A

Major Project Report

on

**Sentiment Analysis using Hadoop Framework and Deep Learning Algorithms**

Submitted in partial fulfilment of the requirements for the award of the Degree of

Bachelor of Technology

By

**Kaparthi Sowmya**

(20EG105416)

**Kasthala Chandu**

(20EG105422)

**Nomula Harshavardhan Reddy**

 (20EG105451)

Under The Guidance

Of

**Dr. P Rathna Sekhar**

Assistant Professor

Department of CSE

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**ANURAG UNIVERSITY**

**VENKATAPUR (V), GHATKESAR (M), MEDCHAL (D), T.S 500088**

**(2023-24)**

**DECLARATION**

We hereby declare that the Report entitled **Sentiment Analysis using Hadoop Framework and Deep Learning Algorithms** submitted for the award of Bachelor of technology Degree is our original work and the Report has not formed the basis for the award of any degree, diploma, associateship or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any degree or diploma.

Place: *Hyderabad*

Date:

Kaparthi Sowmya

(20EG105416)

Kasthala Chandu

(20EG105422)

Nomula Harshavardhan Reddy

(20EG105451)



**CERTIFICATE**

This is to certify that the report entitled **Sentiment Analysis using Hadoop Framework and Deep Learning Algorithms** that is being submitted by **Ms. Kaparthi Sowmya** bearing the hall ticket number **20EG105416**, **Mr. Kasthala Chandu** bearing the hall ticket number **20EG105422**, **Mr. Nomula Harshavardhan Reddy** bearing the hall ticket number **20EG105451** in partial fulfilment for the award of B.Tech degree in Computer Science and Engineering to Anurag University is a record of bonafide work carried out by them under my guidance and supervision.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

|  |  |
| --- | --- |
| Dr. P Rathna Sekhar  Assistant Professor  Department of CSE | Dr. G. Vishnu Murthy  Dean, CSE |

External Examiner 1

**ACKNOWLEDGEMENT**

We would like to express our sincere thanks and deep sense of gratitude to project supervisor **Dr. P Rathna Sekhar, Assistant Professor, Department of CSE** for his constant encouragement and inspiring guidance without which this project could not have been completed. His critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. His patience, guidance and encouragement made this project possible.

We would like express my special thanks to **Dr. V. Vijaya Kumar, Dean School of Engineering, Anurag University,** for his encouragement and timely support in our B. Tech program.

We would like to acknowledge our sincere gratitude for the support extended by **Dr. G. Vishnu Murthy, Dean, Dept. of CSE, Anurag University.** We also express my deep sense of gratitude to **Dr. VVSSS Balaram, Academic co-ordinator**, **Anurag University** and **Dr. Pallam Ravi, Project in-Charge.** Project Co-ordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated us during the crucial stage our project work.

Kaparthi Sowmya

(20EG105416)

Kasthala Chandu

(20EG105422)

Nomula Harshavardhan Reddy

(20EG105451)

**ABSTRACT**

Sentiment analysis, a powerful tool in the era of microblogging, helps us tap into the underlying emotions and opinions expressed on platforms like Twitter. It's like deciphering the language, where every tweet whispers a story – some filled with joy, others with frustration. Analysing these sentiments involves dissecting the data, extracting key characteristics, and then classifying them as positive, negative, or neutral. But Twitter data presents a unique challenge – its brevity and informal nature make it tricky to gauge the true sentiment. This paper proposes a cutting-edge approach to overcome this hurdle. We leverage the power of a distributed processing framework, Hadoop, especially Apache spark, to efficiently handle the vast amount of Twitter data. By meticulously extracting relevant features, we unlock the hidden emotions within each tweet. Then, we employ a sophisticated deep learning algorithm, a deep recurrent neural network (RNN), to assign a sentiment score to each tweet, effectively sorting them into the "positive" or "negative" buckets. The results speak for themselves – our method outperforms traditional approaches, achieving an impressive accuracy, sensitivity, and specificity. In essence, this research unlocks a deeper understanding of the collective mood on Twitter, offering valuable insights for brands, researchers, and anyone curious about the heartbeat of the digital world. Our study focuses on evaluating the performance, scalability, and efficiency of these frameworks in handling real-time sentiment analysis tasks.

**Keywords:** Sentiment Analysis, deciphering, distributed Processing framework, Hadoop, Apache Spark, Deep Learning, Recurrent Neural Networks (RNN), sentiment score.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **SNo** | **CONTENT** | **PAGE NO** |
| 1 | INTRODUCTION   * 1. Sentiment Analysis   2. Hadoop Framework   3. Deep Learning   4. Problem Definition   5. Problem Illustration   6. Objective of the Project | 1  2  3  5  6  7  8 |
| 2 | LITERATURE REVIEW | 9 |
| 3 | PROPOSED METHOD  3.1 Data Collection  3.2 Preprocessing  3.3 Hadoop Setup  3.4 Deep Learning  3.5 Integration with Hadoop  3.6 Real-time Analysis  3.7 Visualization and Reporting | 13  15  15  15  16  16  17  17 |
| 4 | IMPLEMENTATION  4.1 Working Flow  4.2 Attributes  4.3 Experiment Screenshots | 17  17  18  21 |
| 5 | Experiment Set-up  5.1 Software Requirements  5.2 Libraries Used  5.3 Parameters | 25  25  25  27 |
| 6 | Discussion of Results | 29 |
| 7 | Conclusion | 33 |
| 8 | References | 34 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **FIGURE NAME** | **PAGE NO** |
| 1.1 | Machine learning and deep learning | 1 |
| 1.2 | Sentiment Analysis | 2 |
| 1.3 | Hadoop Set-up | 3 |
| 1.4 | Apache Spark | 5 |
| 1.5 | Recurrent Neural Network | 6 |
| 1.6 | Emojis | 8 |
| 3.1 | Architecture | 14 |
| 3.2 | Hadoop Set-up (Map Reduce) | 15 |
| 6.1 | Confusion Matrix | 29 |
| 6.2 | Classification Report | 29 |
| 6.3 | Training and Validation accuracy | 30 |
| 6.4 | Training and Validation loss | 31 |

**LIST OF TABLES**

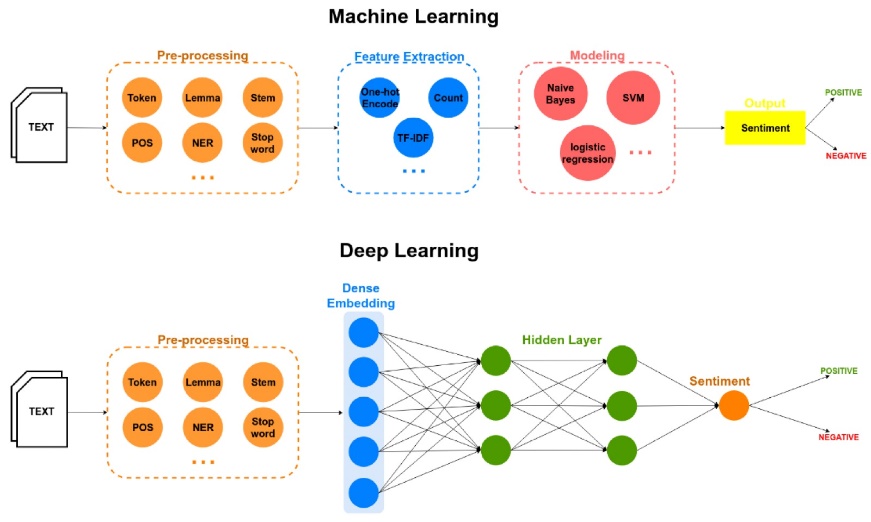
|  |  |  |
| --- | --- | --- |
| **TABLE NO** | **TABLE NAME** | **PAGE NO** |
| I | Metrics of Proposed Method VS Previous Method | 32 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **Abbreviations** | **FullForm** |
| NLP | Natural Language Processing |
| RNN | Recurrent Neural Network |
| CNN | Convolutional Neural Network |
| HDFS | Hadoop Distributed File System |
| JDK | Java Development Kit |
| ANN | Artificial Neural Networks |
| GPU | Graphical Processing Units |
| OMSA | Opinion Mining and Sentiment Analysis |
| APC | Aspect Polarity Classification |
| API | Application Program Interface |
| NLTK | Natural Language ToolKit |
| TPR | True Positive Rate |

**1. INTRODUCTION**

Sentiment analysis, a crucial task in natural language processing (NLP), plays a pivotal role in understanding and extracting sentiments and opinions from textual data. With the exponential growth of online content across social media platforms, e-commerce websites, and customer reviews, the demand for real-time sentiment analysis solutions has surged. Real-time sentiment analysis enables businesses to swiftly respond to customer feedback, monitor brand perception, and gauge market trends, thus gaining a competitive edge in today's dynamic landscape.



**Figure 1.1 Machine Learning and Deep Learning**

Traditional sentiment analysis approaches often face challenges in processing large-scale, real-time data due to their computational overhead and scalability limitations. To address these challenges, emerging technologies such as Hadoop and Apache Spark have garnered attention for their distributed computing capabilities, enabling the processing of massive datasets in parallel. Additionally, deep learning algorithms, particularly Recurrent Neural Networks (RNN), have demonstrated remarkable performance in sentiment analysis tasks by automatically learning hierarchical representations of text.

