Project Cover Sheet

Student IDs: 57142664, 57149200, 57153875, 40156222

Student Names: Chua Ka Hiu, Twan Tsz Yin, Li Yiu Hang, Matthew Zhang

Student Emails: khchua2-c@my.cityu.edu.hk, tytwan2-c@my.cityu.edu.hk, yiuhangli3-c@my.cityu.edu.hk, mattzhang2-c@my.cityu.edu.hk

Title:
The Guardian News Article Analysis with Parallel Computing

Highlights: (maximum: 5 items)
- Parallel Data Fetching
- Text Data Parsing with Apache Spark
- Time Series Prediction with SARIMA
- LDA and Classifier Model Training with Apache Spark
- Parallel Computing Speed Analysis

List of Deliverables

File Name	Description
report.pdf	Main Project Report
2024_2020_merged_2GB.csv	Dataset fetched from the Guardian
article_word_count.py	Python script for bar chars generation & analysis on articles' "Content"
BOW.scala	Scala script for converting text data into Bag of Words
data_cleaning.py	Python script for removing undesired records (short, non_English, missing fields)
data_fetching.py	Python script for fetching data from the Guardian API
random_sample.py	Python script for random sampling
TFIDF.scala	Scala script for TFIDF score computation
wordCloud.py	Python script for generating World Cloud graphs
timeSeries.py	Python script for generating Time Series graphs
sentimentAnalysis.py	Python script for Sentiment Analysis
LDA_Classifiers.ipynb	Python script for data preprocessing, creating LDA word cloud and confusion matrices for logistic regression and naive Bayes
main.scala	Scala script to execute the classifier.scala and lda.scala
classifier.scala	Scala script of the two classifier models: logistic regression and naive Bayes
lda.scala	Scala script of LDA model

------ END ------

The Guardian News Article Analysis with Parallel Computing

Student IDs : 57142664, 57149200, 57153875, 40156222
Student Names : Chua Ka Hiu, Twan Tsz Yin, Li Yiu Hang, Matthew Zhang
Student Emails : khchua2-c@my.cityu.edu.hk, tytwan2-c@my.cityu.edu.hk, yiuhangli3-c@my.cityu.edu.hk, mattzhang2-c@my.cityu.edu.hk

Title:
The Guardian News Article Analysis with Parallel Computing
Highlights: (maximum: 5 items)

- Parallel Data Fetching
- Text Data Parsing with Apache Spark
- Time Series Prediction with SARIMA
- LDA and Classifier Model Training with Apache Spark
- Parallel Computing Speed Analysis

Abstract: (max 300 words)
This study explores the application of data processing techniques mentioned in the lecture to analyze a large dataset of news articles spanning from 2020 to 2024, offering a comprehensive view of recent world events and trends. By leveraging parallel computing frameworks, including Apache Spark, Hadoop, and Pig Latin, alongside natural language processing (NLP) tools and classification models, the research demonstrates the effectiveness of parallelization in enhancing scalability and optimizing computational efficiency. The findings highlight the ability to classify unseen data and extract meaningful insights, showcasing the transformative potential of parallel computing in large-scale text analysis.

The Guardian News Article Analysis with Parallel Computing

Table of Content

The Guardian News Article Analysis with Parallel Computing

1. Introduction

Over the past five years, numerous global events have profoundly shaped world development. The COVID-19 pandemic, the ongoing conflict between Russia and Ukraine, the Middle East crisis, and other significant occurrences have drastically affected people's lives worldwide. In such times, news analysis becomes a crucial tool to gain insights into recent global events and to understand the broader trends shaping our world.

As our world becomes increasingly interconnected, the rapid growth of online platforms and the sheer volume of information have transformed how we consume, interpret, and react to news. This evolution has led to a new paradigm: data-driven news analysis. Leveraging data science and technology, this approach provides deeper insights into the vast amount of information represented in news articles. Data-driven news article analysis utilizes advanced data science techniques and natural language processing (NLP) to extract meaningful insights from news content. A new news section classifier has also been developed to help categorize articles more effectively. It moves beyond traditional qualitative methods by incorporating quantitative tools and algorithms to better understand the context and underlying themes of news pieces.

This analysis uses a dataset from The Guardian, a renowned newspaper based in the United Kingdom. The dataset, comprising approximately 300,000 news articles, was gathered through API calls to The Guardian's open platform: https://open-platform.theguardian.com/.

1.1 Motivation

Data-driven news article analysis is essential because the value of data-driven news article analysis lies in its ability to offer quantitative insights that enhance our qualitative understanding, improving our capacity to make informed decisions. The reasons are as follows:

Deeper Insight: By converting news content into measurable data, both organizations and individuals can gain a more profound comprehension of the information landscape.

Strategic Decision-Making: For professionals across various sectors, data-driven news analysis can inform strategic choices by providing insights into identifying trends and tracking the reach of specific news topics.

Optimized Resource Allocation: Data-driven insights enable organizations to allocate their resources more efficiently, concentrating efforts on areas of significant interest or demand.

1.2 Objective

The primary objective of this study is to uncover and analyze data-driven articles published on The Guardian website over a five-year period from 2019 to 2024. By compiling a dataset of articles, we intend to conduct an analysis to identify significant trends and shifts in the topics that have captured public interest and media attention during this period.

In this project, we have several scopes as follows:

- Trend Analysis - Understand and capture popular topic trends by analyzing articles from different periods
- Event Frequency - Analyze the frequency of specific events over time
- Topic Identification - Predict a topic to a corresponding section
- Sentiment Analysis - Evaluate the sentiment of articles by each pillar

By doing so, we can provide insights for people to gain a deeper understanding of the topics that have consistently captured public attention, helping readers and researchers identify what issues are most relevant to society over the specified period.

2. Data Sources

To study the new articles of the last 5 years, the Guardian provides API that allows us to call for retrieving the information of the news articles, including the articles' published date, URL, and content.   To (objectives of the project), only 5 out of the 19 fields are selected for processing and analysis (see Figure 1). The fetched dataset was 2.27GB in size, containing over 375,000 records.

API Field Name	Local Field Name	Description
webTitle	Title	The title of the article (unstructured dataset)
webUrl	Link	Full URL of the article
webPublicationDate	Published	The publication date and time of the article in ISO 8601 format (single value)
fields -> bodyText	Content	The main text content of the article (unstructured dataset)
pillarName	Pillar	The name of the editorial pillar (e.g., "News," "Opinion," "Lifestyle") (single value)

sectionId	Section	The identifier for the section or category of the article (e.g., "world," "sport") (single value)
Type	Type	Nature of the content (e.g., article, video, gallery) (single value)

Figure 1. Selected article fields for analysis

2.1 Data fetching using a paralleled approach

The Guardian API enforces 200 articles per page, 38,000 articles per day limit, making the process of fetching a dataset spanning 4 years with over 375,000 articles extremely time-consuming, as each page is fetched one by one and each request requires time to respond.

To address the time-consuming issue of data fetching, parallel computing is implemented in the data-fetching process using Python's multiprocessing library. Instead of sequentially fetching articles page by page, each worker is assigned a specific date and is responsible for fetching articles within that date. If there are more than 200 articles on a specific date, the worker will paginate through additional requests until all articles for that date are retrieved, respecting the API's limit of 200 articles per page. After the workers finish obtaining the data, they proceed to write their collected articles to a shared CSV file. To avoid data corruption from concurrent writes, a shared lock is used among all workers to ensure exclusive access, where only one worker can write at any moment, avoiding data corruption from concurrent writes (see Figure 2).

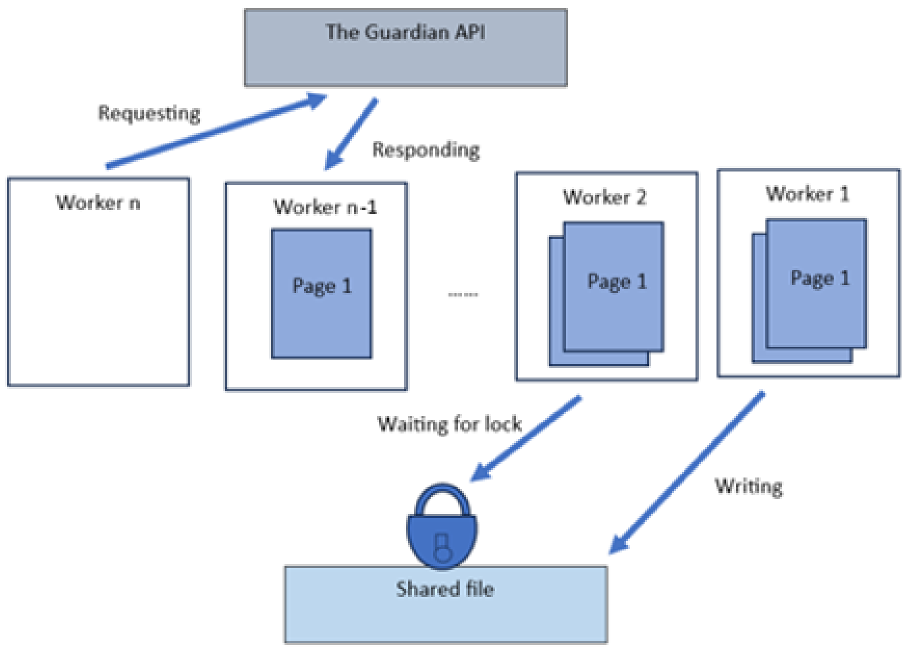


Figure 2. An Overview of the Data-Fetching Process

To address the daily API call limit per registered key, multiple keys are registered using different email accounts. To ensure that data is neither duplicated nor missing across different fetch processes, each fetch operation is controlled and tracked by a specific date range. The start and end dates are set to the first and last days of each month, and the total articles fetched within this period are limited to a maximum of 38,000. This approach keeps data fetching within API limits while allowing efficient tracking and management of each dataset batch.

