Acceptance Testing (UAT) is then carried out to evaluate user satisfaction with the web application.

Chapter 6: Conclusion and future work

6.1 Introduction

This chapter outlines the achievement of objectives, the limitations and constraints encountered, and suggestions for future work, based on the findings during the implementation of the project.

6.2 Objectives achievement

Objective	Description
To implement data collection using web scraping techniques, followed by data cleaning and the application of topic modelling methods to identify and analyse topics in news articles.	Web scraping was implemented to collect business news articles from selected Malaysian news websites. The collected data underwent a data cleaning process, and various topic modelling techniques such as LDA, NMF, BERTopic, and Top2Vec were applied to extract and analyse relevant topics mentioned in Chapter 4.
To evaluate the performance of the topic modelling techniques for obtaining the best topic model on the Malaysia business news.	Through comparative evaluation of the implemented topic modelling techniques. Performance was measured using topic coherence scores, and NMF topic modelling was selected as the best topic model stated in Chapter 5.
To integrate the best topic model into a web-based system that presents topic news articles from different news article websites.	The selected topic model was successfully integrated into a web-based platform named News Analysis Hub, which allows users to analyse text or documents and view topic results with visualizations and downloadable outputs stated in Chapter 4 and 5.

6.3 Limitations and Constraint

One of the primary limitations of this research is restricted to business-related news articles. The system was designed and trained specifically to analyse topics within the business domain, particularly from Malaysian news websites. By focusing on business

news sourced from these sites, the models have learned vocabulary and topic distributions that are specific to this domain. As a result, the system’s performance and accuracy may not be as effective when applied to articles from other categories such as politics, health, or sports.

Another constraint is the topic labelling process. While the topic modelling algorithms used are effective in identifying clusters of related topics, it does not automatically generate descriptive labels for the topics. For example, after the algorithm identifies a cluster of keywords such as “oil,” “Petronas,” “gas,” and “energy,” human expertise is still needed to assign an appropriate label, such as “Oil and Gas”. This manual interpretation may lead to differences in how the results are labelled and understood, which can affect the clarity and consistency of the output.

6.4 Future works

The future work will be focusing on the collection of a larger and more domain expanding the system to support a wider range of news categories such as politics, health, technology, and sports. This will require retraining or testing the topic modelling algorithms on more diverse datasets to ensure accuracy and relevance across multiple domains. Furthermore, future work will focus on labelling keywords using automated approaches such as prompting with Large Language Models (LLMs) and WordNet. This approach will reduce the need for manual labelling, enhance the user experience, and improve the overall clarity of the output.

6.5 Summary

This chapter summarizes the objective achievements, limitations and constraints, and future works of the overall project.

References

- Adila, N. (2022). Implementation of Web Scraping for Journal Data Collection on the SINTA Website. *Sinkron*, 7(4). <https://doi.org/10.33395/sinkron.v7i4.11576>
- Ahmed, H., Jilani, T. A., Haider, W., Hasany, S. N., Abbasi, M. A., & Masroor, A. (2020). Producing standard rules for smart real estate property buying decisions based on web scraping technology and machine learning techniques. *International Journal of Advanced Computer Science and Applications*, 11(3). <https://doi.org/10.14569/ijacsa.2020.0110363>
- Alammary, A. S. (2022). BERT Models for Arabic Text Classification: A Systematic Review. In *Applied Sciences (Switzerland)* (Vol. 12, Issue 11). MDPI. <https://doi.org/10.3390/app12115720>
- Albalawi, R., Yeap, T. H., & Benyoucef, M. (2020). Using Topic Modeling Methods for Short-Text Data: A Comparative Analysis. *Frontiers in Artificial Intelligence*, 3. <https://doi.org/10.3389/frai.2020.00042>
- Albalawi, R., Yeap, T. H., & Benyoucef, M. (2020). Using Topic Modeling Methods for Short-Text Data: A Comparative Analysis. *Frontiers in Artificial Intelligence*, 3. <https://doi.org/10.3389/frai.2020.00042>
- Albalawi, R., Yeap, T. H., & Benyoucef, M. (2020). Using Topic Modeling Methods for Short-Text Data: A Comparative Analysis. *Frontiers in Artificial Intelligence*, 3. <https://doi.org/10.3389/frai.2020.00042>
- Amirifar, T., Lahmiri, S., & Zanjani, M. K. (2023). An NLP-Deep Learning Approach for Product Rating Prediction Based on Online Reviews and Product Features. *IEEE Transactions on Computational Social Systems*. <https://doi.org/10.1109/TCSS.2023.3290558>
- Amirifar, T., Lahmiri, S., & Zanjani, M. K. (2023). An NLP-Deep Learning Approach for Product Rating Prediction Based on Online Reviews and Product Features. *IEEE Transactions on Computational Social Systems*. <https://doi.org/10.1109/TCSS.2023.3290558>
- Angelov, D. (2020). *Top2Vec: Distributed Representations of Topics*. <http://arxiv.org/abs/2008.09470>
- Awangga, R. M., Pane, S. F., & Astuti, R. D. (2019). Implementation of web scraping on GitHub task monitoring system. *Telkomnika (Telecommunication Computing Electronics and Control)*, 17(1). <https://doi.org/10.12928/TELKOMNIKA.v17i1.11613>
- Banerjee, S., & Pan, A. (2024). From Colonial Legacies to Linguistic Inclusion: A BERTopic Enhanced Bibliometric Insight Into Global South Higher Education. *IEEE Access*, 12, 117418–117435. <https://doi.org/10.1109/ACCESS.2024.3447894>
- Chauhan, U., & Shah, A. (2022). Topic Modeling Using Latent Dirichlet allocation: A Survey. *ACM Computing Surveys*, 54(7). <https://doi.org/10.1145/3462478>

- Chauhan, U., & Shah, A. (2022). Topic Modeling Using Latent Dirichlet allocation: A Survey. *ACM Computing Surveys*, 54(7). <https://doi.org/10.1145/3462478>
- Chawla, S., Kaur, R., & Aggarwal, P. (2023). Text classification framework for short text based on TFIDF-FastText. *Multimedia Tools and Applications*, 82(26), 40167–40180. <https://doi.org/10.1007/s11042-023-15211-5>
- Domala, J., Dogra, M., Masrani, V., Fernandes, D., D’Souza, K., Fernandes, D., & Carvalho, T. (2020). Automated Identification of Disaster News for Crisis Management using Machine Learning and Natural Language Processing. *Proceedings of the International Conference on Electronics and Sustainable Communication Systems, ICESC 2020*. <https://doi.org/10.1109/ICESC48915.2020.9156031>
- Egger, R., & Yu, J. (2022). A Topic Modeling Comparison Between LDA, NMF, Top2Vec, and BERTopic to Demystify Twitter Posts. *Frontiers in Sociology*, 7. <https://doi.org/10.3389/fsoc.2022.886498>
- Eren, A. O., & Sert, M. (2023). Automated Audio Captioning With Topic Modeling. *IEEE Access*, 11, 4983–4991. <https://doi.org/10.1109/ACCESS.2023.3235733>
- Frank, E., Oluwaseyi, J., Olaoye, G. O., & Olaoye, G. (2024). *Data preprocessing techniques for NLP in BI*. <https://www.researchgate.net/publication/379652291>
- Gonzalez-Gomez, L. J., Hernandez-Munoz, S. M., Borja, A., Azofeifa, J. D., Noguez, J., & Caratozzolo, P. (2024). Analyzing Natural Language Processing Techniques to Extract Meaningful Information on Skills Acquisition from Textual Content. *IEEE Access*. <https://doi.org/10.1109/ACCESS.2024.3465409>
- Goyal, A., & Kashyap, I. (2023). Comprehensive Analysis of Topic Models for Short and Long Text Data. *International Journal of Advanced Computer Science and Applications*, 14(12). <https://doi.org/10.14569/IJACSA.2023.0141226>
- Grisales, A. M., Robledo, S., & Zuluaga, M. (2023). Topic Modeling: Perspectives From a Literature Review. In *IEEE Access* (Vol. 11). <https://doi.org/10.1109/ACCESS.2022.3232939>
- Grootendorst, M. (2022). *BERTopic: Neural topic modeling with a class-based TF-IDF procedure*. <http://arxiv.org/abs/2203.05794>
- Grybauskas, A., Pilinkienė, V., & Stundžienė, A. (2021). Predictive analytics using Big Data for the real estate market during the COVID-19 pandemic. *Journal of Big Data*, 8(1). <https://doi.org/10.1186/s40537-021-00476-0>
- Gurusamy, V., & Professor, A. (n.d.). *Preprocessing Techniques for Text Mining*.
- Hamamoto, R., Takasawa, K., MacHino, H., Kobayashi, K., Takahashi, S., Bolatkan, A., Shinkai, N., Sakai, A., Aoyama, R., Yamada, M., Asada, K., Komatsu, M., Okamoto, K., Kameoka, H., & Kaneko, S. (2022). Application of non-negative matrix factorization in oncology: One approach for establishing precision medicine. *Briefings in Bioinformatics*, 23(4). <https://doi.org/10.1093/bib/bbac246>

- Hendry, D., Darari, F., Nurfadillah, R., Khanna, G., Sun, M., Condylis, P. C., & Taufik, N. (2021). Topic Modeling for Customer Service Chats. *2021 International Conference on Advanced Computer Science and Information Systems, ICACSIS 2021*. <https://doi.org/10.1109/ICACSIS53237.2021.9631322>
- Jung, H. S., Lee, H., & Kim, J. H. (2023). Unveiling Cryptocurrency Conversations: Insights From Data Mining and Unsupervised Learning Across Multiple Platforms. *IEEE Access*, *11*, 130573–130583. <https://doi.org/10.1109/ACCESS.2023.3334617>
- Kadhim, A. (2018). An Evaluation of Preprocessing Techniques for Text Classification. In *Article in International Journal of Computer Science and Information Security*. <https://sites.google.com/site/ijcsis/>
- Kalra, V., Kashyap, I., & Kaur, H. (2022). Improving document classification using domain-specific vocabulary: hybridization of deep learning approach with TFIDF. *International Journal of Information Technology (Singapore)*, *14*(5), 2451–2457. <https://doi.org/10.1007/s41870-022-00889-x>
- Kastrati, Z., Imran, A. S., Daudpota, S. M., Memon, M. A., & Kastrati, M. (2023). Soaring Energy Prices: Understanding Public Engagement on Twitter Using Sentiment Analysis and Topic Modeling with Transformers. *IEEE Access*, *11*, 26541–26553. <https://doi.org/10.1109/ACCESS.2023.3257283>
- Kathiravan, P., & Haridoss, N. (2018). Preprocessing for Mining the Textual data-A Review. In *International Journal of Scientific Research in Computer Science Applications and Management Studies IJSRCSAMS* (Vol. 7, Issue 5). www.ijsrcsams.com
- Kherwa, P., & Bansal, P. (2020). Topic Modeling: A Comprehensive Review. *EAI Endorsed Transactions on Scalable Information Systems*, *7*(24). <https://doi.org/10.4108/eai.13-7-2018.159623>
- Kim, S. W., & Gil, J. M. (2019). Research paper classification systems based on TF-IDF and LDA schemes. *Human-Centric Computing and Information Sciences*, *9*(1). <https://doi.org/10.1186/s13673-019-0192-7>
- Kmgs, K., & Rahm, R. (2022). Learning to Use Normalization Techniques for Preprocessing and Classification of Text Documents. In *International Journal of Multidisciplinary Studies (IJMS)* (Vol. 9).
- Kumi, S., Snow, C., Lomotey, R. K., & Deters, R. (2024). Uncovering Concerns of Citizens Through Machine Learning and Social Network Sentiment Analysis. *IEEE Access*, *12*, 94885–94913. <https://doi.org/10.1109/ACCESS.2024.3426329>
- Mariam Khader, Arafat Awajan, & Ghazi Al-Naymat. (2018). *The Effects of Natural Language Processing on Big Data Analysis: Sentiment Analysis Case Study*.
- Matta Priya, Sharma Nikita, Sharma Devyani, Pant Bhasker, & Sharma Sachin. (2020). Web Scraping: Applications and Scraping Tools. *International Journal of*

Advanced Trends in Computer Science and Engineering, 9(5).
<https://doi.org/10.30534/ijatcse/2020/185952020>

Mihajlov, T., Nešić, M. I., Stanković, R., & Kitanović, O. (2024). *Topic Modeling of the SrpELTeC Corpus: A Comparison of NMF, LDA, and BERTopic*. 649–653.
<https://doi.org/10.15439/2024F1593>

Mohammadi, S., & Chapon, M. (2020). Investigating the Performance of Fine-Tuned Text Classification Models Based-on Bert. *Proceedings - 2020 IEEE 22nd International Conference on High Performance Computing and Communications, IEEE 18th International Conference on Smart City and IEEE 6th International Conference on Data Science and Systems, HPCC-SmartCity-DSS 2020*, 1252–1257. <https://doi.org/10.1109/HPCC-SmartCity-DSS50907.2020.00162>

Mothe, J., Son, L. H., Nguyen, T. Q. V., Đại học Đà Nẵng. University of Science and Technology, Quỹ phát triển khoa học công nghệ quốc gia (Vietnam), IEEE Vietnam Section., & Institute of Electrical and Electronics Engineers. (n.d.). *Proceedings of 2019 11th International Conference On Knowledge And Systems Engineering : KSE 2019 : October 24-26, 2019, Da Nang, Vietnam*.