In this study, we undertake a comprehensive investigation into real-time sentiment analysis frameworks, integrating Hadoop, Apache Spark, and deep learning algorithms RNN. Our objective is to evaluate the performance, scalability, and efficiency of these frameworks in processing real-time textual data streams for sentiment analysis. By conducting experiments on large-scale datasets and benchmarking various metrics such as processing speed, accuracy, and resource utilization, we aim to provide valuable insights into the comparative strengths and limitations of each approach.

The findings of this study hold significant implications for industries reliant on sentiment analysis, including marketing, customer service, and financial trading. Understanding the trade-offs between different frameworks can aid practitioners in selecting the most suitable solution based on their specific requirements, whether it be real-time response, scalability, or accuracy. Ultimately, our research contributes to advancing the state-of-the-art in real-time sentiment analysis and guides the development of more efficient and effective sentiment analysis systems tailored to modern data-intensive environments.

* 1. **Sentiment Analysis**

Sentiment analysis, also known as opinion mining, is a branch of natural language processing (NLP) concerned with uncovering the emotional tone behind a piece of text. It essentially sifts through written language to determine whether the sentiment expressed is positive, negative, or neutral. This can be incredibly valuable in today's data-driven world, where vast amounts of textual information are constantly generated. Imagine a company that wants to gauge customer satisfaction with a new product launch. Sentiment analysis can be employed to dissect social media comments, reviews, and feedback emails.

By analyzing this text, the company can gain insights into how customers perceive the product, identifying areas of strength and weakness. This allows them to make informed decisions about improvements, marketing strategies, and future product development.Nuances and Context While basic sentiment analysis might categorize text as positive, negative, or neutral, more sophisticated techniques can delve deeper.

They can account for sarcasm, frustration, or amusement, providing a more nuanced understanding of the emotional tone. Additionally, sentiment analysis can incorporate contextual elements. For instance, a critical review mentioning a long wait time at a restaurant might hold a different weight than a similar critique about a doctor's appointment.

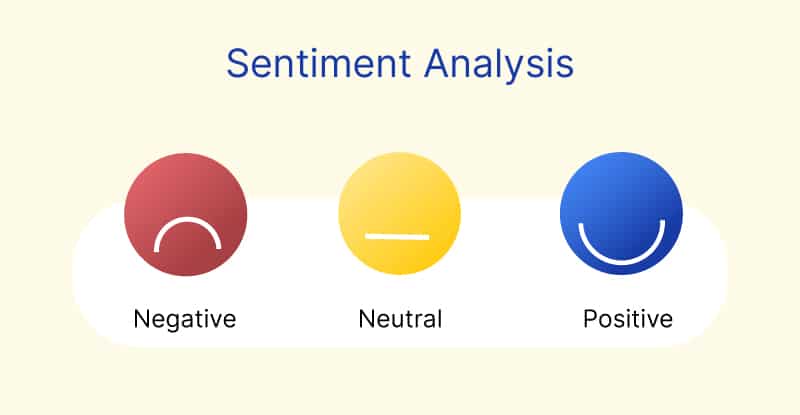


Figure **1.2 Sentiment Analysis**

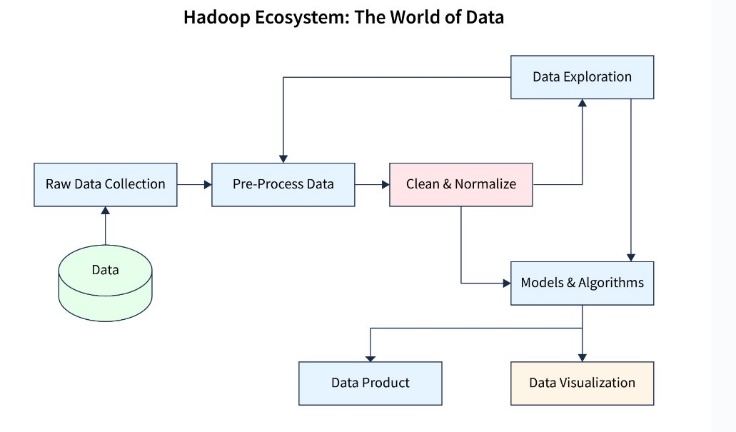
* 1. **Hadoop Framework**

Hadoop is an open-source distributed computing framework designed for processing and storing large volumes of data across clusters of commodity hardware. At the heart of Hadoop are two core components: the Hadoop Distributed File System (HDFS) and the MapReduce programming model.

**Hadoop Distributed File System (HDFS):**

HDFS is a distributed file system that provides high-throughput access to data stored across multiple machines in a Hadoop cluster. It is designed to handle large files by breaking them into smaller blocks (typically 128 MB or 256 MB in size) and distributing these blocks across the cluster. HDFS follows a master-slave architecture, where the NameNode serves as the master node responsible for managing metadata about file locations and block replication, while DataNodes act as slave nodes responsible for storing and serving data blocks.

HDFS ensures fault tolerance and data redundancy through block replication. Each data block is replicated across multiple DataNodes to provide fault tolerance in case of node failures. Additionally, HDFS employs a streaming data access model, where data processing tasks can access data blocks directly from the DataNodes, minimizing data movement and optimizing data locality.



**Figure 1.3 Hadoop Setup**

**Apache Spark**

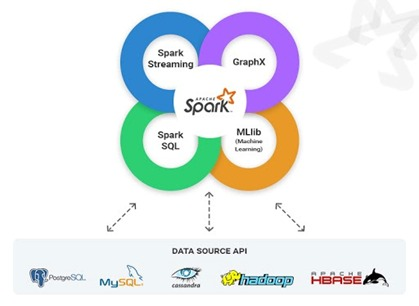
Apache Spark is a powerful distributed computing framework that has gained widespread adoption for processing large-scale data analytics tasks. Unlike traditional batch processing systems, Spark offers the flexibility of in-memory data processing, enabling significantly faster data processing speeds. One of the key features of Spark is its ability to efficiently handle various types of workloads, including batch processing, real-time stream processing, machine learning, and graph processing, all within a unified framework. This versatility makes Spark well-suited for a wide range of data processing tasks across different industries and use cases.

At the core of Apache Spark is its resilient distributed dataset (RDD) abstraction, which represents distributed collections of data that can be processed in parallel across a cluster of machines. RDDs offer fault tolerance by automatically recovering lost data partitions in case of machine failures, ensuring robustness and reliability in distributed data processing. Additionally, Spark provides high-level APIs in languages such as Scala, Java, Python, and R, making it accessible to developers with varying skill sets and programming backgrounds.

One of the distinguishing features of Spark is its support for in-memory data caching, which allows frequently accessed data to be stored in memory across multiple stages of computation. This caching mechanism significantly reduces data access times and enhances overall processing speeds, making Spark well-suited for iterative algorithms and interactive data analysis tasks. Furthermore, Spark's built-in libraries for SQL, streaming, machine learning, and graph processing provide developers with a rich set of tools and functionalities for building complex data analytics pipelines with ease.

Another notable aspect of Apache Spark is its support for stream processing through the Spark Streaming module. Spark Streaming enables real-time data processing by dividing data streams into micro-batches, which are processed using the same RDD abstraction as batch processing. This unified programming model simplifies the development of real-time analytics applications, allowing developers to seamlessly transition between batch and stream processing modes within the same codebase.

Overall, Apache Spark stands out as a versatile and efficient distributed computing framework for big data analytics, offering scalability, speed, and ease of use. Its robust architecture, extensive library ecosystem, and support for diverse workloads make it a popular choice for organizations seeking to extract actionable insights from large-scale datasets in a fast and cost-effective manner.



**Figure 1.4 Apache Spark**

* 1. **Deep Learning**

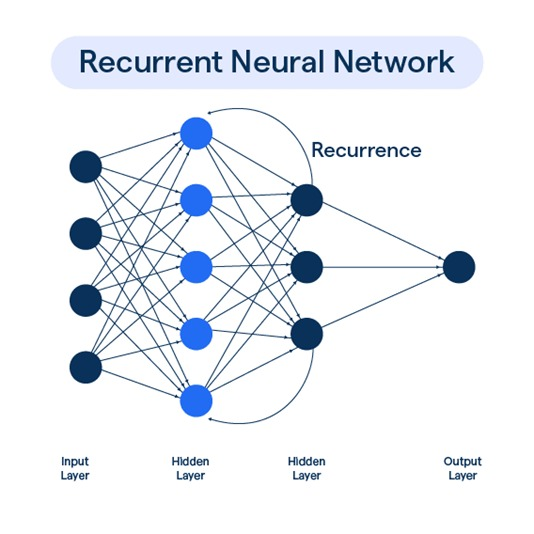
Deep learning is a branch of machine learning which is based on artificial neural networks. It is capable of learning complex patterns and relationships within data. In deep learning, we don’t need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets. Because it is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs).

These neural networks are inspired by the structure and function of the human brain’s biological neurons, and they are designed to learn from large amounts of data. Deep Learning is a subfield of Machine Learning that involves the use of neural networks to model and solve complex problems. Neural networks are modelled after the structure and function of the human brain and consist of layers of interconnected nodes that process and transform data.