By adapting parallel computing in the data-fetching process, some workers were able to write to their corresponding shared CSV file while others waited for server responses. As a result, time can be utilized more efficiently, as writing and fetching can occur simultaneously across multiple processes.
To further demonstrate the impact of parallel computing, comparisons of data-fetching times were made across different numbers of workers on the same amount of data (see Figure 3). Results show that parallel computing can save up to 85.7% of the time required for sequential processing. However, the results also align with Amdahl's Law, as an exponentially decreasing trend in computational time is found as the number of workers increases, reflecting diminishing returns with more workers.

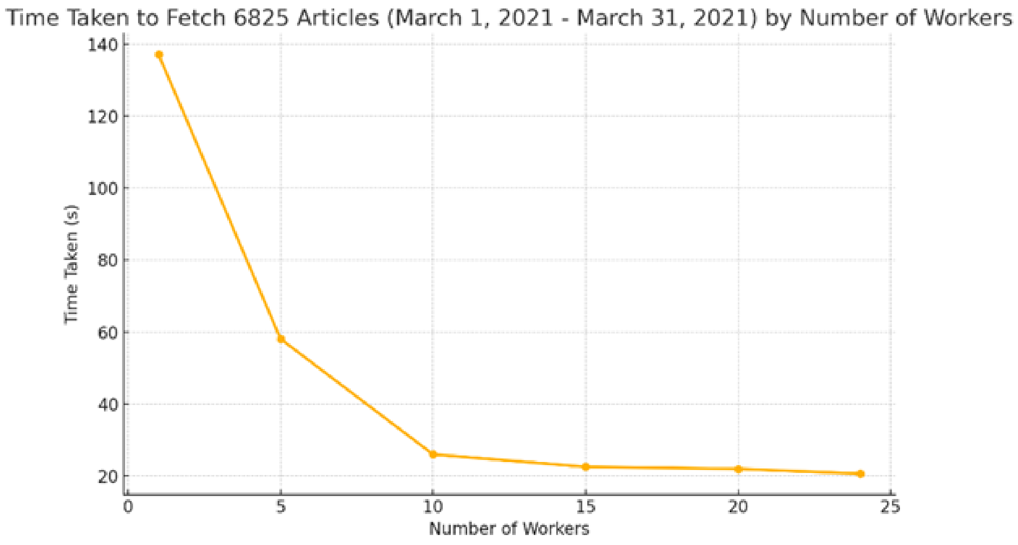


Figure 3. Impact of Worker Count on Data Fetching Time Efficiency

After successfully fetching the data into different CSV batches, they are merged into one.

2.2 Data Cleaning

Due to limited knowledge about the merged CSV file, it was difficult to apply data cleaning directly to the dataset. To gain a general understanding of the dataset, an analysis of the word count for each article's content was conducted using a parallel computing approach (see Figure 4). The parallel computing approach in this task utilizes Python's ProcessPoolExecutor with the chunk size set to (100,000 rows) to process the

CSV file concurrently. By submitting each chunk to the executor, the script was able to perform article content reading in parallel, thereby speeding up the processing time.

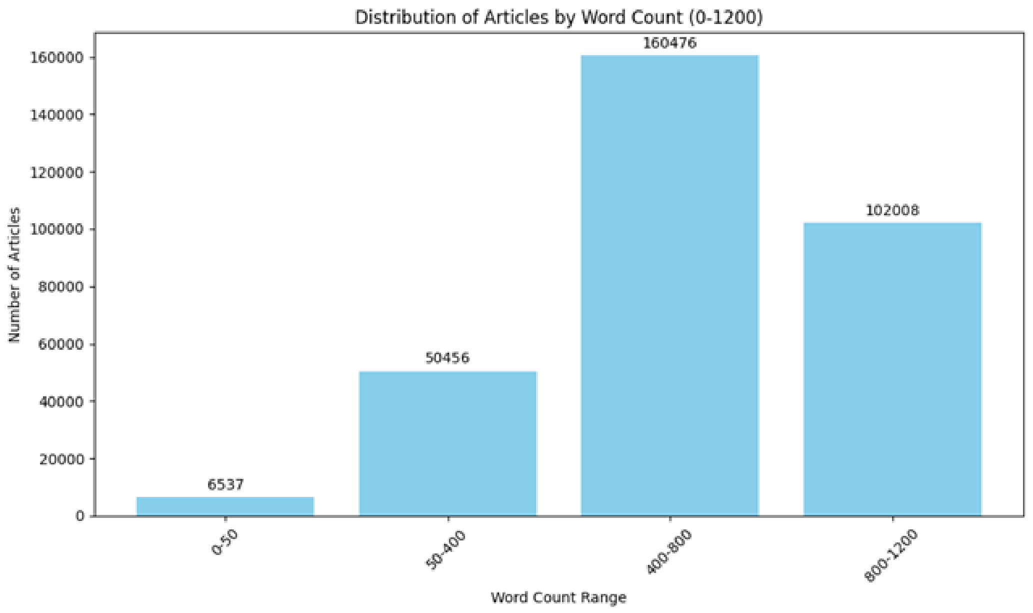


Figure 4. Distribution of Articles by Word Count

The word count analysis revealed a noticeable number of short-length articles (0-50 words) in the dataset. To investigate whether these articles contained sufficient information to be considered newsworthy, a random sample of 10 short articles was manually reviewed. Fortunately, this sample revealed some issues in the current dataset: out of the 20 articles, 19 were non-informative content such as advertisements, and 1 was in a non-English language, resulting in a total of 100% undesirable articles (see Figure 5). With such a high rate of non-informative content in the random samples, it can be concluded that the short articles are mainly filled with irrelevant content.

Issue	Sample
Non-English content	随着中国出台措施，尝试控制和防止新型冠状病毒的传播 …
website redirection message	Click here to access the print version Click here for rules and requests and T&amp;Cs
Advertisement	Molly Oldfield hosts Everything Under the Sun, a weekly podcast answering children’s questions, out now as a book. Does your child have a question? Submit one here

Figure 5. Issues and Samples from Random Sampling

Therefore, the data cleaning process removed a total of 7,719 records with missing fields, content containing fewer than 50 words, or non-English content. For non-English content removal, articles with less than 80% English vocabulary (excluding symbols) were considered non-English.

3. Content and Title Analysis of News Articles

In this section, the Content and Title fields for each news article will be explored and analyzed. To process such a large amount of unstructured data, these data will be transformed into structured data by "Bag of Words", then TF-IDF and Unique Word Count will be performed on Content and Title datasets.

3.1 Bag of Words (BOW)

To enhance the data analysis later on in this project, it is needed to decide on a format for storing BOW clearly and efficiently.

If the BOW output is stored in a Document-Term Matrix with over 375,000 articles, assuming the "Content" field of each article contains 30 unique vocabularies (given the mean word count is 1010.2465, unique vocabulary is around 3% of the article content) and each cell of the CSV file takes 4 bytes, the BOW file output will contain 375,000 rows and 11,250,000 columns with size 16.875 TB, which is unmanageable for the current scale of the project. However, if the BOW is stored as a Sparse Matrix, the output file will contain 378,750,000 rows while having only 3 columns (article_id, word, count), resulting in a 4.54 GB file output (see Figure 6). Therefore, the BOW will be stored as a Sparse Matrix due to its significantly less space usage.

Format	Row Count	Column Count	Size per Cell	Final Size
Document-Term Matrix	375,000	11,250,000	4 bytes	16.875 TB
Sparse Matrix	378,750,000	3	4 bytes	4.54 GB

Figure 6. Document-Term Matrix vs Sparse Matrix Under Assumption

Apache Spark is used to generate the BOW for the Content and Title fields due to its capability for parallel computation and automatic resource management. This allows for efficient utilization of computational resources, making it an ideal choice for processing large datasets effectively. Also, Apache Spark supports many libraries for text processing such as StopWordsRemover and RegexTokenizer, making it a versatile framework for handling BOW tasks.

The process of generating a Bag of Words (BOW) representation using Apache Spark begins with initializing a Spark session, allocating 12 GB for both driver and executor memory. The news article dataset is then loaded into a Spark DataFrame for processing. Preprocessing steps include adding a unique index to each document, converting the text in the "Content" or "Title" column to lowercase for consistency, and tokenizing the text into individual words using a regular expression tokenizer. Common stop words are removed to clean the data further and the CountVectorizer is then applied to convert the tokenized words into a sparse vector representation, capturing the frequency of each word in each document. Afterward, word counts were extracted the word counts from this vector, which were then exploded into individual rows containing the document index, word, and count. Finally, the processed Bag of Words data is saved as a CSV file, and the Spark session is terminated to free up resources.

3.2 TF-IDF Analysis

To gain insights regarding the content and title of the news articles, Term Frequency - Inverse Document Frequency (TFIDF) analysis is performed on the generated BOW. TFIDF is a text data analysis technique that computes a score for each word in BOW (see Eq. 1), where a higher TFIDF score indicates the word appears more often in a document but less often across other documents. By computing and obtaining the words with the highest TFIDF scores, the most significant and distinctive terms that characterize the content or title of each news article can be identified, providing better insight into the unique topics and themes present in the dataset.

$$TF - IDF = \underbrace{\frac{count}{total\ word\ in\ the\ document}}_{\text{Term Freqency}} \times \underbrace{\log(\frac{total\ document}{document\ frequency})}_{\text{Inverse Document Frequency}}$$

Eq 1. TF-IDF Formula

After the TFIDF score is computed for each word in the BOW, the top 100 unique words with the highest TFIDF score are used for observation. To obtain the top TFDIF words, the newly generated output file containing TFIDF scores was passed to Pig Latin, where these words will be abstracted using query language statements (see Figure 7).

