PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research

UNIVERSITY OF ECHAHID
HAMMA LAKHDAR - EL OUED
FACULTY OF EXACT SCIENCES
Computer Science Department

Master Thesis Title:

Big Data Analytics in social media - A Case Study of Twitter

Specialty : Artificial intelligence and data science

Presented by:
KIRAM Mohammed Ali
ABID Imad Eddine

Proposed and Supervised by:
Dr. MAALOUL Kamel

-Academic year: 2025/2026-

Dedication

To my beloved parents, the source of my strength and support, and the reason behind everything I have achieved today.

To my respected teachers, who generously shared their knowledge and guidance.

To everyone who supported and encouraged me, even with a kind word.

To all who believe that success begins with a single step, I dedicate this humble work.

Acknowledgements

We would like to express our sincere gratitude and appreciation to everyone who contributed, directly or indirectly, to the completion of this work.

First and foremost, we would like to thank our supervisor, Dr. MAALOUL Kamel, for his valuable guidance, continuous support, and constructive feedback throughout the realization of this project.

We also extend our sincere thanks to the members of the jury for their time, evaluation, and valuable remarks.

Special thanks go to our teachers at the Department of Computer Science for the knowledge and guidance they provided throughout our academic journey.

Finally, we express our deepest gratitude to our families and friends for their constant encouragement, patience, and moral support.

CONTENTS

Dedication . . . 2
Acknowledgements . . . 3
Abstract . . . 8
Résumé . . . 9
Abstract in Arabic . . . 11

General Introduction 11

I Big Data 15
I.1 Introduction . . . 16
I.2 Definition of Big Data . . . 16
I.2.1 Review of Existing Definitions of Big Data . . . 16
I.2.2 Towards an Approximate Understanding of Big Data . . . 18
I.3 Characteristics of Big Data . . . 18
I.3.1 Core Characteristics of Big Data . . . 18
I.3.2 Complementary Characteristics of Big Data . . . 19
I.4 Concept and Tools of Big Data Analytics . . . 20
I.4.1 Concept of Big Data Analytics . . . 20
I.4.2 Tools of Big Data Analytics . . . 20
I.5 Conclusion . . . 22

II Twitter Big Data Analytics 23
II.1 Introduction . . . 24
II.2 Social Networks . . . 24
II.2.1 Twitter as a Microblogging Platform . . . 25
II.2.2 Twitter APIs . . . 25
II.2.3 Structure and Architecture of a Tweet . . . 26
II.3 Twitter Big Data Analytics . . . 27
II.3.1 Data Collection at Scale . . . 27
II.3.2 Data Preprocessing and Management . . . 28
II.3.3 Content-Based Analytics . . . 28
II.3.4 Similarity Measures and Trend Detection . . . 29
II.3.5 Network-Based Analytics . . . 29
II.3.6 Real-Time Analytics and Streaming Architectures . . . 30
II.3.7 Toward Integrated Analytics Frameworks . . . 30
II.3.8 Ethical and Governance Considerations in Twitter Big Data Analytics . 30

II.4 Conclusion 32

III Design and Implementation 33
III.1 Introduction 34
III.2 System Design 34
III.2.1 Use Case Diagram 34
III.2.2 Class Diagram 35
III.2.3 System Architecture 35
III.3 Development Environment and Tools 39
III.3.1 Big Data Frameworks 39
III.3.2 Data Acquisition and Datasets 41
III.3.3 Programming Languages and Prerequisites 41
III.3.4 Experimental Environment 42
III.4 Implementation 45
III.4.1 Batch Processing 45
III.4.2 Stream Processing 58
III.5 Performance Evaluation 67
III.5.1 Evaluation Metrics 67
III.5.2 Evaluation on Sentiment140 Dataset 68
III.5.3 Evaluation on Twitter 3M Dataset 69
III.5.4 Comparative Analysis 71
III.5.5 Summary 73
III.5.6 Comparison with State-of-the-Art Models 73
III.6 Conclusion 74

General Conclusion 76

LIST OF FIGURES

I.1 The Five V's Characteristics of Big Data 19
I.2 Hadoop Ecosystem 22

III.1 Use Case Diagram of the sentiment analysis system. 35
III.2 Class Diagram of the main logical components of the sentiment analysis pipeline. 35
III.3 Architecture diagrams 36
III.4 UML Sequence Diagram of the complete sentiment analysis pipeline. 37
III.5 Seven-stage workflow roadmap of the sentiment analysis system. 39
III.6 hadoop logo 40
III.7 spark logo 40
III.8 kafka logo 40
III.9 python logo 41
III.10java logo 42
III.11Hadoop HDFS pseudo-distributed cluster startup and daemon verification via jps (NameNode, DataNode, SecondaryNameNode, SparkSubmit) 43
III.12PySpark 3.5.8 session launch with –driver-memory 8g –executor-memory 8g in local[*] mode 43
III.13Live PySpark execution of the Sentiment140 preprocessing pipeline: HDFS data loading, text cleaning, and label remapping 48
III.14Spark MLlib pipeline architectures for the three main model configurations. . . 49
III.15Terminal output of Baseline Logistic Regression training and evaluation on Sentiment140: Accuracy = 0.7744, F1 = 0.7743, AUC = 0.8469 49
III.16Apache Kafka 4.2.0 KRaft mode initialization: UUID generation, storage formatting, and broker startup 59
III.17Kafka topic tweets creation and verification: 1 partition, replication factor 1, bootstrap server localhost:9092 60
III.18Kafka–Spark Structured Streaming pipeline for real-time sentiment analysis. . . 61
III.19Kafka producer output: raw Sentiment140 tweets being streamed to the tweets topic, showing unprocessed content including mentions, URLs, and hashtags . . 63
III.20spark-submit with Kafka connector: dependency resolution of 11 artifacts including spark-sql-kafka-0-10_2.12:3.5.0 64
III.21SparkContext 3.5.8 initialization for streaming: Java 11.0.29, BlockManager registered with 434.4 MiB RAM 65

LIST OF TABLES

III.1 Hardware and software specifications of the experimental environment 42
III.2 Software stack and version configuration 44
III.3 Summary of trained models and their performance metrics 52
III.4 Summary of streaming experiments 67
III.5 Model performance on Sentiment140 dataset (binary classification) 68
III.6 Confusion matrix — Baseline model on Sentiment140 test set 68
III.7 Model performance — balanced multiclass classification (Twitter 3M) 69
III.8 Model performance — unbalanced multiclass classification with class weights (Twitter 3M) 70
III.9 Model performance — binary classification on Twitter 3M dataset (Positive vs. Negative) 70
III.10 Consolidated performance comparison of all trained models 71
III.11 Comparison with state-of-the-art Transformer-based models on binary sentiment classification 74

Abstract

With the rapid expansion of social media usage on the Internet, these platforms have become a massive source of data reflecting users' behaviors, interactions, and interests in the digital world. This data is characterized by the main properties of big data, namely large volume, high velocity, variety, value, and veracity, which makes its analysis require specialized big data analytics techniques and tools.

This thesis aims to study the role of big data analytics in understanding and analyzing data extracted from social media platforms, with a particular focus on Twitter as a case study. The study is based on collecting and processing Twitter data, which are then analyzed using big data analytics approaches, with a focus on content-oriented methods such as sentiment analysis and text classification, applied at scale using distributed computing frameworks.

The results of the study show that big data analytics techniques enable the extraction of valuable information from Twitter data, such as identifying trending topics, analyzing user interaction patterns, and detecting different digital behaviors. The findings also highlight the importance of combining content-based analysis with network-based analysis to achieve a deeper understanding of interaction dynamics within the platform.

This thesis concludes that big data analytics represents an effective tool for understanding social media networks, particularly Twitter , and contributes to supporting various fields such as social, economic, and marketing analysis, while emphasizing the need to address challenges related to data quality and privacy.

Keywords: Big Data Analytics; Social Media; Twitter ; Social Network Analysis; Content Analysis; Network Analysis; User Behavior

Résumé

Avec l'expansion rapide de l'utilisation des réseaux sociaux sur Internet, ces plateformes sont devenues une source massive de données reflétant les comportements, les interactions et les intérêts des utilisateurs dans le monde numérique. Ces données se caractérisent par les principales propriétés du Big Data, à savoir un grand volume, une grande vélocité, une grande variété, une forte valeur et une véracité variable, ce qui rend leur analyse dépendante de techniques et d'outils spécialisés en analyse de données massives.

Cette mémoire vise à étudier le rôle de l'analyse des données massives dans la compréhension et l'analyse des données extraites des réseaux sociaux, en mettant un accent particulier sur Twitter comme étude de cas. L'étude repose sur la collecte et le traitement de données issues de Twitter, qui sont ensuite analysées à l'aide d'approches d'analyse Big Data, avec un accent sur les méthodes orientées contenu, telles que l'analyse des sentiments et la classification de textes, appliquées à grande échelle à l'aide de frameworks de calcul distribué.

Les résultats de l'étude montrent que les techniques d'analyse Big Data permettent d'extraire des informations pertinentes à partir des données de Twitter, telles que l'identification des sujets tendances, l'analyse des schémas d'interaction des utilisateurs et la détection de différents comportements numériques. Les résultats soulignent également l'importance de combiner l'analyse basée sur le contenu avec l'analyse basée sur le réseau afin d'obtenir une compréhension plus approfondie de la dynamique des interactions au sein de la plateforme.

Cette mémoire conclut que l'analyse des données massives constitue un outil efficace pour comprendre les réseaux sociaux, en particulier Twitter, et contribue à soutenir divers domaines tels que l'analyse sociale, économique et marketing, tout en mettant l'accent sur la nécessité de prendre en compte les défis liés à la qualité des données et à la protection de la vie privée.

Mots-clés : Analyse des données massives ; Réseaux sociaux ; Twitter ; Analyse des réseaux sociaux ; Analyse de contenu ; Analyse de réseau ; Comportement des utilisateurs

الملخص

مع التوسع السريع لاستخدام وسائل التواصل الاجتماعي على الإنترنت، أصبحت هذه المنصات مصدراً ضخماً للبيانات التي تعكس سلوكيات المستخدمين وتفاعلاتهم واهتماماتهم في العالم الرقمي. تتميز هذه البيانات بالخصائص الرئيسية للبيانات الضخمة، وهي: الحجم الكبير، والسرعة العالية، والتنوع، والقيمة، والمصداقية، مما يجعل تحليلها يستلزم تقنيات وأدوات متخصصة في تحليل البيانات الضخمة.

تهدف هذه المذكرة إلى دراسة دور تحليلات البيانات الضخمة في فهم وتحليل البيانات المستخرجة من منصات التواصل الاجتماعي، مع التركيز بشكل خاص على منصة تويتر كدراسة حالة. يرتكز البحث على جمع بيانات تويتر ومعالجتها، ثم تحليلها باستخدام مناهج تحليل البيانات الضخمة، مع التركيز على الأساليب الموجهة نحو المحتوى كتحليل المشاعر وتصنيف النصوص، وذلك على نطاق واسع باستخدام أطر الحوسبة الموزعة.

تُظهر نتائج الدراسة أن تقنيات تحليل البيانات الضخمة تُمكّن من استخراج معلومات قيّمة من بيانات تويتر، كتحديد الموضوعات الرائجة، وتحليل أنماط تفاعل المستخدمين، واكتشاف السلوكيات الرقمية المختلفة. كما تبرز النتائج أهمية التحليل القائم على المحتوى للوصول إلى فهم أعمق لديناميكيات التفاعل داخل المنصة.

تخلص هذه المذكرة إلى أن تحليل البيانات الضخمة يمثل أداةً فعّالة لفهم منصات التواصل الاجتماعي، ولا سيما تويتر، ويُسهم في دعم مجالات متعددة كالتحليل الاجتماعي والاقتصادي والتسويقي، مع التأكيد على ضرورة معالجة التحديات المتعلقة بجودة البيانات والخصوصية.

الكلمات المفتاحية: تحليل البيانات الضخمة؛ وسائل التواصل الاجتماعي؛ تويتر؛ تحليل المشاعر؛ الحوسبة الموزعة؛ تحليل المحتوى؛ سلوك المستخدم.

GENERAL INTRODUCTION

Note on Terminology: Throughout this thesis, the platform originally known as Twitter is referred to simply as "Twitter". As of 2023, the platform was officially rebranded as "X" by its owner. This change occurred after the datasets used in this work were collected. All references to "Twitter" therefore refer to the same platform, now known as X.

In the last decade, the digital world has witnessed a rapid evolution in information and communication technologies, where social media platforms have emerged as an unprecedented global phenomenon. These platforms have evolved from simple communication tools into massive digital environments for generating and exchanging data. Platforms such as Twitter , Facebook, Instagram, and YouTube have become primary sources of big data that reflect users' behaviors, interactions, and interests in the digital space.

This data is characterized by the well-known properties of big data (large volume, high velocity, variety, value, and veracity), which makes its analysis dependent on advanced big data analytics tools and techniques. In this context, social media platforms collectively stand out as ideal environments for Big Data analytics due to the richness and diversity of the data they generate. Among these platforms, one stands out as a particularly compelling case study: Twitter , owing to its concise textual nature, the use of hashtags for topic tracking, the ability to measure information diffusion through retweets, and the richness of metadata associated with each tweet. Accordingly, this thesis adopts Twitter as its central case study while situating its findings within the broader landscape of social media analytics.

Review of Previous Studies

The growing interest in employing Big Data analytics to understand interaction dynamics within social media networks has given rise to a rich body of prior work. The following studies illustrate the breadth and relevance of this research field:

- Dealing with Big Data: The Case of Twitter (2013)

  This study presented a Hadoop-based infrastructure to process billions of Dutch tweets under legal constraints, demonstrating the usability of Twitter data for event detection and linguistic and marketing analysis [1].

- Social media and the social sciences: How researchers employ Big Data analytics (2016)

  The study showed how social scientists use big data analytics to gain network-level insights while recommending the combination of quantitative analytics with qualitative methods[2].

- Can Big Data provide good quality statistics? A case study on sentiment analysis on Twitter data (2018)

  The study examined the quality of statistics derived from Twitter sentiment analysis, highlighting issues such as bots, retweets, and coverage errors, and recommending mixed analytical approaches [3].

- Social media (Twitter) Data analysis using maximum entropy classifier on big data processing framework (Case study: Analysis of health condition, education status, states of business) (2018)

  The thesis proposed a distributed Hadoop-based framework using MaxEnt for tweet sentiment classification and achieved high accuracy (90.91%) [4].

- Big Data analysis using a metaheuristic algorithm: Twitter as Case Study (2020)

  The study proposed a hybrid PSOK algorithm combining K-means and PSO for feature optimization in sentiment analysis, achieving superior accuracy using Apache Spark for real-time tweet processing[5].

- Big data analytics meets social media: A systematic review of techniques, open issues, and future directions (2021)

  This systematic review of 74 studies classified social media analytics approaches into content-oriented and network-oriented methods and identified research gaps related to privacy, security, and sentiment analysis challenges [6].

- Analysis and Application of Big Data in Social Media Marketing (2024)

  The study focused on employing big data analytics to improve the effectiveness of social media marketing through accurate user profiling, behavior analysis, and campaign evaluation, while emphasizing data quality and privacy challenges [7].

Objectives of the Study

This study aims to:

- Understand the role of Big Data analytics in analyzing data generated by social media platforms in general.
- Study Twitter as a representative and practical case study for large-scale textual data analysis and sentiment classification, with findings intended to generalize to other platforms.
- Apply distributed computing frameworks to process and analyze Twitter data efficiently.
- Extract valuable information such as sentiment trends, behavioral patterns, and interaction dynamics.
- Highlight challenges related to data quality, class imbalance, and privacy.

Reasons for Choosing the Topic

Objective Reasons:

- The unique data nature of Twitter (concise text, hashtags, retweets, metadata).

- The need for scalable analytical models capable of handling large volumes of real-time data.
- Existing research gaps in applying distributed Big Data frameworks to social media sentiment analysis.
- The possibility of generalizing the methodology to other platforms.

Subjective Reasons:

- Personal interest in big data analytics and digital social phenomena.
- Interdisciplinary integration between computer science, statistics, and natural language processing.
- Increasing professional demand for data analytics skills.

Research Problem

The main research problem can be formulated as follows:

How can Big Data analytics techniques be employed to understand and analyze sentiment patterns and digital behavior on social media platforms at scale, using distributed computing frameworks and machine learning methods — with Twitter serving as the primary case study?

Research Questions

- What type of data does Twitter provide and how can it be exploited analytically?
- How does content-based sentiment analysis contribute to understanding digital topics and user behaviors?
- How can distributed frameworks such as Apache Spark handle large-scale Twitter data efficiently?
- What is the impact of preprocessing quality and class imbalance handling on classification performance?
- What challenges are related to data quality, scalability, and privacy?

Research Hypotheses

- Big Data analytics techniques can extract valuable sentiment information from large-scale Twitter data.
- Distributed computing frameworks such as Apache Spark enable efficient processing of millions of tweets within resource-constrained environments.
- The quality of analytical results depends on effective data preprocessing and appropriate handling of class imbalance.
- The adopted methodology can be generalized to other social media platforms.

Adopted Methodology

The study follows a descriptive, analytical, and applied approach through:

- Collecting large-scale Twitter datasets.
- Data preprocessing and cleaning using distributed PySpark pipelines.
- Applying content-based analytical methods, including sentiment analysis and text classification.
- Training and evaluating multiple machine learning models using Spark MLlib.
- Implementing real-time stream processing using Apache Kafka and Spark Structured Streaming.
- Interpreting results in light of the theoretical framework and previous studies.

Scope of the Study

This study focuses on:

- Social media analytics in general, with Twitter data as the primary case study due to its accessibility, volume, and structural richness.
- Textual content and interactions without multimedia analysis.
- A specific time period for data collection.
- Analytical aspects without deep legal discussions.

Structure of the Thesis

This thesis is organized as follows:

- Chapter One: Big Data
- Chapter Two: Twitter Big Data Analytics
- Chapter Three: Design and Implementation

The thesis ends with a general conclusion, including recommendations and prospects for future research.

CHAPTER I

BIG DATA

I.1 Introduction

In recent years, the concept of Big Data has become central to modern science, industry, and decision-making processes. The unprecedented growth of digital technologies, internet usage, and connected devices has led to the continuous generation of massive amounts of data from diverse sources such as online transactions, emails, sensors, mobile devices, multimedia content, and especially social media interactions. This rapid data growth has exceeded the capabilities of traditional database systems in terms of storage, management, processing, and analysis.

Reports indicate that humanity generated only a few exabytes of data before 2003, while today the same amount can be produced within days [8]. The expansion of the digital world into zettabytes of data, along with billions of mobile subscriptions, social media posts, and connected devices, highlights the scale and complexity of modern data environments. Such an explosion of data has made it necessary to adopt new technologies and analytical approaches capable of extracting meaningful information and knowledge from these vast datasets.