Mridha, M. F., Keya, A. J., Hamid, M. A., Monowar, M. M., & Rahman, M. S. (2021). A Comprehensive Review on Fake News Detection with Deep Learning. *IEEE Access*, 9, 156151–156170. <https://doi.org/10.1109/ACCESS.2021.3129329>

Omar, M., Choi, S., Nyang, D., & Mohaisen, D. (2022). Robust Natural Language Processing: Recent Advances, Challenges, and Future Directions. *IEEE Access*, 10, 86038–86056. <https://doi.org/10.1109/ACCESS.2022.3197769>

Papadia, G., Pacella, M., Perrone, M., & Giliberti, V. (2023). A Comparison of Different Topic Modeling Methods through a Real Case Study of Italian Customer Care. *Algorithms*, 16(2). <https://doi.org/10.3390/a16020094>

Rita, P., Vong, C., Pinheiro, F., & Mimoso, J. (2023). A sentiment analysis of Michelin-starred restaurants. *European Journal of Management and Business Economics*, 32(3). <https://doi.org/10.1108/EJMBE-11-2021-0295>

Sibitenda, H., Diattara, A., Traore, A., Hu, R., Zhang, D., Rundensteiner, E., & Ba, C. (2024). Extracting Semantic Topics About Development in Africa From Social Media. *IEEE Access*, 12, 142343–142359.
<https://doi.org/10.1109/ACCESS.2024.3466834>

Srivastava, R., Bharti, P. K., & Verma, P. (2022). Comparative Analysis of Lexicon and Machine Learning Approach for Sentiment Analysis. In *IJACSA International Journal of Advanced Computer Science and Applications* (Vol. 13, Issue 3). www.ijacsa.thesai.org

Sukmandhani, A. A., Sunjaya, T., Saputro, I. P., & Ohliati, J. (2023). Data Scraping using Python for Information Retrieval on E-Commerce with Brand Keyword. *2023 8th International Conference on Business and Industrial Research, ICBIR 2023 - Proceedings*, 179–183. <https://doi.org/10.1109/ICBIR57571.2023.10147717>

- Supriyono, Wibawa, A. P., Suyono, & Kurniawan, F. (2024). A survey of text summarization: Techniques, evaluation and challenges. *Natural Language Processing Journal*, 7, 100070. <https://doi.org/10.1016/j.nlp.2024.100070>
- Tabassum, A., & Patil, R. R. (2020). A Survey on Text Pre-Processing & Feature Extraction Techniques in Natural Language Processing. *International Research Journal of Engineering and Technology*. www.irjet.net
- Ullah, H., Ullah, Z., Maqsood, S., & Hafeez, A. (2018). Web scraper revealing trends of target products and new insights in online shopping websites. *International Journal of Advanced Computer Science and Applications*, 9(6). <https://doi.org/10.14569/IJACSA.2018.090658>
- Umamaheswaran, S., Dar, V., Sharma, E., & Kurian, J. S. (2023a). Mapping Climate Themes from 2008-2021 - An Analysis of Business News Using Topic Models. *IEEE Access*, 11, 26554–26565. <https://doi.org/10.1109/ACCESS.2023.3256530>
- Umamaheswaran, S., Dar, V., Sharma, E., & Kurian, J. S. (2023b). Mapping Climate Themes from 2008-2021 - An Analysis of Business News Using Topic Models. *IEEE Access*, 11, 26554–26565. <https://doi.org/10.1109/ACCESS.2023.3256530>
- Uwiragiye, E., & Rhinehardt, K. L. (2022). TFIDF-Random Forest: Prediction of Aptamer-Protein Interacting Pairs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(5), 3032–3037. <https://doi.org/10.1109/TCBB.2021.3098709>
- Wani, M. A., Agarwal, N., & Bours, P. (2021). Impact of unreliable content on social media users during COVID-19 and stance detection system. *Electronics (Switzerland)*, 10(1). <https://doi.org/10.3390/electronics10010005>
- Zhou, Z., & Wakabayashi, K. (n.d.). *DEIM2022 E41-3(day4 p10) Topic Modeling using Jointly Fine-tuned BERT for Phrases and Sentences*.

APPENDIX A: SOURCE CODE

Data collection or Web Scraping

1. Business today

```
import requests
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from webdriver_manager.chrome import ChromeDriverManager
import time
import pandas as pd

def scrape_content(url):
    """Scrapes content from a given URL.

    Args:
        url (str): The URL of the page to scrape.

    Returns:
        list: A list containing the scraped content from the specified URL.
    """

    try:
        driver.get(url)
        time.sleep(5) # Adjust sleep time as needed to prevent overwhelming the
        website

        html = driver.page_source
        soup = BeautifulSoup(html, "html.parser")

        # Extract content based on your desired structure (replace placeholders with
        actual selectors)
```

```

    page_title = soup.select_one('h1.entry-title') # Assuming the title is within an
h2 tag

    content_divs = soup.select("div.dable-content-wrapper") # Assuming content is
in divs with this class


    if page_title and content_divs:

        return page_title.text.strip() + "\n" + content_divs[0].text.strip() # Assuming
first div contains desired content

    else:

        return None # Handle cases where title or content elements are not found


except Exception as e:

    print(f'Error scraping content from {url}: {e}')

    return None


def scrape_links_and_content():

    """Scrapes links and content from all pages in the category.

    Returns:

    pd.DataFrame: A DataFrame containing the scraped content and corresponding
URLs.

    """

    all_links = []
    content = []

    for page_num in range(96, 102): # Adjust the range as needed

        print("Processing page", page_num)

        url =
f'https://www.businesstoday.com.my/category/news/business/page/{page_num}'

        driver.get(url)

        time.sleep(5) # Adjust sleep time as needed

```

```

html = driver.page_source
soup = BeautifulSoup(html, "html.parser")

# Extract links from the current page
category_links = [a["href"] for a in soup.select("div.td-image-container a")]

# Filter links for the year
year_links = [link for link in category_links if f'/{2019}/' in link]

# Check if any links were found after filtering
if not year_links:
    print(f"No relevant links found on page {page_num}. Stopping scraping.")
    break # Stop scraping if no relevant links found

# Scrape content from each link
for link in year_links:
    scraped_content = scrape_content(link)
    if scraped_content:
        content.append(scraped_content)
        all_links.append(link)

df = pd.DataFrame(data={"Content": content, "URL": all_links})
return df

if __name__ == "__main__":
    cService = webdriver.ChromeService(ChromeDriverManager().install())
    driver = webdriver.Chrome(service=cService)

    try:
        df = scrape_links_and_content()
        df.to_csv('businesstoday_2019(1).csv', index=False)

```

```
print("Scraping complete.")
except Exception as e:
    print(f'An error occurred during scraping: {e}')
finally:
    driver.quit()
```

2. Daily express

```
import requests
from bs4 import BeautifulSoup
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager
import time
import pandas as pd

cService = webdriver.ChromeService(ChromeDriverManager().install())
driver = webdriver.Chrome(service=cService)

def scrape_content(url):
    """Scrapes content from a given URL.

    Args:
        url (str): The URL of the page to scrape.
        driver (webdriver.Chrome): The WebDriver object.

    Returns:
        tuple: A tuple containing the scraped title, date, and content from the specified URL.
    """
    try:
        driver.get(url)

        time.sleep(5) # Adjust sleep time as needed to prevent overwhelming the
        website
```

```

html = driver.page_source
soup = BeautifulSoup(html, "html.parser")

# Extract content based on your desired structure (replace placeholders with
actual selectors)

page_title = soup.select_one('div.title.titlemobile') # Assuming the title is within
a div with class 'title.titlemobile'

article_date = soup.select_one('div[class*="col-xs-12 col-sm-12 col-md-6 col-lg-6
col-xl-6"]') # Adjust the selector accordingly

content_divs = soup.select("span.newsContent") # Assuming content is in spans
with class 'newsContent'

if page_title and content_divs:
    title_text = page_title.text.strip()
    date_text = article_date.text.strip() if article_date else "" # Extract date text
if available
    content_text = '\n'.join([div.text.strip() for div in content_divs])
    return title_text, date_text, content_text # Return a tuple containing title,
date, and content
else:
    return None, None, None # Handle cases where title or content elements are
not found

except Exception as e:
    print(f"Error scraping content from {url}: {e}")
    return None, None, None

def scrape_links_and_content():
    """Scrapes links, content, and date from all pages in the category.

Returns:
pd.DataFrame: A DataFrame containing the scraped content, URLs, and dates.
    """

```

```

all_links = []
all_dates = []
all_content = []

try:
    for page_num in range(1, 2): # Change according to the page
        print("Processing page", page_num)
        url = f"https://www.dailyexpress.com.my/business/?pageNo={page_num}"
        driver.get(url)
        time.sleep(5) # Adjust sleep time as needed

        html = driver.page_source
        soup = BeautifulSoup(html, "html.parser")

        # Extract links from the current page
        category_links = [a['href'] for a in soup.select("div.title a")]

        # Extract newstag elements from the current page
        newstag_texts = [tag.text for tag in soup.select("div.newstag")]

        # Filter links for the year
        year_links = [link for link, newstag_text in zip(category_links, newstag_texts)
if "2024" in newstag_text]

        # Check if any links were found after filtering
        if not year_links:
            print(f"No relevant links found on page {page_num}. Skipping scraping.")
            continue # Skip to the next page

        # Scrape content, date, and URL from each link
        for link in year_links:

```

```

        full_url = f"https://www.dailyexpress.com.my{link}" # Add base URL to the
relative link

        title, date, content = scrape_content(full_url)

        if title and date and content:

            all_links.append(full_url)

            all_dates.append(date)

            all_content.append(content)

    except Exception as e:

        print(f'An error occurred during scraping: {e}')

    finally:

        driver.quit()

    df = pd.DataFrame(data={"Date": all_dates, "Content": all_content, "URL":
all_links})

    return df

if __name__ == "__main__":

    try:

        df = scrape_links_and_content()

        df.to_csv('dailyexpress_2024(test).csv', index=False)

        print("Scraping complete.")

    except Exception as e:

        print(f'An error occurred during scraping: {e}')

```

3. Malaysia reserve

```

import requests

from bs4 import BeautifulSoup

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

from webdriver_manager.chrome import ChromeDriverManager

import time

```

```

import pandas as pd

def scrape_content(url):
    """Scrapes content from a given URL.

    Args:
        url (str): The URL of the page to scrape.

    Returns:
        list: A list containing the scraped content from the specified URL.
    """

    try:
        driver.get(url)

        time.sleep(5) # Adjust sleep time as needed to prevent overwhelming the
        website

        html = driver.page_source
        soup = BeautifulSoup(html, "html.parser")

        # Extract content based on your desired structure (replace placeholders with
        actual selectors)

        page_title = soup.select_one('h2') # Assuming the title is within an h2 tag
        content_divs = soup.select("div.dable-content-wrapper") # Assuming content is
        in divs with this class

        if page_title and content_divs:
            return page_title.text.strip() + "\n" + content_divs[0].text.strip() # Assuming
            first div contains desired content
        else:
            return None # Handle cases where title or content elements are not found

    except Exception as e:

```

```

    print(f"Error scraping content from {url}: {e}")
    return None

def scrape_links_and_content():
    """Scrapes links and content from all pages in the category.

    Returns:
        pd.DataFrame: A DataFrame containing the scraped content and corresponding
        URLs.
    """

    all_links = []
    content = []

    for page_num in range(740,750):    #change according to the page
        print("Processing page", page_num)
        # page_num += 1
        url = f"https://themalaysianreserve.com/category/business/page/{page_num}"
        driver.get(url)
        time.sleep(5) # Adjust sleep time as needed

        html = driver.page_source
        soup = BeautifulSoup(html, "html.parser")

        # Extract links from the current page
        category_links = [a['href'] for a in soup.select("div.col-xl-4.col-lg-6.col-md-12.col-sm-6 a")]

        # Filter links for the year
        year_links = [link for link in category_links if f"/{2019}/" in link]

        # Check if any links were found after filtering

```

```

if not year_links:
    print(f"No relevant links found on page {page_num}. Skipping scraping.")
    continue # Skip to the next page

# Scrape content from each link
for link in year_links:
    scraped_content = scrape_content(link)
    if scraped_content:
        content.append(scraped_content)
        all_links.append(link)

# Check for next page link (replace with your selector if different)
next_page_button = soup.select_one("div.col-auto a")

if not next_page_button:
    break

page_num += 1

df = pd.DataFrame(data={"Content": content, "URL": all_links})
return df

if __name__ == "__main__":
    cService = webdriver.ChromeService(ChromeDriverManager().install())
    driver = webdriver.Chrome(service=cService)

    try:
        df = scrape_links_and_content()
        df.to_csv('malaysiareserve_2019(14).csv', index=False)
        print("Scraping complete.")
    except Exception as e:

```

```
print(f'An error occurred during scraping: {e}')  
  
finally:  
    driver.quit()
```

4. New Strait Times

```
import requests  
from bs4 import BeautifulSoup  
from selenium import webdriver  
from selenium.webdriver.common.keys import Keys  
from webdriver_manager.chrome import ChromeDriverManager  
import time  
import pandas as pd  
  
cService = webdriver.ChromeService(ChromeDriverManager().install())  
driver = webdriver.Chrome(service = cService)  
category = ["business"]  
  
pages = []  
last_page_scraped = 0 # Initialize the variable to store the last successfully scraped  
page  
  
for catLink in category:  
    url = "https://www.nst.com.my/"  
    fullLink = url + catLink + '?page='  
    #to get the links in one page  
    for p in range(1041, 1073): # Start from the last successfully scraped page  
        link = fullLink + str(p)  
        driver.get(link)  
        time.sleep(5)  
        print("Scraping page:", p)
```

```

html = driver.page_source
soup = BeautifulSoup(html, "html.parser")
all_divs = soup.select("a.d-flex.article.listing.mb-3.pb-3")
if all_divs:
    for i in all_divs:
        tempUrl = i['href'].split('/') + catLink + '/'[1][0:4]
        if "2018" <= tempUrl <= "2023": # Check if the year is between 2018 and
2023
            pages.append(i['href'])
    else:
        print("No articles found on page", p)
        break

last_page_scraped = p # Update the last successfully scraped page

print("Total number of pages scraped:", len(pages))

# to get content and title
content = []
for page in pages:
    link = 'https://www.nst.com.my/business' + page
    driver.get(link)
    time.sleep(5)
    html = driver.page_source
    soup = BeautifulSoup(html, "html.parser")
    all_divs = soup.select("div.dable-content-wrapper")
    page_title = soup.select('h1.page-title.mb-2')
    if page_title and all_divs: # Ensure both title and content are present before
appending
        content.append(page_title[0].text + "\n" + all_divs[0].text)
        print(page_title[0].text + "\n" + all_divs[0].text)
    else:

```

```
print("Skipping:", link)

df = pd.DataFrame(pages, columns=['URL'])
df['Content'] = content
df.to_csv('nst(49).csv')
```