Query Statement	Purpose
data = LOAD 'hdfs://localhost:9000/user/bitnami/part-00000.csv' USING PigStorage(',') AS(id:int, word:chararray, tfidf:float);	Load the BOW from the Distributed file system

grouped_words = GROUP data BY word;	Group the dataset into unique words
max_tfidf = FOREACH grouped_words GENERATE group AS word, MAX(data.tfidf) AS max_tfidf;	Assigning the TFIDF score to each of the unique words based on their highest TFIDF score
sorted_tfidf = ORDER max_tfidf BY max_tfidf DESC;	Sort the unique words by their TFIDF score in descending order
top_100_words = LIMIT sorted_tfidf 100;	Select the top 100 unique words after sorting
STORE top_100_words INTO 'hdfs://localhost:9000/user/bitnami/top_100_words_output' USING PigStorage(',');	Save the abstracted 100 unique words and their corresponding TFIDF scores

Figure 7. Query Statements of Abstracting the Top 100 Unique Words by TFIDF Using Pig Latin

From the output of Pig Latin, the word cloud of the title and content of news articles can be generated, with the size of words proportional to their TFIDF score (see Figure 8). However, the TFIDF word clouds did not yield valuable insights into important events or trends in the world. Instead, they appeared to aggregate words lacking meaningful connections, presenting a collection of random and insignificant terms.

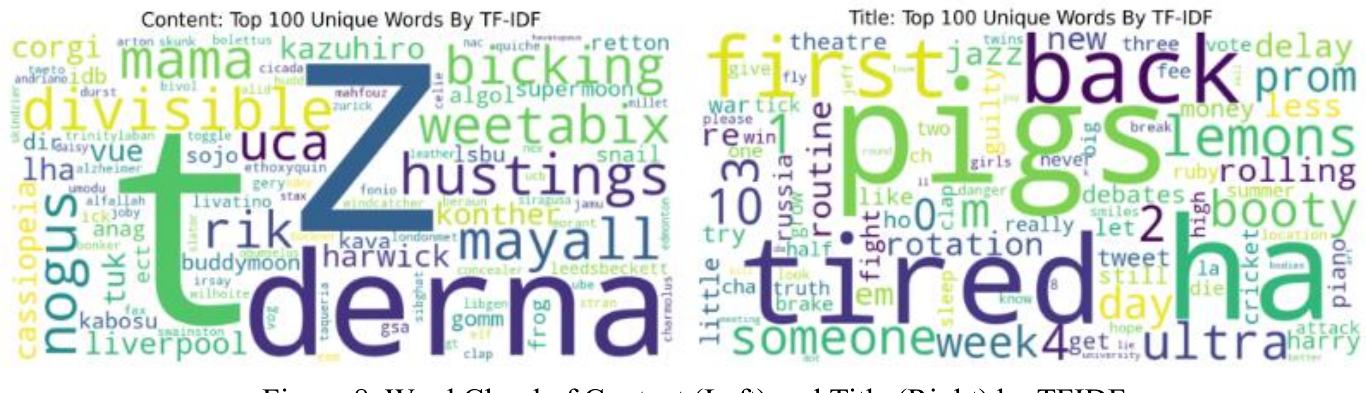


Figure 8. Word Cloud of Content (Left) and Title (Right) by TFIDF

One explanation for such a pattern in TFIDF word clouds could be the definition of TFIDF and the nature of news articles. According to Kwak et al. [1] in their study on the digital news ecosystem, online news platforms have lowered barriers to entry in journalism, broadening the marketplace of ideas, which fostered competition that incentivizes clickbait and plagiarism, undermining the quality of journalism. Therefore, significant events may be reported repeatedly in the news articles over time to attract viewers, which diminishes the uniqueness of their associated keywords. Since TFIDF assigns scores to words based on their uniqueness across documents, it may not effectively capture keywords related to world events or trends due to the repetitive nature of news articles.

3.3 Unique Word Count Analysis

As we found that the result from TF-IDF is not meaningful, we counted the top 100 most frequent unique words instead, which were abstracted by Pig Latin statements (see Figure 9). The top 100 most frequent unique words are separately collected from article topics and contents.

Query Statement	Purpose
data = LOAD 'hdfs://localhost:9000/user/bitnami/part-00000.csv' USING PigStorage(',') AS(id:int, word:chararray, count:int);	Load the BOW from the Distributed file system
grouped_words = GROUP data BY word;	Group the dataset into unique words
sum_count = FOREACH grouped_words GENERATE group AS word, SUM(data.count) AS sum_count;	Sum up the count of each of the unique words
sorted_count = ORDER sum_count BY sum_count DESC;	Sort the unique words by their frequency in descending order
top_100_words = LIMIT sorted_count 100;	Select the top 100 unique words after sorting
STORE top_100_words INTO 'hdfs://localhost:9000/user/bitnami/top_100_words_output' USING PigStorage(',');	Save the abstracted 100 unique words and their corresponding frequency

Figure 9. Query Statements of Abstracting the Top 100 Unique Words by Frequency Using Pig Latin

The results of the top 100 frequent unique words from article contents and titles are shown in Figure 10. From the most frequent words in article contents, some insights are slightly better compared to previous word clouds. We can find some keywords from the word cloud such as “government”, “work” and “covid”. However, the content of articles still struggles to provide obvious keywords because those words are common. From the most frequent words in article titles, insights are more significant as some words, such as “uk”, “us”, “covid” and “trump”, are keywords of articles. These words are crucial for further analysis below.

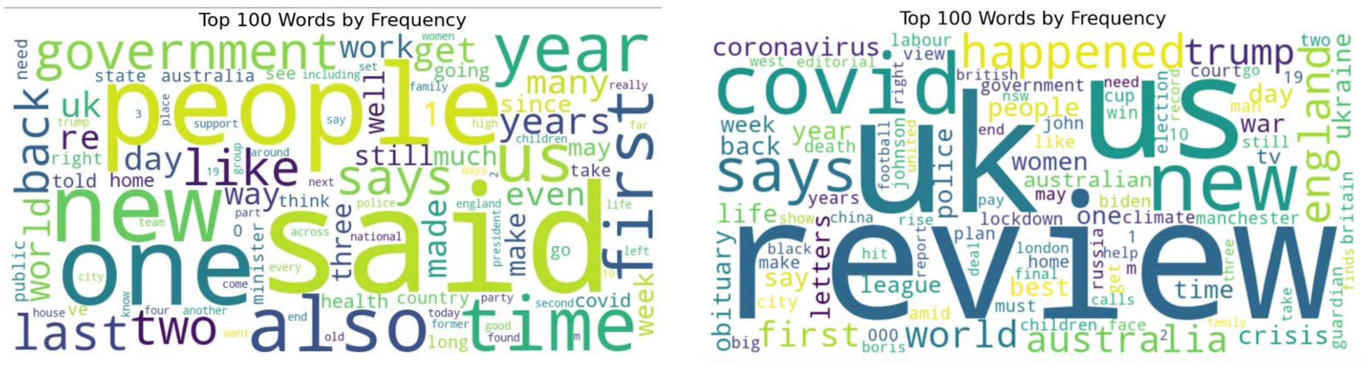


Figure 10. Word Cloud of Content (Left) and Title (Right) by BOW

Besides finding keywords, we also filtered out all country names from the BOW dataset and made a word cloud (see Figure 11) to show the top 100 countries that are frequently mentioned in articles. The countries with the highest frequency mentioned in article titles are namely “us”, “uk”, “ukraine”, ”australia”, “israel” and “china”.