Big Data is not only about the size of data but also about the challenges and opportunities associated with its acquisition, storage, processing, analysis, and interpretation. The data analysis pipeline involves multiple stages including data acquisition, filtering, metadata generation, information extraction, storage design, mining, and visualization. At each stage, challenges arise due to heterogeneity, incompleteness, scale, timeliness, privacy concerns, and system complexity. Therefore, Big Data analytics requires integrated environments that combine database systems, distributed computing frameworks, and advanced analytical algorithms.

Initially, Big Data technologies were embraced by online-native companies such as Google, Facebook, LinkedIn, and eBay, which built their infrastructures around data-driven architectures from the beginning. However, large traditional organizations later faced the challenge of integrating Big Data technologies with existing IT infrastructures and analytical practices. This integration led to the emergence of new perspectives on analytics where Big Data coexists with conventional data sources and methods.

Within this context, understanding the concept, characteristics, and tools of Big Data analytics becomes essential before applying these techniques to specific domains such as social media analytics. This chapter provides the theoretical foundation required for the rest of the thesis by introducing the concept of Big Data, its main characteristics, and the key tools and technologies used in Big Data analytics.

I.2 Definition of Big Data

I.2.1 Review of Existing Definitions of Big Data

The concept of Big Data has been widely discussed in the literature, yet no single, universally accepted definition has emerged. This lack of consensus has led several researchers to analyze previous definitions, classify them into groups, and propose more comprehensive definitions based on common elements.

Two important studies in this context are:

- Defining Big Data (2016)
- A Formal Definition of Big Data Based on its Essential Features (2016)

Each study reviewed existing definitions, classified them according to specific perspectives, and then proposed its own definition.

Classification of Definitions in the First Study

The first study categorized definitions into three groups:

1. Comparative Definitions: Definitions comparing Big Data with traditional relational systems.
2. Attribute-Based Definitions: Definitions based on the famous 3Vs and their extensions.
3. Environmental Definitions: Definitions that consider Big Data within its technical and architectural environment.

Based on this analysis, the study proposed the following definition:

> "The term Big Data describes a data environment in which scalable architectures support the requirements of analytical and other applications which process, with high velocity, high volume data which may have a variety of data formats and which may include high velocity data acquisition." [9]

However, this definition presents certain limitations: it emphasizes the environment, architecture, and processing context of Big Data, yet pays less attention to the notion of value extraction and to the conceptual nature of data as an information asset.

Classification of Definitions in the Second Study

The second study classified existing definitions into four groups according to the focus of each definition:

1. Attributes of Data: Definitions focusing on the characteristics of data such as Volume, Velocity, Variety, Veracity, and Value.
2. Technological Needs: Definitions emphasizing the need for powerful technologies, scalable architectures, and advanced analytical tools.
3. Overcoming Thresholds: Definitions describing Big Data as data that exceeds the capacity of traditional database systems.
4. Social Impact: Definitions highlighting the cultural, social, and organizational impact of Big Data on society and decision-making.

After analyzing these groups, the authors proposed the following definition:

> "Big Data is the Information asset characterized by such a High Volume, Velocity and Variety to require specific Technology and Analytical Methods for its transformation into Value."[10]

However, this definition presents certain limitations: while it successfully combines data characteristics, technological requirements, and value extraction, it focuses mainly on the data itself as an "information asset" and pays less attention to the broader data environment and architectural context in which Big Data operates.

I.2.2 Towards an Approximate Understanding of Big Data

In light of the previous definitions and classifications, we observe that most definitions converge around a set of common elements that characterize Big Data:

- Very large volume of data.
- High velocity of data generation and processing.
- Great variety of data formats (structured, semi-structured, unstructured).
- The need for specific technologies and analytical methods.
- The inability of traditional systems to efficiently handle such data.
- The goal of extracting value and meaningful insights from data.
- The existence of a technical and architectural environment supporting data storage, processing, and analysis.

Therefore, in a simplified and approximate sense, Big Data can be understood as:

> Big Data refers to extremely large, fast, and diverse datasets that cannot be efficiently handled by traditional systems and that require specialized technologies, scalable architectures, and advanced analytical methods in order to extract valuable insights.

I.3 Characteristics of Big Data

Before detailing the characteristics, it is useful to indicate how many characteristics were identified by each source:

- Source 1 – BIG DATA CHARACTERISTICS(2014): identified 6 characteristics.
- Source 2 – A Study of Big Data Characteristics(2016): identified 18 characteristics.
- Source 3 – What makes Big Data, Big Data?(2016): identified 10 characteristics.

From these sources, we can distinguish between core (fundamental) characteristics that appear repeatedly, and complementary characteristics that enrich the understanding of Big Data in practical, managerial, and analytical contexts.

I.3.1 Core Characteristics of Big Data

- Volume Refers to the massive size of data generated and stored, often measured in terabytes or petabytes, which imposes serious storage and processing challenges. [11]
- Velocity Describes the speed at which data is generated, transmitted, processed, and sometimes required to be analyzed in real or near real time. [12]
- Variety Indicates the diversity of data formats such as structured, semi-structured, and unstructured data including text, images, video, and sensor data. [13]
- Veracity / Quality Relates to the accuracy, completeness, and reliability of data, since poor-quality data leads to misleading analysis results. [11]
- Value Represents the potential of data to generate useful insights and actionable knowledge for organizations and decision makers. [13]

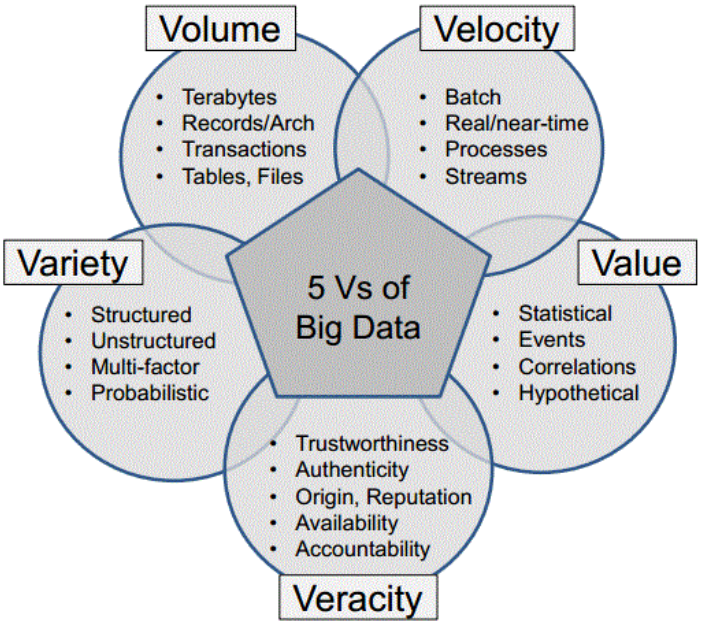


Figure I.1: The Five V's Characteristics of Big Data

I.3.2 Complementary Characteristics of Big Data

- Exhaustivity Big Data systems aim to capture the entire population of data rather than a sample, meaning all events, records, or interactions are recorded. [12]
- Sensitivity Indicates whether data contains sensitive or confidential information that requires legal protection and secure handling. [11]
- Structure Refers to the degree of organization of data into structured or unstructured forms, affecting how easily it can be processed by machines. [11]
- Variability Describes the inconsistency and fluctuation in data flow rates and the presence of noisy or irregular data patterns. [13]
- Volatility Refers to how long data remains useful and the policies governing its retention and deletion. [13]
- Visualization Concerns the ability to represent complex data in graphical or visual forms to support understanding and decision-making. [13]
- Virality Represents how quickly data spreads across users and platforms, especially in social media contexts. [13]
- Viscosity Refers to the delay between an event occurring and the data describing that event becoming available. [13]
- Complexity Indicates the difficulty of linking, correlating, and managing data coming from multiple heterogeneous sources. [13]
- Scalability and Flexibility Describes the ability of systems to scale resources up or down and adapt to varying data loads and evolving data structures. [12]

- Relationality and Indexicality Data is accompanied by metadata (time, location, device) allowing linkage between datasets and identification of data origin.[12]
- Verbosity Refers to redundancy of information collected from multiple sources which may require filtering and cleaning. [13]
- Voluntariness Big Data can be used voluntarily by different organizations for various analytical and decision-making purposes. [13]
- Versatility Denotes the adaptability of Big Data to be used in different contexts and application domains. [13]

I.4 Concept and Tools of Big Data Analytics

The growing volume of data generated every second from digital platforms, especially social media, has made data analysis an essential necessity for understanding behaviors, trends, and interactions in real time. In this context, Big Data analytics emerges as a powerful approach capable of extracting meaningful insights from massive and complex datasets that traditional tools can no longer handle.

I.4.1 Concept of Big Data Analytics

Big Data analytics refers to the process of systematically examining large, diverse, and complex datasets in order to discover hidden patterns, correlations, trends, and useful information that support data-driven decision making [14].

Unlike traditional data analytics, which is usually performed after data is stored and over limited structured datasets, Big Data analytics operates on structured, semi-structured, and unstructured data generated from multiple sources such as social media platforms, sensors, mobile devices, logs, and online transactions, often in real or near real time [15].

From a technological perspective, Big Data analytics relies on modern computational frameworks and distributed processing systems capable of storing and processing data that exceed the capacity of conventional database systems. This enables new analytical approaches that were previously impractical using traditional methods [16].

Big Data analytics aims to transform raw data such as tweets, blog posts, search queries, images, and interaction logs into valuable knowledge through advanced computational techniques, statistical models, and machine learning algorithms to reveal patterns and behavioral trends [16].

I.4.2 Tools of Big Data Analytics

Big Data analytics relies on a rich ecosystem of tools designed to handle data collection, storage, processing, analysis, and visualization across distributed environments. These tools work together to enable scalable and efficient management of massive and heterogeneous datasets.

Apache Hadoop

An open-source framework that provides distributed storage through HDFS and distributed processing using the MapReduce model. It allows large datasets to be stored and processed across clusters of computers reliably.

Apache Spark

A fast, in-memory data processing engine that supports batch processing, real-time streaming, SQL queries, and machine learning. Spark is widely used due to its speed and versatility compared to traditional MapReduce.

Apache Kafka
A distributed event streaming platform used for real-time data ingestion and pipeline building. It enables continuous data flow from multiple sources into analytics systems.

Apache Flink
A real-time stream processing engine designed for high-throughput and low-latency data processing. It is suitable for applications requiring immediate analytics results.

Apache Hive
A data warehouse system built on top of Hadoop that allows querying large datasets using SQL-like language (HiveQL). It simplifies data analysis for users familiar with SQL.

HBase An open-source, distributed NoSQL database modeled after Google Bigtable, running on top of HDFS. It provides real-time read/write access to large datasets and is optimized for random, low-latency queries.

Cassandra A highly scalable, masterless NoSQL database designed for high availability and fault tolerance. It excels at handling massive write workloads across distributed clusters with linear scalability and no single point of failure.

MongoDB
A document-oriented NoSQL database that stores data in flexible JSON-like documents. It is useful for handling unstructured and semi-structured data common in social media analytics.

Elasticsearch
A distributed search and analytics engine used for fast indexing, searching, and analyzing large volumes of text data, logs, and social media content.

Apache NiFi
A data flow automation tool that manages the movement of data between systems. It helps in collecting, routing, and transforming data from multiple sources.

Spark MLlib
A scalable machine learning library built on Apache Spark, providing distributed implementations of common algorithms for classification, regression, clustering, and collaborative filtering. It is designed to handle large-scale data processing across clusters with high speed and fault tolerance.

Scikit-learn
A popular Python library for machine learning that offers simple and efficient tools for data mining and data analysis. It provides implementations of classification, regression, clustering, and dimensionality reduction algorithms, though it is primarily suited for single-node computation.

Jupyter Notebooks
An open-source web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It is widely used for exploratory data analysis, prototyping, and reproducible research in Python and other programming languages.

RStudio
An integrated development environment (IDE) for the R programming language, dedicated to statistical computing and graphics. It provides tools for data manipulation, visualization, modeling, and reporting, making it a standard platform for data scientists and statisticians.

Tableau
A leading business intelligence and data visualization platform that enables users to create interactive dashboards, charts, and reports without requiring programming skills. It connects to various data sources and supports real-time data exploration for informed decision-making.

Power BI
A Microsoft business analytics service that provides interactive visualizations and business intelligence capabilities with an intuitive interface for end-users. It allows users to connect to hundreds of data sources, create reports and dashboards, and share insights across organiza-

tions.

Apache Airflow

A workflow management platform used to schedule, monitor, and automate data processing pipelines in Big Data projects.

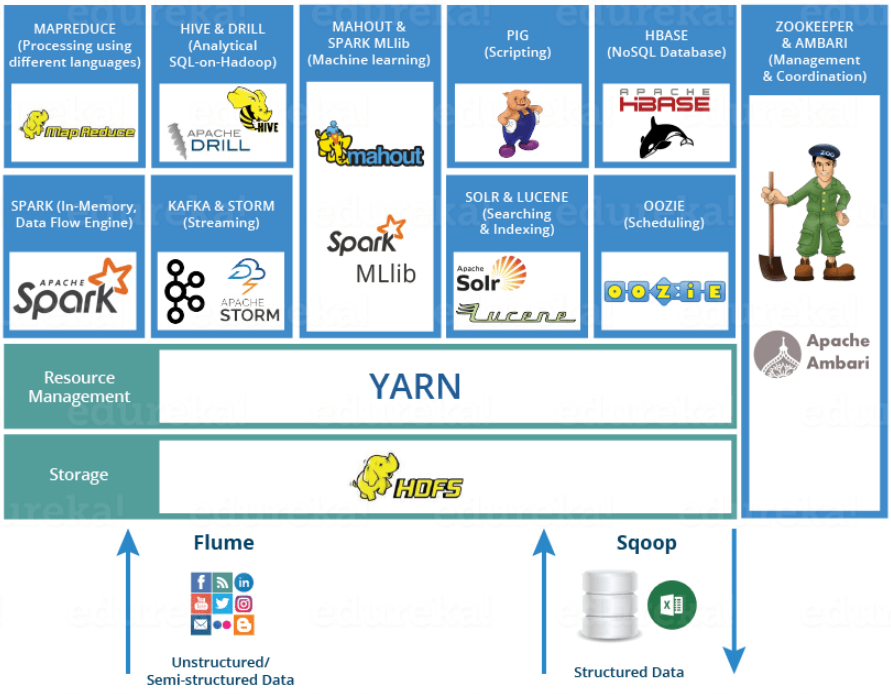


Figure I.2: Hadoop Ecosystem

I.5 Conclusion

This chapter has provided the theoretical foundation necessary for understanding Big Data and its analytical environment before applying these concepts to social media data in the following chapters. It began by introducing the notion of Big Data and reviewing several definitions proposed in the literature, highlighting the common elements that characterize Big Data as a data environment requiring specialized technologies and analytical methods to extract value.

The chapter then examined the main characteristics of Big Data, distinguishing between core characteristics such as Volume, Velocity, Variety, Veracity, and Value, and complementary characteristics that further describe the complexity, scalability, and practical implications of Big Data systems. These characteristics demonstrate why traditional data management tools are no longer sufficient to handle modern data environments.

Finally, the concept of Big Data analytics and its main tools were presented. The discussion showed how distributed frameworks, NoSQL databases, data processing engines, machine learning libraries, and visualization tools work together to enable the extraction of meaningful insights from massive and heterogeneous datasets.

Overall, this chapter establishes a clear understanding of Big Data as both a data phenomenon and a technological ecosystem. This understanding forms the basis for the next chapter, which focuses specifically on applying Big Data analytics to Twitter data as a case study.

CHAPTER II

TWITTER BIG DATA ANALYTICS

II.1 Introduction

The rapid growth of social media platforms has reshaped how people, companies, and institutions communicate in the digital age. Interaction, information sharing, and public discussion now happen largely online. Platforms like Facebook, Instagram, LinkedIn, and YouTube have become part of everyday routines. They generate vast amounts of user-created content every minute.

Posts, images, videos, comments, and connections form continuous streams of textual, visual, and relational data. This data reflects opinions, behaviors, preferences, and patterns of interaction. Social media now stands as one of the most significant sources of Big Data, defined by its scale, speed, and diversity.

Among these platforms, Twitter holds a distinct role because of its real-time and public communication model. Since its launch in 2006, Twitter has grown into a global microblogging network where millions of users publish short messages, known as tweets. People use it to share news, express opinions, comment on events, and engage in ongoing conversations.

Hashtags, mentions, and retweets drive information across the platform at high speed. Topics trend within minutes. Communities form around shared interests. Public reactions unfold in real time and leave behind measurable interaction data. This structure makes Twitter a strong source for analyzing trends, sentiment, public opinion, and patterns of social activity.

The availability of Twitter APIs increases its value for both academic research and industry applications. These interfaces provide structured access to large volumes of real-time and historical data. Researchers and analysts collect tweets, user profiles, and interaction metadata to perform advanced analysis using Big Data tools and machine learning methods.

Twitter data supports work in sentiment analysis, event detection, marketing analytics, disaster management, political communication, and social network analysis. Its open and streaming nature allows large-scale data collection and structured examination.

Given its size, continuous data flow, and structural complexity, Twitter serves as a strong case study for applying Big Data analytics techniques. Combining content-based approaches, such as text mining and sentiment analysis, with network-based methods like interaction and graph analysis creates a deeper understanding of digital behavior. Content shows what users say. Network analysis explains how information spreads.

This chapter focuses on Twitter Big Data Analytics. It begins with an overview of social networks and the defining features of Twitter . It then examines how Twitter APIs and tweet structures support data collection, followed by a discussion of analytical techniques used to extract meaningful insights. Through this framework, Twitter emerges as a central environment for studying large-scale social data using modern analytics methods.

II.2 Social Networks

Social networks have become central platforms for digital communication, enabling individuals, organizations, and institutions to exchange information in real time. These platforms generate massive volumes of user-generated content, including text, images, videos, and interaction metadata. Due to their scale, velocity, and heterogeneity, social networks constitute a major source of Big Data, reflecting public opinion, behavioral patterns, and social interactions. The increasing availability of social media data through Application Programming Interfaces (APIs) has encouraged researchers to adopt computational approaches to study societal phenomena through large-scale digital traces [17]. In this context, social networks provide a dynamic and data-rich environment for applying Big Data analytics techniques to understand communication structures and user behavior.

II.2.1 Twitter as a Microblogging Platform

Twitter was officially launched in 2006 by Jack Dorsey, Biz Stone, and Evan Williams as an online microblogging service designed to enable users to share short, real-time messages [18]. Since its creation, Twitter has evolved into one of the most influential social networking platforms worldwide, playing a significant role in digital communication and information dissemination.

As a microblogging platform, Twitter allows users to publish short posts known as tweets, which were initially limited to 140 characters [1], encouraging concise and immediate expression [18]. This structural constraint promotes rapid sharing of opinions, news updates, and reactions to ongoing events. Over time, the platform expanded its features while maintaining its core principle of short-form communication.