The key characteristic of Deep Learning is the use of deep neural networks, which have multiple layers of interconnected nodes. These networks can learn complex representations of data by discovering hierarchical patterns and features in the data. Deep Learning algorithms can automatically learn and improve from data without the need for manual feature engineering. Deep Learning has achieved significant success in various fields, including image recognition, natural language processing, speech recognition, and recommendation systems. Some of the popular Deep Learning architectures include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Deep Belief Networks (DBNs). Training deep neural networks typically requires a large amount of data and computational resources. However, the availability of cloud computing and the development of specialized hardware, such as Graphics Processing Units (GPUs), has made it easier to train deep neural networks.

**RNN**

Recurrent Neural Network (RNN) is a type of Neural Network where the output from the previous step is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other. Still, in cases when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus, RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is its Hidden state, which remembers some information about a sequence. The state is also referred to as Memory State since it remembers the previous input to the network. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.



**Figure 1.5 Recurrent neural network**

* 1. **Problem Definition**

The existing systems for sentiment analysis often rely on traditional machine learning techniques and may lack scalability and efficiency when dealing with large-scale textual data streams. These systems typically involve preprocessing the data, extracting features, and then applying a classification algorithm to determine the sentiment of each text input. However, there are several disadvantages associated with such approaches:

1. **Limited Scalability**: Traditional sentiment analysis systems may struggle to scale effectively to handle large volumes of textual data in real-time. As the size of the dataset increases, processing times may become prohibitively long, leading to delays in sentiment analysis results.
2. **Lack of Adaptability to Dynamic Data:** Many existing sentiment analysis systems are designed to operate on static datasets and may not adapt well to dynamic data streams, such as social media feeds or online reviews. As a result, the sentiment analysis results may become outdated or less relevant over time.
3. **High Maintenance Costs:** Maintaining and updating traditional sentiment analysis systems can be resource-intensive, requiring ongoing monitoring, tuning, and retraining of machine learning models. Additionally, the need for manual feature engineering adds to the complexity and cost of the system.
   1. **Problem Illustration**

The sentiment analysis based on Twitter analyses the positive, neutral and negative reviews. The major challenge lies in building technology that identified and compiled the overall sentiment. The creation of noise while labelling the data is one of the challenges faced during the sentiment analysis of Twitter data. In abbreviations and slang, limited lexicons of emoticons, insufficient and irregular words expressed by the users in their post resulted in low classifier accuracy in detecting the polarity of the tweets and the incomplete coverage of domain specific words had resulted in incorrect sentiment classification and scoring. In-order to cleanse and analyse the sentiments of user at satisfactory level only a limited automatic and sophisticated Twitter-based content analysis tools were accessible.

1. **Multilingual sentiment analysis**

Multilingual sentiment analysis is the AI-driven process of extracting sentiment from data containing several languages. It is achieved through native language machine learning (ML) models built individually for different languages. A highly varied corpus of manually tagged data is gathered for every language to develop these models. Key processes include:

* Parts-of-speech (POS)tagger
* Lemmatization
* Grammatical constructs
* Polarity

1. **Irony and sarcasm**

The internet is full of irony and sarcasm, and sometimes, it is challenging to understand whether a post is genuine or sarcastic. This poses a significant challenge for sentiment analysis. Irony and sarcasm can skew the otherwise accurate sentiment analysis model and can skew the otherwise accurate sentiment analysis model and turn sentiment analysis results upside down. It may be helpful to use a sarcasm detection tool and then conduct sentiment analysis.

1. **Polarity Classification**

A key aspect of sentiment analysis is polarity classification. Polarity refers to the overall sentiment conveyed by a particular text, phrase or word. Analysing sentiment without context gets difficult as machines cannot learn about contexts if it is not trained explicitly. The most crucial disadvantage that arises from context is changing in polarity. Polarity classification is the task that distinguishes sentences that express positive, negative, or neutral polarities.

1. **Emojis**

Texts published on social media often contain emojis and these can quite significantly influence the final sentiment of the text. That's why a lot of preprocessing should be done related to emojis in order to whitelist and transform them into tokens. There are two basic types of emojis - Western emojis containing only one or two characters and a bit longer combination of characters Eastern emojis are made of. Lists of emojis and their corresponding unicode can be found online and can be very helpful in the preprocessing phase.



**Figure 1.6 Emojis**

* 1. **Objective of the project**

The objective of the project is to develop a cutting-edge approach for sentiment analysis of Twitter data using a distributed processing framework (Hadoop) and deep learning algorithms (Recurrent Neural Networks) to achieve high accuracy, sensitivity, and specificity in real-time processing. The project aims to develop a high-performance sentiment analysis system specifically designed for the unique challenges of Twitter data. In simpler terms, the project aims to build a system that can efficiently analyze the emotions and opinions expressed in tweets, even considering the short and informal language used on Twitter. This system will be more accurate than traditional methods and will be able to handle large amounts of data in real-time.

**2. LITERATURE REVIEW**

Zaharia, Matei, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica, their paper argues that while MapReduce and similar frameworks are successful for large- scale data processing, they struggle with applications that revisit the same data multiple times. This is common in iterative machine learning and interactive data analysis. The paper proposes Spark, a new framework that introduces Resilient Distributed Datasets (RDDs). RDDs are read-only collections of data spread across machines, but can be rebuilt if a part is lost. This allows Spark to outperform Hadoop in iterative jobs by caching data in memory across machines. Additionally, Spark enables interactive data exploration by allowing users to load datasets in memory and query them repeatedly. Overall, Spark offers the scalability and fault tolerance of MapReduce, while adding the ability to efficiently work with data across multiple parallel operations. This makes it a strong candidate for iterative machine learning and interactive data analysis tasks.

Zhang, Lei, Shuai Wang, and Bing Liu., their paper explores the growing use of deep learning in sentiment analysis. Sentiment analysis, also known as opinion mining, is a field of study that focuses on automatically detecting the emotions and opinions expressed in text. The rise of social media has led to a massive amount of opinionated data being available online, and sentiment analysis helps us understand this data. It is useful for many applications, including gauging public opinion on current events or determining customer sentiment towards a product. Traditionally, sentiment analysis relied on machine learning techniques like Support Vector Machines (SVM) and Naive Bayes. These techniques require handcrafted features to be defined from the text data. Deep learning offers a new approach. Deep learning models can automatically learn these features directly from the data. This can lead to more accurate sentiment analysis, especially with complex data like social media posts. This paper provides a survey of recent research in sentiment analysis using deep learning. It reviews the advantages of deep learning for this task and explores different deep learning architectures that have been used.

Li, Chunping, Jianhua Zhang, Guangquan Zhang, and Yongquan Yang says that the rise of the Internet of Things (IoT) and social media has created a vast amount of unstructured data, opening doors for data analysis techniques like Opinion Mining and Sentiment Analysis (OMSA). OMSA plays a valuable role in the big data era, allowing us to categorize opinions and gauge public sentiment. Researchers have developed various OMSA techniques and applied them to diverse datasets and scenarios. This study provides a comprehensive literature review of OMSA. It delves into both the technical aspects (techniques and types) and the non-technical aspects (application areas). Additionally, it highlights the challenges faced in developing OMSA techniques and those encountered during its application. By identifying these challenges, the study paves the way for future research directions.

Sumbaly, Ravi, his master's thesis explores the challenges and opportunities presented by "big data." Big data refers to the massive datasets characterized by high volume, velocity, variety, and veracity. While traditional data management struggles with these characteristics, big data holds immense potential for extracting value from previously untapped sources. The thesis proposes a reference architecture for data management systems that can handle big data. This architecture builds upon existing data warehouse architectures and integrates additional components to address the specific needs of big data. These components leverage technologies like the Apache Hadoop ecosystem and NoSQL databases.

Zaharia, Matei’s paper says that the implementation of MapReduce is optimized to operate over extensive clusters of standard machines, boasting remarkable scalability. Reports indicate that typical MapReduce tasks can handle several terabytes of data across thousands of machines. The system's user-friendliness is evidenced by the numerous MapReduce applications developed and the daily execution of over a thousand MapReduce jobs on expansive computing clusters.

Dua, D. and Graff, C., this study investigates the crucial role of data pre-processing decisions in machine learning model performance. The authors leverage simulations to assess how these choices influence a model's bias, variance, and overall error. the insights gleaned from simulations and benchmark tests are applied to existing machine learning applications. This demonstrates how the proposed approach can lead to enhanced model performance while minimizing development time and processing power requirements. Their study underscores the importance of meticulously considering data pre-processing decisions and their influence on model bias and variance. The findings hold value for data scientists and machine learning practitioners aiming to optimize their models for superior performance.

Malik, M., his work explores the potential of sentiment analysis and opinion mining for gauging public perception on social media. The vast amount of online opinionated data presents a rich opportunity for analysis, particularly with the rise of social media platforms like Twitter, Facebook, and others. The study focuses on Twitter, a prominent microblogging platform, to analyze public sentiment towards lawn tennis. By examining positive, neutral, and negative reviews, the research aims to understand the global reception of this outdoor game and its popularity across different countries. This approach sheds light on the potential of sentiment analysis to assess public opinion on various topics, including sports and leisure activities. It highlights the value of social media data for gauging public interest and understanding trends.