Data preprocessing

```
import os
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import re
from collections import Counter
import time

def preprocess_text(text, lemmatizer):
    text = text.lower()
    text = re.sub(r'^\x00-\x7F'+', ', text) # Remove non-ASCII characters
    text = re.sub(r'^\w\s]', ", ", text) # Remove punctuation
    text = re.sub(r'\d+', ", ", text) # Remove digits
    tokens = text.split()
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    return tokens

if __name__ == '__main__':
    # Make sure to download NLTK stopwords and WordNetLemmatizer
    nltk.download('stopwords')
    nltk.download('wordnet')
```

```

file_path = 'Master List 2022.csv' # Update with your file path

# Check if file exists
if not os.path.exists(file_path):
    raise FileNotFoundError(f'File {file_path} not found.')

data = pd.read_csv(file_path)

# Display the first few rows of the dataframe
print(data.head())

lemmatizer = WordNetLemmatizer()

# Use the column name for article content
data['date'] = pd.to_datetime(data['date'], errors='coerce')

# Drop rows where date parsing failed (NaT values)
data = data.dropna(subset=['date'])

# Separate data into training (January to September) and testing (October to
December) sets
training_data = data[(data['date'].dt.month >= 1) & (data['date'].dt.month <= 9)]
testing_data = data[(data['date'].dt.month >= 10) & (data['date'].dt.month <= 12)]

# Preprocess text for training and testing data
start_time = time.time()

training_data['processed_text'] = training_data['article_content'].apply(lambda x:
preprocess_text(str(x), lemmatizer))

testing_data['processed_text'] = testing_data['article_content'].apply(lambda x:
preprocess_text(str(x), lemmatizer))

end_time = time.time()

```

```

print(f'Preprocessing text took {end_time - start_time:.2f} seconds.')

training_texts = training_data['processed_text'].tolist()
testing_texts = testing_data['processed_text'].tolist()

# Identify rare and frequent words based on training set
start_time = time.time()
all_tokens = [word for text in training_texts for word in text]
rare_words = {word for word in set(all_tokens) if all_tokens.count(word) < 15}

doc_count = len(training_texts)
word_doc_frequency = Counter([word for text in training_texts for word in set(text)])
frequent_words = {word for word, count in word_doc_frequency.items() if count > doc_count * 0.5}

remove_words = rare_words.union(frequent_words)

# Remove identified words from both training and testing sets
training_texts = [[word for word in text if word not in remove_words] for text in training_texts]
testing_texts = [[word for word in text if word not in remove_words] for text in testing_texts]

end_time = time.time()
print(f'Removing rare and frequent words took {end_time - start_time:.2f} seconds.')

# Compile word frequency dictionary for training data
start_time = time.time()
word_freq_training = Counter([word for text in training_texts for word in text])

word_freq_df_training = pd.DataFrame(word_freq_training.items(),
columns=['Word', 'Frequency'])

word_freq_df_training = word_freq_df_training.sort_values(by='Frequency',
ascending=False)

```

```

word_freq_file_path_training = 'Word_Frequencies_Training_Set_2022.csv'
word_freq_df_training.to_csv(word_freq_file_path_training, index=False)
end_time = time.time()
print(f'Saving word frequencies (training) took {end_time - start_time:.2f} seconds.')
print(f'Word frequencies saved to {word_freq_file_path_training}')

# Compile word frequency dictionary for testing data
start_time = time.time()
word_freq_testing = Counter([word for text in testing_texts for word in text])

word_freq_df_testing = pd.DataFrame(word_freq_testing.items(), columns=['Word',
'Frequency'])
word_freq_df_testing = word_freq_df_testing.sort_values(by='Frequency',
ascending=False)
word_freq_file_path_testing = 'Word_Frequencies_Testing_Set_2022.csv'
word_freq_df_testing.to_csv(word_freq_file_path_testing, index=False)
end_time = time.time()
print(f'Saving word frequencies (testing) took {end_time - start_time:.2f} seconds.')
print(f'Word frequencies saved to {word_freq_file_path_testing}')

# Save the processed text to a new CSV file for training and testing data
start_time = time.time()
training_data['processed_text_str'] = training_texts
training_data['processed_text_str'] = training_data['processed_text_str'].apply(lambda x: ' '.join(x))

testing_data['processed_text_str'] = testing_texts
testing_data['processed_text_str'] = testing_data['processed_text_str'].apply(lambda
x: ' '.join(x))

training_processed_file_path = 'Processed_Training_Set_2022.csv'
testing_processed_file_path = 'Processed_Testing_Set_2022.csv'

```

```
training_data.to_csv(training_processed_file_path, index=False)
testing_data.to_csv(testing_processed_file_path, index=False)

end_time = time.time()

print(f'Saving processed text took {end_time - start_time:.2f} seconds.')

    print(f'Processed data saved to {training_processed_file_path} and
{testing_processed_file_path}')
```

Topic modelling

1. LDA

i Train

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from gensim.models.coherencemodel import CoherenceModel
from gensim.corpora.dictionary import Dictionary
from wordcloud import WordCloud
import time
import pickle

# Create a directory to save plots
output_dir = 'plots'
os.makedirs(output_dir, exist_ok=True)

def calculate_coherence_scores(texts, count, count_vectorizer, max_topics, step=1):
    coherence_scores = []
    for n_topics in range(2, max_topics + 1, step):
```

```

start_time = time.time()

lda_model = LatentDirichletAllocation(n_components=n_topics, random_state=42)
lda_topic_matrix = lda_model.fit_transform(count)

topics = []
feature_names = count_vectorizer.get_feature_names_out()
for topic_idx, topic in enumerate(lda_model.components_):
    topics.append([feature_names[i] for i in topic.argsort()[:-10:-1]])

dictionary = Dictionary(texts)
corpus = [dictionary.doc2bow(text) for text in texts]

cm = CoherenceModel(topics=topics, texts=texts, dictionary=dictionary,
coherence='c_v')

coherence_scores.append((n_topics, cm.get_coherence()))

end_time = time.time()

print(f'Calculating coherence for {n_topics} topics took {end_time - start_time:.2f}
seconds.')

return coherence_scores

def plot_word_clouds(topics, feature_names):
    for i, topic in enumerate(topics):
        plt.figure(figsize=(8, 8))

        topic_words = {feature_names[word_index]: topic[word_index] for word_index in
topic.argsort()[-20:][::-1]}

        wordcloud = WordCloud(width=800, height=800, background_color='white',
max_font_size=100).generate_from_frequencies(topic_words)

        plt.imshow(wordcloud, interpolation='bilinear')

        plt.title(f'Topic {i + 1}', fontsize=16)

        plt.axis('off')

```

```

plt.savefig(os.path.join(output_dir, f'topic_{i+1}_wordcloud.png')) # Save each word
cloud

plt.close() # Close the figure to avoid display
print(f'Word cloud for Topic {i+1} saved.')

def plot_top_words(model, feature_names, n_top_words, title):
    n_topics = model.components_.shape[0]

    for topic_idx, topic in enumerate(model.components_):
        plt.figure(figsize=(10, 5))
        top_features_ind = topic.argsort()[-n_top_words:][::-1]
        top_features = [feature_names[i] for i in top_features_ind]
        weights = topic[top_features_ind]

        plt.barh(top_features, weights, height=0.7)
        plt.title(f'Topic {topic_idx + 1}', fontsize=14)
        plt.xlabel('Weight')
        plt.ylabel('Feature')
        plt.gca().invert_yaxis()
        plt.xticks(fontsize=10)
        plt.yticks(fontsize=10)
        plt.grid(axis='x', linestyle='--', alpha=0.7)

        plt.savefig(os.path.join(output_dir, f'topic_{topic_idx+1}_top_words.png')) # Save
each top words plot

        plt.close() # Close the figure to avoid display
        print(f'Top words for Topic {topic_idx+1} saved.')

# Function to save the LDA model
def save_lda_model(lda_model, filename='lda_model_2023.pkl'):
    with open(filename, 'wb') as f:
        pickle.dump(lda_model, f)
    print(f'LDA model saved to {filename}')

```

```

## Function to load the LDA model
def load_lda_model(filename='lda_model.pkl'):
    with open(filename, 'rb') as f:
        lda_model = pickle.load(f)
    print(f'LDA model loaded from {filename}')
    return lda_model

# Function to save the top words to a CSV file
def save_top_words_to_csv(model, feature_names, n_top_words,
filename='top_words_2023.csv'):
    top_words_data = []
    for topic_idx, topic in enumerate(model.components_):
        top_features_ind = topic.argsort()[-n_top_words:][::-1]
        top_features = [feature_names[i] for i in top_features_ind]
        weights = topic[top_features_ind]

        # Append the topic and corresponding words to the list
        for word, weight in zip(top_features, weights):
            top_words_data.append([topic_idx + 1, word, weight])

    # Convert the list into a pandas DataFrame
    top_words_df = pd.DataFrame(top_words_data, columns=['Topic', 'Word', 'Weight'])

    # Save the DataFrame to a CSV file
    top_words_df.to_csv(filename, index=False)
    print(f'Top words for each topic saved to {filename}')

def train_lda_model(training_texts, count_vectorizer, max_topics):
    # Vectorization for training data
    start_time = time.time()

```

```

count_train = count_vectorizer.fit_transform([' '.join(text) for text in training_texts])
end_time = time.time()
print(f'Vectorization took {end_time - start_time:.2f} seconds.')

# Calculate coherence scores for topics ranging from 2 to max_topics
start_time = time.time()
coherence_scores = calculate_coherence_scores(training_texts, count_train,
count_vectorizer, max_topics)
end_time = time.time()
print(f'Coherence scores calculation took {end_time - start_time:.2f} seconds.')

# Find the number of topics with the highest coherence score
optimal_topics, optimal_score = max(coherence_scores, key=lambda x: x[1])
print(f'Optimal number of topics: {optimal_topics}')
print(f'Coherence score for the optimal number of topics: {optimal_score:.4f}') #
Display the exact coherence score

# Optionally plot the coherence scores
topics, scores = zip(*coherence_scores)
plt.plot(topics, scores)
plt.xlabel('Number of Topics')
plt.ylabel('Coherence Score')
plt.title('Coherence Scores by Number of Topics')
plt.savefig(os.path.join(output_dir, 'coherence_scores.png')) # Save the coherence
scores plot
plt.close()
print("Coherence scores plot saved.")

# Train LDA model with the optimal number of topics on the training data
start_time = time.time()
lda_model = LatentDirichletAllocation(n_components=optimal_topics,
random_state=42)
lda_topic_matrix_train = lda_model.fit_transform(count_train)

```

```

end_time = time.time()

print(f'Training LDA model took {end_time - start_time:.2f} seconds.')

# Plot word clouds and top words for training data
plot_word_clouds(lda_model.components_,
count_vectorizer.get_feature_names_out())

plot_top_words(lda_model, count_vectorizer.get_feature_names_out(), 20, f'Top
words per topic for LDA model with {optimal_topics} topics')

# Save top 20 words for each topic to CSV
save_top_words_to_csv(lda_model, count_vectorizer.get_feature_names_out(), 20,
'top_words_2023.csv')

return lda_model, lda_topic_matrix_train, count_vectorizer

if __name__ == '__main__':
    file_path = 'Processed_Training_Data_2023.csv' # Update with your processed
training data file path

# Check if file exists
if not os.path.exists(file_path):
    raise FileNotFoundError(f'File {file_path} not found.')

training_data = pd.read_csv(file_path)

# Display the first few rows of the dataframe
print(training_data.head())

# Handle missing or non-string values in the 'processed_text_str' column
# Replace NaN or non-string entries with empty strings
training_data['processed_text_str'] = training_data['processed_text_str'].fillna("")

# Extract processed text from dataframe

```

```

training_texts = [str(text).split() for text in training_data['processed_text_str']]

# Vectorization and model training
max_topics = 20

count_vectorizer = CountVectorizer(max_df=0.95, min_df=2, stop_words='english',
max_features=20000)

start_time = time.time()

lda_model, lda_topic_matrix_train, count_vectorizer =
train_lda_model(training_texts, count_vectorizer, max_topics)

end_time = time.time()

print(f'Total execution time for training LDA model: {end_time - start_time:.2f}
seconds.')

# Save the trained LDA model to a file
save_lda_model(lda_model, 'lda_model_2023.pkl')

# Save the LDA topic matrix as an .npy file
np.save('lda_topic_matrix_train_2023.npy', lda_topic_matrix_train)
print("LDA topic matrix saved as lda_topic_matrix_train.npy")

# Save the CountVectorizer as a .pkl file
with open('count_vectorizer_2023.pkl', 'wb') as f:
    pickle.dump(count_vectorizer, f)
print("CountVectorizer saved as count_vectorizer.pkl")

```

ii Test

```

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation

```

```

from gensim.models.coherencemodel import CoherenceModel
from gensim.corpora.dictionary import Dictionary
from wordcloud import WordCloud
import time
import pickle

# Create a directory to save plots
output_dir = 'plots'
os.makedirs(output_dir, exist_ok=True)

# Function to load the LDA model
def load_lda_model(filename='lda_model_2019.pkl'):
    with open(filename, 'rb') as f:
        lda_model = pickle.load(f)
    print(f'LDA model loaded from {filename}')
    return lda_model

# Function to load the CountVectorizer
def load_count_vectorizer(filename='count_vectorizer_2019.pkl'):
    with open(filename, 'rb') as f:
        count_vectorizer = pickle.load(f)
    print(f'CountVectorizer loaded from {filename}')
    return count_vectorizer

# Function to calculate and plot coherence scores for test data
def calculate_coherence_for_test(test_texts, lda_model, count_vectorizer, output_dir):
    # Transform the test data using the trained CountVectorizer
    count_test = count_vectorizer.transform([' '.join(text) for text in test_texts])

    # Generate topic distribution for test data
    lda_topic_matrix_test = lda_model.transform(count_test)

```

```

# Extract topics
topics = []
feature_names = count_vectorizer.get_feature_names_out()
for topic_idx, topic in enumerate(lda_model.components_):
    topics.append([feature_names[i] for i in topic.argsort()[: -10: -1]])

# Calculate coherence score for the test set
dictionary = Dictionary(test_texts)
corpus = [dictionary.doc2bow(text) for text in test_texts]
cm = CoherenceModel(topics=topics, texts=test_texts, dictionary=dictionary,
coherence='c_v')
coherence_score = cm.get_coherence()

print(f"Coherence score for test set: {coherence_score:.4f}")

# Save the test topic matrix
np.save(os.path.join(output_dir, 'lda_topic_matrix_test_2019.npy'),
lda_topic_matrix_test)
print("LDA topic matrix for test data saved as lda_topic_matrix_test_2019.npy")

return coherence_score, lda_topic_matrix_test

# Function to plot word clouds
def plot_word_clouds(topics, feature_names):
    for i, topic in enumerate(topics):
        plt.figure(figsize=(8, 8))

        topic_words = {feature_names[word_index]: topic[word_index] for word_index in
topic.argsort()[-20:][::-1]}

        wordcloud = WordCloud(width=800, height=800, background_color='white',
max_font_size=100).generate_from_frequencies(topic_words)

        plt.imshow(wordcloud, interpolation='bilinear')
        plt.title(f'Topic {i + 1}', fontsize=16)

```

```

plt.axis('off')
plt.savefig(os.path.join(output_dir, f'topic_{i+1}_wordcloud.png')) # Save each word
cloud

plt.close()

print(f"Word cloud for Topic {i+1} saved.")