Twitter enables interaction through several mechanisms, including followers, retweets (RT), mentions (@), and hashtags (#) [18]. Unlike traditional social networking platforms that require mutual connections, Twitter operates on an asymmetric follower model, where users can follow others without reciprocal approval. This structure facilitates large-scale information diffusion and enhances the speed at which content spreads across the network [18].

Hashtags allow users to categorize content and aggregate discussions around specific topics, while retweets amplify the reach of messages beyond immediate follower networks [18]. These interaction mechanisms transform Twitter into a dynamic environment for public discourse, making it widely used in areas such as politics, marketing, crisis communication, and disaster response [18].

From a Big Data perspective, Twitter produces massive volumes of user-generated content on a daily basis, characterized by high volume, high velocity, and predominantly textual data streams [19]. These properties make Twitter an appropriate and frequently used case study for applying Big Data analytics techniques, including content analysis, sentiment analysis, trend detection, and social network analysis of user interactions.

II.2.2 Twitter APIs

Application Programming Interfaces (APIs) are standardized interfaces that allow one software component to access the data and functionalities of another through defined protocols. In the context of web services, APIs enable programmatic access to data via HTTP requests, allowing external applications to retrieve and manipulate structured information without interacting directly with the graphical web interface [20, 21].

From a software engineering perspective, APIs emerged as a design principle linked to modularity and interoperability. They separate the internal implementation of a system from its public interface, allowing changes in the private code without affecting external applications that rely on the API [20]. In web-based platforms, APIs function as controlled gateways that regulate access to data while maintaining system stability and scalability.

Twitter was launched in July 2006 as a microblogging service, and only two months after its launch, it made its API publicly available. This early decision to open the platform to third-party developers played a crucial role in its rapid growth and ecosystem expansion [20, 21]. The availability of the Twitter API enabled developers to build external applications, mobile clients, and analytical tools that extended the functionality of the platform beyond its original web interface.

Technically, the Twitter API follows RESTful principles and supports HTTP methods such as GET, POST, and DELETE. GET requests are typically used to retrieve data, while POST requests are required for operations that modify data, such as publishing or deleting tweets [21].

[1] It was officially doubled to 280 characters in November 2017

The API supports multiple data formats including XML, JSON, RSS, and Atom, facilitating integration with different programming environments and applications [21].

Historically, Twitter has provided two main types of APIs: the REST API and the Streaming API. The REST API enables developers to access core functionalities such as retrieving user timelines, posting tweets, accessing follower information, and performing keyword searches. The Streaming API, on the other hand, allows near real-time access to a continuous stream of tweets filtered by keywords, users, or geographic parameters [20, 22].

Each tweet retrieved through the API contains not only the textual content but also rich metadata such as creation time, user profile information, number of followers, retweets, favorites, and sometimes geographic coordinates. This structured metadata makes the Twitter API particularly valuable for research in areas such as credibility analysis, event detection, sentiment analysis, and social network modeling [22].

To ensure fair usage and system stability, Twitter imposes rate limits on API requests. These limits restrict the number of requests that can be made within a given time window, and different policies apply to authenticated and unauthenticated requests [21]. Authentication mechanisms, including OAuth, are used to securely authorize applications to access user data without exposing credentials [21].

Beyond simple data retrieval, the Twitter API provides access to several operational groups of methods, including publishing tweets, retrieving information streams, managing the follow network, exchanging direct messages, administering accounts, and performing keyword searches [21]. These structured methods make it possible to construct comprehensive data collection and analysis pipelines.

From a Big Data perspective, the Twitter APIs represent the primary gateway for acquiring high-volume and high-velocity social media data. By enabling structured, automated, and scalable data collection, they form the foundational layer for Twitter Big Data Analytics systems, supporting both content-oriented and network-oriented analytical approaches.

II.2.3 Structure and Architecture of a Tweet

On Twitter , the tweet is the core unit of communication. At first, tweets were restricted to 140 characters [2], a limit inherited from the SMS protocol and its 140-byte capacity [21]. In reality, this restriction is measured in bytes rather than characters, since Twitter relies on UTF-8 encoding; Unicode symbols can therefore reduce the effective count [21]. Direct messages, however, allowed up to 255 characters [21].

This character cap shaped the way users expressed themselves: tweets became short, immediate, and highly condensed [21]. They are designed for quick sharing rather than lengthy elaboration.

Tweets as Form and Content

While defined by their length, tweets are flexible in meaning and purpose. As Tracing Twitter : the rise of a microblogging platform notes, they can be conversational, expressive, informative, or promotional [23]. Early discussions questioned whether Twitter was mainly a space for everyday social talk or a broader channel for news and information [23].

Pear Analytics offered a practical classification: tweets fall into six groups—news, spam, self-promotion, pointless chatter, conversational exchanges, and retweets. This mix shows how Twitter blends casual social interaction with information sharing [23]. In other words, it functions both as a social network and as a medium for distributing content.

[2]It was officially doubled to 280 characters in November 2017

Status Object: Technical Representation of a Tweet

From the API perspective, each tweet is represented as a Status Object. Stream-related methods return either a single status or a collection grouped under the statuses element [21].

A status object contains the tweet text plus metadata describing its context. Key fields include:

- text: the tweet's visible content.
- created_at: timestamp in GMT.
- id: unique identifier for the tweet.
- source: the application used to post it.
- truncated: shows if the tweet exceeds display limits.
- in_reply_to_status_id: links to another tweet.
- in_reply_to_user_id and in_reply_to_screen_name: identify the user being replied to.
- favorited: indicates whether the tweet was bookmarked.
- user: a nested object with details about the author.

The nested user element makes it possible to access author information directly, without an extra API call [21].

In short, a tweet is both a brief textual message and a structured data object enriched with metadata about time, relationships, and authorship. This dual role—social and technical—explains why tweets are the central unit in Twitter research.

II.3 Twitter Big Data Analytics

The massive growth of data generated by Twitter has positioned the platform as one of the most significant real-world laboratories for Big Data analytics. Millions of tweets are produced daily, forming continuous streams of textual, relational, and temporal data. From a Big Data perspective, Twitter data is characterized by high volume, high velocity, and high variety, requiring scalable architectures, efficient data management strategies, and advanced analytical techniques [24].

Twitter Big Data Analytics refers to the collection, management, processing, and analysis of large-scale Twitter datasets in order to extract meaningful patterns, trends, sentiments, and structural properties of interactions. This process generally involves three major components: data collection, data management and preprocessing, and analytical modeling [24].

II.3.1 Data Collection at Scale

Effective Twitter analytics begins with systematic data collection. Researchers have several API options provided by Twitter , including the Search API, Streaming API, and REST API, each offering different capabilities and constraints [24].

The Search API allows querying recent tweets based on keywords and returns a collection of relevant tweets. The Streaming API enables real-time data capture, filtering tweets by keywords, hashtags, user IDs, or geographic regions. The REST API provides access to specific

user timelines and metadata. However, these APIs impose rate limits and often provide only partial coverage of the full Twitter stream. For instance, the Streaming API typically delivers a 1% sample of the public stream, commonly referred to as the "Spritzer" stream [24].

Due to such limitations, large-scale analytical tasks often require distributed crawling strategies and cloud-based infrastructures to ensure continuous data acquisition over extended periods. Distributed systems have been proposed to enable scalable crawling and improved coverage of communities of interest [24]. The collected data is typically returned in JSON format, which facilitates structured storage and downstream processing.

II.3.2 Data Preprocessing and Management

Raw Twitter data is inherently noisy and requires extensive preprocessing before meaningful analysis can be performed. Tweets frequently contain abbreviations, emoticons, hyperlinks, hashtags, and informal language structures. As emphasized in text analytics research, preprocessing plays a crucial role in transforming unstructured textual data into analyzable formats [25].

Common preprocessing steps include:

- Lowercasing text to ensure uniform representation.
- Removal of URLs, HTML tags, and emoticons.
- Removal of punctuation marks.
- Stop-word removal.
- Removal of extremely frequent or rare terms.
- Stemming and lemmatization to reduce words to their base forms.

These steps enhance the quality of feature extraction and improve the performance of machine learning algorithms [26]. Frameworks such as TrendMiner and TwitIE integrate preprocessing pipelines that include tokenization, part-of-speech tagging, language identification, named entity recognition, and sentiment extraction [24]. The development of modular preprocessing pipelines allows researchers to reuse and extend components for different analytical tasks.

II.3.3 Content-Based Analytics

Content-based analytics focuses on extracting insights directly from tweet text. Two primary approaches dominate this domain: topic modeling and sentiment analysis.

Topic Modeling

Topic modeling aims to automatically discover latent thematic structures in large text corpora. Latent Dirichlet Allocation (LDA) is one of the most widely used probabilistic models for this purpose [26, 19]. In LDA, each tweet is treated as a document composed of a mixture of topics, and each topic is represented as a probability distribution over words.

The LDA workflow typically involves:

1. Text preprocessing (lowercasing, tokenization, stop-word removal, stemming).
2. Vectorization of documents.

3. Iterative probabilistic assignment of words to topics.

4. Estimation of topic-word distributions.

Through this approach, researchers can identify dominant themes such as politics, sports, or health-related discussions, enabling high-level understanding of user discourse [19].

Sentiment Analysis

Sentiment analysis seeks to determine the emotional polarity of tweets. It associates textual content with positive, negative, or neutral sentiment scores [19]. Sentiment can be calculated using lexicon-based methods or machine learning classifiers.

Lexicon-based approaches compare tweet words against predefined dictionaries containing positive and negative scores. However, contextual sensitivity is crucial, as word polarity may vary across domains [19]. Alternatively, supervised classification techniques such as Naïve Bayes or Support Vector Machines (SVM) can be used to automatically label tweets [26].

Sentiment analytics has practical applications in brand monitoring, customer support evaluation, and public opinion analysis. Aggregated sentiment distributions provide insights into customer satisfaction levels and brand perception trends [25].

II.3.4 Similarity Measures and Trend Detection

Trend detection is a central objective in Twitter Big Data Analytics. Several computational techniques are used to identify trending topics and cluster related content.

Hashtag counting represents a basic method for identifying popular themes by measuring frequency of occurrence [26]. Noun counting provides an alternative linguistic-based trend detection strategy.

More advanced techniques include cosine similarity and Jaccard similarity, which measure textual similarity between tweets or between tweets and predefined clusters. Cosine similarity computes the cosine of the angle between two document vectors in multidimensional space, allowing similarity measurement independent of document length [26]. Jaccard similarity measures overlap between sets of terms, supporting classification of tweets into thematic clusters.

Clustering algorithms such as K-means further group tweets or users into coherent categories based on similarity metrics. These clusters reveal behavioral patterns and interest groups among Twitter users [26].

II.3.5 Network-Based Analytics

Beyond textual content, Twitter data can be analyzed through network structures. A network consists of nodes (users or hashtags) and edges (relationships such as retweets or co-occurrences) [19].

A retweet network models information diffusion, where directed edges represent the flow of content from one user to another. Centrality measures are used to identify influential users:

- Degree Centrality (In-degree and Out-degree).
- In-Degree Centrality: number of times a user is retweeted.
- Out-Degree Centrality: number of retweets a user performs.

These measures help identify key actors and opinion leaders within the network [19].

Hashtag co-occurrence networks represent relationships between topics, where edges connect hashtags that appear together within tweets. Weighted edges reflect frequency of co-occurrence, enabling exploration of thematic structures [19].

Network-based analytics complements content-based methods by revealing structural properties of information diffusion and community formation.

While network-based analytics represents a powerful complementary approach, the present work focuses primarily on content-based sentiment analysis, leaving network-level analysis as a direction for future research.

II.3.6 Real-Time Analytics and Streaming Architectures

Given the high velocity of Twitter data, real-time analytics has become increasingly important. Frameworks such as Apache Spark enable structured streaming of tweets, allowing continuous processing of incoming data [26].

In streaming architectures, tweets are written to a TCP socket and processed in micro-batches. Spark processes each batch immediately, updating analytical results dynamically. This approach eliminates delays associated with static data collection and enhances the responsiveness of trend detection systems [26].

Real-time analytics is particularly valuable in crisis detection, disaster monitoring, and event tracking scenarios, where timely insights are critical.

II.3.7 Toward Integrated Analytics Frameworks

Existing systems often focus on isolated components such as data collection, preprocessing, or visualization. However, there is increasing recognition of the need for unified frameworks that integrate crawling, preprocessing, information extraction, querying, and visualization into cohesive platforms [24].

An integrated framework enables analysts to focus on the research objectives rather than infrastructural complexities. Such systems must address scalability, data quality, rate limitations, and real-time processing constraints.

In summary, Twitter Big Data Analytics encompasses a multidisciplinary pipeline combining distributed data acquisition, advanced text mining, similarity measurement, clustering, network modeling, and streaming analytics. The integration of these components allows researchers and practitioners to extract valuable insights from massive Twitter datasets, supporting applications in marketing, public opinion analysis, crisis management, and social behavior modeling.

II.3.8 Ethical and Governance Considerations in Twitter Big Data Analytics

As Big Data analytics on social media platforms becomes increasingly prevalent, it raises critical ethical, legal, and governance challenges that researchers and practitioners must address. This section briefly discusses four key dimensions relevant to Twitter analytics.

General Data Protection Regulation (GDPR)

The General Data Protection Regulation (GDPR), enforced since May 2018 across the European Union, represents one of the most comprehensive legal frameworks governing the collection, storage, and processing of personal data [27]. In the context of Twitter analytics, GDPR introduces several constraints. Even though tweets are publicly accessible, they may still constitute personal data under GDPR when they can be linked to an identifiable individual.

Researchers collecting Twitter data must therefore consider the legal basis for processing, the principle of data minimization, the right to erasure, and restrictions on cross-border data transfers. The tension between the openness of Twitter's API and GDPR compliance remains an active area of legal and technical debate, particularly for large-scale academic and commercial analytics projects.

Data Governance

Data governance refers to the set of policies, processes, and standards that ensure data is managed consistently, securely, and responsibly throughout its lifecycle [28]. In Big Data systems applied to social media, effective data governance encompasses several dimensions: data quality management to ensure accuracy and completeness of collected tweets; access control policies to restrict unauthorized use of sensitive datasets; provenance tracking to maintain transparency about data origins and transformations; and retention policies governing how long data is stored and when it should be deleted. Without robust governance frameworks, Big Data analytics pipelines risk producing unreliable results, violating user privacy, or failing to meet regulatory requirements.

Ethical AI and Responsible Analytics

The integration of machine learning into social media analytics raises important questions about the ethical use of artificial intelligence. Ethical AI principles emphasize transparency, accountability, fairness, and the avoidance of harm in the design and deployment of analytical systems [29]. In the context of Twitter sentiment analysis, these principles translate into concrete requirements: models should be interpretable enough for their decisions to be understood and audited; analytical outputs should not be used to manipulate public opinion or target vulnerable individuals; and the limitations of automated sentiment classification — including its inability to reliably detect sarcasm, cultural nuance, or implicit sentiment — should be clearly communicated to end users. As AI-driven social media analytics is increasingly applied in high-stakes domains such as political communication, public health monitoring, and financial forecasting, adherence to ethical AI frameworks becomes not merely a recommendation but a professional obligation.

Bias in Social Media Data

Social media data, and Twitter data in particular, is subject to multiple forms of bias that can significantly distort analytical results if left unaddressed [30].

Selection bias arises from the fact that Twitter users do not represent the general population: they tend to be younger, more educated, more politically engaged, and concentrated in urban areas, which limits the generalizability of findings to broader society.

Algorithmic bias refers to distortions introduced by Twitter's own recommendation and content moderation systems, which influence what content is amplified or suppressed before it ever reaches researchers.

Temporal bias reflects the uneven distribution of tweet activity across time, with major events generating disproportionate volumes of data that can skew trend detection results.

Label bias is particularly relevant to sentiment analysis: datasets labeled through distant supervision — such as Sentiment140, which uses emoticons as proxies for sentiment — may introduce systematic noise, as the assumed correspondence between emoticons and sentiment polarity is not universally valid.

Acknowledging and mitigating these sources of bias is essential for producing reliable and generalizable insights from Twitter Big Data analytics.

II.4 Conclusion

This chapter has presented a comprehensive overview of Twitter as a rich and dynamic environment for Big Data analytics. Starting from the conceptual foundations of social networks, Twitter was examined both as a microblogging platform and as a large-scale data ecosystem characterized by high volume, velocity, and variety. Its asymmetric follower model, real-time communication structure, and interaction mechanisms such as hashtags, mentions, and retweets make it a unique platform for studying digital communication and information diffusion.

The chapter also highlighted the central role of Twitter APIs as the primary gateway for structured data acquisition. Through REST and Streaming interfaces, researchers can collect tweets, user metadata, and interaction networks in JSON format, enabling large-scale analytical workflows. However, API rate limits and sampling constraints require careful design of data collection strategies and scalable infrastructures.

From an analytical perspective, Twitter Big Data Analytics was presented as a multi-stage pipeline including data collection, preprocessing, content-based analysis, similarity measurement, network modeling, and real-time streaming analytics. Content-based approaches such as topic modeling and sentiment analysis enable extraction of thematic and emotional patterns from textual data. Similarity measures and clustering techniques support trend detection and thematic grouping, while network-based analytics reveals structural properties of user interactions and information diffusion. Furthermore, streaming architectures provide the capability to process Twitter data in real time, which is particularly important for time-sensitive applications such as crisis monitoring and event detection.

Overall, Twitter represents an ideal case study for applying Big Data technologies and computational social science methods. The integration of textual, temporal, and relational dimensions allows for a multidimensional understanding of online behavior. This chapter therefore establishes the theoretical and methodological foundation necessary for conducting large-scale Twitter analytics, which will support the practical and experimental components presented in the following chapters.

CHAPTER III

DESIGN AND IMPLEMENTATION

III.1 Introduction

This chapter presents the design and implementation of the proposed sentiment analysis system. Following a structured approach, it begins with the system design phase, which covers the functional requirements, structural components, and overall architecture. It then describes the development environment and the tools selected for the implementation. Finally, it details the implementation itself, covering both the batch processing pipeline for large-scale historical analysis and the real-time stream processing component, followed by a performance evaluation of all trained models.

The system is built to address the challenges of volume, velocity, and variety inherent in social media data. To achieve this, we rely on distributed computing frameworks and streaming platforms that ensure high performance, fault tolerance, and scalability. The overall workflow includes data collection, preprocessing, feature extraction, model training, prediction, and visualization.

First, we describe the tools and technologies used to develop the system, including distributed storage, data processing frameworks, and messaging systems. Then, we present the global architecture that illustrates how different components interact within the system.

The batch processing phase is detailed next, where we explain how historical Twitter data is collected, cleaned, and transformed before applying sentiment analysis models. This phase also includes the visualization of results to better understand the distribution and trends of sentiments.

