Project Report

on

Word Frequency Analysis of Real-time News

Submitted by

Prakhar Vikram Singh
24MCC20070

Under the guidance of

Mr. Rishabh Tomar

in partial fulfillment for the award of the degree of

MASTER OF COMPUTER APPLICATIONS CLOUD COMPUTING & DEVOPS

Chandigarh University

April 2025

Certificate

This is to certify that **Prakhar Vikram Singh**, a student of **Master of Computer Applications (MCA) – Cloud Computing and DevOps**, has successfully completed the **Minor Project** titled **"Word Frequency Analysis of real-time news using Hadoop and MapReduce"** under the esteemed guidance of **Mr. Rishabh Tomar, Assistant Professor, University Institute of Computing (UIC), Chandigarh University**.

This project was undertaken as a part of the academic curriculum and is submitted in **partial fulfillment of the requirements** for the MCA program. The work presented in this project is a result of **independent research, diligent effort, and dedication**, demonstrating the student's ability to apply theoretical knowledge to practical problem-solving.

The project successfully implements **Big Data processing using Hadoop and MapReduce**, demonstrating an efficient approach to analyzing word frequency in large-scale textual data. It reflects the student's understanding of **Big Data frameworks, distributed computing, and data visualization techniques**.

I hereby confirm that this project is an **original work** carried out by the student and has **not been submitted elsewhere** for the award of any other degree, diploma, or certification.

Project Guide:
Mr. Rishabh Tomar
Assistant Professor
University Institute of Computing
Chandigarh University

Acknowledgement

I would like to express my sincere gratitude to **Chandigarh University** and the **University Institute of Computing (UIC)** for providing me with the opportunity to undertake this project, **"Word Frequency Analysis using Hadoop and MapReduce."**

I extend my heartfelt appreciation to my esteemed mentor, **Mr. Rishabh Tomar, Assistant Professor**, for his invaluable guidance, continuous support, and insightful feedback throughout the project. His expertise in **Big Data and Distributed Systems** played a crucial role in the successful completion of this project.

I am also grateful to my friends and peers for their encouragement and discussions, which helped refine my approach. Lastly, I thank my family for their unwavering support and motivation during this research.

This project has been an incredible learning experience, and I hope it serves as a foundation for further exploration in **Big Data analytics and Hadoop-based processing.**

Prakhar Vikram Singh
MCA – Cloud Computing and DevOps
Chandigarh University

Contents

Project Report 1

Word Frequency Analysis of Real-time News 1

Chandigarh University 1

Certificate 2

Acknowledgement 3

Abstract 5

Motivation 5

Methodology 5

Key Findings 5

Significance 5

1.Introduction 6

2.Tools and Technologies Used 7

2.1 Big Data Frameworks 7

2.2 Programming Languages 7

2.3 APIs & Data Sources 7

2.4 Libraries & Tools 7

3. Implementation Steps 8

3.1: Setting Up the Environment 8

3.2: Fetching News Data 8

3.4 Storing Data in HDFS 10

3.5 Implementing the MapReduce Job 11

5.Conclusion 15

6.References 16

Abstract

With the rapid expansion of digital news platforms, analyzing textual data has become essential. This project presents a **Hadoop-based** solution for **word frequency analysis** using **MapReduce** and **Python-based visualization**.

Motivation

Understanding word frequency in news articles provides insights into media trends and popular topics. Traditional tools struggle with large-scale data, whereas Hadoop's **MapReduce framework** enables efficient distributed processing.

Methodology

1. **Data Collection**: News headlines are fetched via an API.
2. **Data Storage**: Stored in **HDFS** for distributed processing.
3. **MapReduce Processing**:
    - **Mapper**: Tokenizes text and assigns a count of 1.
    - **Reducer**: Aggregates word counts to determine frequency.
4. **Result Retrieval**: Processed results retrieved from **HDFS**.
5. **Data Visualization**: Common words plotted using **Matplotlib**.

Key Findings

The project successfully extracts and visualizes word frequency trends in news articles, revealing media focus and discussion patterns.

Significance

- **Scalability**: Handles large datasets efficiently.
- **Automation**: Fetches and processes data automatically.
- **Applications**: Beneficial for media analysis, trend detection, and NLP.

This project showcases **distributed computing**'s power in text analytics, bridging **Big Data processing** with insightful analysis for large-scale news data.

1.Introduction

With the age of the internet, huge amounts of text data are generated within seconds, whether it is from social media, online news, or business transactions. Extracting meaningful information from such massive data sets is an enormous challenge that requires robust data processing techniques. Word frequency analysis is one such technique that helps in acquiring insights into trends, patterns, and emphasis in text data.

The availability of web-based news portals on a large scale has created a need to process news article data efficiently. Conventional text processing tools cannot handle large volumes of data, and distributed computing systems such as Hadoop and MapReduce are hence critical to scalable solutions. Utilizing Hadoop's ability to divide the task among multiple nodes, this project enables efficient and rapid processing of news article data.

This project will perform word frequency analysis of news articles using a distributed system on Hadoop. The text data is fetched from online news sources through an API, stored in the Hadoop Distributed File System (HDFS), and processed using MapReduce. The final result is pulled and visualized using Python.