# Function to plot top words for each topic
def plot_top_words(model, feature_names, n_top_words, title):
    n_topics = model.components_.shape[0]

    for topic_idx, topic in enumerate(model.components_):
        plt.figure(figsize=(10, 5))
        top_features_ind = topic.argsort()[-n_top_words:][::-1]
        top_features = [feature_names[i] for i in top_features_ind]
        weights = topic[top_features_ind]

        plt.barh(top_features, weights, height=0.7)
        plt.title(f'Topic {topic_idx + 1}', fontsize=14)
        plt.xlabel('Weight')
        plt.ylabel('Feature')
        plt.gca().invert_yaxis()
        plt.xticks(fontsize=10)
        plt.yticks(fontsize=10)
        plt.grid(axis='x', linestyle='--', alpha=0.7)

        plt.savefig(os.path.join(output_dir, f'topic_{topic_idx+1}_top_words.png')) # Save
each top words plot

        plt.close()

        print(f"Top words for Topic {topic_idx+1} saved.")

# Function to save the top words to a CSV file
def save_top_words_to_csv(model, feature_names, n_top_words,
filename='top_words_test_2019.csv'):

```

```

top_words_data = []
for topic_idx, topic in enumerate(model.components_):
    top_features_ind = topic.argsort()[-n_top_words:][::-1]
    top_features = [feature_names[i] for i in top_features_ind]
    weights = topic[top_features_ind]

    # Append the topic and corresponding words to the list
    for word, weight in zip(top_features, weights):
        top_words_data.append([topic_idx + 1, word, weight])

# Convert the list into a pandas DataFrame
top_words_df = pd.DataFrame(top_words_data, columns=['Topic', 'Word', 'Weight'])

# Save the DataFrame to a CSV file
top_words_df.to_csv(filename, index=False)
print(f"Top words for each topic saved to {filename}")

# Function to evaluate LDA model on test data
def evaluate_lda_model(test_file_path):
    if not os.path.exists(test_file_path):
        raise FileNotFoundError(f"Test data file {test_file_path} not found.")

    # Load test data
    test_data = pd.read_csv(test_file_path)
    print(test_data.head())

    # Handle missing or non-string values in the 'processed_text_str' column
    test_data['processed_text_str'] = test_data['processed_text_str'].fillna("")

    # Extract processed text from the dataframe
    test_texts = [str(text).split() for text in test_data['processed_text_str']]

```

```

# Load the trained LDA model and CountVectorizer
lda_model = load_lda_model('lda_model_2019.pkl')
count_vectorizer = load_count_vectorizer('count_vectorizer_2019.pkl')

# Calculate coherence score and save topic matrix for the test data
coherence_score, lda_topic_matrix_test = calculate_coherence_for_test(test_texts,
lda_model, count_vectorizer, output_dir)

# Plot word clouds and top words for test data
plot_word_clouds(lda_model.components_,
count_vectorizer.get_feature_names_out())
plot_top_words(lda_model, count_vectorizer.get_feature_names_out(), 20, 'Top words
per topic for test data')

# Save top 20 words for each topic to CSV
save_top_words_to_csv(lda_model, count_vectorizer.get_feature_names_out(), 20,
'top_words_test_2019.csv')

return coherence_score

if __name__ == '__main__':
    test_file_path = 'Processed_Testing_Data_2019.csv' # Update with your processed
test data file path

# Evaluate LDA model on test data
start_time = time.time()
coherence_score = evaluate_lda_model(test_file_path)
end_time = time.time()

print(f"Total execution time for testing LDA model: {end_time - start_time:.2f}
seconds.")
print(f"Coherence score for test data: {coherence_score:.4f}")

```

2. NMF

i Train

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import NMF
from gensim.models.coherencemodel import CoherenceModel
from gensim.corpora.dictionary import Dictionary
from wordcloud import WordCloud
import time
import pickle

def calculate_coherence_scores(texts, tfidf, tfidf_vectorizer, max_topics, step=2):
    coherence_scores = []
    for n_topics in range(2, max_topics + 1, step):
        start_time = time.time()

        nmf_model = NMF(n_components=n_topics, random_state=42, max_iter=500)
        nmf_topic_matrix = nmf_model.fit_transform(tfidf)

        topics = []
        feature_names = tfidf_vectorizer.get_feature_names_out()
        for topic_idx, topic in enumerate(nmf_model.components_):
            topics.append([feature_names[i] for i in topic.argsort()[:-10:-1]])

        dictionary = Dictionary(texts)
        corpus = [dictionary.doc2bow(text) for text in texts]
```

```

        cm = CoherenceModel(topics=topics, texts=texts, dictionary=dictionary,
coherence='c_v')

        coherence_score = cm.get_coherence()

        coherence_scores.append((n_topics, coherence_score))

    end_time = time.time()

    print(f'Calculating coherence for {n_topics} topics took {end_time - start_time:.2f}
seconds.')

    print(f'Coherence score for {n_topics} topics: {coherence_score:.4f}')

    return coherence_scores

def plot_word_clouds(topics, feature_names, save_dir='wordclouds'):

    os.makedirs(save_dir, exist_ok=True) # Create directory for saving word clouds

    for i, topic in enumerate(topics):

        plt.figure(figsize=(8, 8))

        topic_words = {feature_names[word_index]: topic[word_index] for word_index in
topic.argsort()[-20:][::-1]}

        wordcloud = WordCloud(width=800, height=800, background_color='white',
max_font_size=100).generate_from_frequencies(topic_words)

        plt.imshow(wordcloud, interpolation='bilinear')

        plt.title(f'Topic {i + 1}', fontsize=16)

        plt.axis('off')

        # Save word cloud image

        plt.savefig(os.path.join(save_dir, f'wordcloud_topic_{i+1}.png'))

        plt.close() # Close figure after saving to free up memory

        print(f'Saved word cloud for Topic {i + 1} to {save_dir}')

def plot_top_words(model, feature_names, n_top_words, title, save_dir='top_words'):

    os.makedirs(save_dir, exist_ok=True) # Create directory for saving top words plots

```

```

n_topics = model.components_.shape[0]
for topic_idx, topic in enumerate(model.components_):
    plt.figure(figsize=(10, 5))
    top_features_ind = topic.argsort()[-n_top_words:][::-1]
    top_features = [feature_names[i] for i in top_features_ind]
    weights = topic[top_features_ind]

    plt.barh(top_features, weights, height=0.7)
    plt.title(f'Topic {topic_idx + 1}', fontsize=14)
    plt.xlabel('Weight')
    plt.ylabel('Feature')
    plt.gca().invert_yaxis()
    plt.xticks(fontsize=10)
    plt.yticks(fontsize=10)
    plt.grid(axis='x', linestyle='--', alpha=0.7)

    # Save top words plot image
    plt.savefig(os.path.join(save_dir, f'top_words_topic_{topic_idx+1}.png'))
    plt.close() # Close figure after saving to free up memory
    print(f'Saved top words for Topic {topic_idx + 1} to {save_dir}')

def save_topics_to_csv(model, tfidf_vectorizer, file_path='topics_output_2023.csv'):
    feature_names = tfidf_vectorizer.get_feature_names_out()
    topics = []
    for topic_idx, topic in enumerate(model.components_):
        top_features_ind = topic.argsort()[:-21:-1]
        top_features = [feature_names[i] for i in top_features_ind]
        weights = topic[top_features_ind]

        # Prepare data for CSV
        topic_data = {

```

```

        'Topic': topic_idx + 1,
        'Top 20 Words': ', '.join(top_features),
        'Weights': ', '.join([f'{weight:.3f}' for weight in weights])
    }
    topics.append(topic_data)

df = pd.DataFrame(topics)
df.to_csv(file_path, index=False)
print(f'Topics saved to {file_path}')

def display_topics(nmf_model, tfidf_vectorizer, num_words=20):
    feature_names = tfidf_vectorizer.get_feature_names_out()

    topics = nmf_model.components_ # Get the topic-word matrix
    topic_list = []
    for idx, topic in enumerate(topics):
        top_features_ind = topic.argsort()[::-num_words - 1:-1]
        top_features = [feature_names[i] for i in top_features_ind]
        weights = topic[top_features_ind]

        # Format weights as a comma-separated string
        weights_str = ', '.join([f'{weight:.3f}' for weight in weights])
        top_features_str = ', '.join(top_features)

        topic_list.append({
            'Topic': idx + 1,
            'Top 20 Words': top_features_str,
            'Weights': weights_str
        })
    print(f"Topic {idx + 1}:")
    print(f"Top 20 Words: {top_features_str}")

```

```

    print(f"Weights: {weights_str}")
    print("\n")

    return topic_list

def train_nmf_model(training_texts, tfidf_vectorizer, max_topics):
    start_time = time.time()

    # Vectorization for training data
    tfidf_train = tfidf_vectorizer.fit_transform([' '.join(text) for text in training_texts])

    # Calculate coherence scores for topics ranging from 2 to max_topics
    coherence_scores = calculate_coherence_scores(training_texts, tfidf_train,
    tfidf_vectorizer, max_topics)

    # Find the number of topics with the highest coherence score
    optimal_topics = max(coherence_scores, key=lambda x: x[1])[0]
    print(f'Optimal number of topics: {optimal_topics}')

    # Plot the coherence scores and save the plot
    topics, scores = zip(*coherence_scores)
    plt.plot(topics, scores)
    plt.xlabel('Number of Topics')
    plt.ylabel('Coherence Score')
    plt.title('Coherence Scores by Number of Topics')
    plt.savefig('coherence_scores_plot.png')
    plt.close() # Close figure after saving to free up memory
    print('Saved coherence scores plot to coherence_scores_plot.png')

    # Train NMF model with the optimal number of topics on the training data
    nmf_model = NMF(n_components=optimal_topics, random_state=42)

```

```

nmf_topic_matrix_train = nmf_model.fit_transform(tfidf_train)

# Extract topics
feature_names = tfidf_vectorizer.get_feature_names_out()

# Plot word clouds and top words for training data and save the plots
plot_word_clouds(nmf_model.components_, feature_names)
plot_top_words(nmf_model, feature_names, 20, f'Top words per topic for NMF model
with {optimal_topics} topics')

# Save topics to CSV
save_topics_to_csv(nmf_model, tfidf_vectorizer, file_path='topics_output_2023.csv')

# Display topics using the display_topics function
display_topics(nmf_model, tfidf_vectorizer)

end_time = time.time()
print(f'Training and evaluation took {end_time - start_time:.2f} seconds.')

return nmf_model, nmf_topic_matrix_train

if __name__ == '__main__':
    start_time = time.time()

    file_path = 'Processed_Training_Data_2023.csv' # Update with your processed
training data file path

    # Check if file exists
    if not os.path.exists(file_path):
        raise FileNotFoundError(f'File {file_path} not found.')

    training_data = pd.read_csv(file_path)

```

```

# Display the first few rows of the dataframe
print(training_data.head())

# Extract processed text from dataframe, handling non-string values
training_texts = []
for text in training_data['processed_text_str']:
    if isinstance(text, str): # Check if the value is a string
        training_texts.append(text.split())
    else:
        training_texts.append([]) # Add an empty list if the value is not a string

# Vectorization and model training
max_topics = 20

tfidf_vectorizer = TfidfVectorizer(max_df=0.85, min_df=1, stop_words='english',
max_features=30000)

nmf_model, nmf_topic_matrix_train = train_nmf_model(training_texts,
tfidf_vectorizer, max_topics)

# Save the model and topic matrices for further use
model_file_path = 'nmf_model_2023.pkl'
topic_matrix_train_file_path = 'nmf_topic_matrix_train_2023.npy'
tfidf_vectorizer_file_path = 'tfidf_vectorizer_2023.pkl'

np.save(topic_matrix_train_file_path, nmf_topic_matrix_train)
with open(model_file_path, 'wb') as file:
    pickle.dump(nmf_model, file)

with open(tfidf_vectorizer_file_path, 'wb') as file:
    pickle.dump(tfidf_vectorizer, file)

end_time = time.time()

```

```

print(f'NMF model saved to {model_file_path}')
print(f'Training topic matrix saved to {topic_matrix_train_file_path}')
print(f'TF-IDF vectorizer saved to {tfidf_vectorizer_file_path}')
print(f'Total execution time: {end_time - start_time:.2f} seconds.')