After that, we introduce the stream processing component, which enables real-time sentiment analysis using a continuous flow of data. This part highlights how incoming tweets are processed instantly and how predictions are generated and stored.

Finally, we evaluate the performance of the implemented system using appropriate metrics such as accuracy and AUC, and we discuss the obtained results. The chapter concludes with a summary of the implementation and key findings.

III.2 System Design

This section presents the design of the proposed sentiment analysis system before describing its technical implementation. It covers the functional requirements through a use case diagram, the structural organization of the main components through a class diagram, and the overall system architecture.

III.2.1 Use Case Diagram

The use case diagram (Figure III.1) identifies the two primary actors interacting with the system and the main functions each actor can perform.

- Data Engineer: responsible for data ingestion, preprocessing, model training, and pipeline deployment.
- Analyst: responsible for exploring results, visualizing data through Power BI, and evaluating model performance.

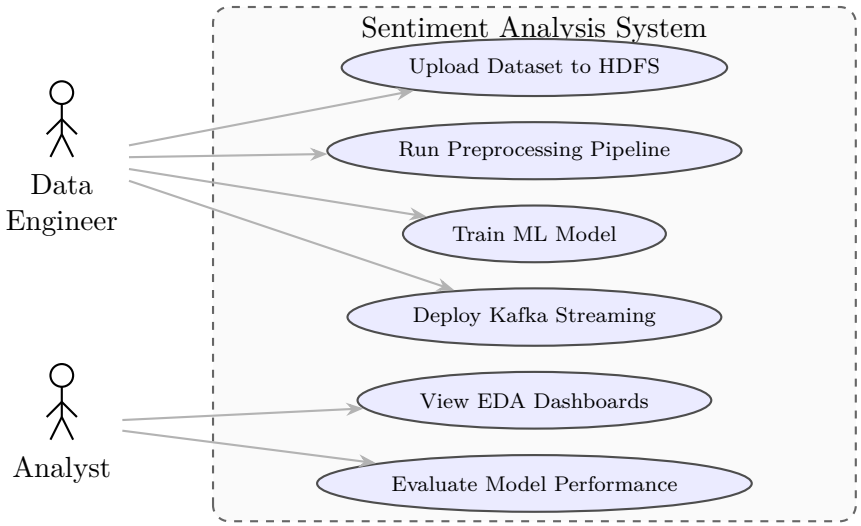


Figure III.1: Use Case Diagram of the sentiment analysis system.

III.2.2 Class Diagram

The class diagram (Figure III.2) presents the main logical components of the pipeline and their relationships at a conceptual level.

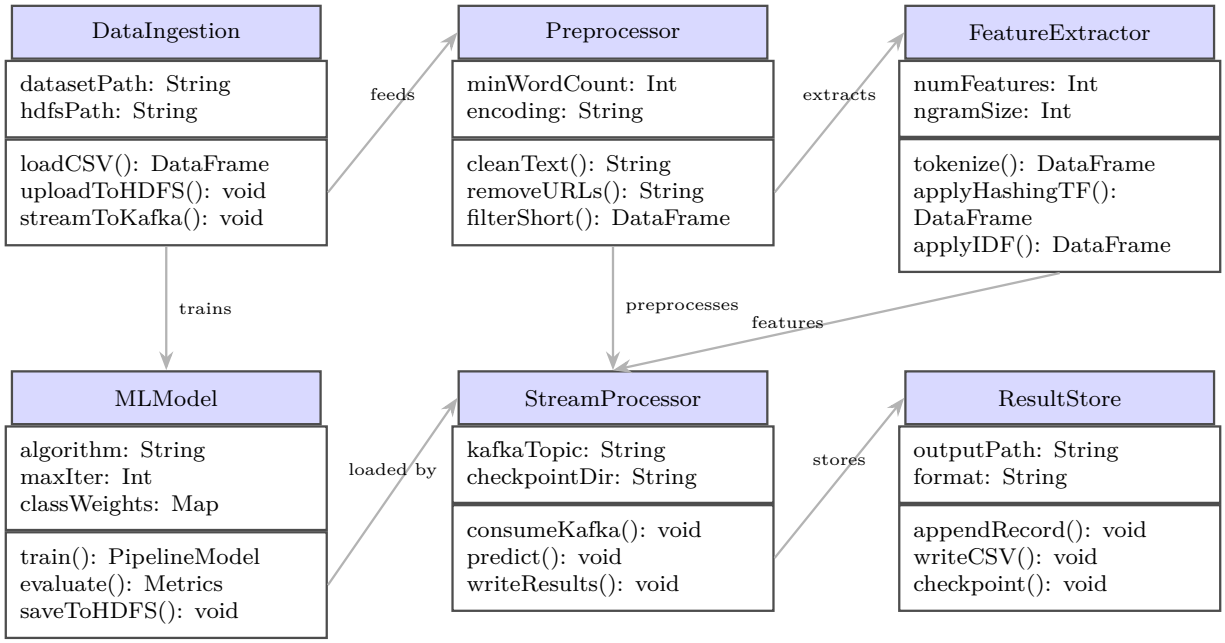


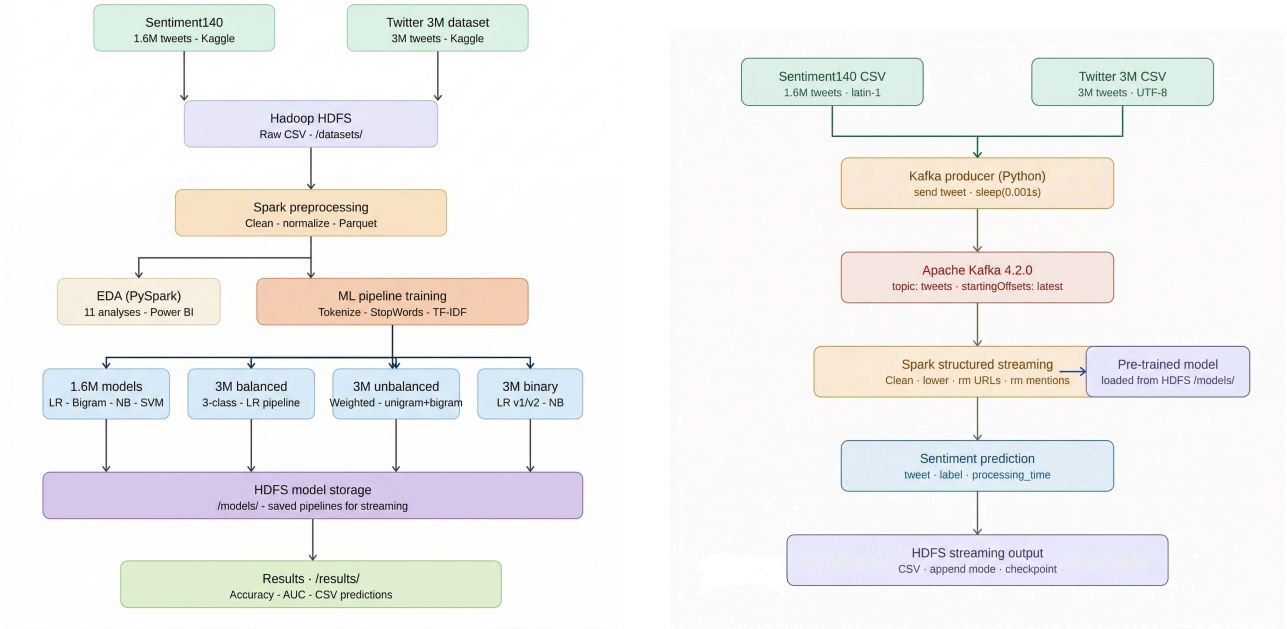
Figure III.2: Class Diagram of the main logical components of the sentiment analysis pipeline.

III.2.3 System Architecture

The architecture of the proposed sentiment analysis system is designed to handle both large-scale batch processing and real-time stream processing in a unified and scalable manner. It follows a modular approach, where each component is responsible for a specific task in the data pipeline, ensuring flexibility, fault tolerance, and high performance.

Overview

The system is composed of five main layers: data ingestion, storage, processing, machine learning, and output. These layers interact seamlessly to transform raw textual data into meaningful sentiment predictions, supporting both offline batch analysis and real-time streaming inference.



(a) Batch Processing (b) Stream Processing

Figure III.3: Architecture diagrams

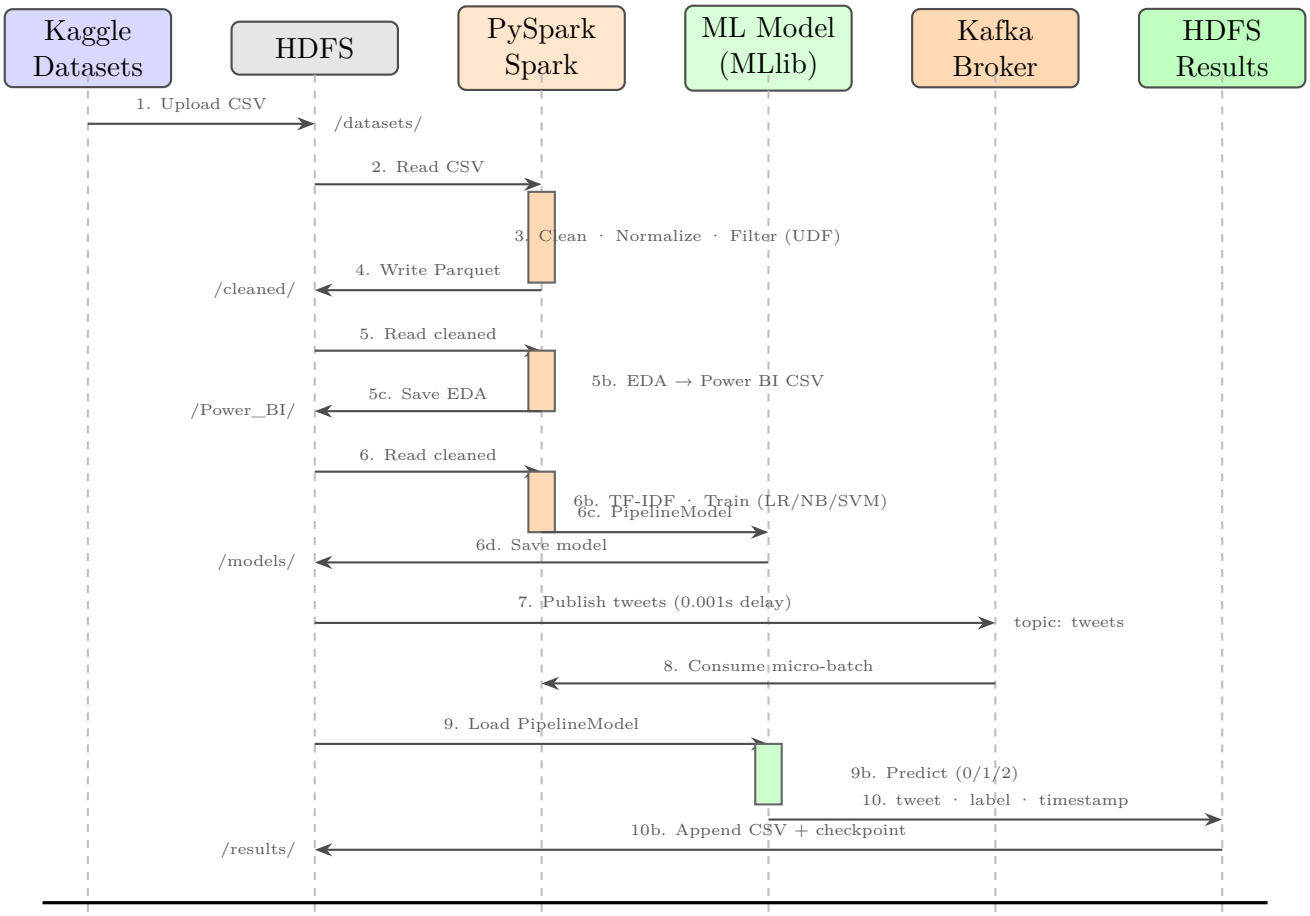


Figure III.4: UML Sequence Diagram of the complete sentiment analysis pipeline.

Data Ingestion Layer

The data ingestion layer is responsible for feeding the system with input data in two distinct modes.

For batch processing, two large-scale datasets were loaded from Kaggle and transferred into HDFS. The first is the Sentiment140 dataset containing 1.6 million annotated tweets, and the second is the Twitter Sentiment Dataset containing 3 million labelled rows. Both datasets were used to train and evaluate multiple machine learning models under different configurations.

For stream processing, Apache Kafka was used as a real-time data ingestion platform. A custom Python-based Kafka producer reads tweets from a CSV file and continuously sends them to a Kafka topic named tweets, simulating a live Twitter data stream. A time.sleep(0.001) delay was applied between messages to control the flow rate. Both the Sentiment140 dataset and the 3 million dataset were used as data sources for the streaming simulation.

Storage Layer

The storage layer is built on top of Hadoop HDFS (version 3.3.6), which serves as the centralized and distributed storage system. All datasets, intermediate results, trained models, and streaming outputs are stored in HDFS under organized paths. Raw datasets are stored under /datasets/, cleaned data in Parquet format under /cleaned/ and /processed/, trained pipeline models under /models/, and streaming prediction results under dedicated directories such as /Model_1.6_million_and_dataset_1.6_million/results. EDA outputs intended for visualization were stored under /Power_BI/ as single-partition CSV files.

Processing Layer

The processing layer is powered by Apache Spark 3.5.8 using PySpark, which handles both batch and streaming computations.

In batch mode, Spark processes large volumes of historical data using DataFrames. Text cleaning operations include lowercasing, removal of URLs, mentions (@user), hashtag symbols, non-alphabetic characters, and trimming of extra whitespace. After cleaning, data was stored in Parquet format for efficient downstream access.

In streaming mode, Spark Structured Streaming consumes real-time data from the Kafka topic tweets. Incoming tweets undergo the exact same preprocessing pipeline applied during training to ensure consistency between training and inference phases. The results are then passed directly to the pre-loaded machine learning model for inference.

Machine Learning Layer

The machine learning layer is implemented using Spark MLlib. A Pipeline approach is adopted to streamline the different stages of text processing and classification. Several models were trained and evaluated across two datasets and three classification scenarios.

Models on Sentiment140 (1.6 Million Tweets)

Four models were trained on the Sentiment140 dataset for binary classification (Positive vs. Negative):

- Model V1 — Logistic Regression (Unigrams): Tokenization, stop-word removal, HashingTF (numFeatures=20000), IDF, and Logistic Regression.
- Model V2 — Logistic Regression (Bigrams): Tokenization, stop-word removal, NGram (n=2), HashingTF (numFeatures=30000), IDF, and Logistic Regression.
- Model V3 — Naïve Bayes: Tokenization, stop-word removal, HashingTF (numFeatures =20000), IDF, and Naïve Bayes classifier.
- Model V4 — Linear SVM: Tokenization, stop-word removal, HashingTF (numFeatures =25000), IDF, and LinearSVC (maxIter=20, regParam=0.1).

Models on Twitter 3 Million Dataset

Three classification scenarios were explored on the 3 million tweet dataset:

- Balanced Multiclass (3 classes): The dataset was balanced by downsampling the Negative and Positive classes to match the Neutral class size. A Logistic Regression pipeline with HashingTF (numFeatures=30000) was trained on this balanced split.
- Unbalanced Multiclass (3 classes): Models were trained with class weights to handle imbalance. Two versions were built: one using CountVectorizer with IDF and Logistic Regression (maxIter=30), and an enhanced version adding bigrams, combined unigram and bigram features via VectorAssembler (maxIter=50).
- Binary Classification (Positive vs. Negative): The Neutral class was dropped to focus on binary sentiment. Three sub-models were trained: Logistic Regression V1 (maxIter=20), Logistic Regression V2 (maxIter=30, regParam=0.01), and Naïve Bayes, all using HashingTF with IDF.

All trained pipeline models were saved to HDFS under the /models/ directory for later use in the streaming phase.

Output Layer

The output layer is responsible for storing and presenting the results of the sentiment analysis.

In batch processing, predictions and performance metrics (Accuracy and AUC) were evaluated using Spark MLlib evaluators. Prediction results were saved to HDFS as CSV or Parquet files under /results/.

In streaming processing, pre-trained models were loaded from HDFS and applied directly to incoming Kafka messages. Results were continuously written to HDFS in CSV format. Each record includes the processed tweet text, the predicted sentiment label, and the processing timestamp. Multiple streaming experiments were conducted combining different trained models with different input datasets, stored under clearly named HDFS directories.

Workflow Summary

The overall workflow is illustrated as a seven-stage roadmap in Figure III.5, covering the full pipeline from raw data ingestion to real-time streaming predictions.

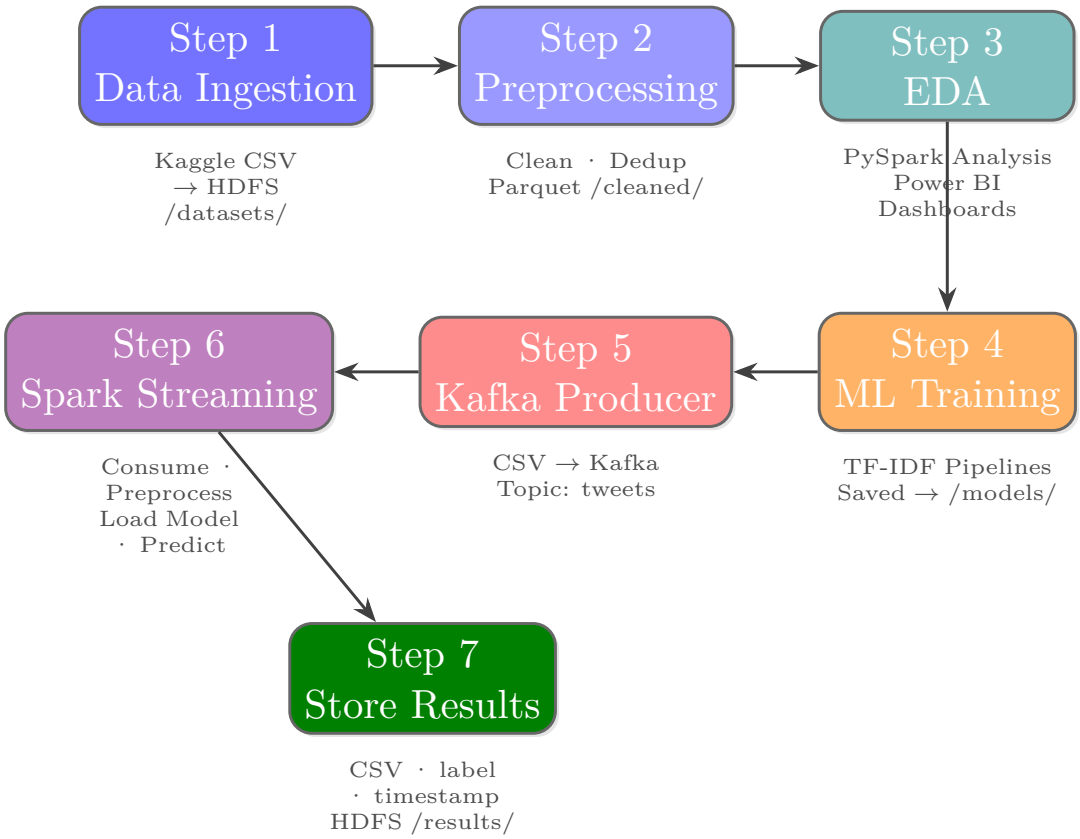


Figure III.5: Seven-stage workflow roadmap of the sentiment analysis system.