Nagarajan, Gandhi, their study presented herein offers a comprehensive analysis of leveraging Twitter data to discern the political inclinations of users through sentiment extraction and classification methodologies. Through a fusion of natural language processing techniques, sentiment analysis algorithms, and machine learning approaches, the research endeavours to accurately categorize user profiles based on expressed sentiments within tweets. The research extends its utility by predicting sentiment towards uncharted keywords, thereby enabling a nuanced understanding of user biases towards political events or social issues. By harnessing a robust dataset encompassing 1.72 million tweets from over 10,000 profiles, the study demonstrates a remarkable proficiency in identifying user political orientations, attaining a commendable 99% confidence level.

Rodrigues, this study offers a significant advancement in sentiment analysis for Twitter data. The HL-NBC method, with its combined topic classification and hybrid classification model, provides a powerful tool for extracting insights from the ever-growing stream of Twitter content. The HL-NBC approach outperforms existing methods like Lexicon and Naive Bayes classifiers, achieving an accuracy of 82% in sentiment classification. Beyond improved accuracy, the HL-NBC method is significantly faster (93% improvement) compared to traditional methods, making it efficient for handling large datasets.

B. Pang’s study evaluates three machine learning methods—Naive Bayes, maximum entropy classification, and support vector machines—against human-produced baselines, particularly focusing on sentiment classification using movie reviews as data. Despite the demonstrated superiority of machine learning techniques over human baselines, the results reveal a notable performance gap between sentiment and topic-based classification. This discrepancy underscores the inherent complexity of sentiment analysis, characterized by nuanced expressions and contextual subtleties. Through an in-depth analysis, the paper endeavours to elucidate the intricacies of sentiment classification, shedding light on factors that contribute to its heightened difficulty compared to traditional categorization tasks.

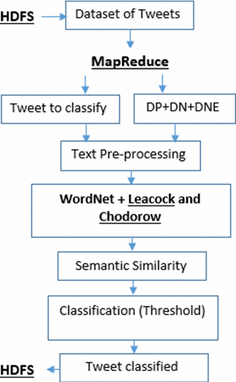
H. Liu, I. Chatterjee, M. Zhou, X. S. Lu, and A. Abusorrah, The study of sentiment analysis plays a crucial role in understanding public opinion, market research, brand reputation, customer experiences, and social media influence. Its application spans various industries, prompting the development of methods tailored to different needs for aspect granularity, including document, sentence, and aspect-based analysis. In addressing aspect-based sentiment analysis, recent studies have focused on three main methodologies: lexicon-based approaches, traditional machine learning techniques, and deep learning methods. This survey article offers a comprehensive overview of state-of-the-art deep learning methodologies within this domain.Benchmark datasets and evaluation metrics are essential components in assessing the performance of deep learning methods for aspect-based sentiment analysis. This review incorporates a discussion on commonly used datasets, evaluation metrics, and the comparative performance of existing deep learning techniques. Despite the advancements made in deep learning methodologies, there remain challenges and opportunities for further research. The literature review concludes by highlighting existing limitations and proposing future research directions in the field of aspect-based sentiment analysis.

P. Vyas, M. Reisslein, B. P. Rimal, G. Vyas, G. P. Basyal, and P. Muzumdar. Their study specifies that emergence of social media platforms, particularly microblogging sites like Twitter, has significantly impacted information sharing in modern society, especially during notable events such as the COVID-19 pandemic. In response to this trend, researchers have developed automated frameworks aimed at extracting sentiments from tweets, enabling a deeper understanding of societal sentiments during profound events like the COVID-19 crisis. Evaluation of such frameworks typically involves measures such as precision, accuracy, recall, and F1 score, providing insights into the effectiveness and reliability of sentiment analysis techniques. In the case of the mentioned framework, results show a predominance of positive and neutral sentiments among tweets related to COVID-19, with positive sentiments comprising 38.5% and neutral sentiments 34.7%.Moreover, the framework's performance evaluation highlights the Long Short-Term Memory (LSTM) neural network as the preferred machine learning technique, achieving an accuracy rate of 83%. This finding underscores the effectiveness of LSTM networks in capturing complex patterns within tweet data and accurately classifying sentiments expressed by users. G. Zhao, Y. Luo, Q. Chen, and X. QianG. Their work explores a novel approach to enhance aspect-based sentiment analysis (ABSA). ABSA aims to identify aspects (e.g., "battery life") and their corresponding sentiment (positive, negative) in reviews. It highlights the challenges of ABSA, which typically involves two separate tasks: Aspect Term Extraction (ATE): Identifying the specific aspects mentioned in text. Aspect Polarity Classification (APC): Classifying the sentiment towards those aspects. The passage criticizes current methods that train these tasks independently, neglecting the natural connection between them. To address this, the authors propose a multitask learning model that tackles both ATE and APC simultaneously. This allows the model to leverage the ATE task as an auxiliary tool, enabling. The proposed solution utilizes a multi-head attention mechanism to link dependency sequences with aspect extraction. This not only combines ATE and APC tasks but also emphasizes important dependency relationships. By focusing on words closely related to aspects, the model achieves better sentiment classification. The passage concludes by mentioning successful evaluations on benchmark datasets, demonstrating the effectiveness of their approach in establishing a stronger connection between ATE and APC for improved ABSA performance.

**3. PROPOSED METHOD**

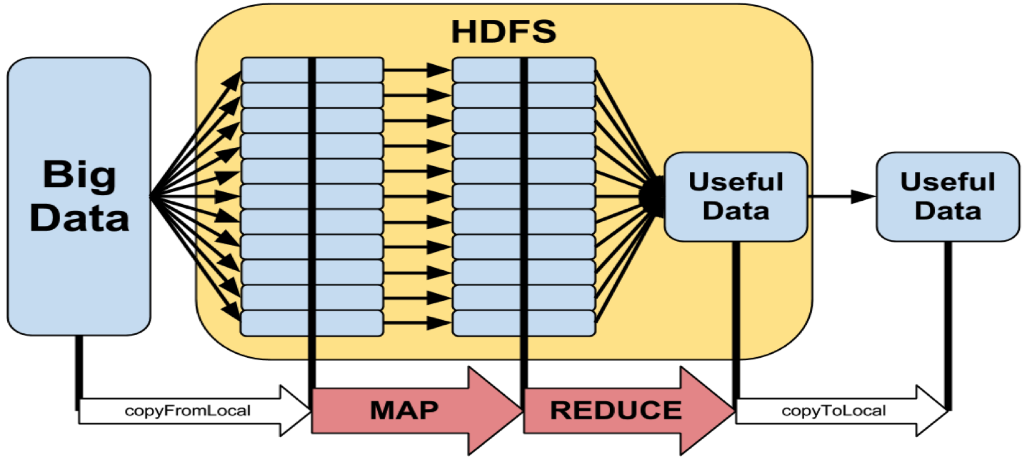
Our proposed system architecture for real-time sentiment analysis integrates several key components to enable efficient processing of large-scale textual data streams. At the core of the architecture lies the utilization of Hadoop and Apache Spark, two distributed computing frameworks renowned for their scalability and processing capabilities. Hadoop facilitates distributed storage and processing, enabling the system to manage vast volumes of data efficiently. Meanwhile, Apache Spark plays a crucial role in data preprocessing and feature extraction, essential steps in the sentiment analysis pipeline.

Furthermore, our architecture incorporates deep learning algorithms, specifically Recurrent Neural Network for sentiment classification. RNNs excel at capturing intricate patterns and relationships within textual data, making them well-suited for sentiment analysis tasks. By leveraging the power of deep learning, our system aims to achieve higher accuracy in sentiment classification compared to traditional machine learning techniques.



**Figure 3.1 Architecture**

The flow of data within the system begins with the ingestion of textual data streams, which are then processed and analyzed in real-time. Hadoop handles the distributed storage of raw data, while Spark facilitates data preprocessing tasks such as tokenization, normalization, and feature extraction. The pre-processed data is then fed into the RNN model, which classifies the sentiment of each text input as positive, negative, or neutral.

Overall, our system architecture offers a scalable and efficient solution for real-time sentiment analysis, capable of handling the demands of modern data-intensive environments. By integrating distributed computing frameworks like Hadoop and Spark with deep learning techniques such as CNNs, our architecture enables accurate and timely sentiment analysis on large-scale textual data streams.

**Figure 3.2 Hadoop Setup (Map Reduce)**

Performing real-time sentiment analysis using the Hadoop framework and Deep Learning involves several steps:

**3.1 DATA COLLECTION**

Gathering real-time data from various sources such as social media, news feeds, or any other relevant platforms. The data will contain text that needs to be analysed for sentiment. Social media platforms like Twitter function as live feeds, where users express their sentiments and opinions through text-based content. Fortunately, many of the platforms offer APIs (Application Programming Interfaces) that facilitate programmatic data collection. APIs grant access to user-generated content, making them ideal for large-scale sentiment analysis endeavours.