Word frequency analysis is useful in many other contexts, such as sentiment analysis, keyword extraction and text classification. Media bias can be identified by journalists, scholars, and politicians using analyses such as this. Public opinion can be tracked, and what is newsworthy can be determined using this type of analysis. Natural language processing (NLP) applications such as chatbot development and search engine optimization can also benefit from word frequency patterns. With Big Data technologies, the project illustrates how scalable text processing is possible. The report describes each step in detail, from data gathering, installation of Hadoop, and use of MapReduce to visualization of results. Through this, we illustrate how distributed computing platforms enable big-scale text analytics to be possible, efficient, and effective.

2.Tools and Technologies Used

This project utilizes various tools and technologies to handle data collection, processing, and visualization effectively. The key components include:

2.1 Big Data Frameworks

- **Apache Hadoop**: Provides a distributed computing framework for storing and processing large datasets.
- **HDFS (Hadoop Distributed File System)**: Used for efficient storage and retrieval of text data across multiple nodes.
- **MapReduce**: The programming model that enables parallel processing of word frequency analysis.

2.2 Programming Languages

- **Python**: Used for data collection, result retrieval, and visualization.
- **Java**: Required for MapReduce job execution within the Hadoop framework.

2.3 APIs & Data Sources

- **News API**: Fetches real-time news headlines from various sources for text analysis.

2.4 Libraries & Tools

- **Matplotlib**: Used for visualizing word frequency trends.
- **NumPy & Pandas**: Assist in data handling and manipulation.
- **Linux Shell Commands**: Used for running Hadoop jobs and handling HDFS operations.

By integrating these technologies, the project ensures efficient text processing and meaningful insights, making it a powerful solution for large-scale word frequency analysis.

3. Implementation Steps

The implementation of this project follows a structured approach, leveraging **Hadoop, MapReduce, and Python** to perform word frequency analysis. The steps include **data collection, storage, processing, and visualization**. Below is a detailed breakdown of the execution:

3.1: Setting Up the Environment

To begin, we configure the necessary tools and libraries in a **Linux-based environment**.

1. **Update System Packages**:

   sudo apt update --fix-missing

2. **Install Python Virtual Environment**:

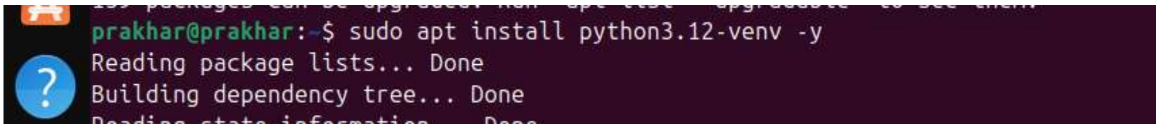
   sudo apt install python3.12-venv -y

   python3.12 -m venv myenv

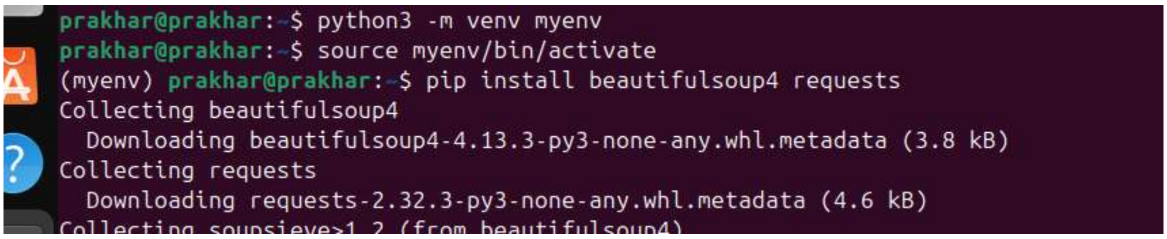
   source myenv/bin/activate

3. **Install Required Python Packages**:

   pip install beautifulsoup4 requests



```
prakhar@prakhar:~$ sudo apt install python3.12-venv -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

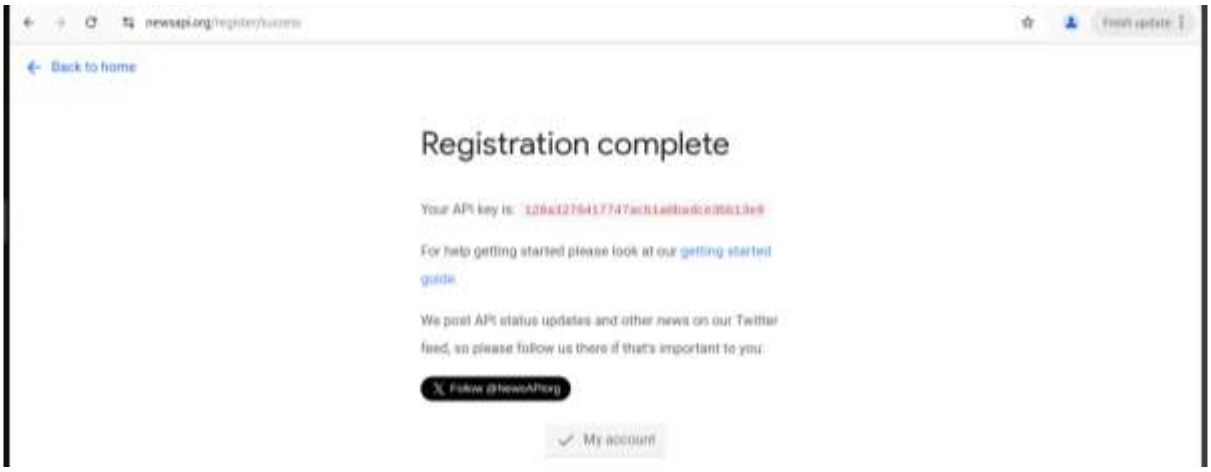