III.3 Development Environment and Tools

III.3.1 Big Data Frameworks

Apache Hadoop 3.3.6

Hadoop is widely considered the undisputed standard for big data storage. In our project, we utilized Hadoop primarily for its Hadoop Distributed File System (HDFS). Given the massive scale of our datasets (millions of rows), storing them on a local, traditional file system



Figure III.6: hadoop logo

was not feasible. HDFS allowed us to store the cleaned Parquet files reliably and served as the foundational layer from which our analytical engines retrieved data.

Apache Spark 3.5.8 and PySpark



Figure III.7: spark logo

While Hadoop is excellent for storage, we selected Apache Spark as our primary processing engine due to its exceptional speed. Unlike traditional MapReduce which writes intermediate data to disk, Spark processes data in-memory using Resilient Distributed Datasets (RDDs) and DataFrames, making it significantly faster for iterative machine learning tasks.

Since Spark is natively built on Scala and runs on the Java Virtual Machine (JVM), writing raw Spark code can be complex. To bypass this, we exclusively used PySpark, the official Python API for Spark. PySpark allowed us to leverage the simplicity and rich syntax of Python while utilizing Spark's distributed processing power under the hood. Furthermore, we heavily relied on Spark MLlib — Spark's scalable machine learning library — to build our NLP pipelines, apply TF-IDF, and train our Logistic Regression models without worrying about low-level JVM memory management.

Apache Kafka 4.2.0



Figure III.8: kafka logo

To bridge the gap between static batch processing and real-world scenarios, we integrated Apache Kafka to simulate real-time stream processing. Kafka is a distributed event-streaming platform that acts as a highly reliable message broker. In our simulation architecture, we utilized the following core concepts:

- Topic: We created specific channels where the continuous stream of tweets was published.
- Producer: We built a script that acts as a producer, fetching data and sending it to the Kafka topic in real-time. To ensure every ingested tweet was uniquely tracked and to avoid duplication, we assigned a UUID (Universally Unique Identifier) to each message.
- Consumer: On the other side, our Spark Streaming application acted as the consumer, subscribing to the topic, fetching the live messages, and immediately passing them to our pre-trained machine learning model for real-time sentiment prediction.

III.3.2 Data Acquisition and Datasets

Due to the strict limitations and high costs associated with the modern Twitter API, extracting millions of tweets directly from the platform was not a viable option. Therefore, we relied on Kaggle, the world's premier platform for data science and machine learning, which hosts massive, high-quality, pre-collected datasets essential for training robust big data models.

For this project, we utilized two primary datasets:

- Sentiment140 Dataset (1.6 Million Tweets): This highly popular dataset contains 1.6 million tweets automatically extracted using the Twitter API. The tweets are annotated with binary sentiment labels (0 for negative, 4 for positive). It contains six fields: target (polarity), ids, date, flag (query), user, and text. We used this dataset in our initial phases to train baseline models and test our data cleaning pipelines.

- Twitter Sentiment Dataset (3 Million Labelled Rows): To push the limits of our Spark cluster and perform a true big data training run, we transitioned to a much larger dataset containing 3 million rows. This dataset features three sentiment labels (0 = Negative, 1 = Positive, 2 = Neutral). However, the class distribution is severely imbalanced: the Negative and Positive classes each contain approximately 1.5 million tweets, while the Neutral class comprises only around 10,000 tweets — representing less than 0.35% of the total data. This extreme imbalance introduces two fundamental issues. First, any model trained on this raw distribution risks developing a strong prior bias toward the majority classes, effectively suppressing the gradient contribution of the minority Neutral class during optimization. Second, as confirmed by the EDA vocabulary analysis (Section III.4.1), the Neutral class in this dataset is dominated by domain-specific financial terminology (e.g., stocks, SPX, AAPL, trading), suggesting that it represents a topically biased subset rather than a linguistically neutral sentiment category. Including such a semantically skewed minority class would introduce systematic noise into the sentiment signal. For these reasons, the Neutral class was excluded in the binary classification scenario (Scenario 3), allowing the model to focus on the primary sentiment discrimination task with a balanced and linguistically coherent training set.

III.3.3 Programming Languages and Prerequisites

Building a big data environment requires careful configuration of underlying languages and dependencies. Our environment was built using:

Python 3.10



Figure III.9: python logo

Python served as the main programming language for writing data preprocessing scripts, orchestrating the PySpark pipelines, and evaluating the models. Python's readability and massive ecosystem made it the ideal choice for all analytical and machine learning tasks.

Java Development Kit (JDK)

Because Hadoop, Spark, and Kafka are all JVM-based, managing Java versions was a strict technical requirement. We configured our environment to use Java 11 to ensure full compatibility with the Apache Hadoop and Spark ecosystem, while Java 17 was required to properly run the latest version of Apache Kafka (4.2.0) for our streaming simulation.



Figure III.10: java logo

III.3.4 Experimental Environment

All experiments were conducted on a single personal laptop configured as a pseudo-distributed Big Data environment, simulating a single-node cluster. Table III.1 summarizes the hardware and software specifications of the experimental machine.

Table III.1: Hardware and software specifications of the experimental environment

Component	Specification
Machine	Lenovo ThinkPad X390
Operating System	Pop!_OS (Linux, Ubuntu 22.04 base)
CPU	Intel Core i5 WhiskeyLake-U (8 logical cores)
RAM	15 GiB
GPU	Intel UHD Graphics 620 (integrated, no CUDA)
Storage	Local SSD
Deployment Mode	Single-node pseudo-distributed cluster

Hadoop HDFS Cluster Startup

Figure III.11 shows the successful startup of the Hadoop HDFS pseudo-distributed cluster, confirming that all three required daemons — NameNode, DataNode, and SecondaryNameNode — are running simultaneously on the local machine. The jps command output verifies that the SparkSubmit process is also active, confirming the co-existence of both Hadoop and Spark on the same node.

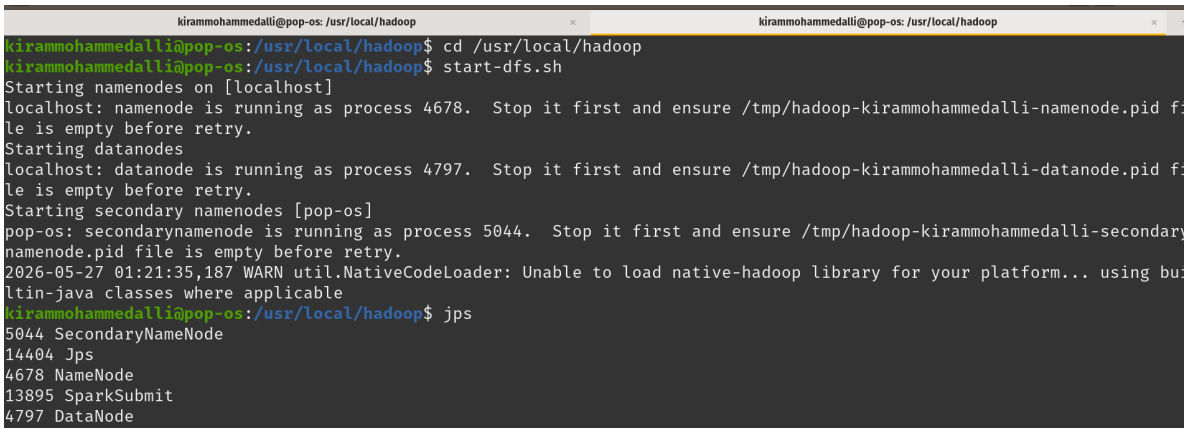


Figure III.11: Hadoop HDFS pseudo-distributed cluster startup and daemon verification via jps (NameNode, DataNode, SecondaryNameNode, SparkSubmit)

PySpark Session Initialization

Figure III.12 confirms the successful launch of a PySpark session with 8 GiB driver memory and 8 GiB executor memory. The Spark context runs in local[*] mode, utilizing all 8 available logical cores, and the Spark Web UI is accessible at http://pop-os:4040 for real-time job monitoring.

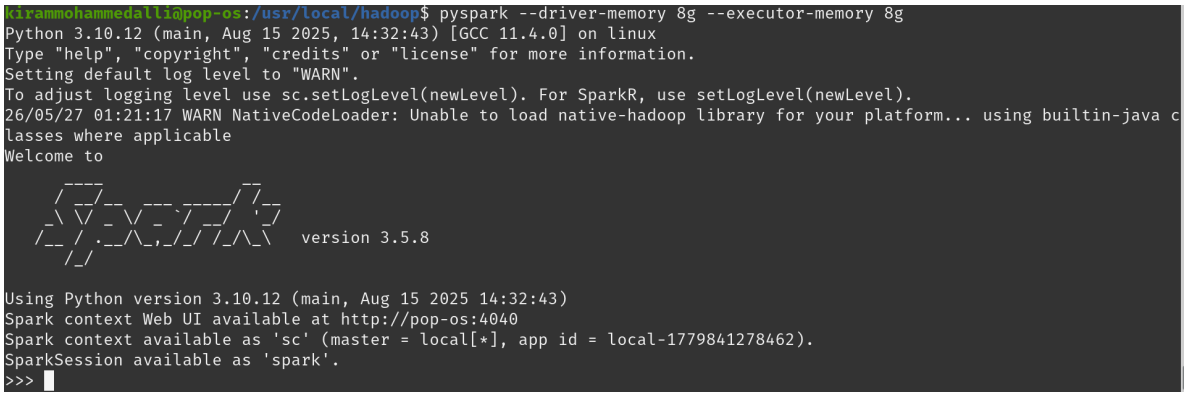


Figure III.12: PySpark 3.5.8 session launch with –driver-memory 8g –executor-memory 8g in local[*] mode

Software Stack Configuration

Table III.2 details the software versions and Java environment configuration used across the different components of the pipeline.

Table III.2: Software stack and version configuration

Component	Version / Configuration
Apache Hadoop	3.3.6 — pseudo-distributed mode (HDFS)
Apache Spark	3.5.8 — local[*] mode via PySpark
Apache Kafka	4.2.0 (KRaft mode, no ZooKeeper)
Python	3.10
Java (Hadoop/Spark)	OpenJDK 11
Java (Kafka)	OpenJDK 17.0.17 (2025-10-21)
Spark Driver Memory	8 GiB
Spark Executor Memory	8 GiB

Cluster Deployment Mode

The system operates in pseudo-distributed mode, where all Hadoop and Spark daemons run on the same physical machine while preserving the distributed architecture semantics. The HDFS cluster is started via the following daemon sequence:

1. hdfs –daemon start namenode
2. hdfs –daemon start datanode
3. hdfs –daemon start secondarynamenode

Spark sessions are launched with explicit memory allocation to maximize the available 15 GiB RAM:

pyspark –driver-memory 8g –executor-memory 8g

This configuration allocates 8 GiB to the Spark driver process and 8 GiB to the executor, leaving the remaining memory for the operating system and HDFS daemons.

Kafka KRaft Configuration

Apache Kafka 4.2.0 was deployed in KRaft mode (Kafka Raft metadata mode), which eliminates the dependency on Apache ZooKeeper. This represents a significant architectural simplification for single-node deployments. The Kafka setup procedure for each streaming experiment follows these steps:

1. Generate a unique cluster UUID:

bin/kafka-storage.sh random-uuid

2. Format the storage directory with the generated UUID:

bin/kafka-storage.sh format –standalone -t <UUID> -c config/server.properties

3. Start the Kafka broker:

   bin/kafka-server-start.sh config/server.properties

4. Create the tweets topic with a single partition and replication factor of 1 (appropriate for single-node deployment):

   bin/kafka-topics.sh –create –topic tweets –bootstrap-server localhost:9092 –partitions 1 –replication-factor 1

Implications for Big Data Scalability

It is important to contextualize the experimental environment within the Big Data paradigm. A single-node pseudo-distributed setup with 8 logical cores and 15 GiB RAM represents a resource-constrained environment compared to production multi-node clusters. Several implications follow directly from this configuration.

First, no GPU acceleration was available throughout all experiments. The Intel UHD Graphics 620 is an integrated graphics unit without CUDA support, which excludes the use of GPU-accelerated deep learning frameworks. This constraint directly motivates the choice of CPU-optimized distributed algorithms (Spark MLlib TF-IDF + Logistic Regression) over Transformer-based models, which require GPU infrastructure for practical training on million-scale datasets.

Second, Spark runs in local[*] mode, utilizing all 8 available logical cores for parallel task execution. While this is not equivalent to a multi-node cluster, it preserves the Spark DataFrame and MLlib API semantics, ensuring that the implemented pipelines are directly portable to a production cluster environment without code modification.

Third, memory allocation of 8 GiB per Spark process required careful management during training on the 3-million-row dataset. Parquet format was used throughout the pipeline to minimize memory footprint through columnar compression, and intermediate results were persisted to HDFS rather than kept in memory to prevent out-of-memory errors during multi-stage pipeline execution.

These constraints confirm that the performance results reported in Section III.5 were obtained under resource-limited conditions, and that deployment on a genuine multi-node cluster would be expected to reduce processing time significantly while maintaining equivalent model accuracy.

III.4 Implementation

This section describes the practical realization of the system components described in the design phase. It is divided into two main parts: batch processing for offline large-scale analysis, and stream processing for real-time inference.

III.4.1 Batch Processing

Introduction

The batch processing phase constitutes the core analytical component of this project. It encompasses the full pipeline from raw data collection to model training and evaluation, applied to two large-scale Twitter datasets. All processing was performed using Apache Spark via PySpark, leveraging its distributed computing capabilities to handle millions of records efficiently.

Data Collection

Due to the strict limitations and high costs associated with the modern Twitter API, tweets were not collected directly from the platform. Instead, two large-scale, pre-annotated datasets were obtained from Kaggle and stored in HDFS under the /datasets/ directory.

Sentiment140 Dataset

The first dataset is the well-known Sentiment140 corpus, containing 1,600,000 tweets extracted via the Twitter API and annotated automatically using distant supervision. The dataset contains six fields: target (sentiment polarity), ids (tweet ID), date, flag (query), user, and text. The original labelling scheme uses 0 for negative and 4 for positive tweets; this was remapped to a binary scheme (0 = Negative, 1 = Positive) during preprocessing. The dataset is perfectly balanced, containing exactly 800,000 tweets per class.

Twitter 3 Million Dataset

The second dataset contains 3,000,000 labelled tweets covering three sentiment classes: 0 (Negative), 1 (Positive), and 2 (Neutral). However, the class distribution is highly imbalanced: the Negative and Positive classes each contain approximately 1.5 million tweets, while the Neutral class contains only around 10,000 tweets. This imbalance required dedicated handling strategies during the model training phase, as described in Section III.4.1.

Preprocessing

Before training any machine learning model, both datasets underwent a rigorous data cleaning and preparation pipeline implemented in PySpark. The preprocessing steps were designed to remove noise, normalize the text, and ensure data quality.

Stop Words Configuration

In the preprocessing pipelines, stop-word removal was applied selectively. Two distinct configurations were used depending on the model variant:

- Spark MLlib Default Stop Words: For models V1–V4 on Sentiment140 and all Twitter 3M models, the built-in English stop-word list provided by StopWordsRemover in Spark MLlib was used. This list contains 318 common English words, including common pronouns, articles, prepositions, and conjunctions (e.g., the, a, is, in, at, which, on).
- Custom Stop Words (EDA only): For the Exploratory Data Analysis phase (Section III.4.1), a domain-specific extended stop-word list was applied to the frequency analysis. This custom list augmented the default Spark list with Twitter-specific tokens that carry no analytical value, including: rt (retweet marker), via, amp (HTML entity residue), lol, im, ur, u, r, got, like, get, just, go, know, time, day, one, now, today, will.

It is worth noting that, as discussed in Section III.5, the Baseline model — which omits stop-word removal entirely — achieved the highest performance on the Sentiment140 dataset. This finding underscores the importance of domain-specific evaluation before applying standard stop-word lists to short social media text.

Preprocessing of the Sentiment140 Dataset

The raw CSV file was loaded into Spark with ISO-8859-1 encoding (latin-1), as the file contains non-ASCII characters. Columns were renamed to meaningful names: target, ids, date, flag, user, and text. The following cleaning steps were then applied sequentially:

- Duplicate removal: Rows with duplicate tweet IDs were dropped using dropDuplicates (["ids"]).
- Null filtering: Rows with missing text or target values were removed.
- Label remapping: The original label 4 (positive) was remapped to 1, keeping 0 as negative, resulting in a standard binary classification scheme.
- Text cleaning (UDF): A custom Python UDF was applied to each tweet, performing the following operations:
  - Removal of URLs (http, www).
  - Removal of @mentions.
  - Removal of the # symbol while preserving hashtag words.
  - Removal of HTML entities (&, <, etc.).
  - Stripping of non-ASCII characters.
  - Removal of special characters, keeping only letters, digits, spaces, and basic punctuation.
  - Collapsing of multiple whitespace characters.
  - Conversion to lowercase.
- Short tweet filtering: Tweets with fewer than 3 words after cleaning were discarded.
- Metadata cleaning: The user column was normalized to lowercase and stripped of special characters. The flag column was uppercased and trimmed.

The cleaned dataset was saved back to HDFS as a CSV file under /datasets/twitter/ with UTF-8 encoding, with a word_count column added for downstream EDA use.

Preprocessing of the Twitter 3 Million Dataset

The 3 million tweet dataset was loaded from HDFS with UTF-8 encoding and multiline support. Columns were renamed defensively to id, tweet, and sentiment. The following cleaning steps were applied:

- Leaked header removal: Rows where the tweet column contained the literal string "tweet" — a common artifact of concatenated CSV exports — were filtered out.
- Type casting: The sentiment column was cast to integer and filtered to retain only valid labels $\{0, 1, 2\}$.
- Null and empty filtering: Rows with null or empty tweet values were dropped.
- Text cleaning (UDF): The same text normalization pipeline described above was applied, including URL removal, mention removal, hashtag normalization, HTML entity stripping, ASCII encoding, and whitespace collapsing.