```

ii Test

```

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from gensim.models.coherencemodel import CoherenceModel
from gensim.corpora.dictionary import Dictionary
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import NMF
from wordcloud import WordCloud
import pickle
import time

def plot_word_clouds(nmf_matrix, feature_names, save_dir=None):
    """
    Generate and save word clouds for each topic.
    """
    os.makedirs(save_dir, exist_ok=True)

    for idx, topic in enumerate(nmf_matrix):
        word_cloud = WordCloud(background_color='white',
                                max_words=50).generate_from_frequencies(
            dict(zip(feature_names, topic))
        )

        plt.figure(figsize=(10, 10))

```

```

plt.imshow(word_cloud, interpolation='bilinear')
plt.axis("off")
plt.title(f"Topic {idx + 1}", fontsize=16)

# Save the word cloud if save directory is specified
if save_dir:
    plt.savefig(os.path.join(save_dir, f'topic_{idx + 1}_wordcloud.png'))
# plt.show()

def plot_top_words(nmf_model, feature_names, num_top_words, title, save_dir=None):
    """
    Plot the top words for each topic.
    """
    os.makedirs(save_dir, exist_ok=True)

    fig, axes = plt.subplots(1, nmf_model.components_.shape[0], figsize=(30, 15),
                             sharex=True)
    axes = axes.flatten()

    for topic_idx, topic in enumerate(nmf_model.components_):
        top_feature_indices = topic.argsort()[::-num_top_words - 1:-1]
        top_features = [feature_names[i] for i in top_feature_indices]
        weights = topic[top_feature_indices]

        ax = axes[topic_idx]
        ax.barh(top_features, weights, height=0.7)
        ax.set_title(f'Topic {topic_idx + 1}', fontdict={'fontsize': 16})
        ax.invert_yaxis()
        ax.tick_params(axis='both', which='major', labelsize=12)
        for i in 'top right left'.split():
            ax.spines[i].set_visible(False)

```

```

fig.suptitle(title, fontsize=22)
fig.tight_layout()
fig.subplots_adjust(top=0.90)

# Save the figure if save directory is specified
if save_dir:
    plt.savefig(os.path.join(save_dir, 'top_words_per_topic.png'))
# plt.show()

def display_new_topics(new_nmf_matrix, tfidf_vectorizer, num_words=20,
output_csv_path=None):
    """
    Display and save the newly extracted topics from the unseen data.
    """
    feature_names = tfidf_vectorizer.get_feature_names_out()

    topic_data = []
    for idx, topic in enumerate(new_nmf_matrix):
        # Sort topics for the document and pick the top N strongest topics
        # Here N corresponds to the number of topics from the training model (e.g., 19)
        top_topic_indices = np.argsort(topic)[-num_words:] # Top N topics for this
document
        top_features = [feature_names[i] for i in top_topic_indices]
        weights = topic[top_topic_indices]

        print(f"Document {idx + 1}:")
        print(f"Top Topics: {' '.join(top_features)}")
        print(f"Weights: {' '.join([f'{weight:.3f}' for weight in weights])}")
        print("\n")

    # Collect data for saving to CSV

```

```

topic_data.append({
    "Document": idx + 1,
    "Top Topics": ', '.join(top_features),
    "Weights": ', '.join([f'{weight:.3f}' for weight in weights])
})

# Save to CSV if output path is provided
if output_csv_path:
    topic_df = pd.DataFrame(topic_data)
    topic_df.to_csv(output_csv_path, index=False)
    print(f"New topics saved to {output_csv_path}")

def test_nmf_model(test_texts, model_file_path, tfidf_vectorizer_file_path,
output_dir='test_output'):
    """
    Apply the pre-trained NMF model to the new dataset and extract new topics.
    """
    # Load the pre-trained model and TF-IDF vectorizer
    with open(model_file_path, 'rb') as file:
        nmf_model = pickle.load(file)

    with open(tfidf_vectorizer_file_path, 'rb') as file:
        tfidf_vectorizer = pickle.load(file)

    # Transform the new dataset using the pre-trained TF-IDF vectorizer
    tfidf_test = tfidf_vectorizer.transform(test_texts)

    # Apply the NMF model to the new data to extract topics
    nmf_topic_matrix_test = nmf_model.transform(tfidf_test) # The document-topic
distribution for the test data
    # print(nmf_topic_matrix_test)

```

```

# Plot word clouds and top words for the new data
feature_names = tfidf_vectorizer.get_feature_names_out()
print(feature_names)

plot_word_clouds(nmf_topic_matrix_test, feature_names,
save_dir=os.path.join(output_dir, 'wordclouds'))

plot_top_words(nmf_model, feature_names, 20, f'Top words per topic for new data',
save_dir=os.path.join(output_dir, 'top_words'))

# Save the test topic matrix for further analysis
np.save(os.path.join(output_dir, 'nmf_topic_matrix_test.npy'),
nmf_topic_matrix_test)
print(f'Test topic matrix saved to {output_dir}/nmf_topic_matrix_test.npy')

# Display and save new topics to CSV
display_new_topics(nmf_topic_matrix_test, tfidf_vectorizer, num_words=20,
output_csv_path=os.path.join(output_dir, 'nmf_top_words.csv'))

if __name__ == '__main__':
    start_time = time.time()

    # Load the unseen test data for prediction
    file_path = 'Processed_Testing_Data_2023.csv' # Update with the path to your test
data
    model_file_path = 'nmf_model_2023.pkl'
    tfidf_vectorizer_file_path = 'tfidf_vectorizer_2023.pkl'

    # Check if the file exists
    if not os.path.exists(file_path):
        raise FileNotFoundError(f'File {file_path} not found.')

    # Load the test data
    test_data = pd.read_csv(file_path)

```

```

# Extract the processed text from the dataframe (assumed to be in 'processed_text_str'
column)

test_texts = test_data['processed_text_str'].fillna("") # Fill empty cells with an empty
string

# Create the output directory for saving results
output_dir = 'test_output'
os.makedirs(output_dir, exist_ok=True)

# Test the NMF model on the test dataset to extract new topics
test_nmf_model(test_texts, model_file_path, tfidf_vectorizer_file_path, output_dir)

end_time = time.time()
print(f'Testing and evaluation took {end_time - start_time:.2f} seconds.')

```

3. Top2vec

i Train

```

import os
import pandas as pd
import time
from top2vec import Top2Vec
import nltk
import gensim
from gensim.models.coherencemodel import CoherenceModel
import matplotlib.pyplot as plt
import umap
import hdbscan

def main():
    # Load the data
    df = pd.read_csv('Master_List_2023_Jan_to_Sep.csv')

```

```

# Ensure all entries in 'article_content' are strings and handle missing values
df['article_content'] = df['article_content'].astype(str).fillna("")

# Convert the text to a list
docs_filtered = df['article_content'].values.tolist()

# Define stopwords and tokenization functions
nltk.download('stopwords')
stopwords = set(nltk.corpus.stopwords.words('english'))

def remove_stopwords(text):
    return [word for word in text if word not in stopwords]

def tok(text):
    text = gensim.utils.simple_preprocess(text)
    text = remove_stopwords(text)
    return text

# Debug: Print some tokenized documents
for i, doc in enumerate(docs_filtered[:1]):
    print(f"Document {i}: {tok(doc)}")

# Measure the execution time for training the model
start_time = time.time()

# Train the doc2vec model
topic_model = Top2Vec(
    documents=docs_filtered,
    embedding_model='doc2vec', # Generate document embeddings using doc2vec
    speed="deep-learn",

```

```

tokenizer=tok,
ngram_vocab=True,
ngram_vocab_args={"connector_words":
"phrases.ENGLISH_CONNECTOR_WORDS"}
)

end_time = time.time()
execution_time = end_time - start_time
print("Execution time for training the model:", execution_time)

# Create the directory if it doesn't exist
model_dir = 'model'
os.makedirs(model_dir, exist_ok=True)
model_path = os.path.join(model_dir, 'topic2vec_filtered_2023(1)')

# Save the model for later use
topic_model.save(model_path)
print(f"Model saved to {model_path}")

# Get the number of topics
print("Number of topics:", topic_model.get_num_topics())

# Get topic sizes
topic_sizes, topic_nums = topic_model.get_topic_sizes()
print("Topic sizes:", topic_sizes)

# Get topic words, word scores, and topic numbers
topic_words, word_scores, topic_nums = topic_model.get_topics()

# Save topics and their top 20 words to a CSV file
topics_df = pd.DataFrame()

```

```

for topic_num in topic_nums:
    topic_words_top20 = topic_words[topic_num][:20]
    topics_df[f'Topic {topic_num}'] = topic_words_top20

topics_csv_path = os.path.join(model_dir, 'topics_train_2023(1).csv')
topics_df.to_csv(topics_csv_path, index=False)
print(f'Topics and their top 20 words saved to {topics_csv_path}')

# Calculate coherence score
texts = [tok(doc) for doc in docs_filtered]
coherence_model = CoherenceModel(topics=topic_words, texts=texts,
dictionary=gensim.corpora.Dictionary(texts), coherence='c_v')
coherence_score = coherence_model.get_coherence()
print("Coherence score:", coherence_score)

# Apply UMAP for dimensionality reduction
umap_model = umap.UMAP(n_neighbors=15, n_components=5, min_dist=0.0,
metric='cosine')
umap_embeddings = umap_model.fit_transform(topic_model.document_vectors)

# Apply HDBSCAN for clustering
hdbscan_model = hdbscan.HDBSCAN(min_cluster_size=15, metric='euclidean',
cluster_selection_method='eom')
hdbscan_labels = hdbscan_model.fit_predict(umap_embeddings)

# Debug: Print the number of unique clusters
print(f'Number of clusters found by HDBSCAN: {len(set(hdbscan_labels))}')

# Generate and display word clouds for all topics
for topic in topic_nums:
    plt.figure(figsize=(10, 10))
    topic_model.generate_topic_wordcloud(topic)

```

```

plt.title(f'Topic {topic} Word Cloud')
plt.axis('off')
plt.show()

# Plot UMAP embeddings colored by HDBSCAN clusters
plt.figure(figsize=(10, 8))
plt.scatter(umap_embeddings[:, 0], umap_embeddings[:, 1], c=hdbscan_labels,
cmap='Spectral', s=5)
plt.colorbar()
plt.title("UMAP Projection of Documents with HDBSCAN Clusters")
plt.xlabel("UMAP Dimension 1")
plt.ylabel("UMAP Dimension 2")
plt.savefig(os.path.join(model_dir, 'umap_hdbscan_plot_2023(1)_train.png'))
plt.show()

print(f"UMAP and HDBSCAN plot saved to {os.path.join(model_dir,
'umap_hdbscan_plot_2023(1)_train.png')}")

if __name__ == '__main__':
    main()

```

ii Test

```

import os
import pandas as pd
from top2vec import Top2Vec
import nltk
import gensim
from gensim.models.coherencemodel import CoherenceModel
import matplotlib
matplotlib.use('Agg') # Use a non-interactive backend
import matplotlib.pyplot as plt
import umap
import hdbscan

```

```

from wordcloud import WordCloud

def main():
    # Load the saved Top2Vec model
    model_dir = 'model'
    model_path = os.path.join(model_dir, 'topic2vec_filtered_2022')
    topic_model = Top2Vec.load(model_path)
    print(f'Model loaded from {model_path}')

    # Load the testing data
    test_df = pd.read_csv('Master_List_2022_Oct_to_Dec.csv')

    # Ensure all entries in 'article_content' are strings and handle missing values
    test_df['article_content'] = test_df['article_content'].astype(str).fillna("")

    # Convert the text to a list
    test_docs = test_df['article_content'].values.tolist()

    # Define stopwords and tokenization functions
    nltk.download('stopwords')
    stopwords = set(nltk.corpus.stopwords.words('english'))

    def remove_stopwords(text):
        return [word for word in text if word not in stopwords]

    def tok(text):
        text = gensim.utils.simple_preprocess(text)
        text = remove_stopwords(text)
        return text

    # Tokenize the test documents

```

```

test_docs_tokenized = [tok(doc) for doc in test_docs]

# Infer topics for each document in the testing data
inferred_topics = []
inferred_scores = []
for doc in test_docs:
    # Infer the most similar topic for each document
    results = topic_model.query_topics(doc, num_topics=1) # Query for the top topic
    topic_num = results[0][0] # Extract topic number from results
    similarity_score = results[1][0] # Extract similarity score from results
    inferred_topics.append(topic_num)
    inferred_scores.append(similarity_score)

# Store the inferred topics and scores in the DataFrame
test_df['Inferred_Topic'] = inferred_topics
test_df['Topic_Similarity_Score'] = inferred_scores

# Save the results to a CSV file
results_csv_path = os.path.join(model_dir, '2022_testing_data_results.csv')
test_df.to_csv(results_csv_path, index=False)
print(f"Testing results saved to {results_csv_path}")

# Calculate coherence score for the testing data
topic_words, _, _ = topic_model.get_topics()
coherence_model_test = CoherenceModel(
    topics=topic_words,
    texts=test_docs_tokenized,
    dictionary=gensim.corpora.Dictionary(test_docs_tokenized),
    coherence='c_v'
)
coherence_score_test = coherence_model_test.get_coherence()

```

```

print("Coherence score for testing data:", coherence_score_test)

# Save the top 20 words for each topic
topics_df = pd.DataFrame()
for topic_num in range(len(topic_words)):
    topics_df[f'Topic {topic_num}'] = topic_words[topic_num][:20]

topics_csv_path = os.path.join(model_dir, '2022_test_topics_top20_words.csv')
topics_df.to_csv(topics_csv_path, index=False)
print(f'Top 20 words for each topic saved to {topics_csv_path}')

# Generate and save word clouds for each topic
wordcloud_dir = os.path.join(model_dir, 'wordclouds')
os.makedirs(wordcloud_dir, exist_ok=True)

for topic_num in range(len(topic_words)):
    wordcloud = WordCloud(width=800, height=800,
background_color='white').generate(" ".join(topic_words[topic_num][:20]))
    plt.figure(figsize=(8, 8))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(f'Topic {topic_num} Word Cloud')
    plt.savefig(os.path.join(wordcloud_dir, f'topic_{topic_num}_wordcloud.png'))
    plt.close()
print(f'Word clouds saved in {wordcloud_dir}')

# Apply UMAP for dimensionality reduction
umap_model = umap.UMAP(n_neighbors=15, n_components=2, min_dist=0.0,
metric='cosine')
umap_embeddings = umap_model.fit_transform(topic_model.document_vectors)

# Apply HDBSCAN for clustering

```

```

    hdbscan_model = hdbscan.HDBSCAN(min_cluster_size=15, metric='euclidean',
cluster_selection_method='eom')

    hdbscan_labels = hdbscan_model.fit_predict(umap_embeddings)

    # Plot the UMAP embeddings with HDBSCAN labels
    plt.figure(figsize=(10, 10))

    plt.scatter(umap_embeddings[:, 0], umap_embeddings[:, 1], c=hdbscan_labels,
cmap='Spectral', s=5)

    plt.colorbar()

    plt.title("UMAP Projection of Testing Data with HDBSCAN Clusters")

    plt.savefig(os.path.join(model_dir, 'umap_hdbscan_plot_2022_test.png'))

    plt.close()

    print(f'UMAP and HDBSCAN plot saved to {os.path.join(model_dir,
'umap_hdbscan_plot_2022_test.png')}')

if __name__ == '__main__':
    main()

```

4. BERTopic

```

import pandas as pd
import numpy as np
import time
from sentence_transformers import SentenceTransformer
from umap import UMAP
from hdbscan import HDBSCAN
from bertopic import BERTopic
from gensim.models.coherencemodel import CoherenceModel
from gensim.corpora.dictionary import Dictionary
from sklearn.feature_extraction.text import CountVectorizer
from wordcloud import WordCloud
import seaborn as sns
import matplotlib.pyplot as plt