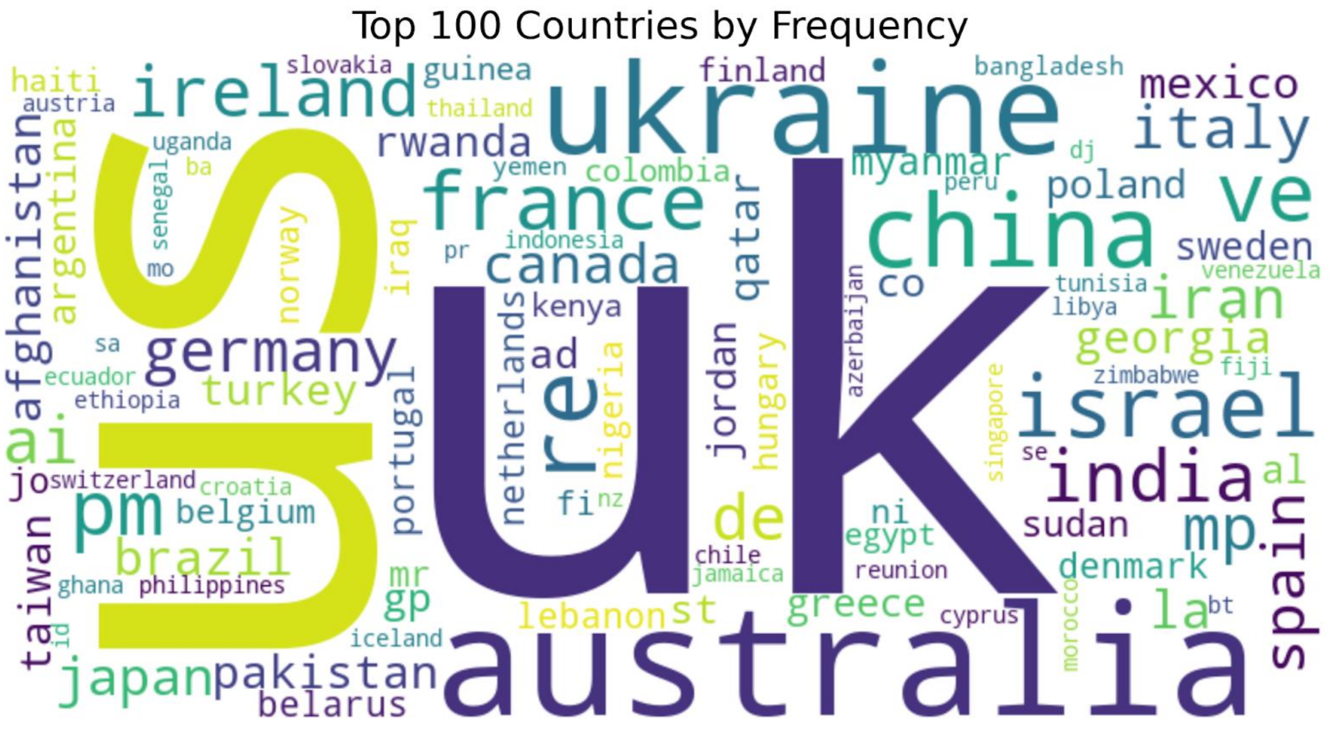


Figure 11. Word Cloud of Top 100 Countries Mentioned in Article Titles

3.4 Time Series Analysis

Based on the findings from word cloud pictures, we want to analyse the news trends by selecting keywords from news article titles. Thereby we processed keyword matching with news article titles from the dataset. The process steps are as follows:

1. Identify and select several keywords from word clouds
2. Decide a time frame from November 2019 to April 2024
3. The program with parallel computing will record articles if their titles match the keywords
4. For each period, the program will count the number of news articles that keywords are included
5. Generate time series graphs

Here are some time series showing the trend of news articles.

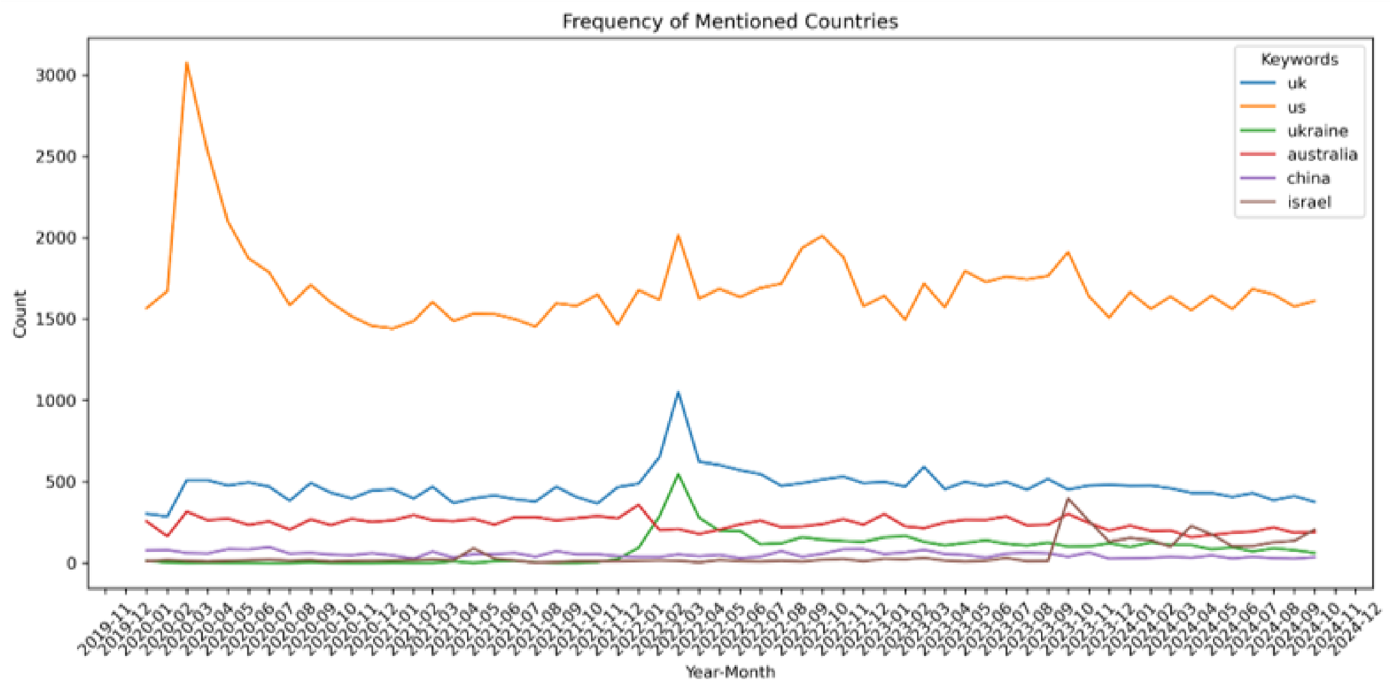


Figure 12. Word Cloud of Top 100 Countries Mentioned in Articles

In Figure 12, the United States has the highest number of these article trends, followed by the United Kingdom and Australia. In the meantime, there is one significant observation in February 2022. Countries such as the United States, the United Kingdom, and Ukraine experienced similar peaks during this period, indicating that some special events closely related to these countries occurred. Also, in October 2023, countries such as the United States and Israel experienced similar peaks during this period, showing there is a relationship between them.

Based on the observations in Figure 12, we found some relationship between those countries from the article trends. Therefore, we further selected more non-country name keywords to get more insights from articles. To have a deeper insight, we just selected Ukraine for a specific country keyword and included other keywords namely “covid”, “war”, “trump”, “biden ” and “election” according to the word cloud in Figure 10. The result of keyword frequencies of article trends is shown in Figure 13.

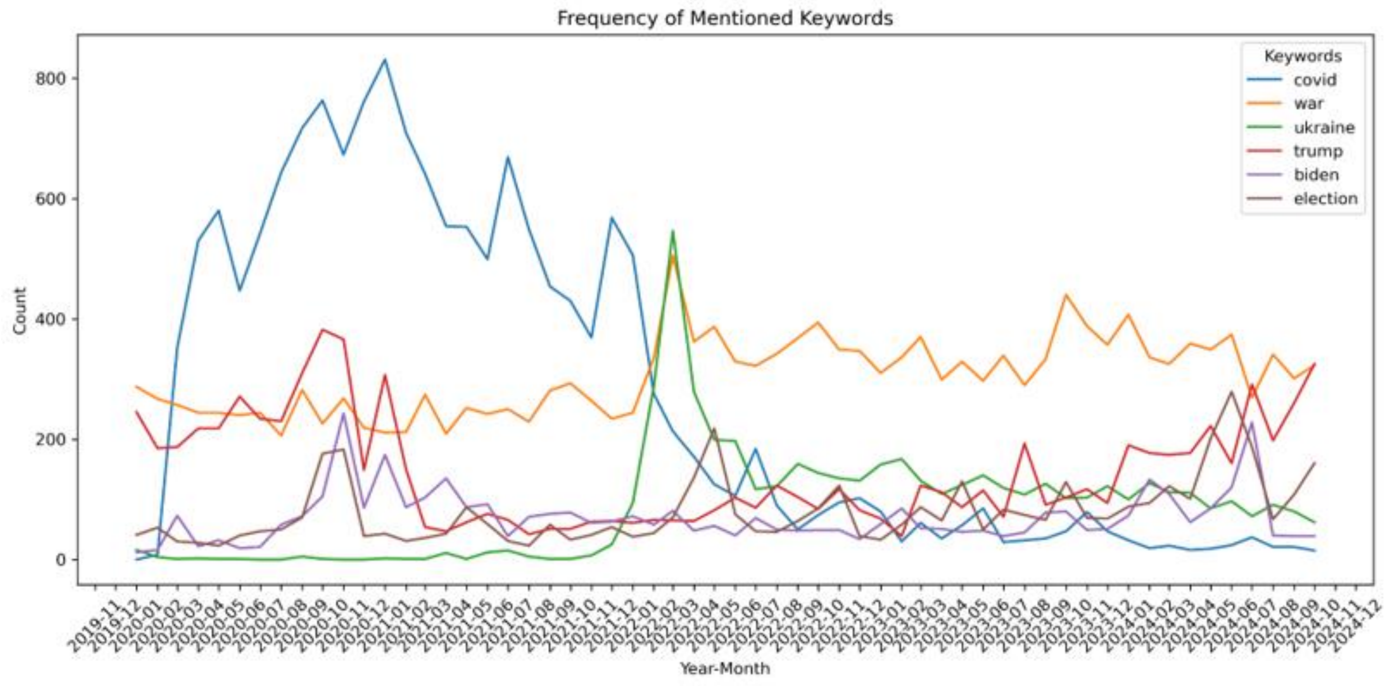


Figure 13. Word Cloud of Top 100 Keywords Mentioned in Articles

In Figure 13, we observed that the keyword “covid” has the highest trend from November 2019 to February 2022; however, we cannot find any keyword related to covid, meaning that this is just a specific event. Also, there is a significant peak for the keywords “Ukraine” and “war” in February 2022 as their peaks are very close to each other during the period. A possible finding is that the special event experienced by the United States, the United Kingdom, and Ukraine is related to the war, explaining there is a close relationship between these countries and the keyword “war” during the period. Besides, we observed that the keywords “trump”, “biden” and “election” have a relationship as they have similar peaks from August 2020 to February 2021 and May 2024 to June 2024. A possible explanation is that election events drew people’s attention. Trump and Biden were critical representatives of the events during these periods.