**3.2. DATA PRE-PROCESSING**

Data pre-processing is a crucial step in sentiment analysis, especially when dealing with unstructured text data from sources like social media. The process involves several techniques to clean and prepare the data for analysis. The steps involved are:

* **Text Cleaning:** This includes removing special characters, correcting typos, and converting all text to lowercase to maintain consistency.
* **Tokenization**: Breaking down the text into individual words or tokens to analyze them separately.
* **Stop Words Removal:** Eliminating common words like ‘the’, ‘is’, ‘at’, which don’t contribute to sentiment.
* **Stemming and Lemmatization:** Reducing words to their root form to ensure that different forms of a word are analyzed as one.
* **Vectorization**: Converting text into numerical data that machine learning models can understand. Common methods include Bag-of-Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), and word embeddings.
* **Feature Selection:** Choosing the most relevant features that contribute to the sentiment of the text.

**3.3. HADOOP SETUP**

The Hadoop framework handles the issues with parallel executions, like recovery between the tasks, detection of failure and automatic tasks synchronization and dealt with large-scale data processing. In Hadoop, the data are processed by the user using MapReduce models. Thus, the Hadoop framework is used for effective analysis of sentiment using twitter data. Sentiment analysis is a procedure for analyzing and classifying the opinion from text data. The sentimental analysis provided an overview of the public opinion about the certain topics. Here, the sentiment analysis is done using the Hadoop framework and deep learning classifier. Initially, the input twitter data is subjected to Hadoop cluster to distribute data for the extraction of features. The extraction of feature is carried out in the mapper phase. In the mapper phase, the significant features, like all-caps, emoticon, hashtag, elongated units, sentiment lexicon, negation, and punctuation are extracted from the twitter data. The obtained features are then fed in the shuffle, where lists of unique features are selected. The list of unique features is fed to the reducer. In the reducer phase, the features are classified using deep recurrent neural network classifier, which classifies the features into two classes, namely positive review and negative review. Figure 1 portrays the schematic view for analyzing sentiments.

**3.4. DEEP LEARNING**

It is training a deep learning model for sentiment analysis. This involves using frameworks like TensorFlow to build and train a deep learning model such as a recurrent neural network (RNN) or a convolutional neural network (CNN) for sentiment analysis. RNNs are well-suited for sentiment analysis, but real-time applications require some tweaks. But the challenge is unlike traditional analysis (feeding a whole dataset), real-time involves a continuous stream of data (tweets, chats). The model needs to analyze this data quickly, providing sentiment analysis with minimal delay. The solution is We can't wait for all the data to arrive. The data stream is split into smaller chunks (mini-batches). The RNN processes each batch and updates its internal parameters after each pass, continuously learning and adapting to the sentiment flow. RNN considers Latency and data processing.

**3.5 INTEGRATION WITH HADOOP**

Integrating the deep learning model with Hadoop for real-time analysis can be done using tools like Apache Spark, which provides APIs for running computations on large datasets in a distributed manner.

* Kafka collects real-time news data from various sources. It ensures seamless data flow for instant analysis.
* Hadoop stores and manages vast news datasets efficiently. Its distributed file system (HDFS) scales effortlessly.
* Spark cleanses and pre-processes raw news data. It transforms data for sentiment analysis in near real-time.
* Our trained model discerns sentiments within news articles. It adapts to varying tones and contexts.
* The trained model is deployed within the Hadoop ecosystem, where it can analyze new data in real-time and provide sentiment predictions.
* Hadoop’s distributed nature allows the deep learning model to scale across multiple nodes, handling large-scale data efficiently.

**3.6. REAL-TIME ANALYSIS**

* As the data is distributed across the cluster, it can be processed in parallel by different nodes. Here, a deep learning model can be utilized to perform complex analyses on the incoming data. This could involve tasks such as anomaly detection, pattern recognition, sentiment analysis, or any other task that the deep learning model is trained for.
* Hadoop's distributed processing framework, typically MapReduce or Apache Spark, enables parallel execution of tasks across the nodes of the cluster. This allows for efficient processing of large volumes of data in real-time. As the data is processed by the deep learning model, insights or actions can be derived based on the analysis performed.
* The results of the analysis can be fed back into the system for further refinement of the deep learning model. This continuous feedback loop helps improve the accuracy and effectiveness of the real-time analysis over time.

**3.7. VISUALIZATION AND REPORTING**

* We will use graphical representations to showcase sentiment data. Using barcharts or heat maps. Later we’ll summarize the findings from the sentiment analysis in a report that includes an executive summary, methodology, results and conclusions.

**4. IMPLEMENTATION**

**4.1 WORKING FLOW**

The workflow of the project involves several key stages, beginning with data collection and ending with real-time sentiment analysis using Hadoop, Apache Spark, and RNNs. The project begins with the collection of a Twitter dataset, sourced from Kaggle or similar repositories. This dataset serves as the foundation for the sentiment analysis task, containing a diverse range of textual data reflecting users' opinions and sentiments on various topics.

Once the Twitter dataset is collected, the next step involves preprocessing and cleaning the data to ensure its suitability for sentiment analysis. This preprocessing stage typically includes tasks such as tokenization, removing stopwords, handling special characters, and converting text to lowercase. These steps help to standardize the textual data and improve the accuracy of sentiment analysis algorithms by reducing noise and irrelevant information.

After preprocessing, the cleaned Twitter dataset is then ingested into the Hadoop Distributed File System (HDFS) for distributed storage and processing. Hadoop provides a scalable and fault-tolerant infrastructure for handling large volumes of data across clusters of commodity hardware. The dataset is partitioned and distributed across multiple nodes in the Hadoop cluster, ensuring efficient data storage and retrieval.

Next, Apache Spark is utilized for data processing and feature extraction. Spark's high-level APIs and in-memory processing capabilities enable efficient data manipulation and transformation, making it well-suited for complex analytics tasks. In this project, Spark is used to preprocess the Twitter dataset further, extract relevant features from the text data, and prepare it for sentiment analysis.

The final stage of the workflow involves performing real-time sentiment analysis using Recurrent Neural Networks (RNNs). RNNs are deep learning algorithms capable of learning complex patterns and relationships within textual data, making them well-suited for sentiment analysis tasks. The pre-processed Twitter dataset is fed into the RNN model, which classifies the sentiment of each text input as positive, negative, or neutral in real-time. The output of the sentiment analysis model provides valuable insights into users' opinions and sentiments on the topics discussed on Twitter.

Overall, the workflow of the project encompasses data collection, preprocessing, distributed storage and processing using Hadoop, data manipulation and feature extraction using Apache Spark, and real-time sentiment analysis using Recurrent Neural Networks, culminating in the extraction of valuable insights from Twitter data.

**4.2** **ATTRIBUTES**

**1. appName():** It is used to said the application name.

**2. builder():** It is used to create a SparkSession.

**3. getOrcreate():** It returns a SparkSession if it exists, otherwise it creates a new session.

**4.** **tweets\_csv:** It used for reading the input file which contains positive or negative tweets. The file is named as tweets.csv in which it contains columns such as itemID, Sentiment, SentimentSource, SentimentText .

**5. dividedData():** It is used for splitting the data into training and testing data, in which it uses randomSplit() which is function in PySpark, used to randomly split a dataset into two or more subsets with a specified ratio.

**6. KerasTokenizer():** Tokenizers convert raw string input into integer input suitable for a Keras Embedding layer. They can also convert back from predicted integer sequences to raw string output. All tokenizers subclass keras\_nlp. tokenizers. Tokenizer , which in turn subclasses keras.

**7. flatMap():** flatMap can be used as a way to add and remove items (modify the number of items) during a map . In other words, it allows you to map many items to many items (by handling each input item separately), rather than always one-to-one. In this sense, it works like the opposite of filter.

**8. Collect():** Collect() is the function, operation for Resilient Distributed Datasets (RDD) or Dataframe that is used to retrieve the data from the Dataframe.

**9. pad\_sequences():** pad\_sequences is a function commonly used in natural language processing (NLP) and other sequence-based tasks to ensure that input sequences have the same length by padding them with a specific value (often a special token or zeros) or truncating them if they are too long.

**10. embedding\_dim():** It talks about size of each embedding vector(as all the learned vectors will have a fixed size).

**11. rnn\_units():** It processes input data sequentially while maintaining an internal state. This internal state allows the network to remember information from previous time steps, making it suitable for processing sequential data like time series or text.

**12. Sequential():** A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor. The sequential model allows us to specify a neural network, precisely, sequential: from input to output, passing through a series of neural layers, one after the other.

**13. sigmoid():** The Sigmoid function performs the role of an activation function in machine learning which is used to add non-linearity in a machine learning model. Basically, the function determines which value to pass as output and what not to pass as output.

**14. Dense():** Tensorflow dense is the type of layer and function available in Neural networks while implementing Artificial Intelligence and deep learning in a python programming language. Deep connections exist between the neurons in the neural network in dense layers.

**15. epoch():** An epoch in machine learning means one complete pass of the training dataset through the algorithm. This epoch's number is an important hyperparameter for the algorithm. It specifies the number of epochs or complete passes of the entire training dataset passing through the training or learning process of the algorithm.

**16. validation\_split():** it is the parameter specifying how big chunk of training data will be used for validation. It's a float value between 0 and 1. Validation data is not used for the training, but to evaluate the loss and the accuracy. For example: validation\_split=0.3 will cause that 30% of the training data will be used for validation.