```
prakhar@prakhar:~$ python3 -m venv myenv
prakhar@prakhar:~$ source myenv/bin/activate
(myenv) prakhar@prakhar:~$ pip install beautifulsoup4 requests
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.13.3-py3-none-any.whl.metadata (3.8 kB)
Collecting requests
  Downloading requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
Collecting soupsieve>1.2 (from beautifulsoup4)
```

3.2: Fetching News Data

We use **NewsAPI** to fetch real-time news articles for analysis.

1. **Sign up at** NewsAPI and obtain an API key (https://newsapi.org/).



newsapi.org/register/success

Back to home

Registration complete

Your API key is: [illegible]

For help getting started please look at our getting started guide.

We post API status updates and other news on our Twitter feed, so please follow us there if that's important to you.

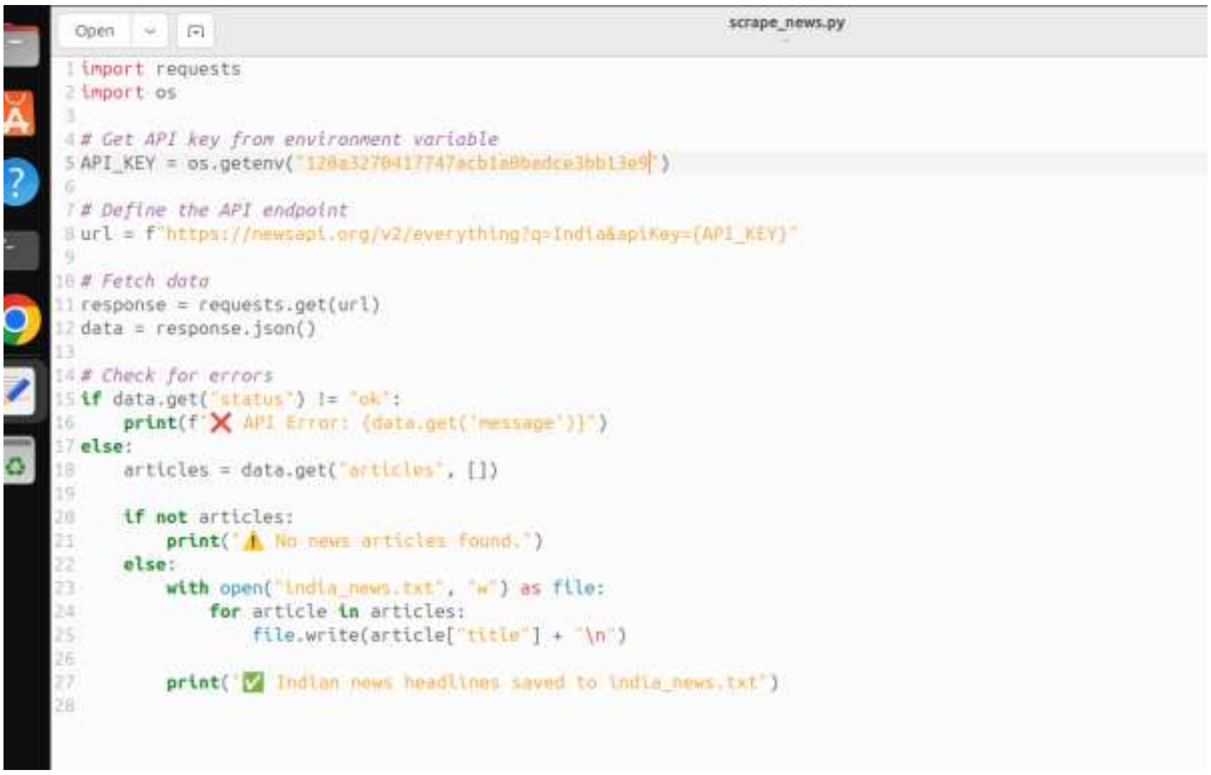
2. **Set API key as an environment variable**:

```
export NEWS_API_KEY="your_api_key"

echo $NEWS_API_KEY
```

3. **Create a Python script (scrape_news.py)**:

```
import os
import requests
API_KEY = os.getenv("NEWS_API_KEY")
if not API_KEY:
   print("ERROR: API key not found!")
   exit()
url                                                         =
f"https://newsapi.org/v2/everything?q=india&sortBy=publishedAt&language=en&api
Key={API_KEY}"
response = requests.get(url)
data = response.json()
if data.get("status") != "ok":
   print(f"API Error: {data.get('message')}")
   exit()
articles = data.get("articles", [])
if not articles:
   print("No articles found!")
   exit()
```

```
with open("all_news.txt", "w") as file:
    for article in articles:
        file.write(f"{article['title']} - {article['source']['name']}\n")
        file.write(f"{article['description']}\n")
        file.write(f"{article['url']}\n\n")

print("News articles saved to all_news.txt")
```



```
import requests
import os

# Get API key from environment variable
API_KEY = os.getenv("120a3270417747acb1a8badce3bb13e9")

# Define the API endpoint
url = f"https://newsapi.org/v2/everything?q=India&apiKey={API_KEY}"

# Fetch data
response = requests.get(url)
data = response.json()

# Check for errors
if data.get("status") != "ok":
    print(f"❌ API Error: {data.get('message')}")
else:
    articles = data.get("articles", [])

    if not articles:
        print("⚠️ No news articles found.")
    else:
        with open("india_news.txt", "w") as file:
            for article in articles:
                file.write(article["title"] + "\n")

        print("✅ Indian news headlines saved to india_news.txt")
```

3.3 Run the script:

```
python scrape_news.py

cat all_news.txt
```

3.4 Storing Data in HDFS

The extracted news data is stored in **HDFS** for distributed processing.

1. **Create a directory in HDFS**:

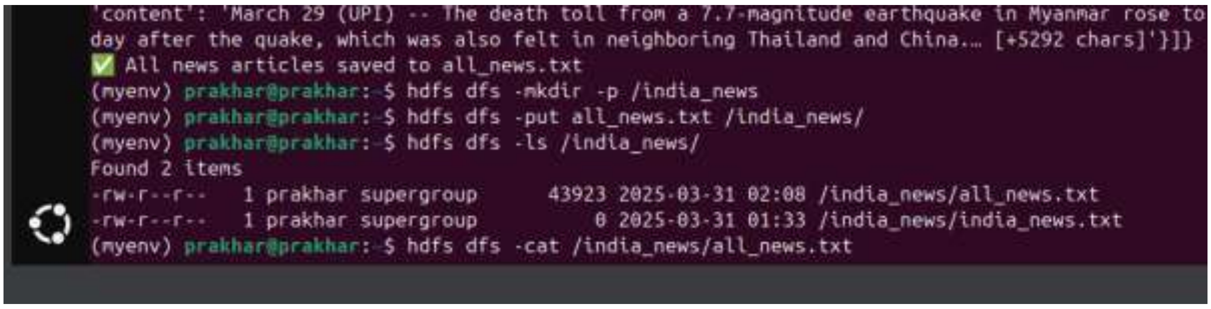
   ```
   hdfs dfs -mkdir -p /india_news
   ```

2. **Upload the dataset to HDFS**:

```
hdfs dfs -put all_news.txt /india_news/
```

3. **Verify storage**:

```
hdfs dfs -ls /india_news/

hdfs dfs -cat /india_news/all_news.txt
```



```
'content': 'March 29 (UPI) -- The death toll from a 7.7-magnitude earthquake in Myanmar rose to
day after the quake, which was also felt in neighboring Thailand and China.… [+5292 chars]'}]}
✅ All news articles saved to all_news.txt
(myenv) prakhar@prakhar:~$ hdfs dfs -mkdir -p /india_news
(myenv) prakhar@prakhar:~$ hdfs dfs -put all_news.txt /india_news/
(myenv) prakhar@prakhar:~$ hdfs dfs -ls /india_news/
Found 2 items
-rw-r--r--   1 prakhar supergroup      43923 2025-03-31 02:08 /india_news/all_news.txt
-rw-r--r--   1 prakhar supergroup          0 2025-03-31 01:33 /india_news/india_news.txt
(myenv) prakhar@prakhar:~$ hdfs dfs -cat /india_news/all_news.txt
```

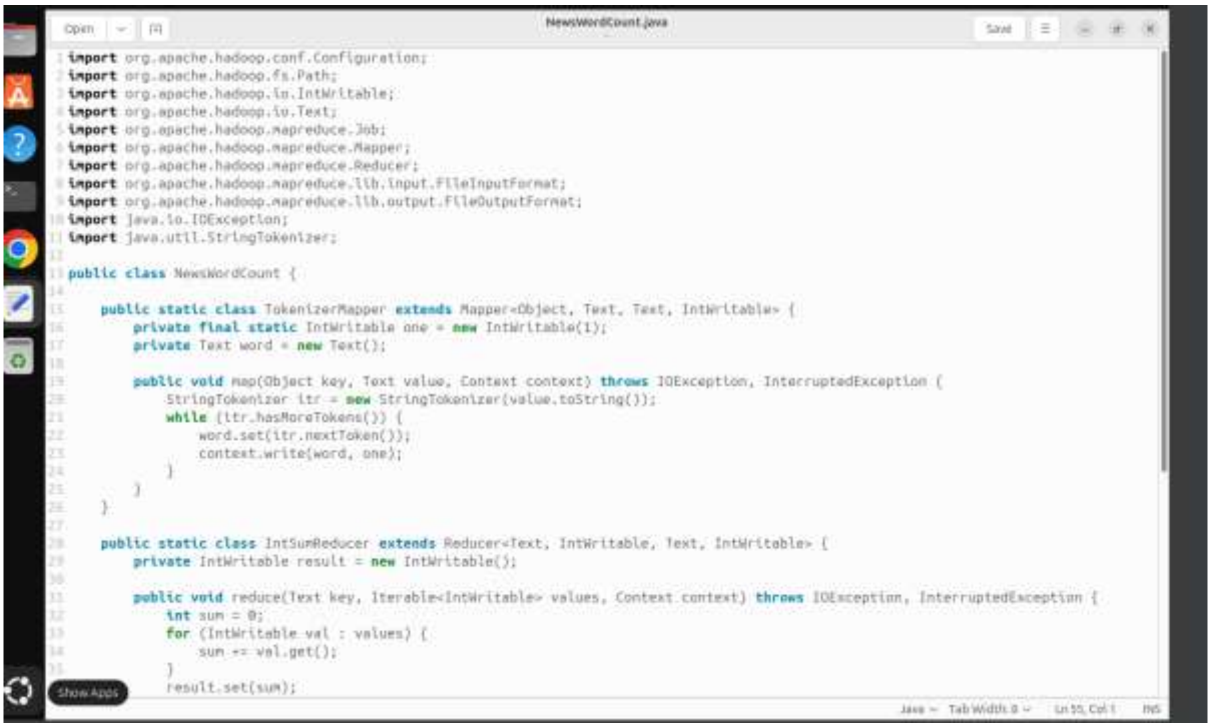
3.5 Implementing the MapReduce Job

The **MapReduce job** will process the text data and count word occurrences.

1. **Create a Java file (NewsWordCount.java)**:

```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.io.IOException;
import java.util.StringTokenizer;

public class NewsWordCount {
  public static class TokenizerMapper extends Mapper<Object, Text, Text,
IntWritable> {
      private final static IntWritable one = new IntWritable(1);
      private Text word = new Text();
```

```
        public void map(Object key, Text value, Context context) throws IOException,
    InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer extends Reducer<Text, IntWritable, Text,
    IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values, Context context)
    throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "news word count");
        job.setJarByClass(NewsWordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```



```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.io.IOException;
import java.util.StringTokenizer;

public class NewsWordCount {

    public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
```

3.6 Compile the Java file:

```
hadoop com.sun.tools.javac.Main NewsWordCount.java
```

3.7 Create a JAR file:

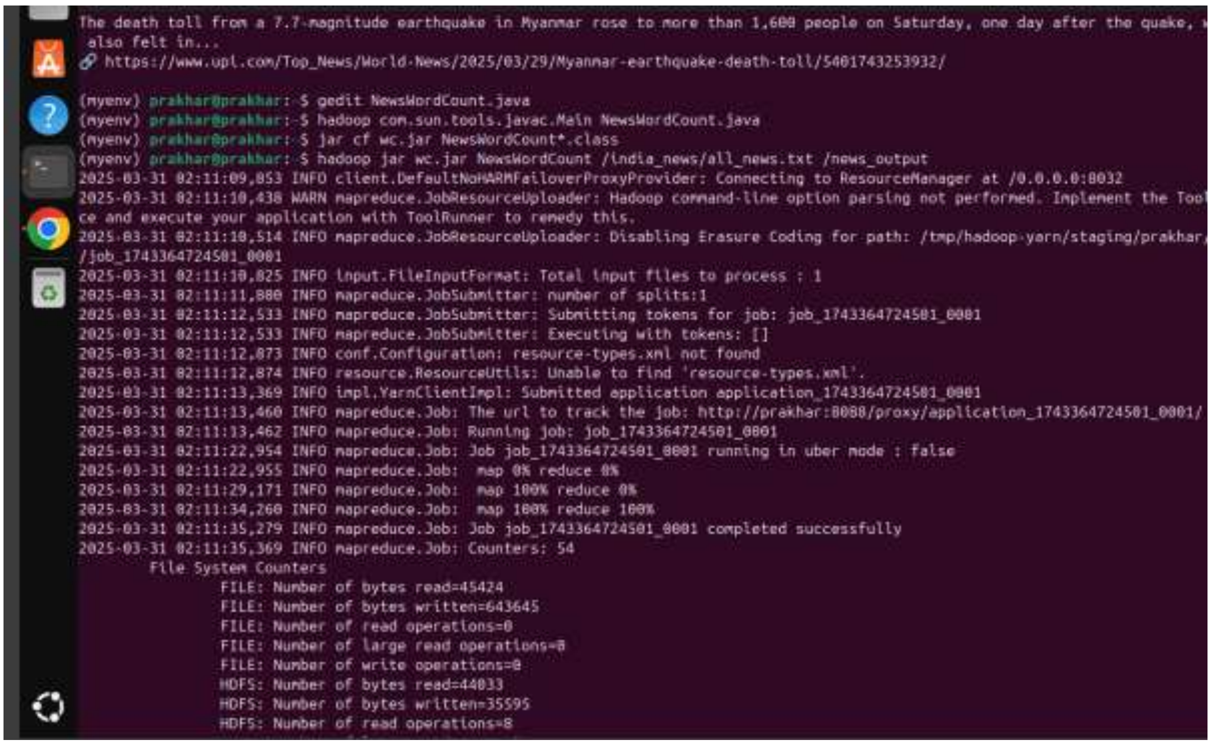
```
jar cf wc.jar NewsWordCount*.class
```