```

```

if __name__ == "__main__": # Add this block to handle multiprocessing correctly
    # Record the start time
    start_time = time.time()

    # Load CSV file
    file_path = "Processed_Testing_Data_2020.csv"
    df = pd.read_csv(file_path)
    docs = df['article_content'].astype(str).tolist()

    # Create embeddings
    embedding_model = SentenceTransformer('sentence-transformers/all-MiniLM-L6-
v2')
    embeddings_start = time.time()
    embeddings = embedding_model.encode(docs, show_progress_bar=True)
    print(f'Embeddings created in {time.time() - embeddings_start:.2f} seconds')

    # Save embeddings (adjust path)
    np.save('embeddings_test_2020.npy', embeddings)

    # Train UMAP and HDBSCAN
    umap_model = UMAP(n_neighbors=10, n_components=2, metric='cosine')
    hdbscan_model = HDBSCAN(min_cluster_size=10, metric='euclidean',
cluster_selection_method='eom')

    # Train BERTopic model
    topic_model_start = time.time()
    topic_model = BERTopic(
        embedding_model=embedding_model,
        umap_model=umap_model,
        hdbscan_model=hdbscan_model,
        vectorizer_model=CountVectorizer(stop_words="english")

```

```

).fit(docs, embeddings=embeddings)

print(f'BERTopic model trained in {time.time() - topic_model_start:.2f} seconds")

# Get topics and save top 20 words
topics = topic_model.get_topics()
top_20_words_per_topic = {}

# Modify this section to retrieve the top 20 words instead of 10
for topic_num, topic in topics.items():
    top_20_words = topic[:20] # Extract the top 20 words
    top_20_words_str = ', '.join([f'{word} ({weight:.4f})' for word, weight in
top_20_words])
    top_20_words_per_topic[topic_num] = top_20_words_str

    topic_words_df = pd.DataFrame.from_dict(top_20_words_per_topic, orient='index',
columns=['Top 20 Words'])
    topic_words_df.index.name = 'Topic'

    csv_file_path = "top_words_test_2020.csv"
    topic_words_df.to_csv(csv_file_path)
    print(f'Topics and top 20 words saved to {csv_file_path}')

# Generate word clouds for each topic
wordcloud_start = time.time()
for topic_num, topic_words in topics.items():
    word_freq = {word: weight for word, weight in topic_words[:20]} # Use the top 20
words for wordcloud

    wordcloud = WordCloud(
        width=800,
        height=400,
        background_color='white'

```

```

).generate_from_frequencies(word_freq)

# Plot the wordcloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title(f"WordCloud for Topic {topic_num}", fontsize=16)

# Save each wordcloud as an image
wordcloud_file_path = f"wordcloud_topic_{topic_num}.png"
plt.savefig(wordcloud_file_path, bbox_inches='tight')
print(f"WordCloud for Topic {topic_num} saved to {wordcloud_file_path}")

# plt.show()

print(f"Word clouds generated in {time.time() - wordcloud_start:.2f} seconds")

# Get topic info and coherence score
coherence_start = time.time()
tokenized_docs = [doc.split() for doc in docs]
dictionary = Dictionary(tokenized_docs)
corpus = [dictionary.doc2bow(doc) for doc in tokenized_docs]

topic_words = [[word for word, _ in topic_model.get_topic(topic_num)[:20]] for
topic_num in topics.keys()] # Adjust coherence for top 20 words
coherence_model = CoherenceModel(topics=topic_words, texts=tokenized_docs,
dictionary=dictionary, coherence='c_v')
coherence_score = coherence_model.get_coherence()
print(f"Coherence Score: {coherence_score:.4f}")
print(f"Coherence model calculated in {time.time() - coherence_start:.2f} seconds")

# Save topic model

```

```

topic_model.save(
    path="model_dir_test_2020",
    serialization="safetensors",
    save_ctfidf=True,
    save_embedding_model="sentence-transformers/all-MiniLM-L6-v2"
)

# Visualize with UMAP and Seaborn
visualization_start = time.time()
reduced_embeddings = umap_model.fit_transform(embeddings)
clusters = hdbscan_model.fit(reduced_embeddings).labels_

# Adjust visualization settings for color mapping
df_viz = pd.DataFrame({"x": reduced_embeddings[:, 0], "y": reduced_embeddings[:, 1],
"Topic": topic_model.topics_})
df_viz["Length"] = [len(doc) for doc in docs]
df_viz["Topic"] = df_viz["Topic"].astype("category")

# Plot UMAP
plt.figure(figsize=(16, 16))
sns.scatterplot(data=df_viz, x="x", y="y", hue="Topic",
palette=sns.color_palette("hsv", len(df_viz["Topic"].unique())))

# Save plot
fig_file_path = "umap_plot_test_2020.png"
plt.savefig(fig_file_path, bbox_inches='tight')
print(f"Plot saved to {fig_file_path}")

plt.show()
print(f"Visualization completed in {time.time() - visualization_start:.2f} seconds")

# Total execution time

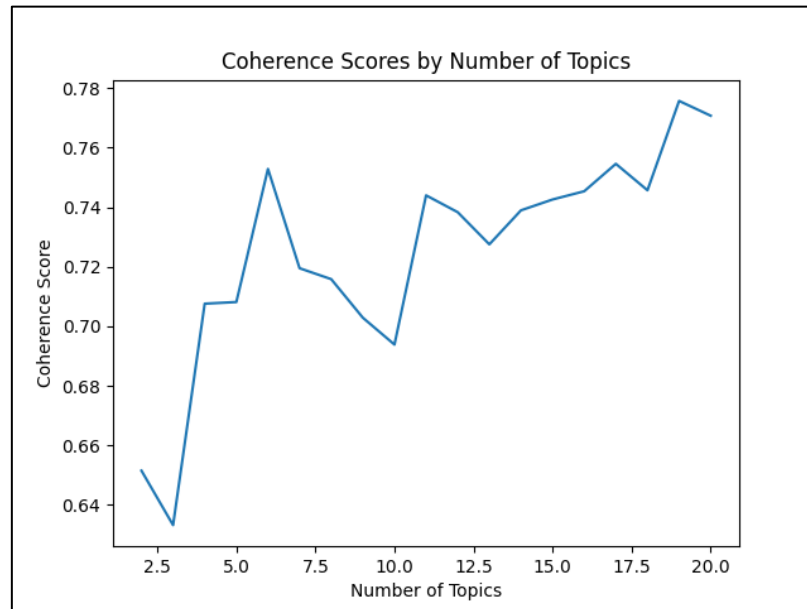
```

```
print(f"Total execution time: {time.time() - start_time:.2f} seconds")
```

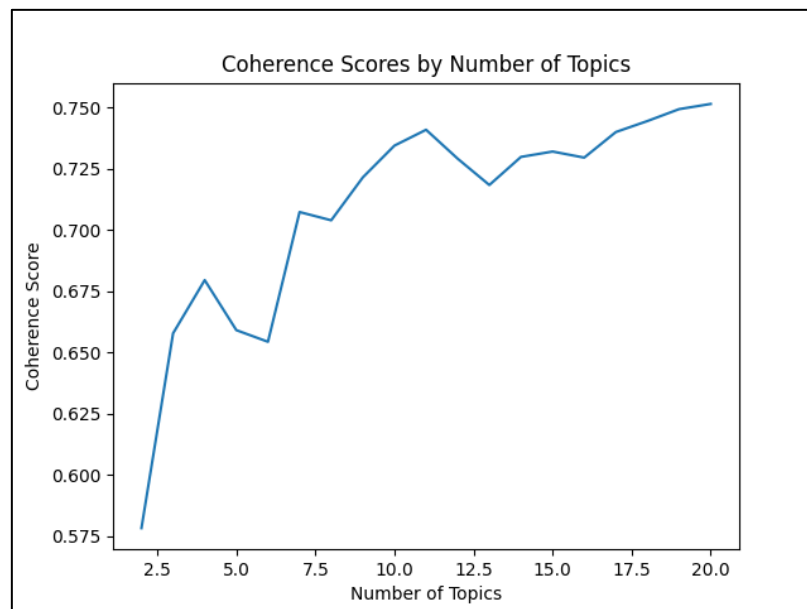
APPENDIX B: TOPIC MODELLING

Coherence score graph of NMF model.

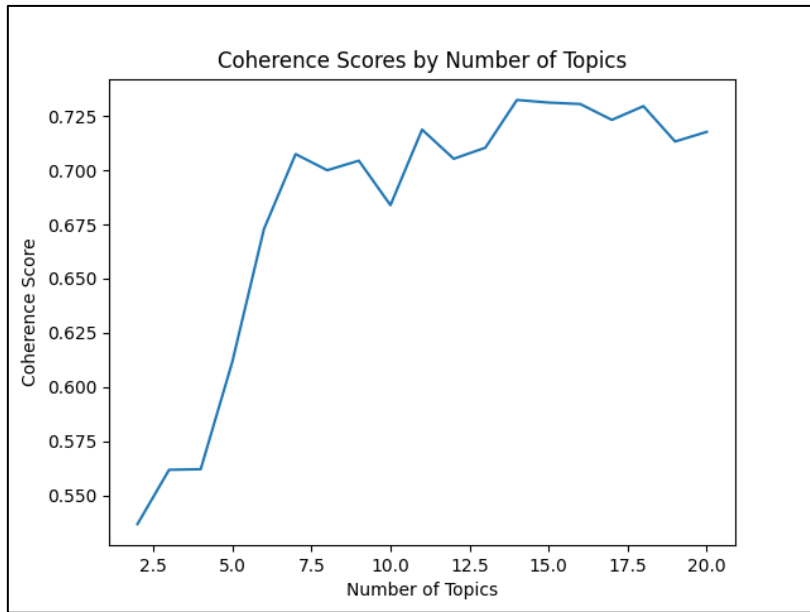
1. Model 2019



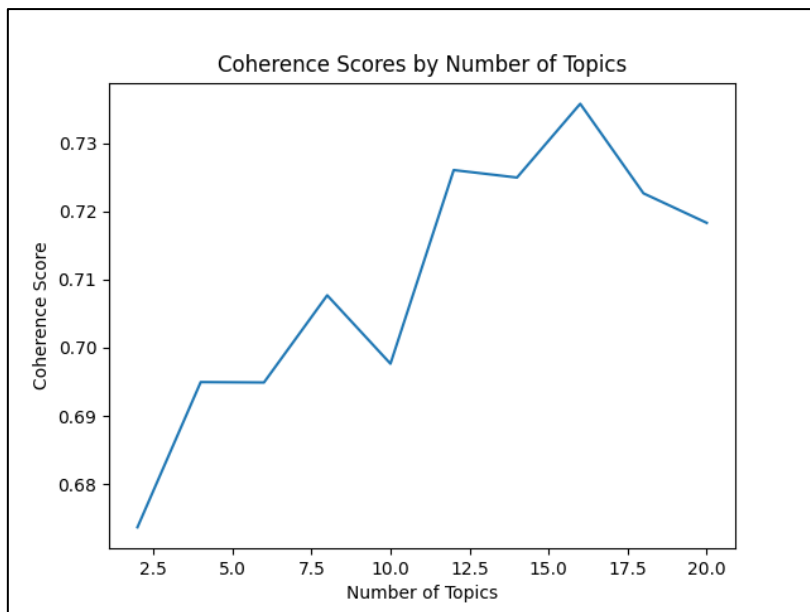
2. Model 2020



3. Model 2022

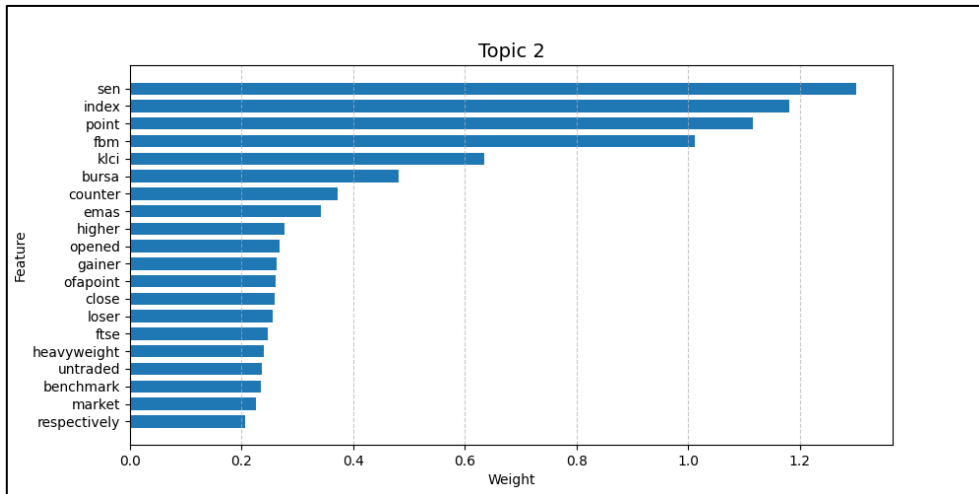


4. Model 2023

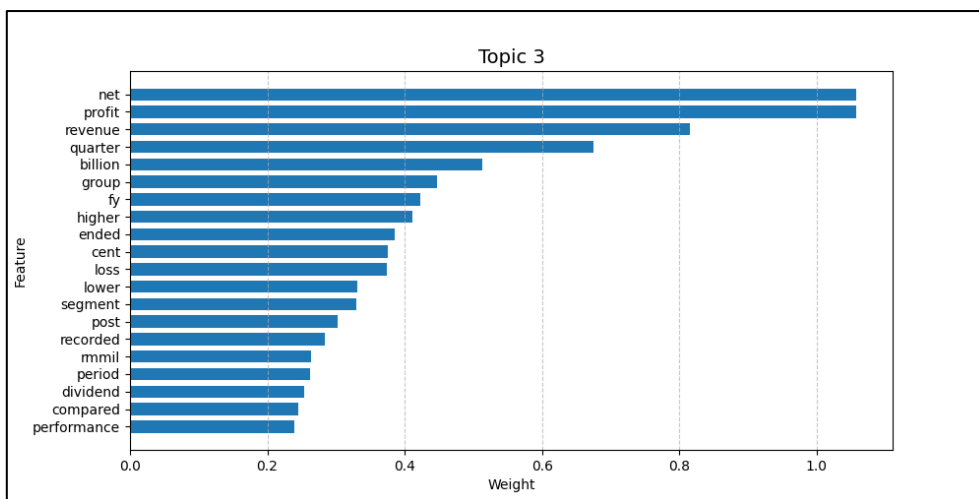


Top words bar chart of 2021 model.