**17. model.save():** It is used to save a deep learning model in PyTorch. This method saves the entire model, including the model architecture and weights, in a format that can be loaded later to make predictions.

**18. model.predict():** given a trained model, predict the label of a new set of data. This method accepts one argument, the new data X\_new (e.g. model. predict(X\_new)), and returns the learned label for each object in the array.

**19. confusion\_matrix():** A confusion matrix is a performance evaluation tool in machine learning, representing the accuracy of a classification model. It displays the number of true positives, true negatives, false positives, and false negatives.

**20. classification\_report():** A classification report is a text summary that shows the main metrics for each class of a machine learning model. It usually includes the precision, recall, F1-score, and support for each class, as well as the weighted average of these metrics across all classes.

**21. legend():** A legend is a predefined function legend() that creates an area on the graph which describes all the elements of a graph.

**22. spark.stop():** It is used to stop the session.

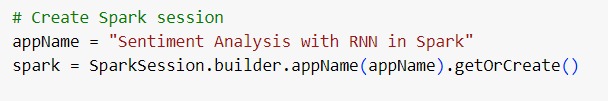
**4.3 EXPERIMENT SCREENSHOTS**



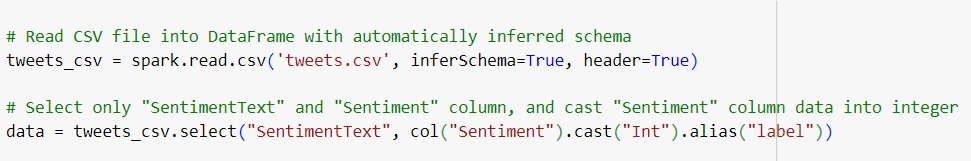
**Installing PySpark**: The “!pip install pyspark” command is used to install the PySpark library, which is an interface for Apache Spark in Python. It allows for big data processing with the power of Spark's distributed computing capability.

**Importing Libraries:** The code imports various Python libraries such as numpy for numerical operations, SparkSession from PySpark for initializing Spark, Tokenizer for text processing, and Keras for building neural networks.

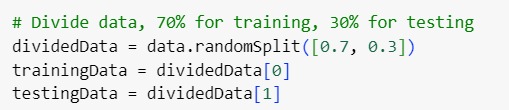
The matplotlib library is imported for plotting graphs, and sklearn.metrics is used for evaluating the performance of machine learning models with functions like confusion\_matrix and classification\_report.



It is a Python code for initializing a Spark session, which is the first step in creating a Spark application. The code sets the application name to "Sentiment Analysis with RNN in Spark," indicating that the application will perform sentiment analysis using a Recurrent Neural Network (RNN) within the Spark framework. This session creation is essential for distributing data processing tasks across a cluster, enabling large-scale data analysis and machine learning operations.



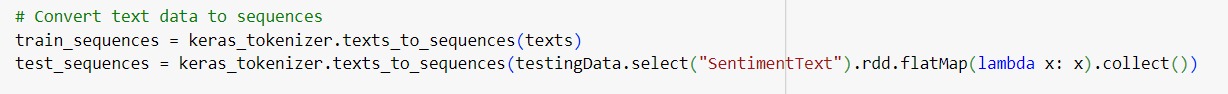
It contains Python code for reading and processing a CSV file into a DataFrame, and then selecting specific columns for further analysis. The code snippet suggests that the data being handled is related to sentiment analysis, as indicated by the selection of "SentimentText" and "Sentiment" columns, with the latter being cast to an integer type. This is a common step in preparing data for machine learning tasks, where text data is used to train models to classify sentiments as positive, negative, or neutral. The use of a DataFrame indicates that the code is likely part of a larger data processing or machine learning pipeline, possibly using the PySpark framework for scalable data analysis.



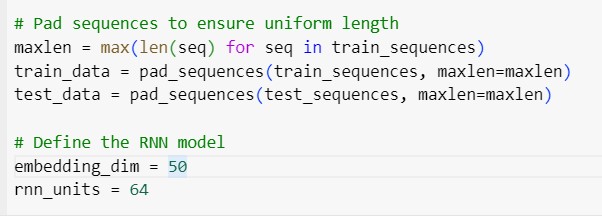
This code snippet that is used for splitting a dataset into training and testing sets. The code uses the randomSplit() method to divide the data, allocating 70% for training and 30% for testing. This is a common practice in machine learning to evaluate the performance of a model, ensuring that it can generalize well to new, unseen data. The trainingData set is used to train the model, while the testingData set is used to test and validate the model's predictions.



This is setting up a tokenizer using Keras, a popular deep learning library. The tokenizer is being defined and then fitted on text data extracted from a DataFrame column named "SentimentText". This is a common preprocessing step in natural language processing (NLP) tasks, where text data is converted into a format that can be fed into machine learning models for training. In this case, the tokenizer will create a dictionary of words based on the frequency of their occurrence in the data and transform the text into sequences of integers, where each integer represents a unique word.

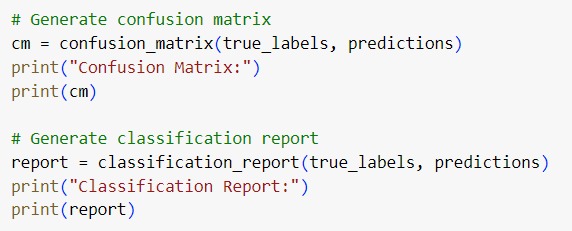


This code is for converting text data into sequences, which is a common preprocessing step in natural language processing (NLP). The code uses a Keras tokenizer to transform the text from the "SentimentText" column of a DataFrame into numerical sequences. These sequences are then used for training and testing in machine learning models, such as Recurrent Neural Networks (RNNs), which are effective for tasks like sentiment analysis. Essentially, this process translates words into numbers that the model can understand and learn from.



This involves padding sequences to ensure they have uniform length and defining the parameters for a RNN model. The code is preparing the data for input by making sure all sequences are the same length.. It also sets the dimensions for the embedding layer and the RNN units.

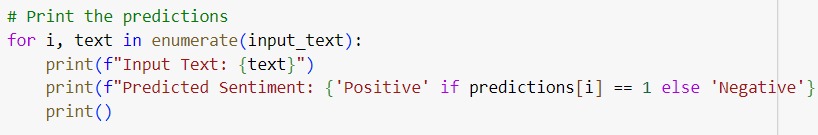
The confusion\_matrix function is called with true\_labels and predictions as arguments, suggesting that the model’s predictions are being compared against the true labels to assess performance.



The confusion matrix is a table that allows visualization of the performance of an algorithm. Each row of the matrix represents the instances in an actual class, while each column represents the instances in a predicted class.



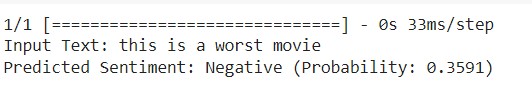
Giving the input text to categorize into positive or negative.



Printing the predictions for a given set of input texts. The predictions are determined by the values in a list called predictions, where a value of 1 indicates a positive sentiment and any other value indicates a negative sentiment. The use of Python's f-string formatting allows for the dynamic insertion of the input text and the predicted sentiment into the print statements.



This command is used to stop the Spark session. This command is typically called at the end of a Spark program to close the session and free up resources. It ensures allocated resources are properly released after the computations are completed.



This is the result of the sentiment analysis where the input text was “this is a worst movie.” The sentiment analysis algorithm has determined that the sentiment of this text is negative, with a probability of 0.3591. This means that the algorithm is 35.91% confident that the sentiment expressed in the text is negative.

**5.EXPERIMENTAL SETUP**

**5.1 SOFTWARE REQUIREMENTS:**

**5.1.1 Hadoop:** The project requires installation and configuration of the Hadoop distributed computing framework to facilitate distributed storage and processing of large-scale data. We used Hadoop version Hadoop 3.3.0. which is the latest version. Hadoop 3 has advantages like Min Java version is JDK 8.0, supports erasure coding, distributed scheduling, revision of YARN timeline service, opportunistic containers, revoke daemon, intra-DataNode balancer, more than two NameNodes, etc.

**5.1.2 Apache Spark:** Apache Spark needs to be installed to leverage its in-memory data processing capabilities for efficient data manipulation and feature extraction. We used Apache 3.4.1 as per latest edition. Spark Connect is a new client-server architecture introduced in Spark 3.4 that decouples Spark client applications and allows remote connectivity to Spark clusters. The separation between client and server allows Spark and its open ecosystem to be leveraged from anywhere, embedded in any application. In Spark 3.4, Spark Connect provides DataFrame API coverage for PySpark and DataFrame/Dataset API support in Scala.

**5.1.3 Deep Learning Framework:** A deep learning framework such as TensorFlow or PyTorch is required for implementing Recurrent Neural Networks (RNNs) for real-time sentiment analysis. A recurrent neural network (RNN) is a deep learning model that is trained to process and convert a sequential data input into a specific sequential data output. Sequential data is data—such as words, sentences, or time-series data—where sequential components interrelate based on complex semantics and syntax rules.