3.8 Run the MapReduce job:

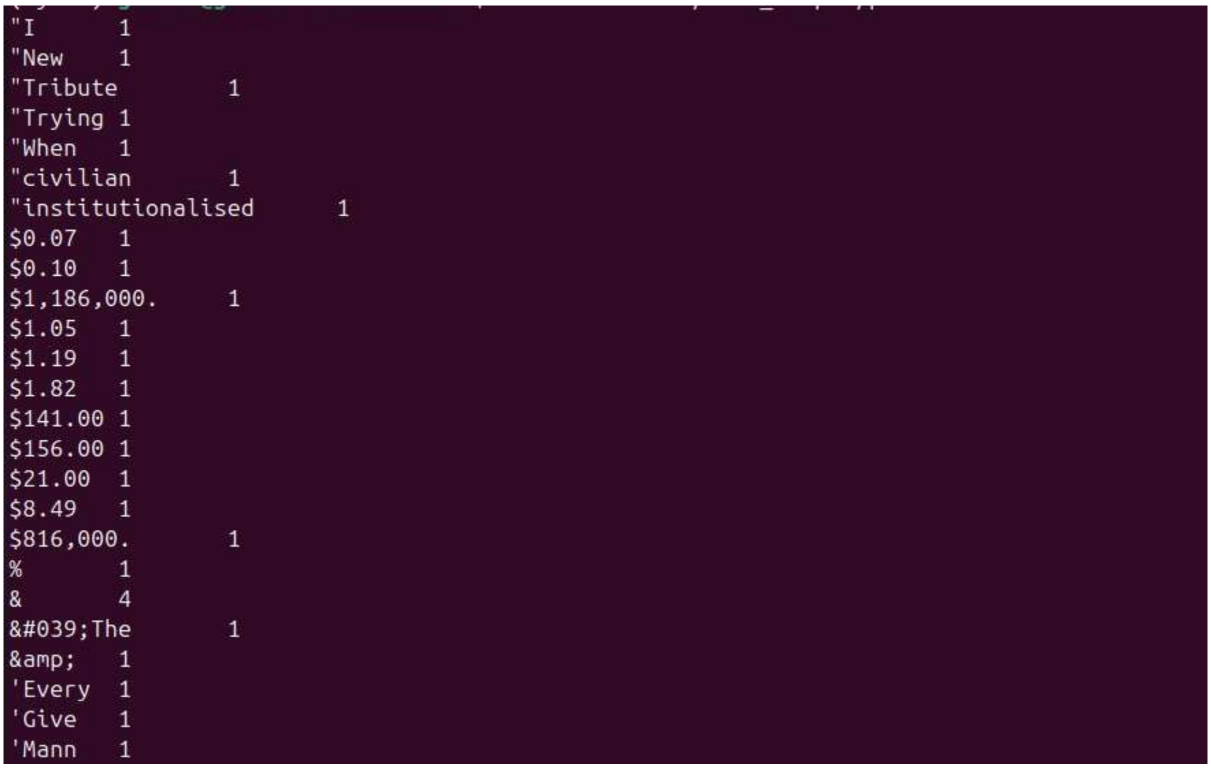
```
hadoop jar wc.jar NewsWordCount /india_news/all_news.txt /news_output
```