- Duplicate removal: Exact duplicates based on the (tweet, sentiment) pair were removed.
- Short tweet filtering: Tweets with fewer than 3 words were discarded.
- ID reassignment: Clean sequential IDs were assigned using monotonically_increasing_id().

The cleaned dataset was saved to HDFS under /datasets/twitter3m/ as a single-partition CSV file.

Preprocessing Execution in PySpark

Figure III.13 illustrates the live execution of the preprocessing pipeline in a PySpark interactive shell. The screenshot confirms the actual reading of the Sentiment140 dataset from HDFS (hdfs://localhost:9000/datasets/twitter/), column renaming, text cleaning via regexp_replace, label remapping from 4 to 1, and the construction of the final cleaned DataFrame — all executed on the distributed Spark engine.

```
kirammohammedalli@pop-os:/usr/local/hadoop$ pyspark --driver-memory 8g --executor-memory 8g
Python 3.10.12 (main, Aug 15 2025, 14:32:43) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
26/05/26 18:22:25 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java c
lasses where applicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.5.8
      /_/

Using Python version 3.10.12 (main, Aug 15 2025 14:32:43)
Spark context Web UI available at http://pop-os:4040
Spark context available as 'sc' (master = local[*], app id = local-1779816145893).
SparkSession available as 'spark'.
>>> df = spark.read.csv("hdfs://localhost:9000/datasets/twitter/training.1600000.processed.noemoticon.csv", header=Fals
e, inferSchema=True)
>>> # Rename columns
>>> df = df.toDF("label", "id", "date", "query", "user", "text")
>>> # Clean text: lowercase + remove URLs + remove non-alphabetic characters
>>> from pyspark.sql.functions import lower, regexp_replace
>>> df_clean = df.withColumn("clean_text", lower(regexp_replace("text", "http\\S+", "")))
>>> df_clean = df_clean.withColumn("clean_text", regexp_replace("clean_text", "[^a-zA-Z\\s]", ""))
>>> # Create sentiment column (4 → 1, 0 stays 0)
>>> from pyspark.sql.functions import when
>>> df_final = df_clean.withColumn("sentiment", when(df_clean.label == 4, 1).otherwise(0))
```

Figure III.13: Live PySpark execution of the Sentiment140 preprocessing pipeline: HDFS data loading, text cleaning, and label remapping

Sentiment Analysis

The sentiment analysis phase consists of building and evaluating machine learning pipelines using Spark MLlib. All pipelines share a common NLP structure: tokenization, stop-word removal, feature extraction via TF-IDF (either HashingTF or CountVectorizer combined with IDF), and a classification algorithm. The models were trained and evaluated across two datasets and three classification scenarios. For binary classification tasks, performance is reported using Accuracy, F1-Score, and AUC. For multiclass tasks, AUC is not applicable in Spark MLlib's multiclass setting; Accuracy, F1-Score, and Precision are reported instead.

Models on Sentiment140 (1.6 Million Tweets)

Five models were trained on the preprocessed Sentiment140 dataset for binary sentiment classification (Positive vs. Negative). The dataset was split into 80% training and 20% test sets using a fixed random seed of 42.

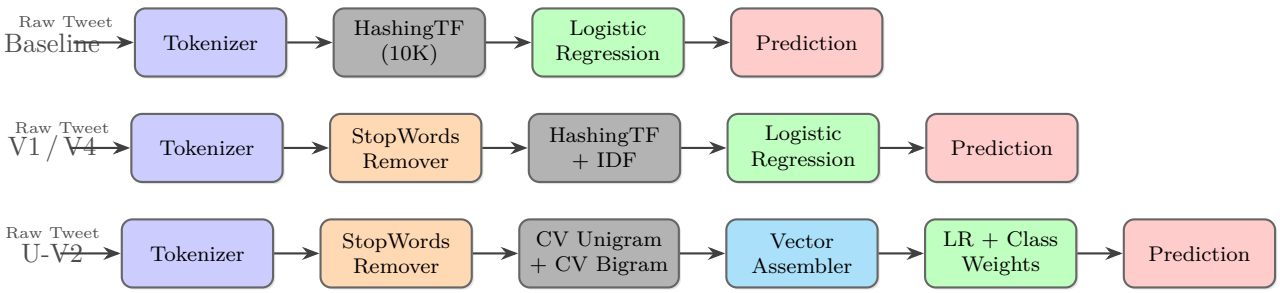


Figure III.14: Spark MLlib pipeline architectures for the three main model configurations.

Baseline model — Logistic Regression (no stop-word removal) As a first exploration, a minimal pipeline was built without stop-word removal: Tokenizer → HashingTF (numFeatures=10000) → Logistic Regression. Despite its simplicity, this baseline achieved the highest performance on the Sentiment140 dataset, with Accuracy 77.44%, F1-Score 77.44%, and AUC 0.847. This result suggests that for this dataset, stop-words carry discriminative sentiment signals that are lost when removed.

Baseline Model Training and Evaluation Output

Figure III.15 presents the actual terminal output of the Baseline model training and evaluation on the Sentiment140 dataset. The results confirm the reported metrics: Accuracy = 0.7744, F1-Score = 0.7743, and AUC = 0.8469, directly validating the values reported in Table III.5. The pipeline (Tokenizer → HashingTF → Logistic Regression) is clearly visible in the code, and the evaluators used are MulticlassClassificationEvaluator and BinaryClassificationEvaluator from pyspark.ml.evaluation.

```
>>> # Tokenize
>>> from pyspark.ml.feature import Tokenizer
>>> tokenizer = Tokenizer(inputCol="clean_text", outputCol="words")
>>> df_words = tokenizer.transform(df_final)
>>> # HashingTF
>>> from pyspark.ml.feature import HashingTF
>>> hashingTF = HashingTF(inputCol="words", outputCol="features", numFeatures=10000)
>>> df_features = hashingTF.transform(df_words)
>>> # Train/Test split
>>> train, test = df_features.randomSplit([0.8, 0.2], seed=42)
>>> # Logistic Regression
>>> from pyspark.ml.classification import LogisticRegression
>>> lr = LogisticRegression(featuresCol="features", labelCol="sentiment")
>>> model = lr.fit(train)
>>> predictions = model.transform(test)
>>> # Evaluate
>>> from pyspark.ml.evaluation import MulticlassClassificationEvaluator, BinaryClassificationEvaluator
>>> accuracy = MulticlassClassificationEvaluator(labelCol="sentiment", predictionCol="prediction", metricName="accuracy
").evaluate(predictions)
>>> f1 = MulticlassClassificationEvaluator(labelCol="sentiment", predictionCol="prediction", metricName="f1").evaluate(
predictions)
>>> auc = BinaryClassificationEvaluator(labelCol="sentiment", rawPredictionCol="rawPrediction", metricName="areaUnderRO
C").evaluate(predictions)
>>>
>>> print("Accuracy =", accuracy)
Accuracy = 0.7744214189892774
>>> print("F1 Score =", f1)
F1 Score = 0.7743637772315854
>>> print("AUC      =", auc)
AUC      = 0.8469717253603968
```

Figure III.15: Terminal output of Baseline Logistic Regression training and evaluation on Sentiment140: Accuracy = 0.7744, F1 = 0.7743, AUC = 0.8469

Model V1 — Logistic Regression with unigrams The first pipeline introduced stop-word removal: Tokenizer → StopWordsRemover → HashingTF (numFeatures=20000) → IDF → Logistic Regression. Accuracy: 76.51%, F1-Score: 76.50%, AUC: 0.839. The slight drop compared

to the baseline confirms that removing stop-words slightly reduced discriminative power for this dataset.

Model V2 — Logistic Regression with bigrams An NGram (n=2) stage was inserted between StopWordsRemover and HashingTF (numFeatures=30000), replacing unigram tokens with bigram pairs. Accuracy: 65.24%, F1-Score: 65.20%, AUC: 0.718. The significant drop in performance indicates that discarding unigrams entirely and relying solely on bigrams is detrimental for this dataset, likely due to increased data sparsity at the bigram level.

Model V3 — Naïve Bayes Stop-word removal was retained, and Logistic Regression was replaced by a Naïve Bayes classifier: Tokenizer → StopWordsRemover → HashingTF (numFeatures =20000) → IDF → Naïve Bayes. Accuracy: 74.50%, F1-Score: 74.50%, AUC: 0.499. While the accuracy remains reasonable, the near-random AUC (0.499) reveals that Naïve Bayes produces unreliable probability estimates, confirming that its conditional independence assumption is poorly suited to tweet-level text features.

Model V4 — Linear SVM A LinearSVC classifier (maxIter=20, regParam=0.1) was used with HashingTF (numFeatures=25000) and IDF. Accuracy: 76.56%, F1-Score: 76.54%, AUC: 0.839. Performance is comparable to V1, confirming that linear classifiers perform similarly on this task when given equivalent features.

Models on Twitter 3 Million Dataset

Three classification scenarios were explored on the 3 million tweet dataset, each addressing the challenge of class imbalance differently.

Scenario 1 — Balanced multiclass classification Given the extreme imbalance in the dataset (approximately 1.5M Negative, 1.5M Positive, and only ∼10,000 Neutral tweets), a downsampling strategy was applied to balance all three classes. The Negative and Positive classes were randomly sampled to match the Neutral class size, producing a balanced dataset of approximately 30,000 tweets total. Three models were trained on this balanced split:

- Balanced Model V1: Tokenizer → StopWordsRemover → HashingTF (numFeatures=30000) → Logistic Regression (maxIter=20). Accuracy: 67.90%, F1-Score: 68.23%, Precision: 68.68%. This is the best-performing model under the balanced training strategy.
- Balanced Model V2: Tokenizer → StopWordsRemover → NGram (n=2) → HashingTF (numFeatures=100000) → IDF → Logistic Regression (maxIter=30, reg-Param=0.01). Accuracy: 50.22%, F1-Score: 50.65%, Precision: 51.91%. The poor performance indicates that relying solely on bigrams with a small balanced dataset leads to degenerate training, likely converging to a near-constant prediction.
- Balanced Model V3: Tokenizer → StopWordsRemover → Word2Vec (vectorSize=200, minCount=2) → Random Forest (numTrees=100, maxDepth=10). Accuracy: 67.82%, F1-Score: 67.71%, Precision: 67.83%. Comparable to V1, demonstrating that dense word embeddings with tree-based classifiers reach a similar performance ceiling under the data constraint imposed by downsampling.

The moderate accuracy figures (67–68%) are expected given the severe reduction in training data after downsampling, and the inherent difficulty of distinguishing Neutral from Positive and Negative tweets.

Scenario 2 — Unbalanced multiclass classification with class weights To exploit the full dataset without discarding data, class weights were computed as $w_c = \frac{N}{k \cdot n_c}$, where N is the total number of samples, $k = 3$ is the number of classes, and n_c is the count of class c. These weights were passed to the Logistic Regression weightCol parameter. Two models were trained:

- Unbalanced Model V1: Tokenizer → StopWordsRemover → CountVectorizer (vocabSize =30000) → IDF → Logistic Regression (maxIter=30) with class weights. Accuracy: 77.17%, F1-Score: 77.29%, Precision: 77.62%.
- Unbalanced Model V2 (enhanced): An extended pipeline combining both unigram and bigram TF-IDF representations. Unigram features were extracted using CountVectorizer (vocab-Size=40000, minDF=3) followed by IDF (minDocFreq=3). Bigram features were extracted via NGram (n=2), CountVectorizer (vocabSize=20000, minDF=5), and IDF (minDoc-Freq=5). Both feature sets were concatenated using VectorAssembler, and Logistic Regression was trained with maxIter=50, regParam=0.01, and class weights. Accuracy: 79.15%, F1-Score: 79.24%, Precision: 79.49%. This is the best-performing model across all multiclass scenarios.

Scenario 3 — Binary classification (Positive vs. Negative) The Neutral class was entirely removed from the 3 million dataset, reducing the problem to binary classification. The cleaned binary dataset was saved in Parquet format under /cleaned/twitter_binary_cleaned. Four models were trained:

- Binary Model V1: Tokenizer → StopWordsRemover → HashingTF (numFeatures=2^{18}) → IDF → Logistic Regression (maxIter=20). Accuracy: 78.39%, F1-Score: 78.39%, AUC: 0.850. This is the best-performing binary model on this dataset.
- Binary Model V2: Same architecture with reduced features (numFeatures=2^{17}), L2 regularization (regParam=0.01, elasticNetParam=0), and maxIter=30. Accuracy: 78.06%, F1-Score: 78.06%, AUC: 0.854. Although slightly lower in accuracy than V1, this model achieves the highest AUC (0.854), indicating better probability calibration and ranking ability.
- Binary Model V3 — Naïve Bayes: HashingTF (numFeatures=2^{17}), IDF, and Naïve Bayes (smoothing=1.0). Accuracy: 75.75%, F1-Score: 75.74%, AUC: 0.522. As observed on Sentiment140, Naïve Bayes produces near-random AUC values, confirming the unreliability of its posterior probability estimates regardless of the dataset.
- Binary Model V4: HashingTF (numFeatures=2^{17}), IDF, and Logistic Regression with ElasticNet regularization (maxIter=50, regParam=0.01, elasticNetParam=0.1). Accuracy: 77.07%, F1-Score: 77.04%, AUC: 0.847.

All trained pipeline models were saved to HDFS under the /models/ directory for subsequent use in the real-time streaming phase. Table III.3 summarizes the performance of all trained models across both datasets and all scenarios.

Table III.3: Summary of trained models and their performance metrics

Dataset	Scenario	Model	Algorithm	Accuracy	F1	AUC / Precision
Sentiment140 (1.6M)	Binary	Base.	LR · HashingTF (10K)	77.44%	77.44%	0.847
		V1	LR · HashingTF (20K) + IDF	76.51%	76.50%	0.839
		V2	LR · Bigrams (30K) + IDF	65.24%	65.20%	0.718
		V3	Naïve Bayes · HashingTF (20K)	74.50%	74.50%	0.499
		V4	LinearSVC · HashingTF (25K)	76.56%	76.54%	0.839
Twitter 3M	Balanced (3-class)	B-V1	LR · HashingTF (30K)	67.90%	68.23%	Precision: 0.687
		B-V2	LR · Bigrams (100K) + IDF	50.22%	50.65%	Precision: 0.519
		B-V3	Word2Vec + Random Forest	67.82%	67.71%	Precision: 0.678
	Unbalanced (3-class)	U-V1	LR · CV (30K) + weights	77.17%	77.29%	Precision: 0.776
		U-V2	LR · Uni+Bigrams + weights	79.15%	79.24%	Precision: 0.795
	Binary	Bin-V1	LR · HashingTF (2^{18})	78.39%	78.39%	0.850
		Bin-V2	LR + L2 · HashingTF (2^{17})	78.06%	78.06%	0.854
		Bin-V3	Naïve Bayes · HashingTF (2^{17})	75.75%	75.74%	0.522
		Bin-V4	LR + ElasticNet · HashingTF (2^{17})	77.07%	77.04%	0.847

Visualization

To gain a deeper understanding of the data prior to model training, an Exploratory Data Analysis (EDA) phase was conducted using PySpark on both datasets. The analytical outputs were stored in HDFS and subsequently imported into Microsoft Power BI for interactive visualization.

EDA on Sentiment140 Dataset

Eleven analytical outputs were generated from the cleaned Sentiment140 dataset and saved as single-partition CSV files under /Power_BI/:

- EDA-01: Label distribution — tweet counts and percentages per sentiment class.
- EDA-02: Word count bucket distribution — grouping tweets into ranges (1–5, 6–10, ..., 31+ words) per sentiment.
- EDA-03: Top 50 most frequent words per sentiment class, after removing a custom stopword list.
- EDA-04: Daily tweet counts per sentiment class.
- EDA-05: Monthly tweet counts per sentiment class.
- EDA-06: Hourly tweet counts per sentiment class, revealing posting time patterns.
- EDA-07: Top 30 most frequent bigrams per sentiment class, extracted after stopword filtering.

EDA on Twitter 3 Million Dataset

Eight analytical outputs were generated from the cleaned 3 million dataset and saved under /Power_BI_3m/:

- File 01: Sentiment distribution with counts and percentages for all three classes.
- File 02: Tweet length statistics per class (character and word level, including standard deviation).
- File 03: Tweet length histogram buckets (20-character intervals).
- Files 04–06: Top 50 most frequent words for Negative, Positive, and Neutral tweets respectively. @users across the dataset.
- File 07: Summary KPI card — total tweets, unique tweets, average lengths, and per-class counts and percentages.