**5.2 LIBRARIES USED:**

Various Python libraries such as Pandas, NumPy, NLTK (Natural Language Toolkit), and Scikit-learn may be necessary for data preprocessing, feature extraction, and model evaluation.

**5.2.1 Pandas**

Pandas is a Python library that helps you work with data in a way that's simple and intuitive. It introduces two main data structures: DataFrame, which is like a table with rows and columns, and Series, which is like a single column of data. With Pandas, you can easily load data from various file formats like CSV or Excel, manipulate it by filtering, sorting, and summarizing, handle missing values, and perform calculations. It's particularly handy for tasks like cleaning messy data, exploring datasets, and preparing them for analysis or visualization. Whether you're a beginner or an experienced data scientist, Pandas makes it easy to get your data into shape so you can focus on extracting insights and making decisions.

**5.2.2 NumPy:**

NumPy is a fundamental Python library widely used for numerical computing tasks. It offers robust support for handling large, multi-dimensional arrays and matrices efficiently. With its extensive collection of mathematical functions, NumPy is indispensable in various scientific disciplines, including machine learning, data analysis, physics, and engineering. NumPy's another significant advantage is enabling arithmetic operations on arrays with different shapes by automatically adjusting the smaller array's shape to match the larger array's shape, if possible.

**5.2.3 KerasTokenizer:**

Keras Tokenizer is a utility in the Keras deep learning library for tokenizing text data, which involves converting text into numerical sequences that can be fed into neural networks. It enables preprocessing text data by tokenizing strings into individual words or characters, converting them to lowercase, and optionally filtering out punctuation or other special characters. Tokenizer also allows users to limit the vocabulary size, handle out-of-vocabulary tokens, and pad sequences to ensure uniform input dimensions. This functionality is crucial for natural language processing (NLP) tasks such as text classification, sentiment analysis, and machine translation, as it facilitates the transformation of raw text data into a format suitable for deep learning models.

**5.2.4 Scikit-learn:**

Scikit-learn, often referred to as sklearn, is a comprehensive open-source machine learning library for the Python programming language. It provides simple and efficient tools for data mining and data analysis, covering a wide array of machine learning algorithms and techniques, including classification, regression, clustering, dimensionality reduction, model selection, and preprocessing. With a consistent and user-friendly interface, scikit-learn allows users to easily implement and experiment with various machine learning models and workflows. It also emphasizes code readability, documentation, and ease of use, making it a popular choice for both beginners and experienced practitioners in the field of machine learning and data science.

**5.2.5 NLTK (Natural Language ToolKit):**

NLTK (Natural Language Toolkit) is a comprehensive open-source library for natural language processing (NLP) tasks in Python. It offers a wide range of tools and resources for tasks such as tokenization, stemming, lemmatization, part-of-speech tagging, named entity recognition, sentiment analysis, and more. NLTK provides access to numerous corpora and lexical resources, making it suitable for both educational purposes and real-world NLP applications. With its intuitive interface and extensive documentation, NLTK is widely adopted by researchers, students, and professionals alike, empowering them to explore, analyze, and process textual data effectively in various domains.

**5.3 PARAMETERS**

The analysis of the proposed Hadoop based deep RNN method is performed considering performance measures, like classification accuracy, precision, recall, F1 score.

**Classification accuracy** is a metric used to evaluate the performance of a classification model. It is calculated as the number of correct predictions divided by the total number of predictions made, expressed as a percentage illustrated in Equation (1).

Accuracy = % (1)

**F1 score** is a machine learning evaluation metric that measures a model’s accuracy. It combines the precision and recall scores of a model, illustrated in (2).

F1-score **=**  (2)

**Precision** refers to the amount of information that is conveyed by a number in terms of its digit. It shows the closeness of two or more measurement to each other. It is independent of accuracy, illustrated in (3).

Precision = (3)

**Recall**, also known as the true positive rate (TPR), is the percentage of data samples that a machine learning model correctly identifies as belonging to a class of interest—the “positive class”—out of the total samples for that class, illustrated in (4).

Recall = (4)

where,

A **true positive** is an outcome where the model correctly predicts the positive class.

A **true negative** is an outcome where the model correctly predicts the negative class.

A **false positive** is an outcome where the model incorrectly predicts the positive class.

A **false negative** is an outcome where the model incorrectly predicts the positive class.

**6.DISCUSSION OF RESULTS**

The confusion matrix (Figure 6.1) is a table that allows visualization of the performance of an algorithm. Each row of the matrix represents the instances in an actual class, while each column represents the instances in a predicted class. This matrix is particularly useful because it shows not only the correct predictions but also the types of errors made by the classifier. In the code, the confusion\_matrix function is called with true\_labels and predictions as arguments, suggesting that the model’s predictions are being compared against the true labels to assess performance.

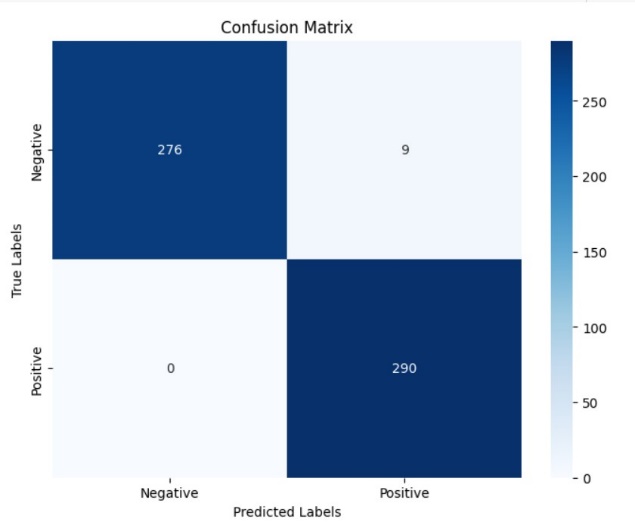


Figure 6.1 Confusion Matrix

The below Fig 4.2 depicts the overall classification report of the proposed method.

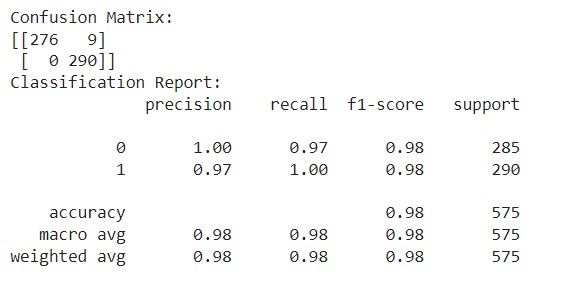


Figure 6.2 Classification Report

This is a confusion matrix and a classification report from a binary classification model

Confusion Matrix: This matrix shows the number of correct and incorrect predictions made by the model. It indicates that the model predicted 276 true positives and 290 true negatives, with only 9 false positives and no false negatives, which suggests a high level of accuracy.

Classification Report: This report provides metrics such as precision, recall, and f1-score for both classes (0 and 1). Both classes have high scores across all metrics, with precision and recall being perfect (1.00) for class 0 and nearly perfect for class 1 (0.97). The f1-score, which is a balance between precision and recall, is also high for both classes (0.98), indicating a well-performing model.

Overall, the model’s accuracy is shown to be 0.98, which means it is correct 98% of the time. The macro and weighted averages for precision, recall, and f1-score are also 0.98, further confirming the model’s strong performance. This kind of analysis is crucial in evaluating the effectiveness of machine learning models

The below Fig 4.3 and Fig 4.4 shows the training and validation loss & accuracy, which are based on the above results.

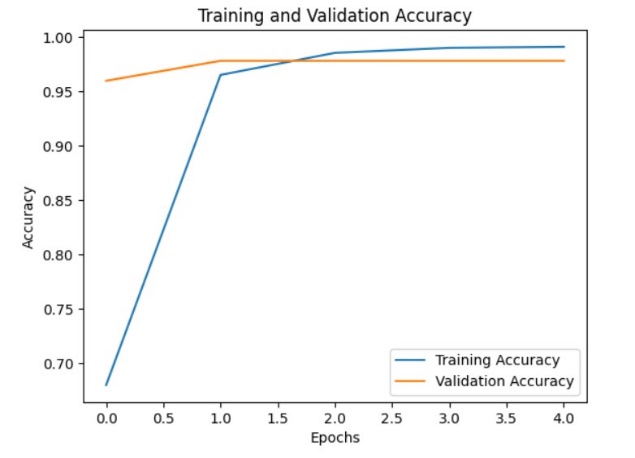


Figure 6.3 training and validation accuracy

This graph is expressing about “Training and Validation Accuracy,” which represents the performance of a machine learning model over several epochs.

Epochs: The x-axis shows the number of epochs, ranging from 0 to 4, which are iterations over the entire data set during the training process.

Accuracy: The y-axis measures the accuracy, ranging from 0.70 to 1.00, indicating the proportion of correct predictions made by the model.

Training Accuracy (Blue Line), shows a sharp increase in accuracy during the initial epochs, which then plateaus near perfect accuracy (1.00).

Validation Accuracy (Orange Line), represents the accuracy on a separate set of data not seen by the model during training. It increases more gradually, suggesting the model is learning and generalizing well, but not overfitting.

Overall, the graph indicates that the model is performing well, with high accuracy on both the training and validation sets. This kind of visualization is crucial for understanding model performance and diagnosing issues like overfitting or underfitting.