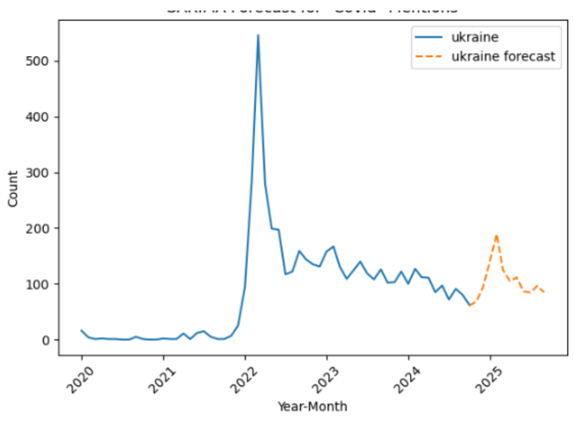
From time series graphs, we can observe some similar trends in articles with keywords to get insights. There are relationships between keywords, and these relationships may disclose significant information about events during the periods. For example, the Ukraine War happened in February 2022, and countries such as the United States, the United Kingdom, and Ukraine are critical participants in this event. Also, for the election events from February 2021 to May 2024 and from May 2024 to June 2024, Trump and Biden are critical representatives in these events as they are mentioned in many articles during the periods. Therefore, article trends can reflect the phenomena of the event’s popularity. The trends may also provide more information about the events by showing relationships between keywords.

3.5 Time Series for Trend Prediction

To predict the trend of keywords in a further year, we have applied the time series forecasting model using SARIMA, which stands for Seasonal Autoregressive Integrated Moving Average. The model handles data with seasonal patterns and the formula is as follows:

$$y_t = c + \sum_{n=1}^{p} \alpha_n y_{t-n} + \sum_{n=1}^{q} \theta_n \epsilon_{t-n} + \sum_{n=1}^{P} \phi_n y_{t-sn} + \sum_{n=1}^{Q} \eta_n \epsilon_{t-sn} + \epsilon_t$$

Eq 2. SARIMA Formula where p is the number of higher-order AR terms, q is the number of lagged forecasting error MA terms, P is the number of seasonal AR terms and Q is the number of seasonal MA terms.



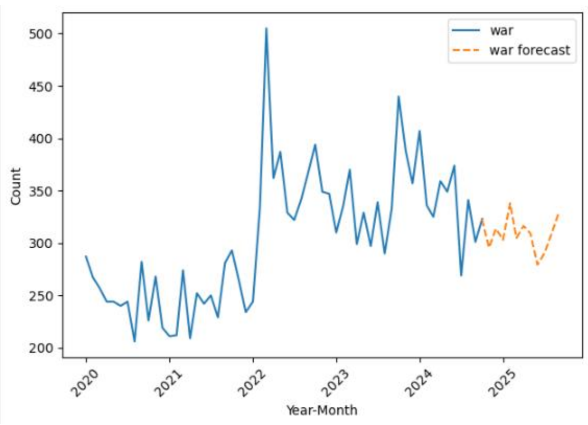


Figure 14. Trend Prediction of keywords “ukraine” (Left) and “war” (Right) by SARIMA

Based on the graphs in Figure 14, we can analyze the trend of the keywords “ukraine” and “war” in 2025. For the keyword “ukraine,” the historical trend shows a significant spike in interest around 2022, likely due to geopolitical events. This spike is followed by a decline and subsequent stabilization. For the keyword “war,” the historical trend displays fluctuations with noticeable peaks and troughs over time. This suggests that the topic of war may continue to be significant, possibly due to ongoing or emerging conflicts.

3.6 Sentiment Analysis

To draw people’s attention, most online news platforms have developed strategies to express opinions regarding news entities. The sentiment can be reflected in articles. In this way, we can analyze the titles of articles to understand what kinds of articles are more likely to cater to readers’ interests.

In this experiment, we applied TextBlob from the Python library. Five dimensions comprising arts, lifestyle, news, opinion, and sport were selected for sentiment analysis from news articles. We analyzed the polarity of each article's title, which sentiment associated with the entity is positive, negative, or neutral. The process steps are as follows:

1. Text Preprocessing
2. Calculate the Polarity of sentiment words
3. Calculate the Total Sentiment Score of the Article Title Text
4. Show Sentiment Results in a table

Here is the table showing the sentiment score results of news article texts separated by article pillars.

Pillar	**Total Articles**	**Positive**	**Negative**	**Natural**	**Average Polarity**
Arts	63533	24226 (38.1%)	11040 (17.3%)	28267 (44.5%)	0.09257
Lifestyle	29389	11185 (38.1%)	4650 (15.8%)	13554 (46.1%)	0.11020
News	196101	50275 (25.6%)	38141 (19.4%)	107685 (54.9%)	0.01703
Opinion	25021	8410 (33.6%)	5874 (23.5%)	10737 (42.9%)	0.03013
Sport	52891	18095 (34.2%)	7792 (14.7%)	27004 (51.1%)	0.07768

Figure 14. Table of Article Sentiment Analysis

In Figure 14, the majority of articles in each category tend to have a neutral sentiment. For articles with a positive sentiment, the pillars of Arts and Lifestyle have the highest percentage (38.1%), followed by the pillar of Sport (34.2%). For articles with a negative sentiment, the pillar of Opinion has the highest percentage (23.5%), followed by the pillar of News (19.4%). For articles with a natural sentiment, the pillar of News has the highest percentage (54.9%), followed by the pillar of Sport (51.1%). Among the pillars, the pillar of Lifestyle has the highest average polarity, suggesting it generally has more positive content. However, the pillar of News has the lowest average polarity, indicating a more neutral or slightly negative sentiment. From this table, the sentiment distribution separated by article pillars can help us understand how media platforms tailor content to audience preferences and enhance engagement.

4. Content and Title Analysis of News Articles

Latent Dirichlet Allocation (LDA) is a powerful topic modeling technique used to discover abstract topics within a collection of documents. It works by identifying groups of words that frequently occur together, thus revealing underlying themes or topics in the text. In the context of news analysis, LDA is useful for uncovering the major topics discussed in different news sections, providing insights into the prevalent themes and key issues.

By finding the top 100 keywords in the news dataset based on their LDA probability, we can gain different insights into the dataset, giving an overview of its content.

4.1 Workflow

To find the top keywords, Spark is used to speed up the computation time of the LDA algorithm and word vectorization. Here is a simple workflow of the code:

1. Load the CSV file into a DataFrame (df), then select the relevant columns: Title and Section.
2. Start the Spark Session, removing stop words and symbols, and vectorizing the "Title" and "Section" using CountVectorizer.
3. Fit the LDA model (import from Spark's library) to the entire dataset using the feature vectors to identify topics within the dataset (k=5 topics).
4. Obtain the vocabulary list from the CountVectorizer model.
5. Describe the topics by using the LDA model to assign probabilities to terms within each topic.
6. Aggregate the term probabilities across all the topics to determine the overall significance of each word.
7. Map each word to its accumulated probability and extract the top 100 most significant keywords.
8. Save the top 100 keywords, along with their respective probabilities.

4.2 Result Evaluation

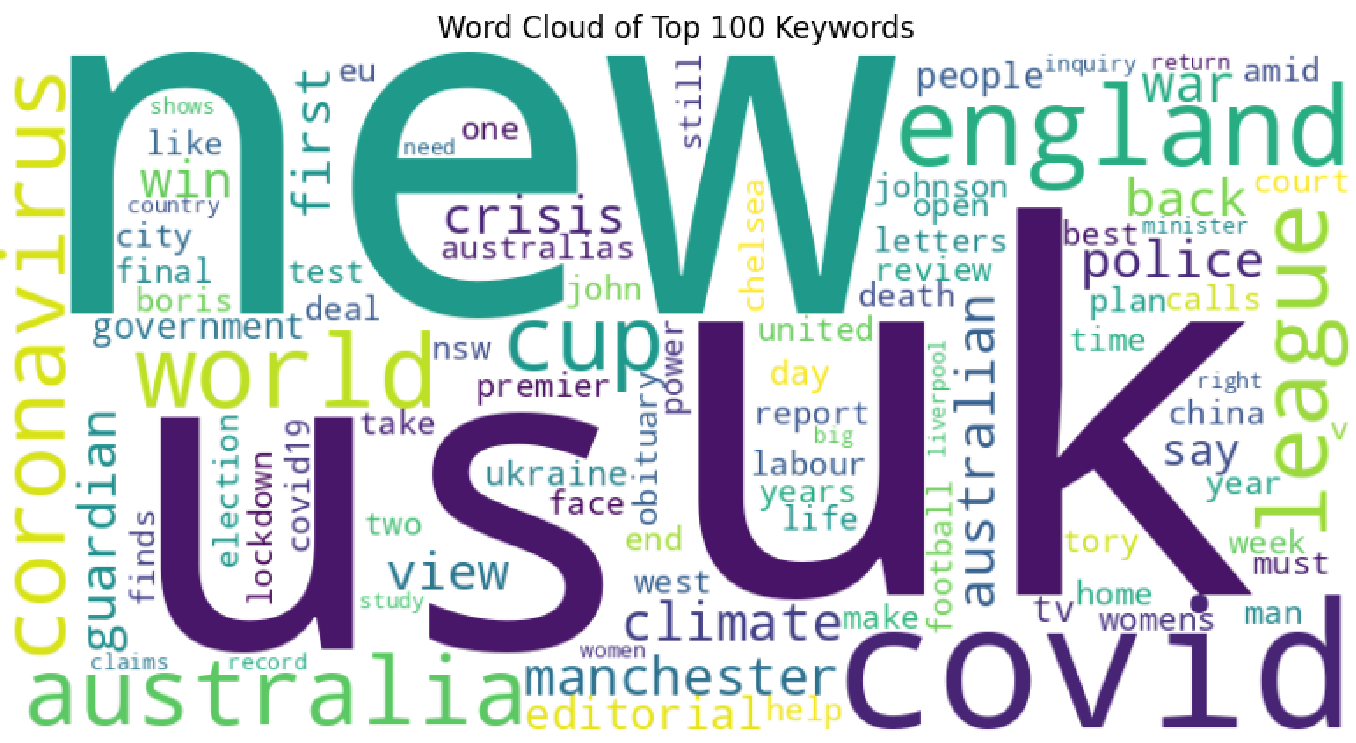


Figure 15. Word Cloud of Top 100 keywords based on the LDA model

From the word cloud, prominent keywords such as "UK," "new," "COVID," "coronavirus," and "world" reflect major topics covered in the news articles. This visualization effectively highlights the key themes and allows us to quickly identify the most discussed issues, such as the ongoing pandemic, geopolitical events, and regional news. The presence of terms like "Ukraine," "crisis," and "climate" further underscores the diversity of coverage in different global events and pressing issues.