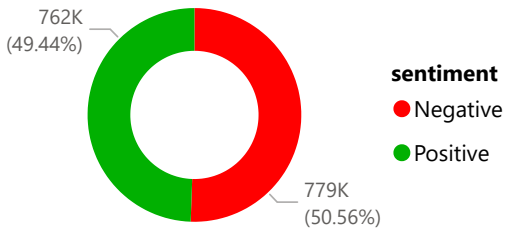
The following pages present selected visualizations produced from the EDA phase for both datasets.

sentiment

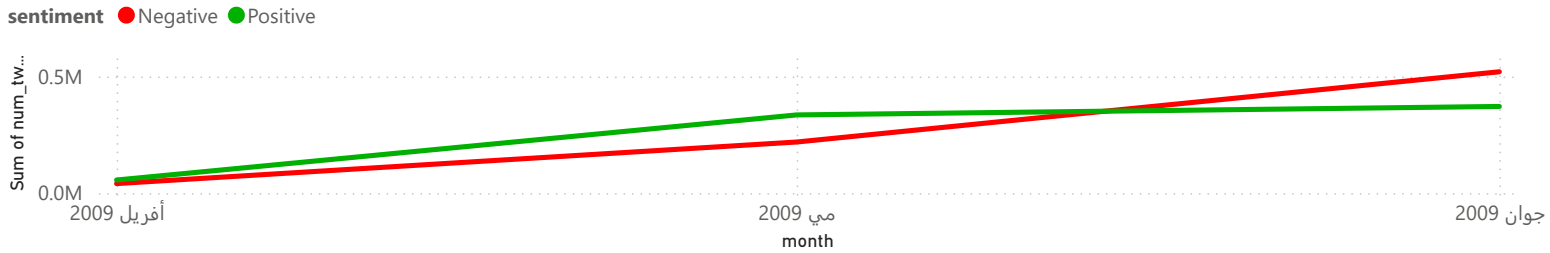
- [ ] Negative
- [ ] Positive

Total Tweets	Positive Tweets	Negative Tweets	Positive Rate	Positive Rate
1.54M	762K	779K	49.44%	50.56%
Total Tweets	Positive Tweets	Negative Tweets	Positive Rate %	Negative Rate %

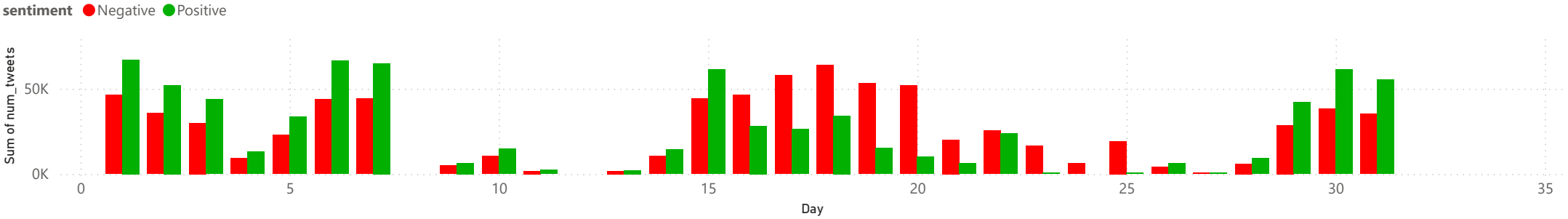
Sentiment Distribution



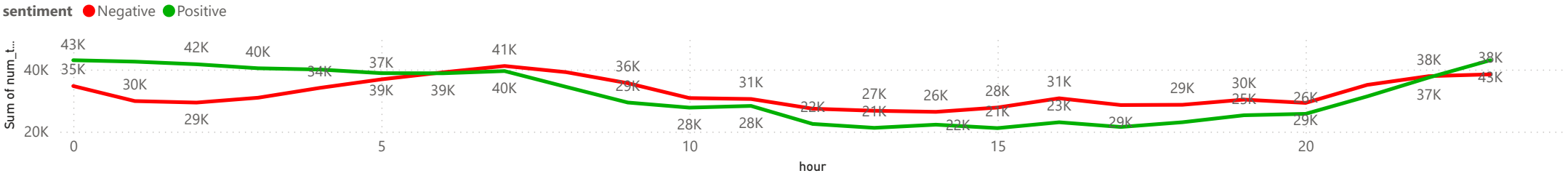
Monthly Sentiment Trend

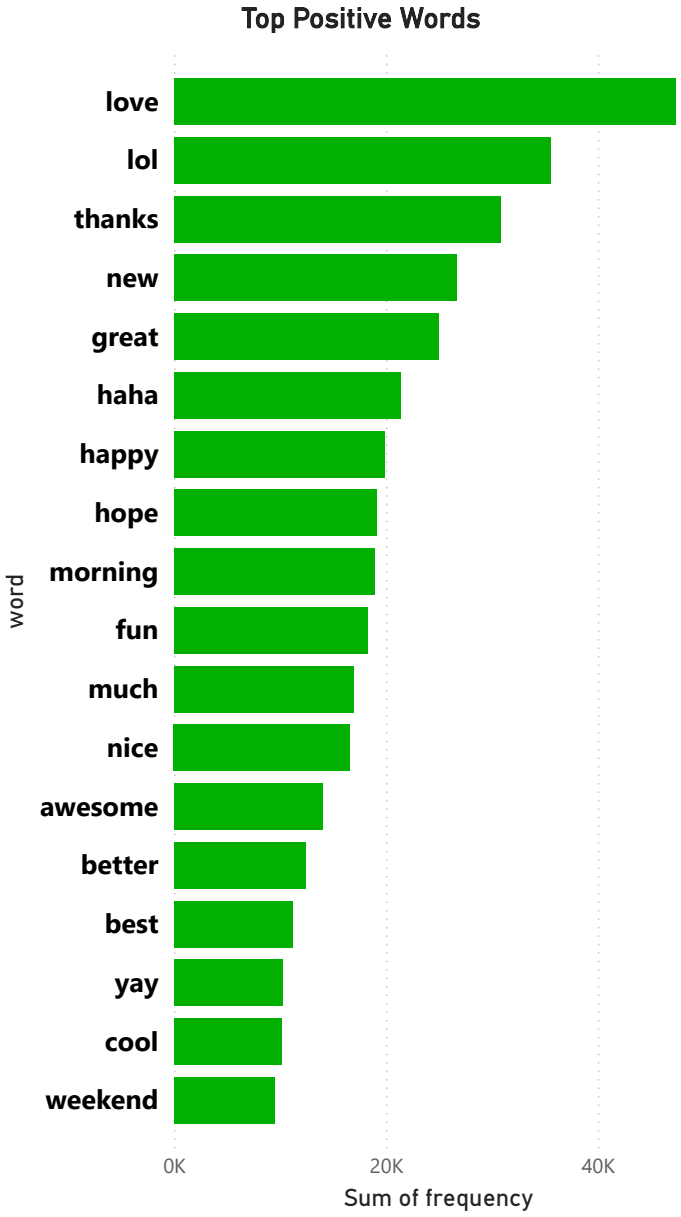


Tweet Activity by Day

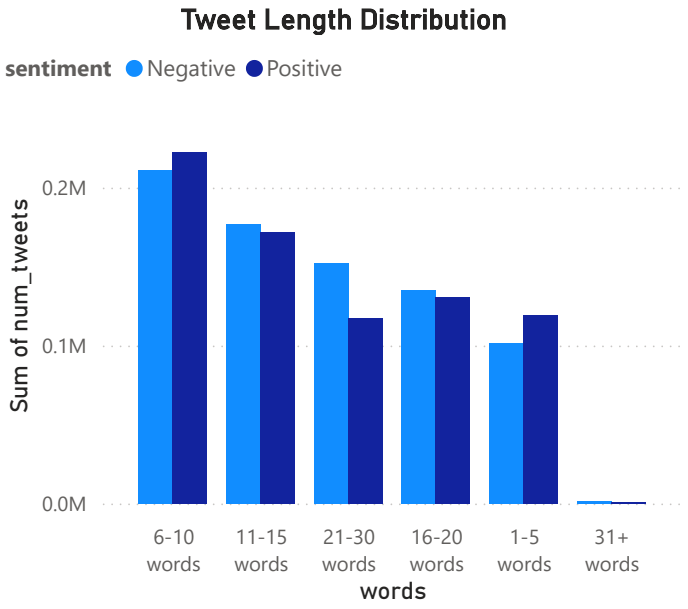


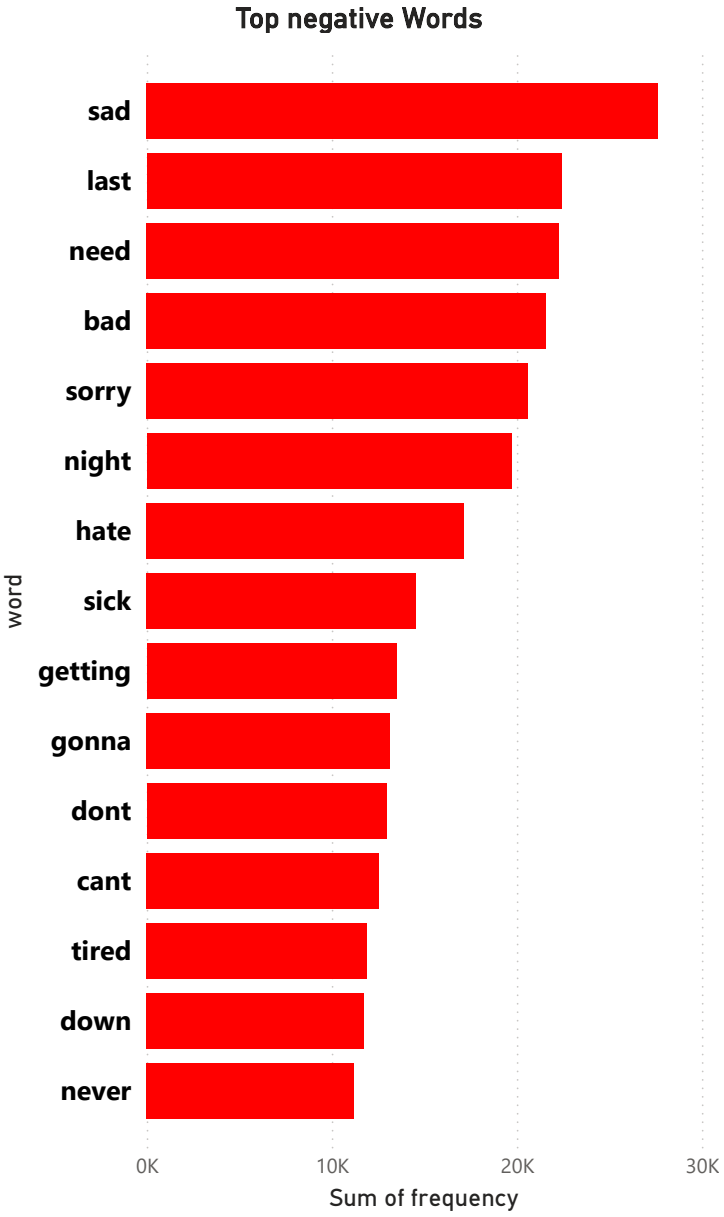
Tweet Activity by Hour




Top Positive Words
word
love
lol
thanks
new
great
haha
happy
hope
morning
fun
much
nice
awesome
better
best
yay
cool
weekend
0K
20K
40K
Sum of frequency


Top Bigrams
bigram
last night
can't wait
looking for...
getting ready
i'm sorry
i'm gonna
next week
i'm sure
feel better
cant wait
can't sleep
happy birth...
can't believe
0K
10K
Sum of frequency


Tweet Length Distribution
sentiment
Negative
Positive
Sum of num_tweets
0.2M
0.1M
0.0M
6-10 words
11-15 words
21-30 words
16-20 words
1-5 words
31+ words
words


Top negative Words
word
sad
last
need
bad
sorry
night
hate
sick
getting
gonna
dont
cant
tired
down
never
0K
10K
20K
30K
Sum of frequency

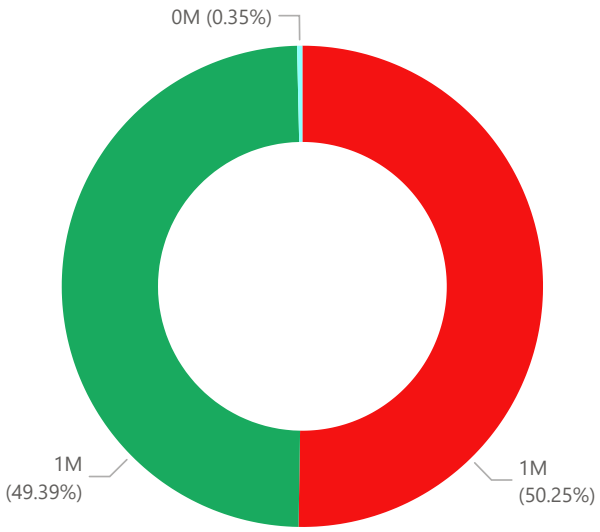
Total Tweets	Positive Tweets	Negative Tweets	Neutral Tweets	Positive Rate %
2.93M	1.45M	1.47M	10.39K	49.39%

Negative | Neutral | Positive

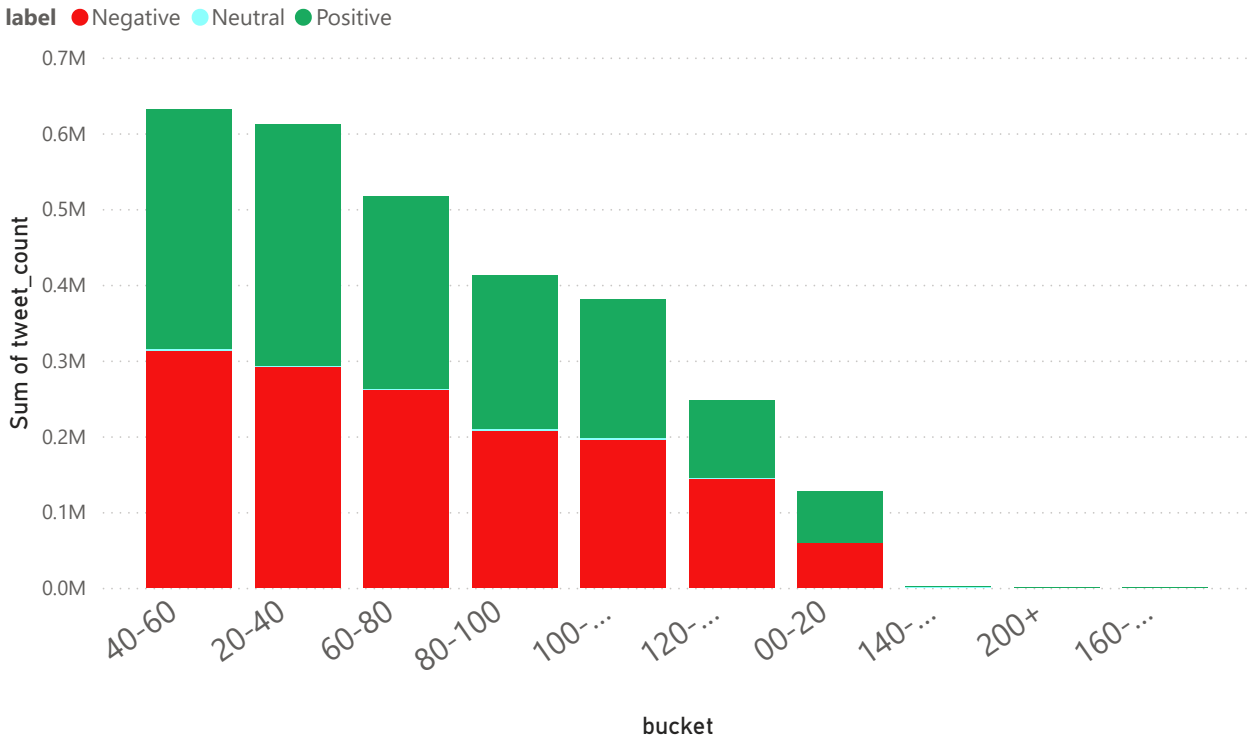
Tweet Length Statistical Summary

14.15	3.00	60.00	7.43
Average of avg_word_count	Average of min_word_count	Average of max_word_count	Average of stddev_word_count

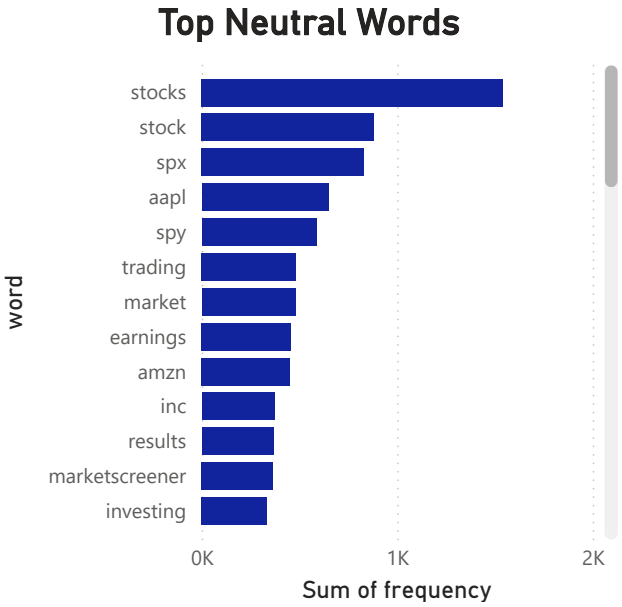
Sentiment Distribution (3-Class)

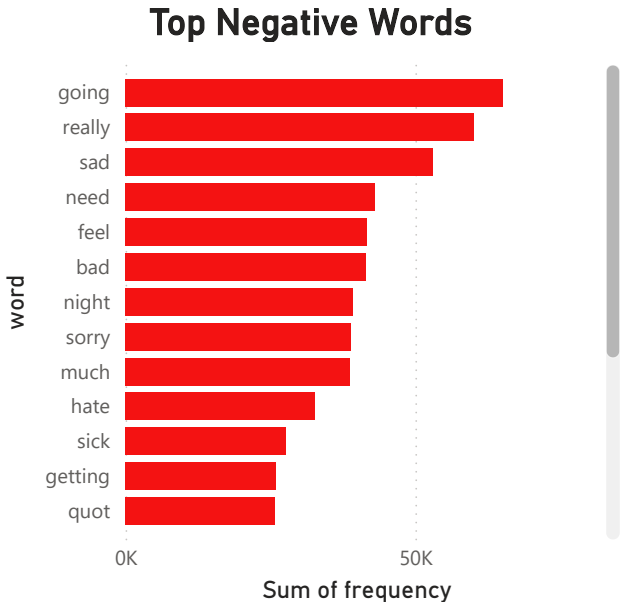


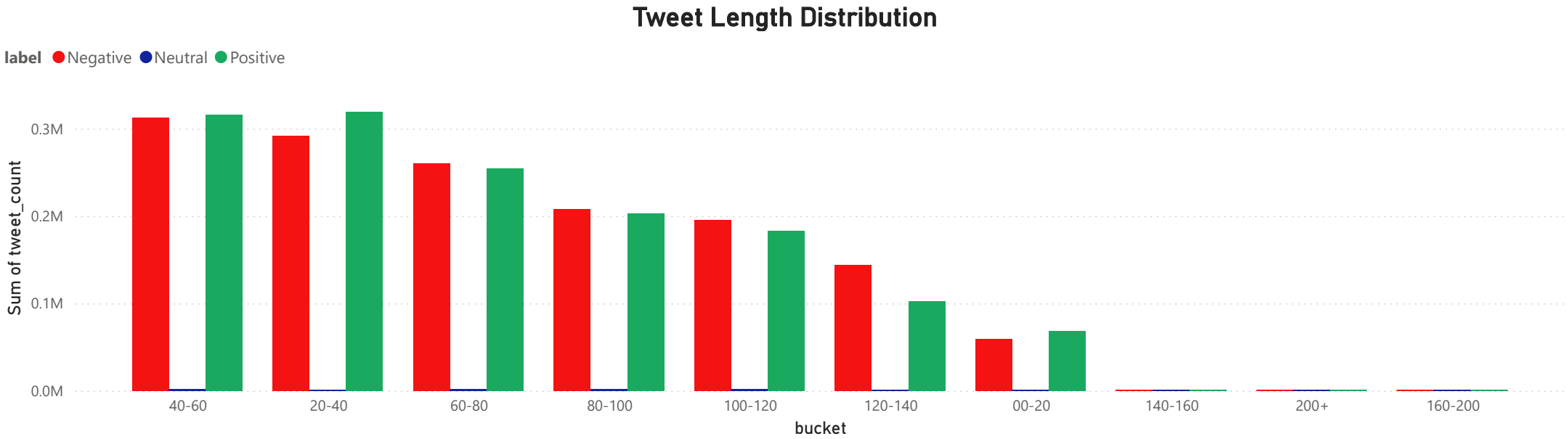
Tweet Length Distribution




Top Positive Words
good
love
thanks
great
happy
morning
hey
better
best
yay
cool
weekend
word
0.0M
0.1M
Sum of frequency


Top Neutral Words
stocks
stock
spx
aapl
spy
trading
market
earnings
amzn
inc
results
marketscreener
investing
word
0K
1K
2K
Sum of frequency


Top Negative Words
going
really
sad
need
feel
bad
night
sorry
much
hate
sick
getting
quot
word
0K
50K
Sum of frequency


Tweet Length Distribution
label
Negative
Neutral
Positive
Sum of tweet_count
0.3M
0.2M
0.1M
0.0M
40-60
20-40
60-80
80-100
100-120
120-140
00-20
140-160
200+
160-200
bucket

III.4.2 Stream Processing

Introduction

The stream processing phase extends the batch-trained sentiment analysis models to a real-time inference scenario. Rather than analyzing static datasets, this phase simulates a live Twitter data stream where tweets arrive continuously and are classified instantaneously. The pipeline integrates Apache Kafka as a distributed message broker with Spark Structured Streaming as the real-time processing engine, forming a complete end-to-end streaming architecture.