Figure 6.4 training and validation loss

This depicts a graph showing the training and validation loss of a machine learning model over a series of epochs.

Training Loss (Blue Line), starts at a higher loss value and decreases sharply, indicating rapid learning during the initial phase of training. It then levels off, suggesting that the model has learned as much as it can from the training data.

Validation Loss (Orange Line), starts slightly lower and decreases at a more gradual rate. This slower decline indicates that the model is generalizing the patterns it learned during training to new, unseen data.

The graph is a common way to visualize the performance of a machine learning model, with the goal being to minimize the loss, which represents the error between the predicted values and the actual values. The ideal scenario is when both lines decrease and then level off close to each other, which would indicate a well-fitted model.

Table 1 presents the average accuracy scores and other parameters obtained by using previous method and proposed method applied on the same dataset. Our observation indicates that the proposed method excels, achieving an accuracy score of 0.91 surpassing other methods.

TABLE I

|  |  |  |
| --- | --- | --- |
| **parameter** | **Previous method** | **Proposed method** |
| Accuracy | 98.6% | 98.7% |
| Recall | 0.97 | 0.98 |
| F1**-**Score | 0.98 | 0.99 |
| Precision | 0.98 | 0.99 |

The sentiment analysis results show a comparison between a previous method and a proposed method.

**Accuracy:** The proposed method shows a slight improvement in accuracy, increasing from 98.6% to 98.7%. This indicates that the proposed method is slightly better at correctly identifying sentiments.

**Recall:** There is an increase in recall from 0.97 to 0.98 with the proposed method. This means it is better at identifying all relevant instances of the positive class in the dataset.

**F1-Score:** The F1-Score, which balances precision and recall, has improved from 0.98 to 0.99. This suggests that the proposed method has a better overall performance in terms of precision and recall.

**Precision:** Precision has also increased from 0.98 to 0.99. This means the proposed method has a higher likelihood of labelling an instance as positive only if it is truly positive.

Overall, the proposed method shows marginal but consistent improvements across all metrics, suggesting it may be more effective for sentiment analysis tasks.

**7. CONCLUSION**

This research has demonstrated the effectiveness of this approach, showcasing how it can be used to achieve accurate and timely sentiment analysis on streaming data sources. By harnessing the scalability and parallel processing capabilities of Hadoop, coupled with the sophisticated analysis provided by deep learning models, this approach offers substantial advantages in industries such as marketing, customer service, and social media monitoring. One of the key strengths of this approach is its ability to handle large volumes of data in real-time. Hadoop's distributed file system (HDFS) allows for the storage and processing of massive datasets across a cluster of commodity hardware, enabling the analysis of data streams as they are generated. This is particularly important in applications such as social media monitoring, where the volume of data being generated can be enormous and traditional analysis techniques may struggle to keep up. Additionally, the use of deep learning models adds a layer of complexity and sophistication to the analysis. Deep learning algorithms, such as recurrent neural networks (RNNs), are well-suited to capturing the nuanced and context-dependent nature of sentiment in text data. By training these models on large datasets, they can learn to identify subtle patterns and correlations that may not be apparent to traditional machine learning algorithms. Furthermore, the integration of Hadoop with deep learning enables the analysis to be performed in a highly parallelized manner. This means that different parts of the analysis can be carried out simultaneously across multiple nodes in the Hadoop cluster, significantly reducing the time taken to process the data. By demonstrating the effectiveness of integrating scalable big data technologies like Hadoop with advanced machine learning techniques like deep learning, this research opens up new possibilities for applications in diverse fields.

**8. REFERENCES**

[1] Zaharia, Matei, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. "Spark: Cluster computing with working sets." HotCloud 10, no. 10-10 (2010): 95.

[2] Zhang, Lei, Shuai Wang, and Bing Liu. "Deep learning for sentiment analysis: A survey." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8, no. 4 (2018): e1253.

[3] Li, Chunping, Jianhua Zhang, Guangquan Zhang, and Yongquan Yang. "Big data sentiment analysis: Challenges and opportunities." IEEE Access 8 (2020): 15384-15403.

[4] Sumbaly, Ravi. Apache Flume: Distributed log collection for Hadoop. "O'Reilly Media, Inc.", 2013.

[5] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: Simplified data processing on large clusters." Communications of the ACM 51, no. 1 (2008): 107-113.

[6] Dua, D. and Graff, C. "UCI machine learning repository." University of California, Irvine, School of Information and Computer Sciences (2019).

[7] Malik, M., Naaz, S. and Ansari, I.R., "Sentiment Analysis of Twitter Data Using Big Data Tools and Hadoop Ecosystem," In proceedings of International Conference on ISMAC in Computational Vision and Bio-Engineering, Springer, pp. 857-863, 2018.

[8] Nagarajan, S.M. and Gandhi, U.D., "Classifying streaming of Twitter data based on sentiment analysis using hybridization," Neural Computing and Applications, vol.31, no.5, pp.1425-1433, 2019.

[9] Rodrigues, A.P. and Chiplunkar, N.N., "A new big data approach for topic classification and sentiment analysis of Twitter data," Evolutionary Intelligence, pp.1-11, 2019.

[10] B. Pang, L. Lee, S. Vaithyanathan, and S. Jose, “sentiment classification using machine learning techniques,” in Proceedings of theACL-02 conference on Empirical methods in natural language processing, vol.10, pp. 79–86.

[11] G. Zhao, Y. Luo, Q. Chen, and X. Qian, ‘‘Aspect-based sentiment analysis via multitask learning for online reviews,’’ Knowl.-Based Syst., vol. 264, Mar. 2023, Art. no. 110326, doi: 10.1016/j.knosys.2023.110326.

[12] [1]. Agarwal, A., Xie, B., Vovsha, I., Rambow, O. and Passonneau, R., "Sentiment analysis of twitter data," In Proceedings of the Workshop on Language in Social Media, pp. 30-38, 2011.

[13] Kharde, V. and Sonawane, P., "Sentiment analysis of twitter data: a survey of techniques," arXiv preprint arXiv:1601.06971, 2016.

[14] E l Alaoui, I., Gahi, Y. and Messoussi, R., "Big Data Quality Metrics for Sentiment Analysis Approaches," In Proceedings of the 2019 International Conference on Big Data Engineering, ACM, pp. 36-43, 2019.

[15] Asghar, M.Z., Kundi, F.M., Ahmad, S., Khan, A. and Khan, F., "TSAF: Twitter sentiment analysis framework using a hybrid classification scheme," Expert Systems, vol.35, no.1, pp.12233, 2018 978-1-7281-4142-8/$31.00 ©2020 IEEE 5

[16] R. Alfred, W. W. Yee, Y. Lim, and J. H. Obit, “Factors affectingsentiment prediction of malay news headlines using machine learningapproaches,” Commun. Comput. Inf. Sci., vol. 652, pp. 289–299, 2016.

[17] J. Leskovec, D. Huttenlocher, and J. Kleinberg, “Predicting positive andnegative links in online social networks,” in Proceedings of the 19thinternational conference on World wide web, pp. 641–650, 2010.

[18] Tan, Y.S., Tan, J., Chng, E.S., Lee, B.S., Li, J., Date, S., Chak, H.P., Xiao, X. and Narishige, A., 2013. Hadoop framework: impact of data organization on performance. Software: Practice and Experience, 43(11), pp.1241-1260.

[19] Plunz, R.A., Zhou, Y., Vintimilla, M.I.C., Mckeown, K., Yu, T., Uguccioni, L. and Sutto, M.P., "Twitter sentiment in New York City parks as measure of well-being," Landscape and Urban Planning, vol.189, pp.235-246, 2019. [13]. Rodrigues, A.P. and Chiplunkar, N.N., "A new big data approach for topic classification and sentiment analysis of Twitter data," Evolutionary Intelligence, pp.1-11, 2019.

[20] Chan, C.C. and Liszka, K.J., "Application of rough set theory to sentiment analysis of microblog data," In Rough Sets and Intelligent Systems- Professor Zdzisaw Pawlak in Memoriam, Springer, pp. 185-202, 2013.

[21] A. Lin, “Improved Twitter Sentiment Analysis Using Naive Bayes andCustom Language Model,” arXivPrepr. arXiv1711.11081, 2017.

[22] Khan, M. and D. B. Kalra (2018). "AN INSPECTION ON BIG DATA COMPUTING." International Journal of Engineering & Science Research(Special Issue/Article No-52): 326-329.

[23] Duyu Tang, Bing Qin, Furu Wei, Li Dong, Ting Liu, & Ming Zhou, “A Joint Segmentation and Classification Framework for Sentence Level Sentiment Classification,” IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol.23, no.11, pp.1750–1761, 2015.

[24] Asghar, M.Z., Khan, A., Khan, F. and Kundi, F.M., "Rift: a rule induction framework for twitter sentiment analysis," Arabian Journal for Science and Engineering, vol.43, no.2, pp.857-877, 2018.

[25] J. Leskovec, D. Huttenlocher, and J. Kleinberg, “Predicting positive andnegative links in online social networks,” in Proceedings of the 19thinternational conference on World wide web, pp. 641–650, 2010.