5. News Sections Classifiers

Classification is a supervised learning technique that classifies undefined data into labeled categories. By training a classifier to classify news articles into different sections, it aids in categorizing unseen data in future studies.

Logistic regression model is a classification model that is used to predict the probability that an instance belongs to a particular class. It works well when the relationship between features and the target variable is roughly linear, making it simple and effective for binary classification tasks.
Naive Bayes is a probabilistic classification algorithm based on Bayes' Theorem. It assumes that all features are independent of each other, which makes it computationally efficient and particularly useful for text

classification problems.

These two models are powerful for text classification because they offer complementary strengths. Both models are computationally simple compared to deep neural networks while maintaining good accuracy. Additionally, the Spark machine learning library supports both models, reducing computational time by utilizing parallel computing techniques.

5.1 Workflow

To train a classifier, Spark is used to speed up the computation time of the LDA algorithm and word vectorization. Here is a simple workflow of the code:

1. Set a threshold to drop the section that the total number of articles in that section is lower than the threshold.
2. Start the Spark Session, removing stop words and symbols, and vectorizing the "Title" and "Section".
3. Convert the "Section" label into a numeric label using Stringindex
4. Split the data into a training set and testing set (8:2) and further collect 100 titles in each section for a more balanced evaluation as our dataset is heavy in "world" section
5. Put the word vectors into the Logistic Regression and Naive Bayes models (Import from spark's library)
6. Train the models, and use 3-fold cross-validation to obtain the performance of the models
7. Output the accuracy and confusion matrix of the models for evaluation

5.2 Model Evaluation

Accuracies / Models	Logistic Regression Model	Naive Bayes Model
Training Accuracy	78.1%	76.3%
Testing Accuracy	69.6%	70.0%

Figure 16. Table of Accuracy of Logistic Regression Model and Naive Bayes Model

From Figure 16, both models obtained similar results, the training accuracy of the logistic regression model is slightly higher than naive Bayes model while the testing accuracy of the logistic regression model is slightly lower than naive Bayes model.

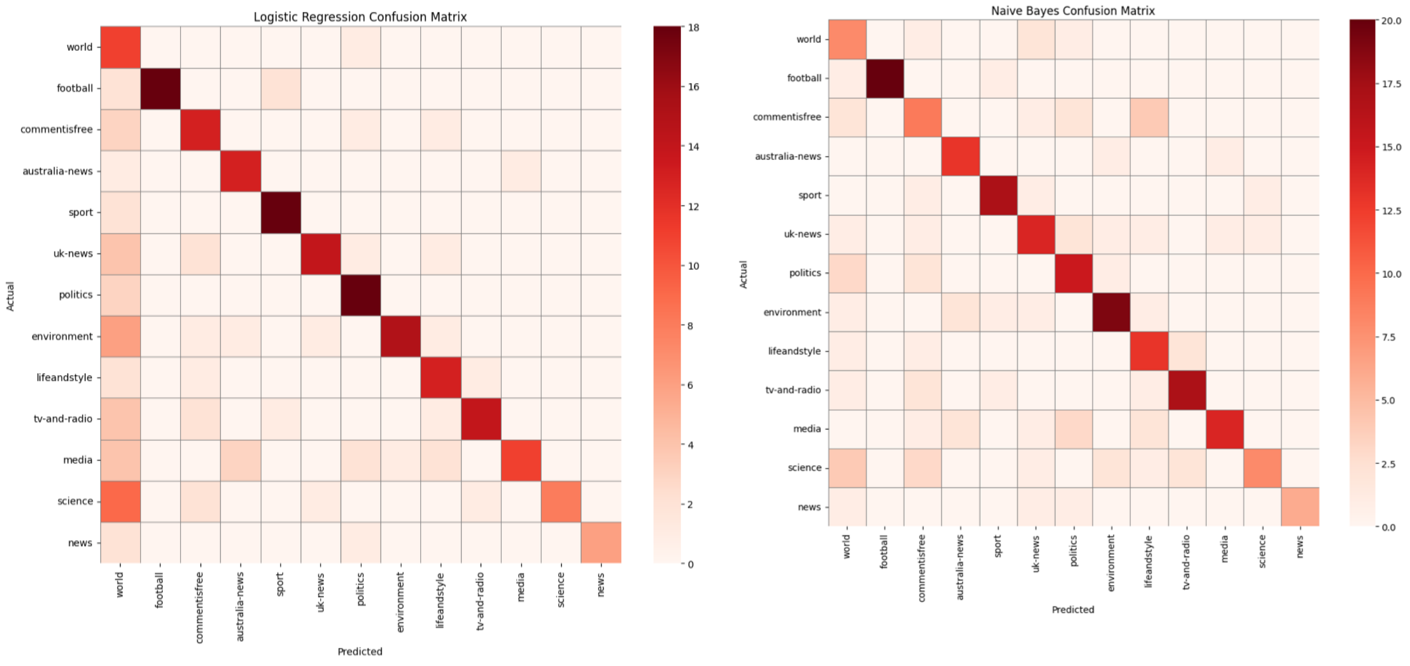


Figure 16. Confusion Matrix of Logistic Regression (Left) and Naive Bayes (Right)

In Figure 16, most sample points are concentrated along the diagonal, indicating that both models performed well in general. However, both models struggled to classify articles in the 'world' section, possibly due to the broad range of topics covered under this category. In the logistic regression confusion matrix, the deeper color in the first row compared to the Naive Bayes matrix suggests that logistic regression had more difficulty accurately classifying articles into the 'world' section.

6. Parallel Computing Analysis

6.1 Efficiency Analysis of Bag of Words and TF-IDF

A key aspect of our project was to analyze the performance benefits of using parallel computing for processing large datasets. The experiment compared two main approaches: Python for sequential processing and Apache Spark/Scala for optimized parallel computing.

6.2 Bag of Words (BOW) Processing

The Bag of Words (BOW) processing task was implemented in both Spark/Scala and Python. The performance comparison showed that the Python implementation, when run on our dataset of ~375k data points, required over 3000 seconds. Meanwhile, the Spark implementation completed the task in under 400 seconds, achieving a nearly 10x speedup.

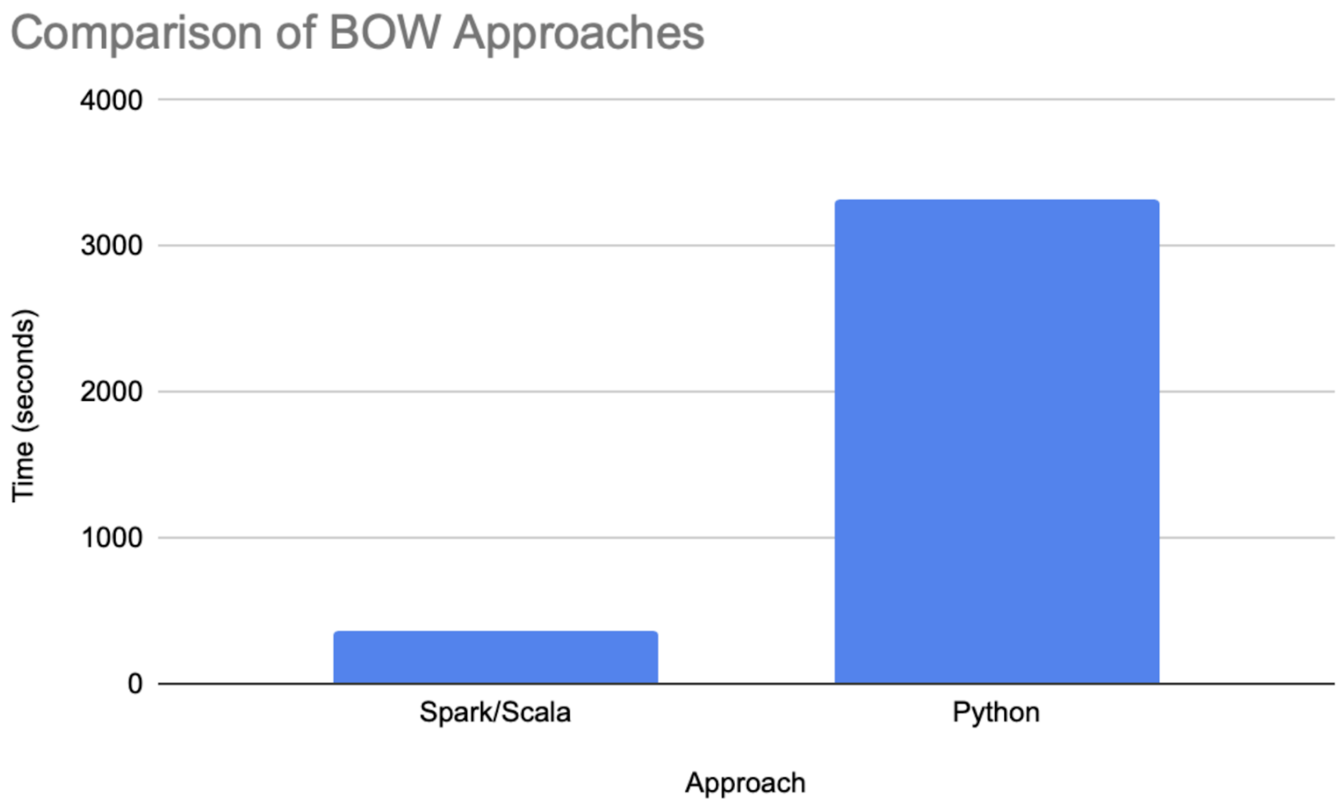


Figure 15. Efficiency Comparison for BOW Approaches

A flame graph was generated for the Python implementation to analyze its sequential execution. This revealed major bottlenecks in operations like removing duplicates and merging data, which contributed to most of the overall runtime.