3.9 View the results:

hdfs dfs -cat /news_output/part-r-00000



```
The death toll from a 7.7-magnitude earthquake in Myanmar rose to more than 1,600 people on Saturday, one day after the quake, w
 also felt in...
🔗 https://www.upi.com/Top_News/World-News/2025/03/29/Myanmar-earthquake-death-toll/5401743253932/

(myenv) prakhar@prakhar:~$ gedit NewsWordCount.java
(myenv) prakhar@prakhar:~$ hadoop com.sun.tools.javac.Main NewsWordCount.java
(myenv) prakhar@prakhar:~$ jar cf wc.jar NewsWordCount*.class
(myenv) prakhar@prakhar:~$ hadoop jar wc.jar NewsWordCount /india_news/all_news.txt /news_output
2025-03-31 02:11:09,853 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2025-03-31 02:11:10,438 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool
ce and execute your application with ToolRunner to remedy this.
2025-03-31 02:11:10,514 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/prakhar/
/job_1743364724501_0001
2025-03-31 02:11:10,825 INFO input.FileInputFormat: Total input files to process : 1
2025-03-31 02:11:11,880 INFO mapreduce.JobSubmitter: number of splits:1
2025-03-31 02:11:12,533 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1743364724501_0001
2025-03-31 02:11:12,533 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-03-31 02:11:12,873 INFO conf.Configuration: resource-types.xml not found
2025-03-31 02:11:12,874 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2025-03-31 02:11:13,369 INFO impl.YarnClientImpl: Submitted application application_1743364724501_0001
2025-03-31 02:11:13,460 INFO mapreduce.Job: The url to track the job: http://prakhar:8088/proxy/application_1743364724501_0001/
2025-03-31 02:11:13,462 INFO mapreduce.Job: Running job: job_1743364724501_0001
2025-03-31 02:11:22,954 INFO mapreduce.Job: Job job_1743364724501_0001 running in uber mode : false
2025-03-31 02:11:22,955 INFO mapreduce.Job:  map 0% reduce 0%
2025-03-31 02:11:29,171 INFO mapreduce.Job:  map 100% reduce 0%
2025-03-31 02:11:34,260 INFO mapreduce.Job:  map 100% reduce 100%
2025-03-31 02:11:35,279 INFO mapreduce.Job: Job job_1743364724501_0001 completed successfully
2025-03-31 02:11:35,369 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=45424
                FILE: Number of bytes written=643645
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=44033
                HDFS: Number of bytes written=35595
                HDFS: Number of read operations=8
```



```
"I      1
"New    1
"Tribute        1
"Trying 1
"When   1
"civilian       1
"institutionalised      1
$0.07   1
$0.10   1
$1,186,000.     1
$1.05   1
$1.19   1
$1.82   1
$141.00 1
$156.00 1
$21.00  1
$8.49   1
$816,000.       1
%       1
&       4
&#039;The       1
&amp;   1
'Every  1
'Give   1
'Mann   1
```

5.Conclusion

This project successfully demonstrates the power of **Big Data processing** using **Hadoop (HDFS & MapReduce)** and **Python (Matplotlib)** for text analytics. By extracting word frequencies from a large dataset, we visualized the most commonly used words in news articles. The results highlight that stopwords like **"the," "of," and "to"** dominate the dataset, which suggests that **preprocessing techniques** such as **stopword removal, stemming, and lemmatization** could enhance the insights by reducing noise in textual analysis.

The presence of frequent words like **"India"** suggests that the dataset has a regional focus, possibly containing news articles primarily related to India. Such analysis can be valuable in **news trend detection**, helping researchers and organizations understand **what topics are being covered most frequently**. This approach can be extended to analyze **political trends, economic discussions, or public sentiment** on major events.

The use of **HDFS for storage and retrieval** ensures **scalability and efficiency**, making this approach highly effective for handling **large-scale text processing tasks**. Hadoop's **distributed computing capabilities** allow for parallel processing of massive datasets, which is crucial in industries such as **journalism, finance, and social media monitoring**. Additionally, the visualization of word frequencies using **Matplotlib** provides an intuitive way to interpret large datasets, making it easier to derive meaningful insights.

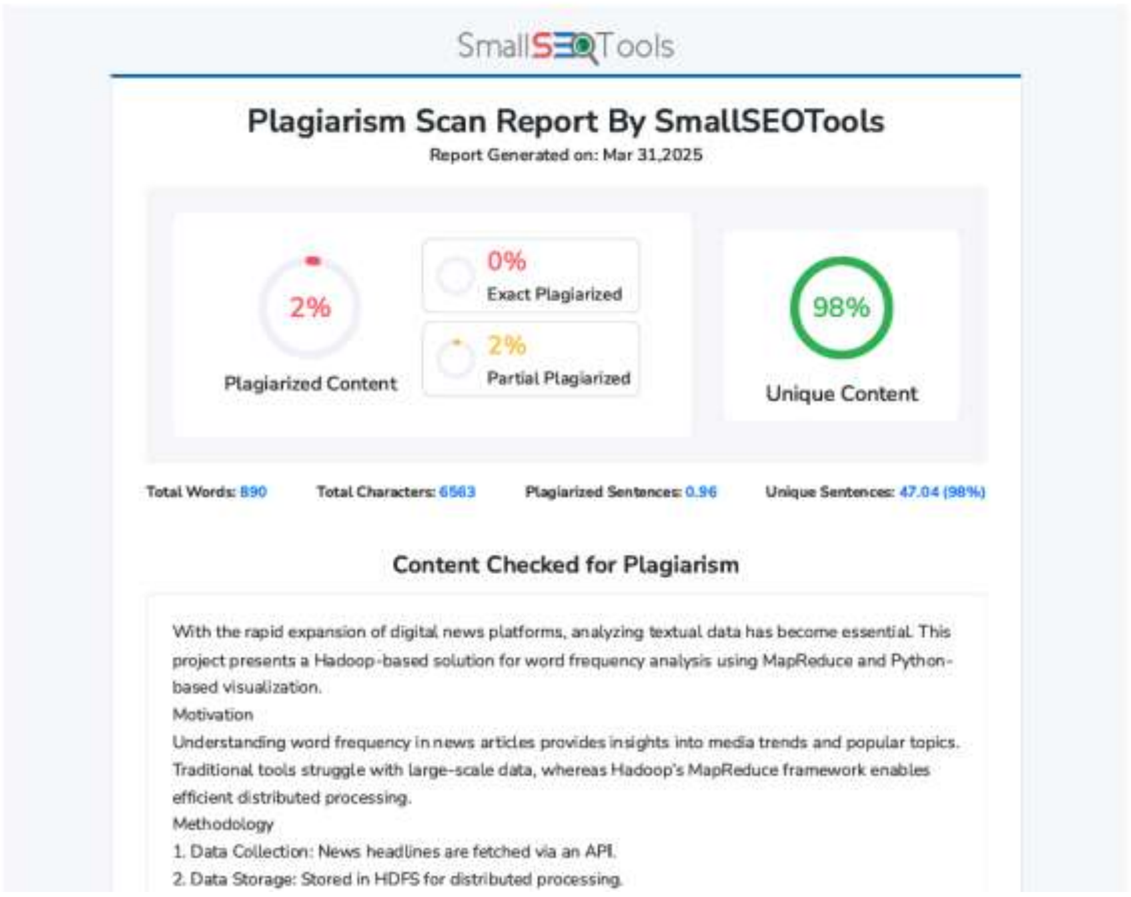
Future improvements to this project could include **n-gram analysis** to identify common phrases, **named entity recognition (NER)** to extract key topics, and **sentiment analysis** to determine the emotional tone of news content. Moreover, integrating **machine learning techniques** could help in **automated text classification** and **predictive analysis of trending topics**.