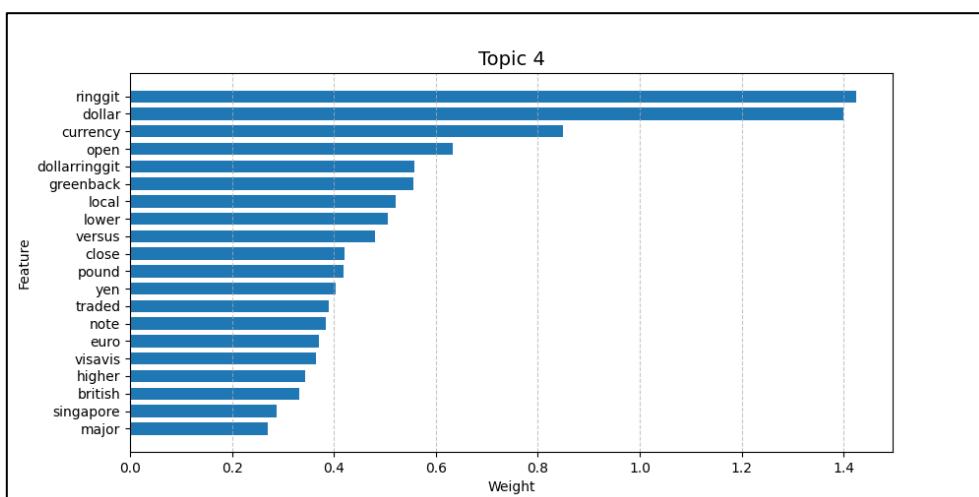
1. Topic 2



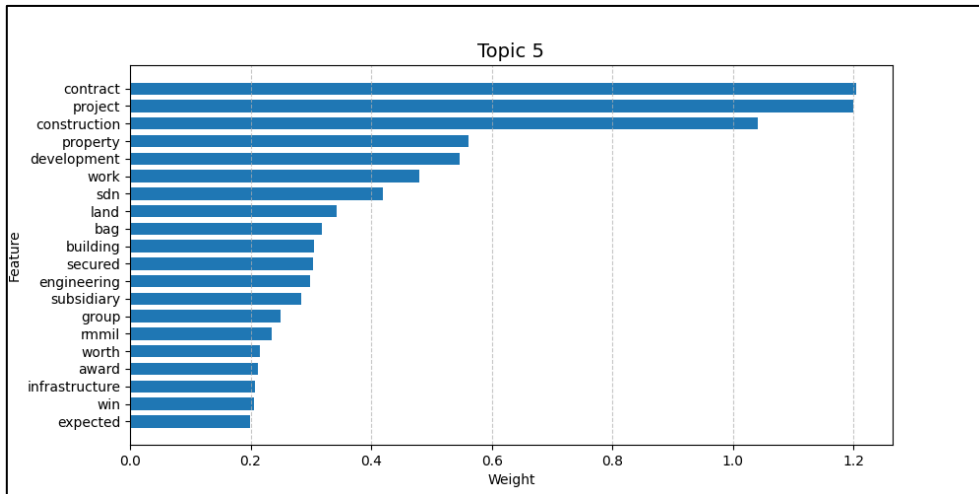
2. Topic 3



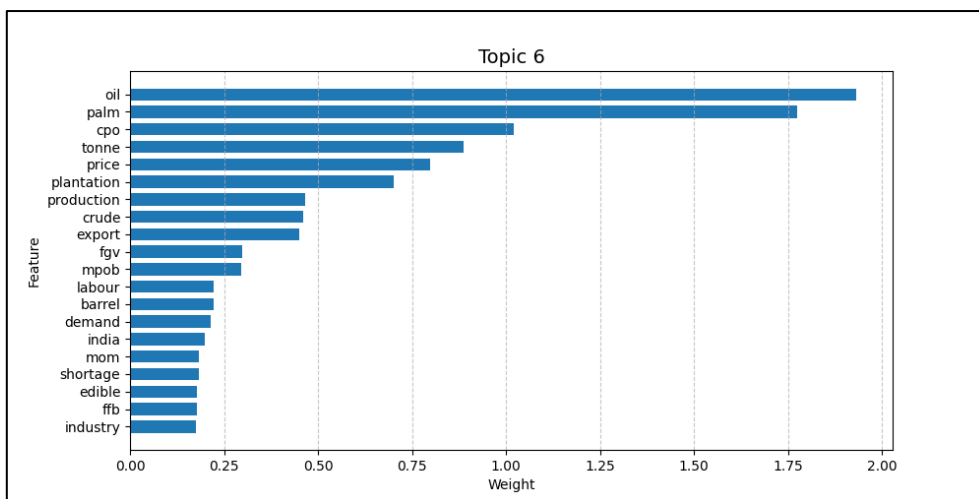
3. Topic 4



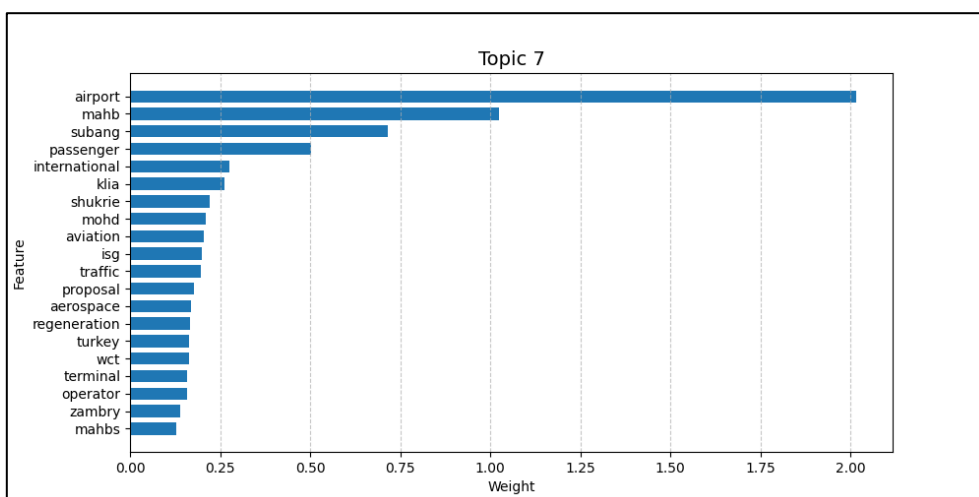
4. Topic 5



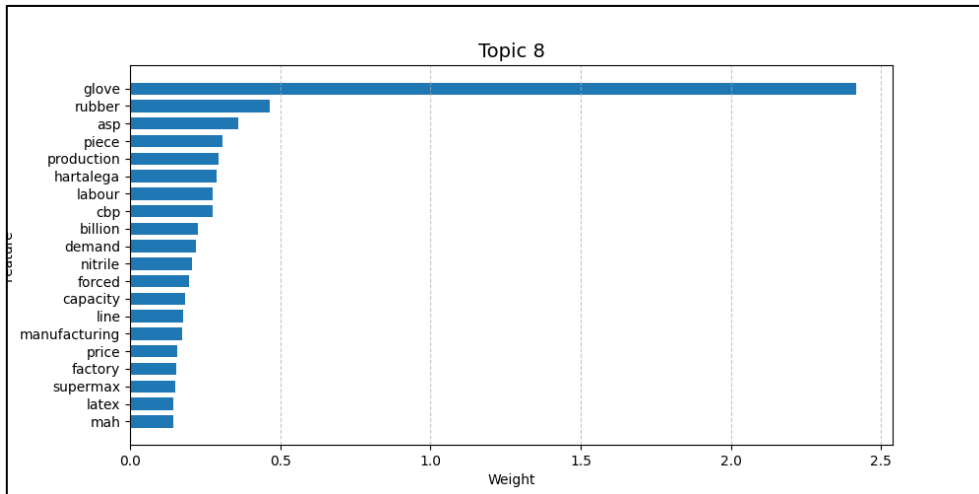
5. Topic 6



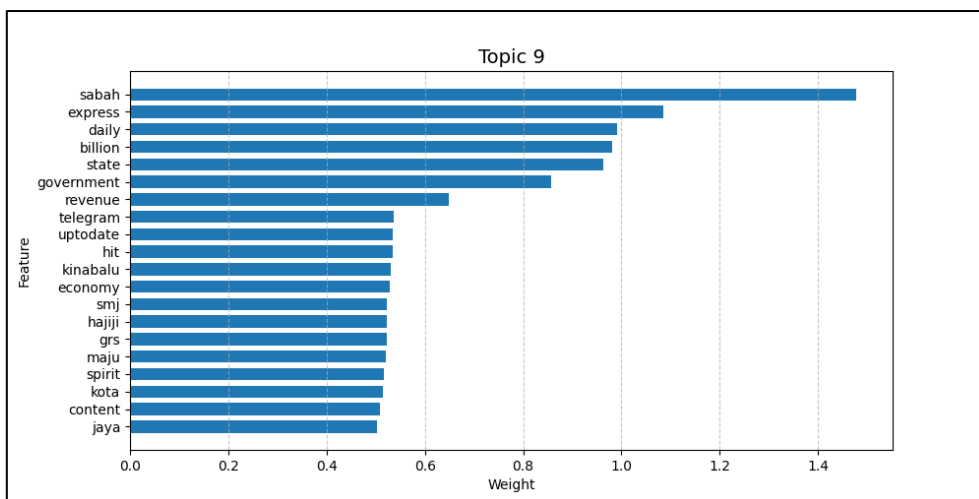
6. Topic 7



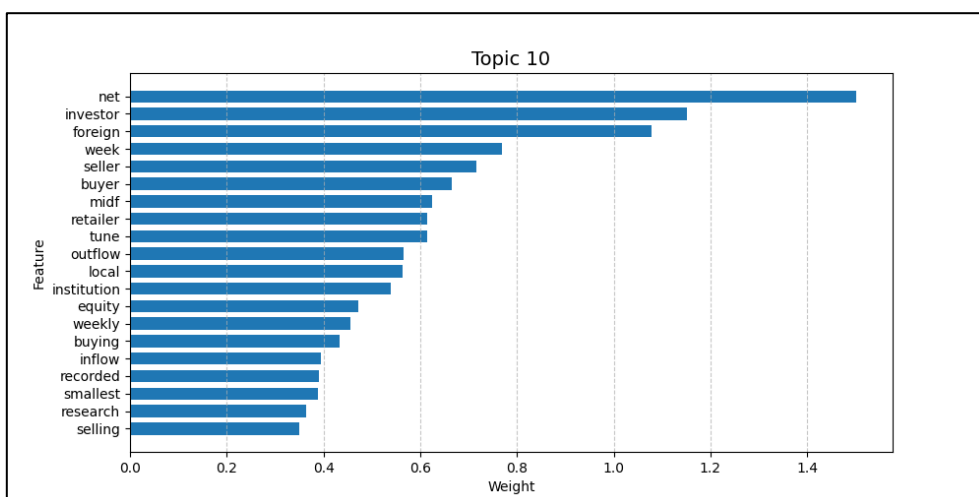
7. Topic 8



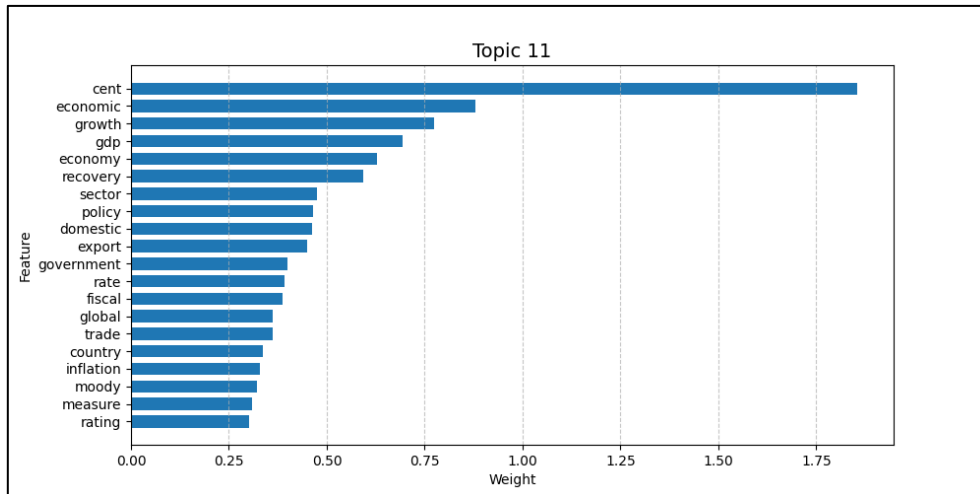
8. Topic 9



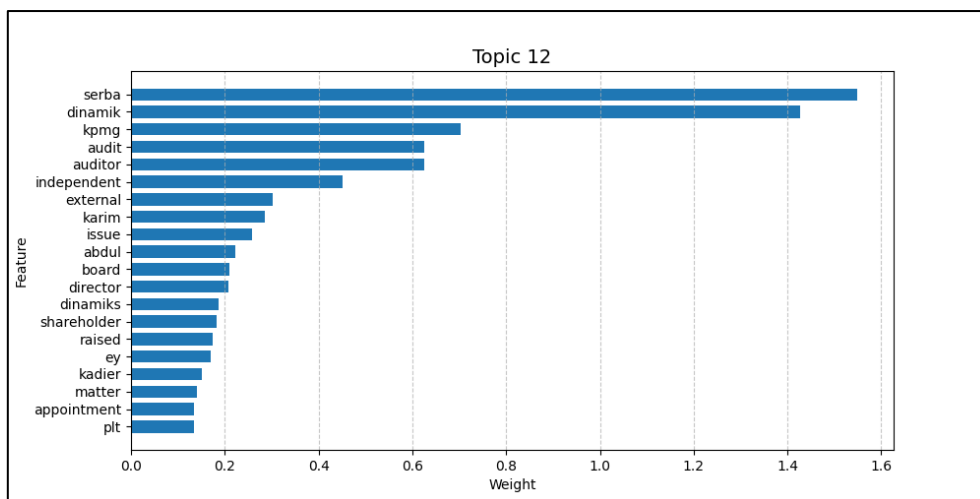
9. Topic 10



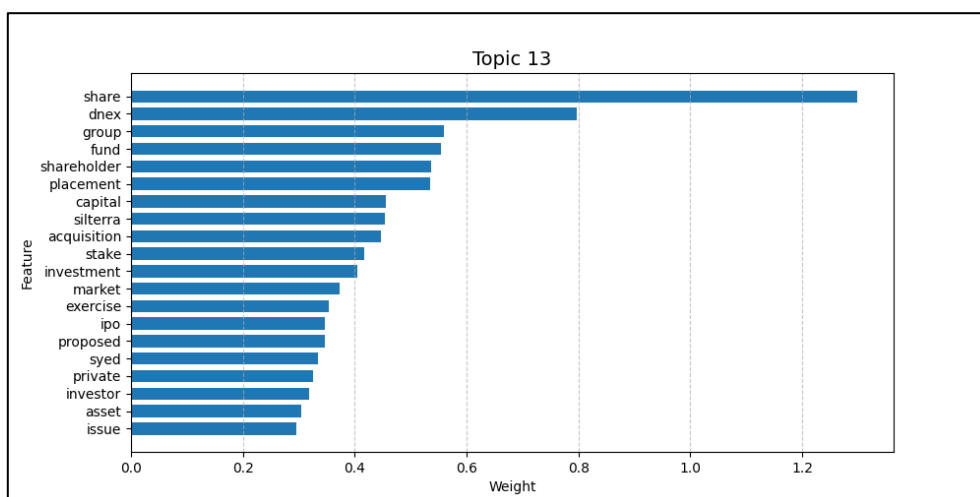
10. Topic 11



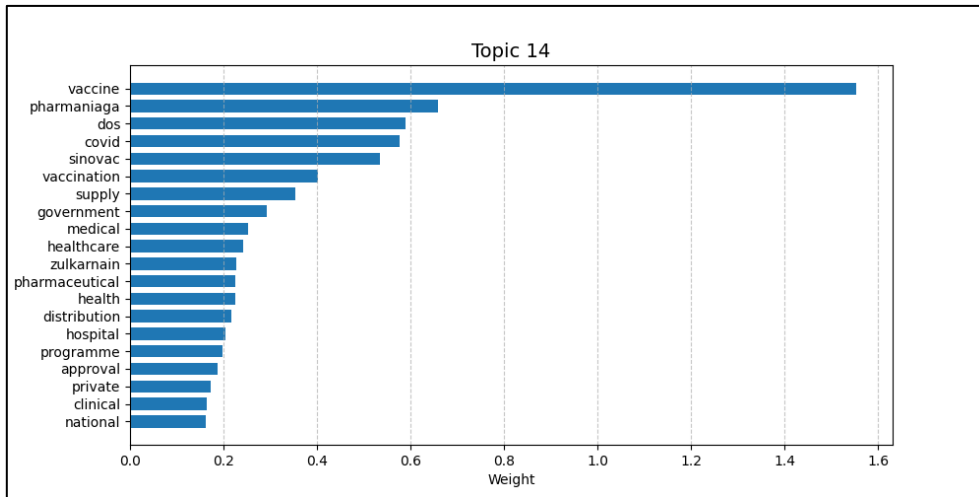
11. Topic 12



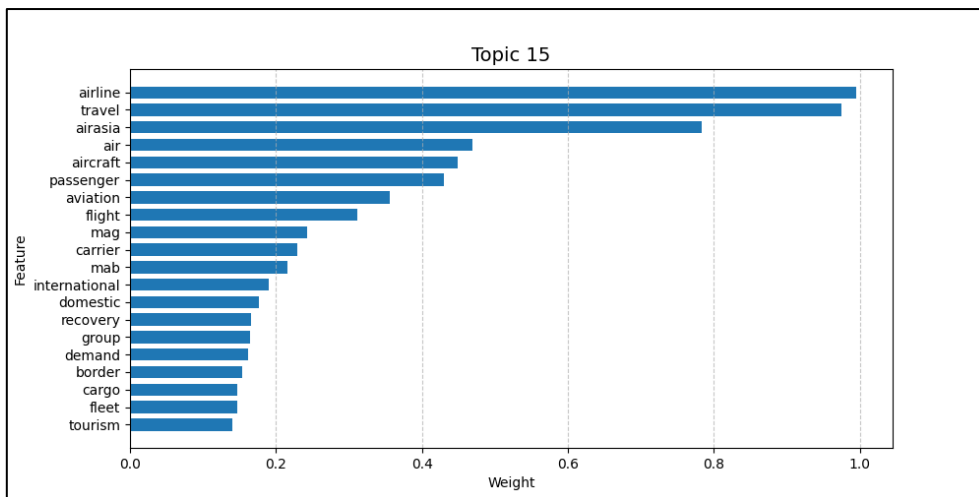
12. Topic 13



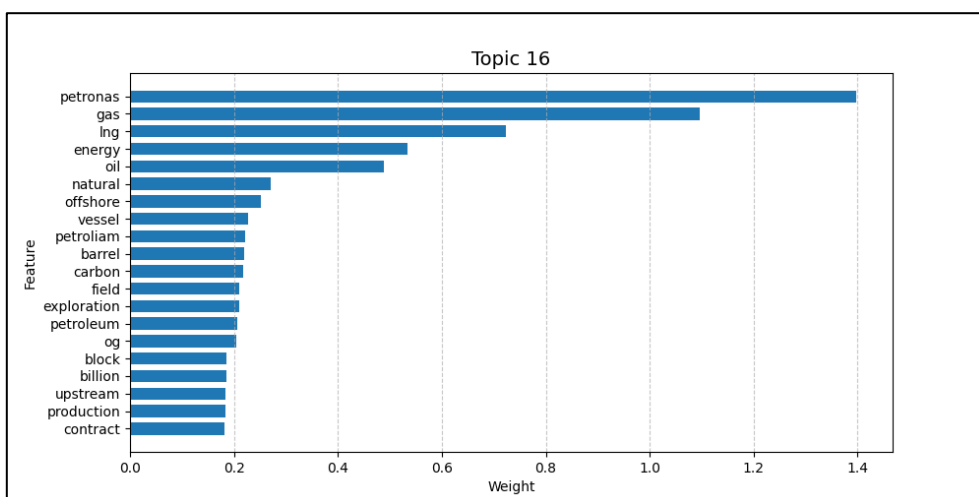
13. Topic 14



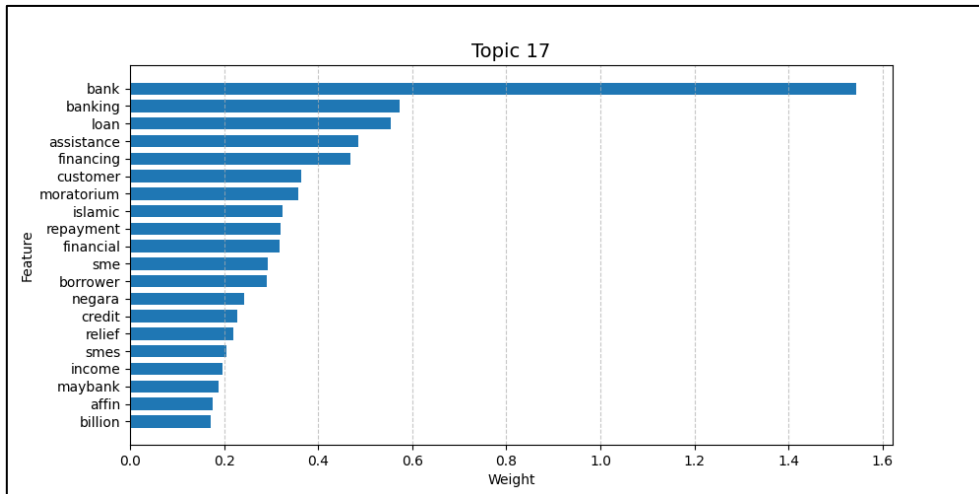
14. Topic 15



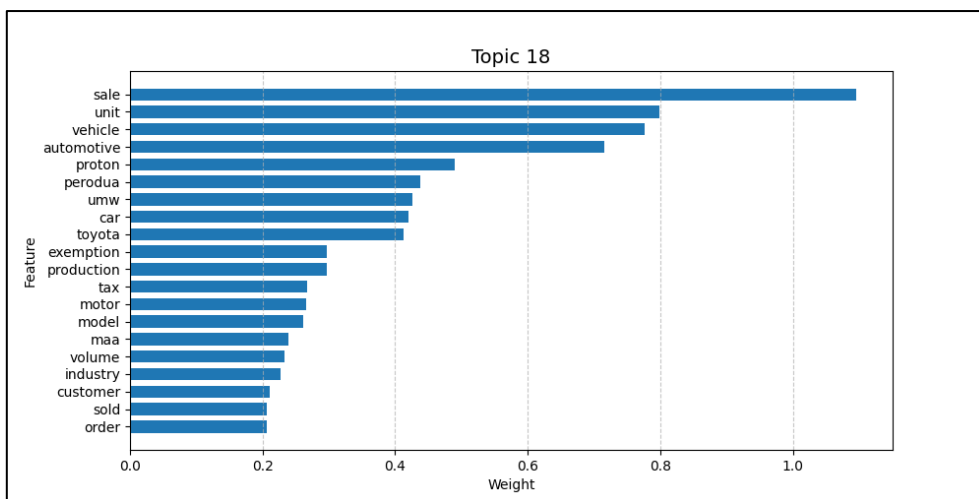
15. Topic 16



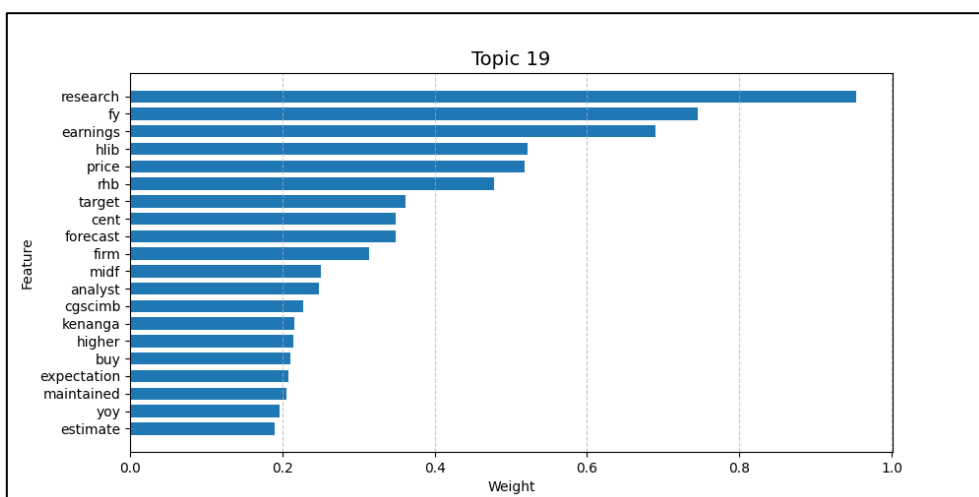
16. Topic 17



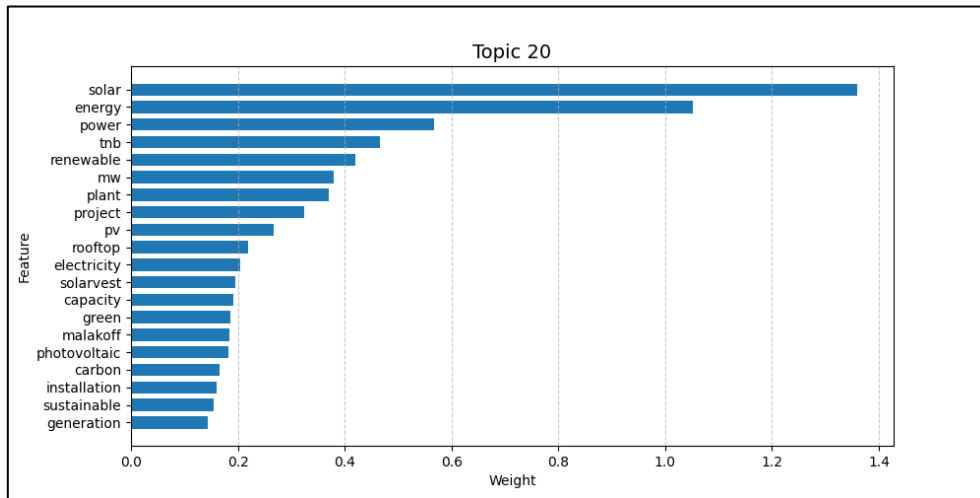
17. Topic 18



18. Topic 19



19. Topic 20



APPENDIX C: USER ACCEPTANCE TEST

User Acceptance Test (UAT)

Section 1: Demographic Information

Please enter your faculty:

Your answer

Please select your current year of study:

☐ First year

☐ Second year

☐ Third year

☐ Fourth year

Gender

Section 2: Web Application Usability Scale

A) Analyse Documents

1. The interface for uploading and viewing documents is simple and easy to use.

☐ Strongly disagree

☐ Disagree

☐ Neutral

☐ Agree

☐ Strongly agree

2. The provided file format or template was clear and easy to use.

☐ Strongly disagree

☐ Disagree

☐ Neutral

☐ Agree

Please select your current year of study:

☐ First year

☐ Second year

☐ Third year

☐ Fourth year

Gender

☐ Female

☐ Male

[Back](#) [Next](#) [Clear form](#)