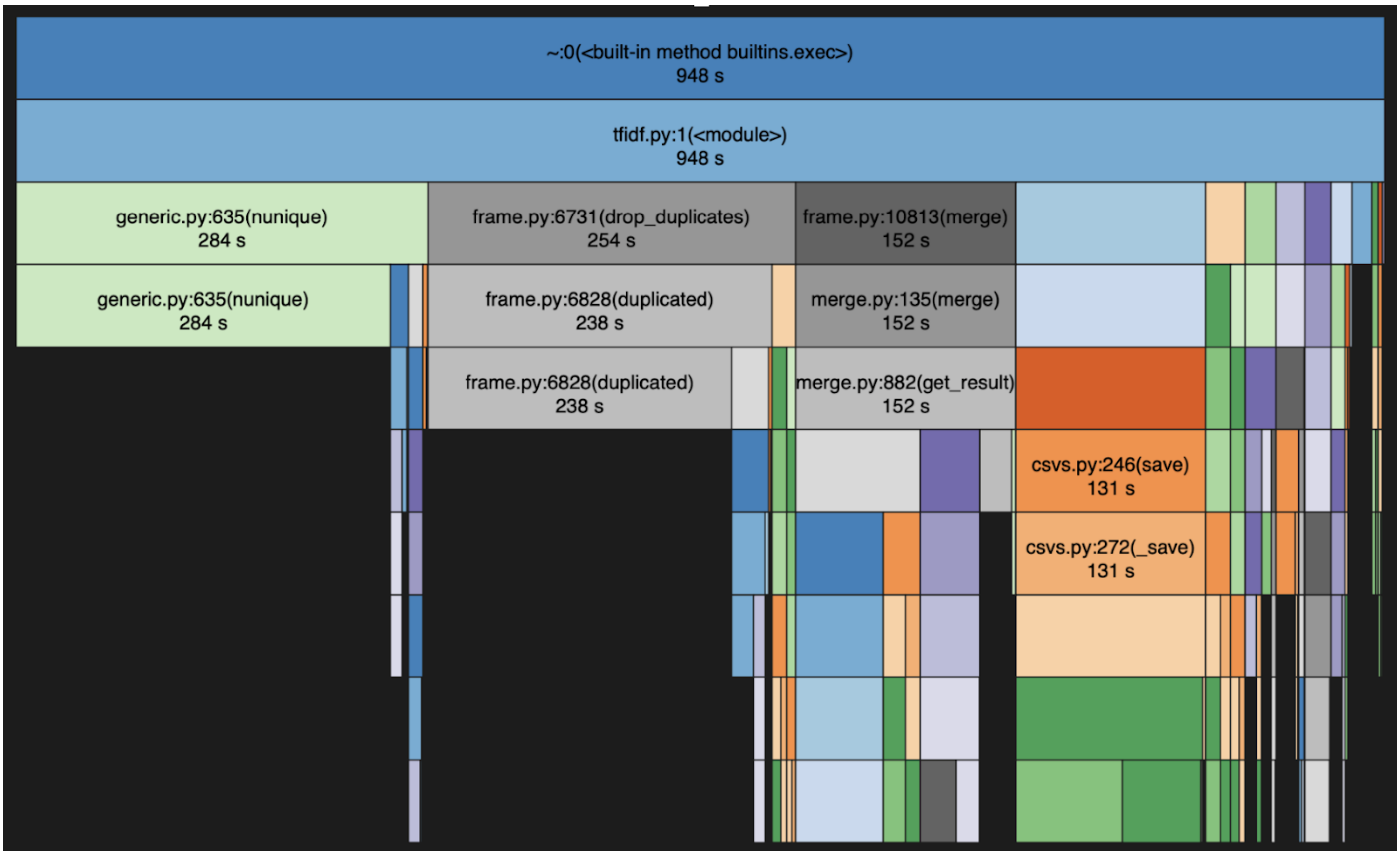


Figure 16. Flame Graph of BOW Python Execution

6.3 Term Frequency-Inverse Document Frequency (TF-IDF) Processing

Similarly to the BOW analysis, the TFIDF analysis showed a significant speedup when executed in Spark compared to Python (see Figure 17). This significant improvement is mainly caused by Spark's ability to efficiently parallelize independent steps, as visualized in the Directed Acyclic Graph (DAG) generated by Spark's execution engine (see Figure 18). The DAG highlights the execution flow and dependencies between operations by showing which tasks can be run concurrently.