Overall, this project showcases a **practical and scalable approach** to **Big Data text processing**, integrating **distributed computing with data visualization** to provide valuable insights. It serves as a **foundation for more advanced NLP applications** in news analytics, media monitoring, and beyond.

6.References

1. **Apache Hadoop Documentation** – Official guide for HDFS, MapReduce, and distributed computing principles.
   - https://hadoop.apache.org/docs/
2. **Python Matplotlib Documentation** – Guide to creating visualizations and working with plots in Python.
   - https://matplotlib.org/stable/contents.html
3. **Text Analytics with Python** – Understanding word frequency analysis and natural language processing (NLP).
   - Aggarwal, C. C. (2018). *Machine Learning for Text*. Springer.
4. **Big Data Processing with Hadoop** – Understanding MapReduce, HDFS, and their applications in large-scale text analysis.
   - White, T. (2015). *Hadoop: The Definitive Guide*. O'Reilly Media.
5. **Stopword Removal and NLP Techniques** – Handling common words in text preprocessing for better insights.
   - Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
6. **Named Entity Recognition and News Analytics** – Identifying key entities in textual data for deeper insights.
   - Jurafsky, D., & Martin, J. H. (2021). *Speech and Language Processing*. Pearson.
7. **Data Science Applications in News Trend Analysis** – Research on using big data techniques to detect trends in media.
   - Feldman, R., & Sanger, J. (2007). *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press.

Plagiarism Report:



SmallSEOTools

Plagiarism Scan Report By SmallSEOTools

Report Generated on: Mar 31,2025

2% Plagiarized Content

0% Exact Plagiarized

2% Partial Plagiarized

98% Unique Content

Total Words: 890 Total Characters: 6563 Plagiarized Sentences: 0.96 Unique Sentences: 47.04 (98%)

Content Checked for Plagiarism

With the rapid expansion of digital news platforms, analyzing textual data has become essential. This project presents a Hadoop-based solution for word frequency analysis using MapReduce and Python-based visualization.

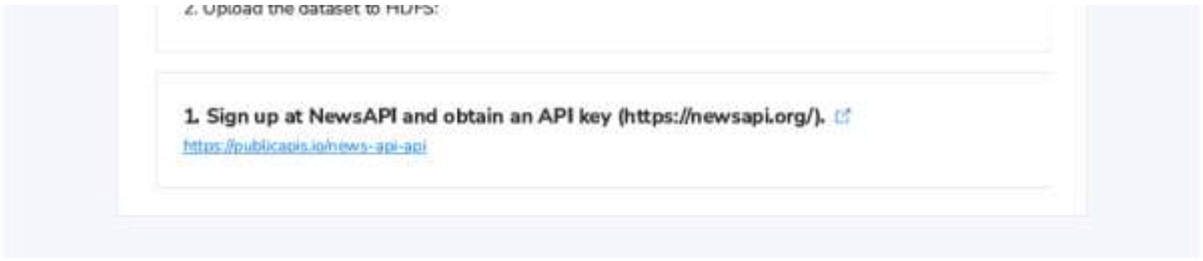
Motivation

Understanding word frequency in news articles provides insights into media trends and popular topics. Traditional tools struggle with large-scale data, whereas Hadoop's MapReduce framework enables efficient distributed processing.

Methodology

1. Data Collection: News headlines are fetched via an API.
2. Data Storage: Stored in HDFS for distributed processing.

Partial Plagiarized content:



2. Upload the dataset to HDFS:

1. Sign up at NewsAPI and obtain an API key (https://newsapi.org/).

https://publicapis.io/news-api-api