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. - [Contact form owner](#) - [Terms of Service](#) - [Privacy Policy](#)

Does this form look suspicious? [Report](#)

Google Forms

2. The provided file format or template was clear and easy to use.

☐ Strongly disagree

☐ Disagree

☐ Neutral

☐ Agree

☐ Strongly agree

3. Uploading documents or templates was smooth and without issues.

☐ Strongly disagree

☐ Disagree

☐ Neutral

☐ Agree

☐ Strongly agree

4. The analysis results were presented clearly and were easy to interpret.

☐ Strongly disagree

☐ Disagree

☐ Neutral

☐ Agree

☐ Strongly agree

5. I was able to download the analysis results easily.

☐ Strongly disagree

☐ Disagree

☐ Neutral

☐ Agree

☐ Strongly agree

[Back](#) [Next](#) [Clear form](#)

Section 2: Web Application Usability Scale

B) Analyse Sentences

1. Entering or pasting the text for analysis was straightforward.

☐ Strongly disagree

☐ Disagree

☐ Neutral

☐ Agree

☐ Strongly agree

2. I understood how the input text was handled or processed before analysis.

☐ Strongly disagree

☐ Disagree

☐ Neutral

☐ Agree

2. I understood how the input text was handled or processed before analysis.

☐ Strongly disagree

☐ Disagree

☐ Neutral

☐ Agree

☐ Strongly agree

3. The analysis results were presented clearly and were easy to interpret.

☐ Strongly disagree

☐ Disagree

☐ Neutral

☐ Agree

☐ Strongly agree

4. The results were easy to download without any issues.

☐ Strongly disagree

☐ Disagree

☐ Neutral

☐ Agree

☐ Strongly agree

5. I found the single-text analysis process to be effective and user-friendly.

☐ Strongly disagree

☐ Disagree

☐ Neutral

☐ Agree

☐ Strongly agree

Back

Submit

Clear form