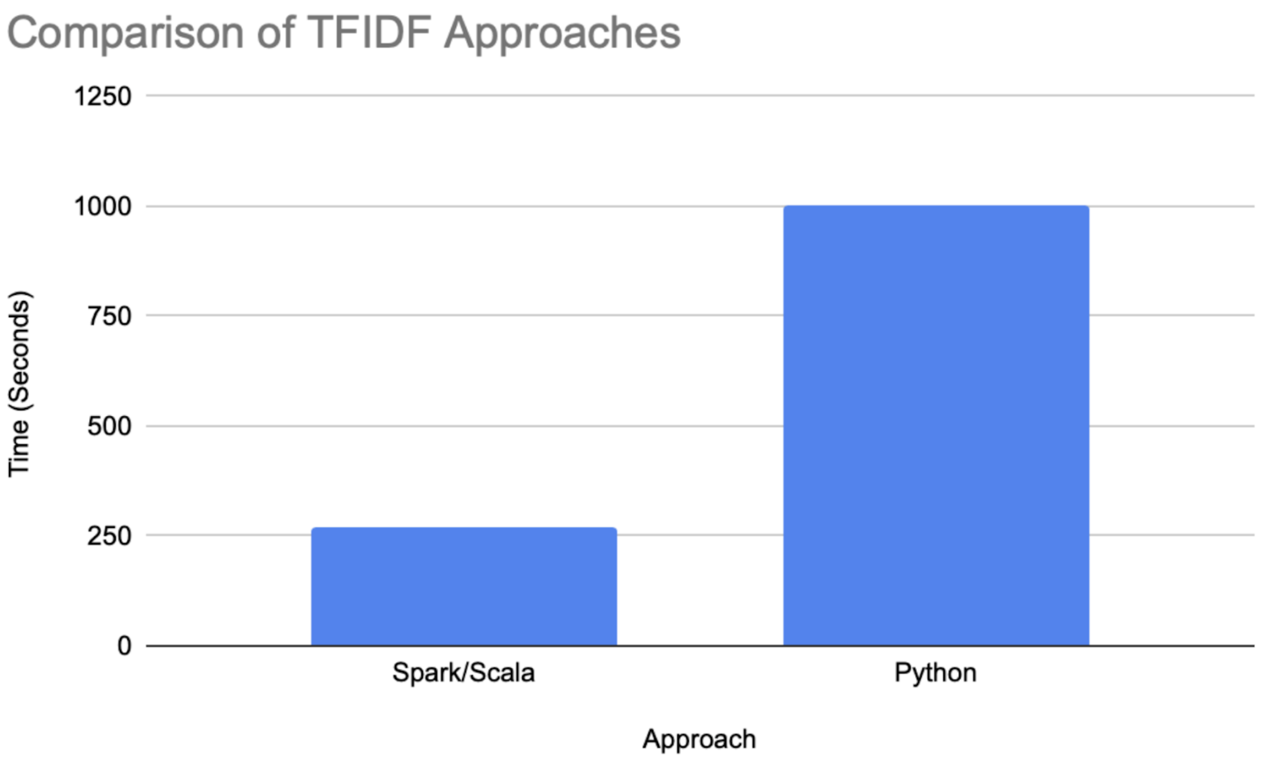


Figure 17. Efficiency Comparison for TF-IDF Approaches

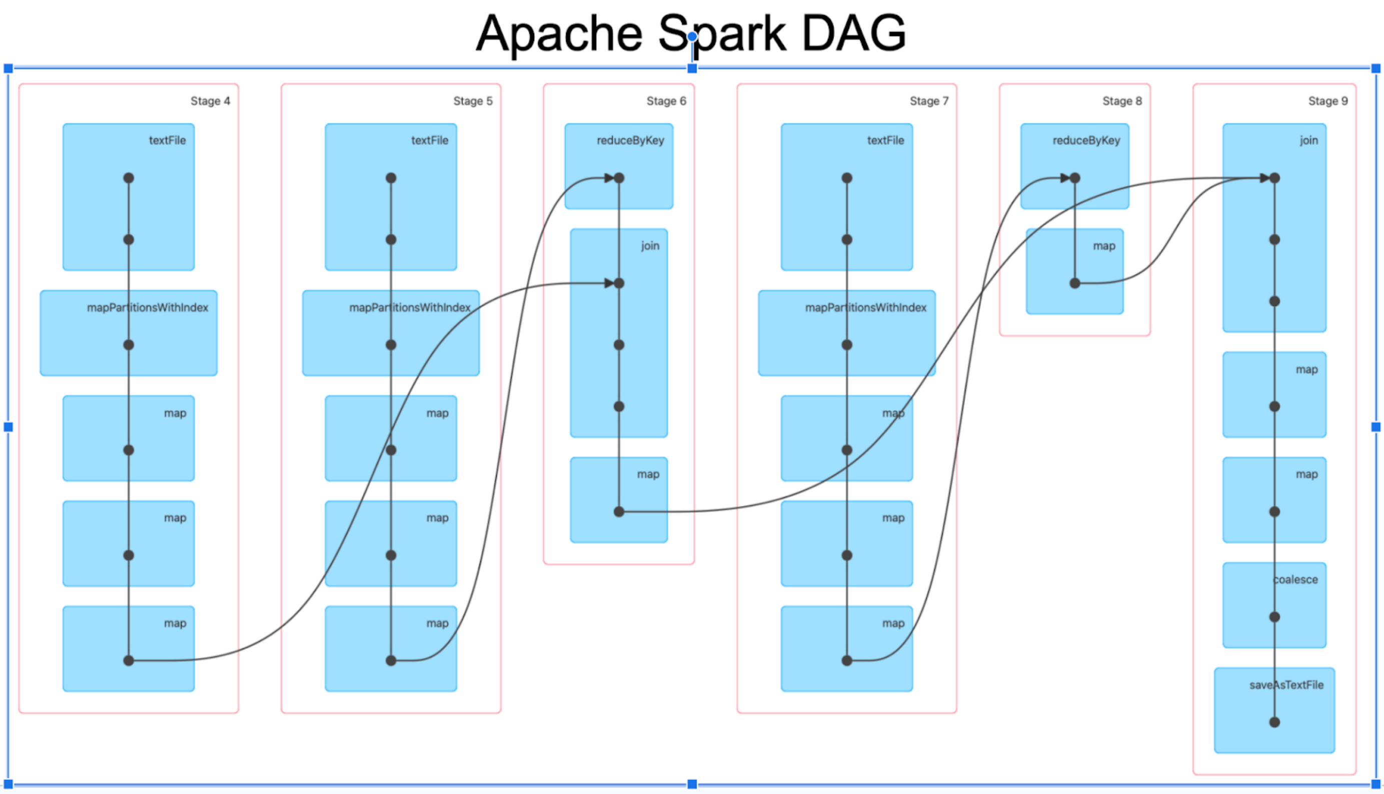


Figure 18. Apache Spark Directed Acyclic Graph

By profiling the TF-IDF computation performed by Apache Spark, we confirmed that the framework effectively parallelized tasks across multiple executors. By analyzing the Scala code profile (see Figure 19), we can observe that Spark efficiently divided the work into stages, such as tokenization, mapping, and reduction by key. The profiling data highlights the concurrent execution of tasks in a topologically sorted order of the DAG. This insured that independent tasks were processed simultaneously, significantly reducing the runtime compared to sequential approaches.

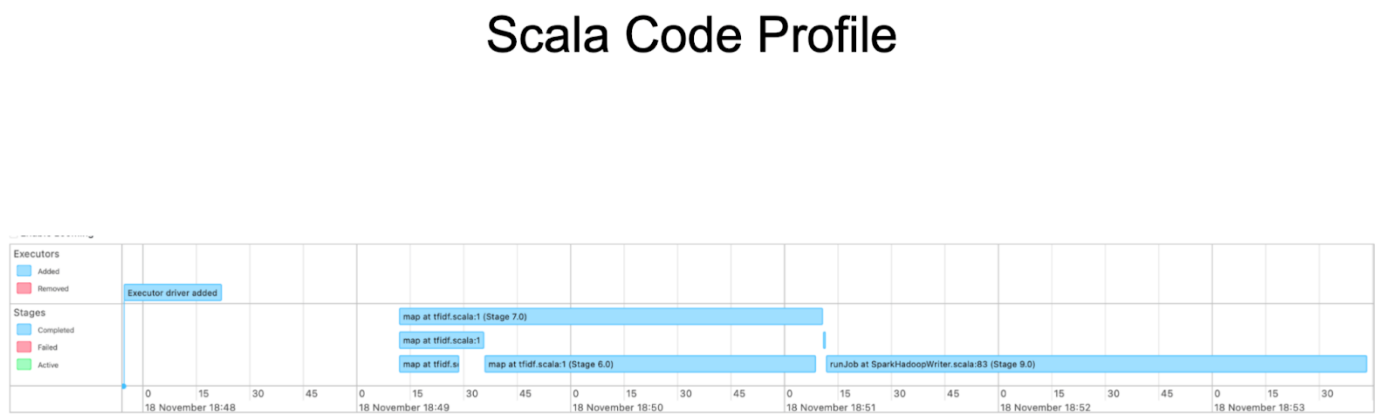


Figure 19. Scala Code Profile

7. Conclusion

In conclusion, this project demonstrates the application of various text data processing and analysis techniques, mostly enhanced by parallelization, on a large news article dataset fetched from *The Guardian*. The findings highlight several key insights:

1. Implementing different parallelization approaches in processing large-scale text datasets can significantly reduce computational time by utilizing local computing resources more effectively compared to sequential computation.
2. Unique word frequency has proven to provide better insights than TF-IDF due to the repetitive reporting nature of news articles. Keywords derived from unique word frequency are more closely aligned with current world events, making them more relevant and insightful.
3. Through NLP approaches in article text analysis, the topics of articles can reflect trends in event popularity and the significance of events across various keywords, uncovering relationships between keywords and providing clear, predictive insights for policy-making and decision planning.
4. LDA analysis created some different insight of the news article content compared to unique word frequency, providing a different scope of global event

5. News section classifier provides a different point of view of classifying new article, showing some shortage of text classification.

8. References

1. K. T. Kwak, S. C. Hong, and S. W. Lee, "An Analysis of a Repetitive News Display Phenomenon in the Digital News Ecosystem," *Sustainability*, vol. 10, no. 12, Art. no. 4736, 2018. [Online]. Available: https://doi.org/10.3390/su10124736.

9. Appendix

Individual Contribution

Individual Contribution Table

Student Name	Participation Tasks	Participation Percentage
Twan Tsz Yin	• Data Fetching • Data Processing • BOW • TFIDF Analysis	31%
Chua Ka Hiu	• LDA Analysis • News Sections Classification Modeling • Parallel computing	31%
LI Yiu Hang	• Word Cloud Analysis • Time Series Analysis • Time Series Prediction • Sentiment Analysis	31%

Matthew Zhang	• Parallel Computing Analysis	7% • Not responsive • Barely participated in the regular group meetings • Did not demonstrate individual problem solving ability • Unable to complete assigned tasks • Unable to provide source code after completing task

Individual Contribution Statement

STUDENT ID:57149200
STUDENT NAME: Twan Tsz Yin
My role in this project involved data fetching, parsing of text data for the “Title” and “Content” into Bag of Words, TFIDF computation, and keyword abstraction.

In data fetching, I designed a parallel fetching approach with workers fetching a small amount of data per request to bypass the restrictions while enhancing the data fetching process. After the abstracted data was fetched locally, general data cleaning was performed to ensure the data quality for other groupmates to perform their analysis. This process involved word counting, categorization, and random sampling using parallel approaches.

In the text data parsing, I decided to transform the text data in “Title” and “Content” into a Bag of Words and represent it as a sparse matrix due to its theoretically better space efficiency. Apache Spark was used to perform these tasks for parallelization.

The TFIDF score for each unique word in every article was also computed using Apache Spark to enhance parallelization and reduce computational time.

For keyword abstraction, the datasets were moved to the provided virtual machine, where Pig Latin was used to group, rank, and abstract the top unique words by TFIDF score or frequency, and also to facilitate computation speed in distributed file systems.

Overall, this project provided me with valuable experience in implementing different data processing techniques within distributed file systems to enhance task parallelization and computational efficiency, introducing me to a new area I had never experienced before.

STUDENT ID: 57142664
STUDENT NAME: Chua Ka Hiu, Angus

Throughout the project, my main contributions included implementing the Latent Dirichlet Allocation (LDA) analysis and developing the news section classifier. This involved using data filtering techniques, Apache Spark, and natural language processing (NLP). The project allowed me to practice and apply the knowledge I gained from coursework, while also providing valuable experience in NLP, data analysis, and parallel computing. This experience deepened my understanding of parallel computing concepts and demonstrated the power of distributed computing. Additionally, I learned a great deal from collaborating with my teammates, both in terms of dividing tasks effectively and engaging in meaningful project discussions.

STUDENT ID: 40156222
STUDENT NAME: Matthew Zhang

Throughout the project, I contributed to the Bag of Words and TF-IDF analysis. This involved writing and profiling both sequential and parallelized executions to highlight the benefits of Spark's parallel computation. I gained a lot of valuable hands-on experience with Spark, Scala and profiling tools, and got the opportunity to apply the concepts I learned in CS4480. This experience helped me deepen my understanding of distributed computing frameworks and their real-world applications in processing large datasets. I really enjoyed collaborating with my teammates, and I am happy that we achieved our goals together.

STUDENT ID: 57153875
STUDENT NAME: LI Yiu Hang

During the project, I made contributions to the implementation of the Word Cloud, Time Series and Sentiment Analysis. This opportunity allowed me to gain valuable hands-on experience by applying the NLP concepts and techniques I learned in CS4480 to the project. Throughout the whole process, I witnessed how we can implement parallel computing, MapReduce, Hadoop, and Spark into data analysis projects. Collaborating with my teammates was a memorable experience, and together we achieved the project's goals.