The core principle of this phase is the reuse of pre-trained models: all machine learning pipelines were trained offline during the batch processing phase and saved to HDFS. They are then loaded directly into the streaming application, enabling real-time predictions without any retraining overhead. A critical constraint respected throughout all streaming experiments is preprocessing consistency: the exact same text cleaning operations applied during training must be reapplied to each incoming tweet before inference, to ensure that the feature space seen by the model matches what it was trained on.

Streaming Architecture

The streaming pipeline follows a six-stage architecture as illustrated in Figure III.3b:

1. Dataset: A pre-existing CSV dataset (either Sentiment140 or the 3 million tweet dataset) is read from the local file system to simulate a continuous tweet source.

2. Kafka Producer: A Python script reads tweets from the dataset row by row and publishes them to a Kafka topic named tweets, with a time.sleep(0.001) delay between messages to simulate a realistic ingestion rate.

3. Apache Kafka: The Kafka broker (version 4.2.0) receives the published messages and makes them available to downstream consumers. The startingOffsets option is set to latest so that only new incoming messages are processed.

4. Spark Structured Streaming: A PySpark streaming application subscribes to the Kafka topic, casts the binary message value to a string, and applies the same text preprocessing pipeline used during training.

5. ML Model: The pre-trained PipelineModel is loaded from HDFS and applied to each incoming micro-batch of cleaned tweets, producing a sentiment prediction label for each record.

6. HDFS Storage: The results — comprising the tweet text, the predicted sentiment label, and a processing_time timestamp — are written continuously to HDFS in CSV format using append output mode, with checkpoint locations maintained for fault tolerance.

Kafka Broker Startup and Topic Creation

Figures III.16 and III.17 document the complete Kafka KRaft initialization process. Figure III.16 shows the storage formatting with a randomly generated cluster UUID, the KRaft metadata version negotiation (metadata.version 4.2-IV1), and the successful broker startup sequence confirming that the BrokerServer transitioned from STARTING to STARTED state and is awaiting socket connections on 0.0.0.0:9092. The log also confirms the creation of the tweets-0 partition in /tmp/kraft-combined-logs.



Figure III.16: Apache Kafka 4.2.0 KRaft mode initialization: UUID generation, storage formatting, and broker startup

Figure III.17 shows the creation and verification of the tweets Kafka topic with 1 partition and replication factor 1, appropriate for single-node deployment. The kafka-topics.sh –list command confirms the topic is active and ready to receive messages from the producer.

```
kirammohammedalli@pop-os:~/kafka_2.13-4.2.0$ /usr/lib/jvm/java-17-openjdk-amd64/bin/java -version
openjdk version "17.0.17" 2025-10-21
OpenJDK Runtime Environment (build 17.0.17+10-Ubuntu-122.04)
OpenJDK 64-Bit Server VM (build 17.0.17+10-Ubuntu-122.04, mixed mode, sharing)
kirammohammedalli@pop-os:~/kafka_2.13-4.2.0$ JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64 \
bin/kafka-topics.sh \
--create \
--topic tweets \
--bootstrap-server localhost:9092 \
--partitions 1 \
--replication-factor 1
Created topic tweets.
kirammohammedalli@pop-os:~/kafka_2.13-4.2.0$ JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64 \
bin/kafka-topics.sh \
--list \
--bootstrap-server localhost:9092
tweets
```

Figure III.17: Kafka topic tweets creation and verification: 1 partition, replication factor 1, bootstrap server localhost:9092

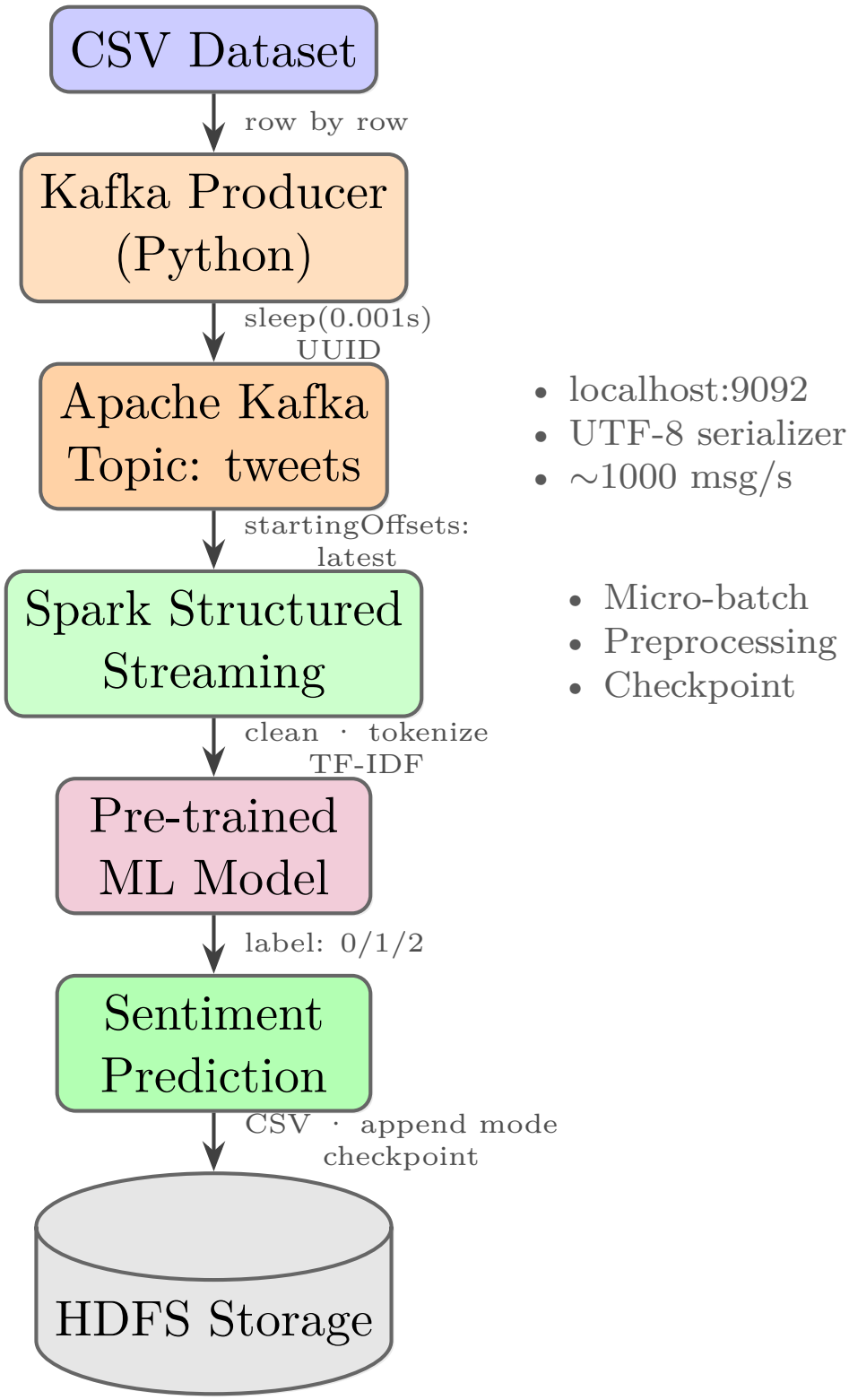


Figure III.18: Kafka–Spark Structured Streaming pipeline for real-time sentiment analysis.

Kafka Producer

The Kafka producer component is a lightweight Python script responsible for simulating a live tweet stream by feeding data from a CSV file into the Kafka topic. The producer is configured with a value_serializer that encodes each tweet as a UTF-8 string before transmission. A fixed inter-message delay of 1 millisecond (time.sleep(0.001)) was applied to control the message rate and avoid overwhelming the Kafka broker.

Six Kafka producer scripts were used across the streaming experiments, each simulating a live tweet stream by continuously reading records from a CSV dataset and publishing them to the Kafka topic tweets. All producers share the same core configuration: a bootstrap server at localhost:9092, a UTF-8 value serializer, and a fixed inter-message delay of 1 millisecond to simulate a realistic ingestion rate.

- Producer 1 — Sentiment140 → 1.6M model: Streams tweets from the Sentiment140 dataset to the model trained on the same dataset. This experiment evaluates in-distribution streaming performance, where the data source matches the model's training distribution.
- Producer 2 — Twitter 3M (full) → 1.6M model: Streams tweets from the full Twitter 3M dataset to the model trained on Sentiment140. This experiment assesses the cross-dataset generalization capability of the 1.6M-trained model when exposed to a different data distribution.
- Producer 3 — Twitter 3M (full) → 3M balanced model: Streams tweets from the full Twitter 3M dataset to the multiclass model trained on the balanced version of the same dataset. This evaluates real-time inference for the three-class classification scenario under balanced training conditions.
- Producer 4 — Twitter 3M (full) → 3M unbalanced model: Streams tweets from the full Twitter 3M dataset to the multiclass model trained with class weights on the unbalanced version of the same dataset. This evaluates real-time inference under the unbalanced training strategy, which retained the full data distribution during training.
- Producer 5 — Sentiment140 → 3M binary model: Streams tweets from the Sentiment140 dataset to the binary model trained exclusively on the Positive and Negative classes of the Twitter 3M dataset. Since both datasets are binary in nature, this experiment tests how well the 3M binary model generalizes to a temporally and stylistically different tweet corpus.
- Producer 6 — Twitter 3M (binary) → 3M binary model: Streams only the Positive and Negative tweets from the Twitter 3M dataset to the binary model trained on the same data. This experiment validates in-distribution streaming performance for the binary classification scenario.

Kafka Producer Execution

Figure III.19 shows the execution of the Kafka producer script (producer_tweets.py), confirming that real tweet data from the Sentiment140 dataset is being read and published to the Kafka tweets topic. The raw tweet content — including mentions, URLs, and informal language — is visible before preprocessing, confirming that the producer feeds unprocessed data to Kafka and that all cleaning operations are applied downstream by the Spark Structured Streaming consumer.

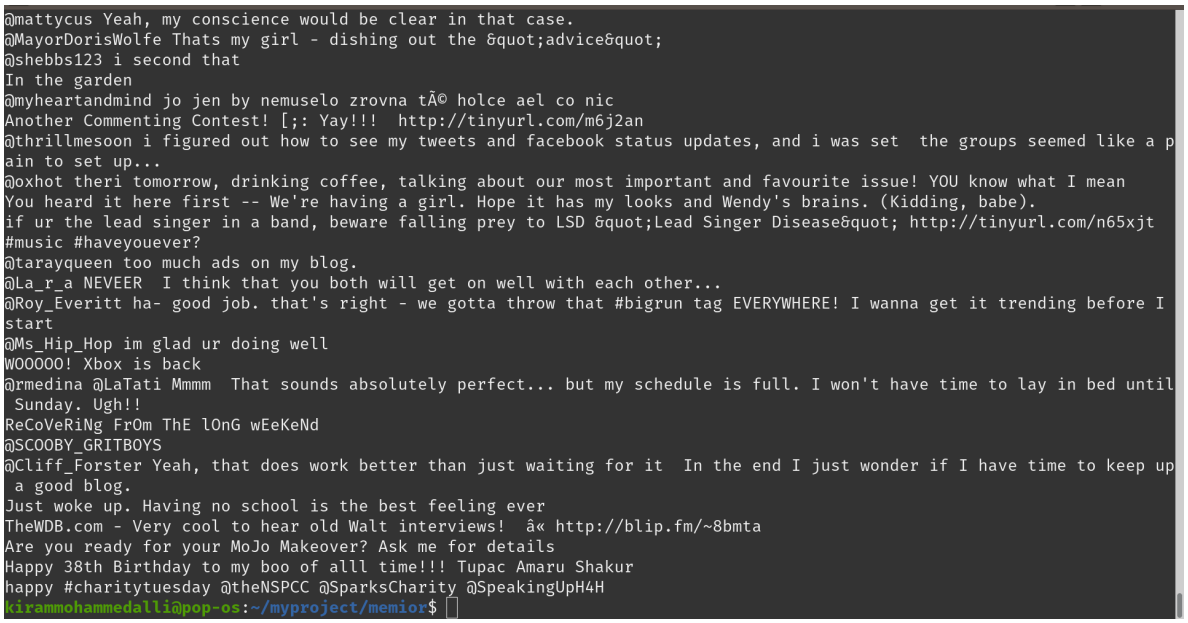


Figure III.19: Kafka producer output: raw Sentiment140 tweets being streamed to the tweets topic, showing unprocessed content including mentions, URLs, and hashtags

Spark Structured Streaming Consumer

The Spark Structured Streaming consumer subscribes to the tweets Kafka topic and processes incoming messages in micro-batches. Each incoming message is cast from binary Kafka format to a UTF-8 string via selectExpr("CAST(value AS STRING) as tweet"). The preprocessing and inference steps are then applied in a fully declarative Spark DataFrame API style, ensuring compatibility with the streaming execution engine.

The preprocessing applied within the streaming context mirrors exactly the steps performed during batch training:

- Conversion to lowercase.
- Removal of URLs (http, www).
- Removal of @mentions.
- Normalization of hashtags (symbol removed, word preserved).
- Removal of all non-alphabetic characters.
- Trimming of leading and trailing whitespace.
- Filtering of empty tweets after cleaning.

A processing_time column is added using current_timestamp() to record the exact moment each micro-batch was processed. The output stream is written to HDFS using writeStream in append output mode with CSV format, with a dedicated checkpoint directory to enable recovery in case of failure. The maxRecordsPerFile option was set to 50,000 where applicable to avoid generating excessively large output files.

Spark Streaming Job Submission and Initialization

Figures III.20 and III.21 document the actual submission and initialization of the Spark Structured Streaming job consuming from Kafka. Figure III.20 shows the spark-submit command with the spark-sql-kafka connector package, and the Ivy dependency resolution confirming that all 11 required artifacts — including kafka-clients-3.4.1, lz4-java, and snappy-java — were successfully resolved from Maven Central.

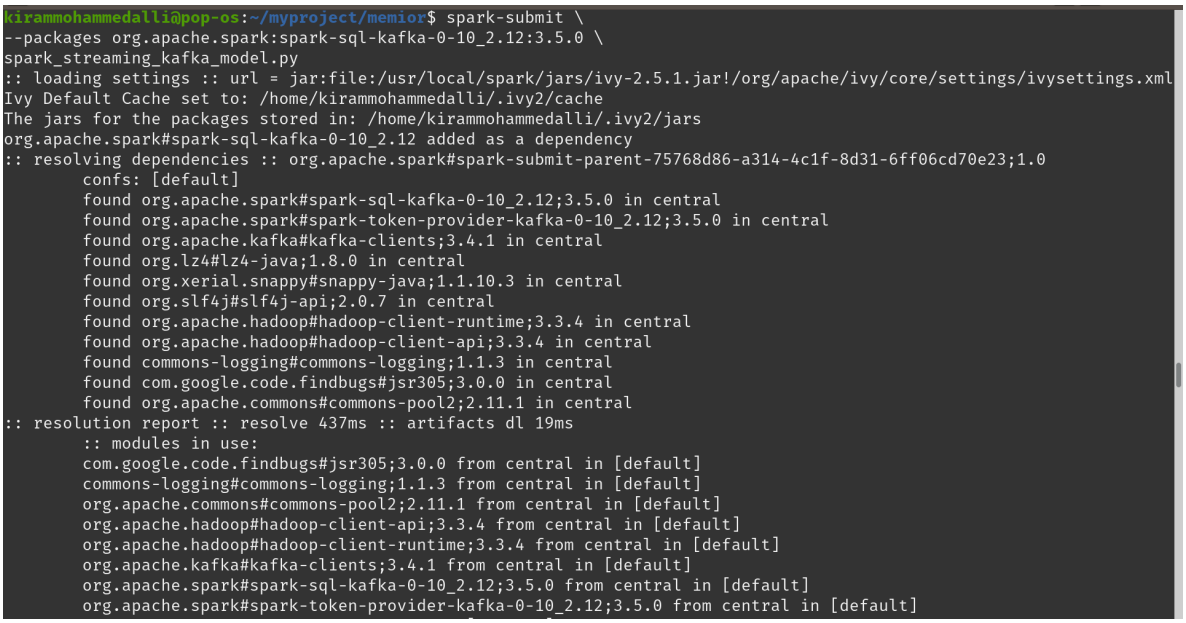


Figure III.20: spark-submit with Kafka connector: dependency resolution of 11 artifacts including spark-sql-kafka-0-10_2.12:3.5.0

Figure III.21 confirms the successful initialization of SparkContext version 3.5.8 on Linux/amd64 with Java 11.0.29, and the registration of the BlockManager with 434.4 MiB RAM — confirming that the streaming application is running and connected to the Kafka topic.

```
:: resolution report :: resolve 437ms :: artifacts dl 19ms
        :: modules in use:
        com.google.code.findbugs#jsr305;3.0.0 from central in [default]
        commons-logging#commons-logging;1.1.3 from central in [default]
        org.apache.commons#commons-pool2;2.11.1 from central in [default]
        org.apache.hadoop#hadoop-client-api;3.3.4 from central in [default]
        org.apache.hadoop#hadoop-client-runtime;3.3.4 from central in [default]
        org.apache.kafka#kafka-clients;3.4.1 from central in [default]
        org.apache.spark#spark-sql-kafka-0-10_2.12;3.5.0 from central in [default]
        org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.5.0 from central in [default]
        org.lz4#lz4-java;1.8.0 from central in [default]
        org.slf4j#slf4j-api;2.0.7 from central in [default]
        org.xerial.snappy#snappy-java;1.1.10.3 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   11  |   0   |   0   |   0   ||   11  |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-75768d86-a314-4c1f-8d31-6ff06cd70e23
        confs: [default]
        0 artifacts copied, 11 already retrieved (0kB/11ms)
26/05/27 01:49:20 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java c
lasses where applicable
26/05/27 01:49:21 INFO SparkContext: Running Spark version 3.5.8
26/05/27 01:49:21 INFO SparkContext: OS info Linux, 6.17.4-76061704-generic, amd64
26/05/27 01:49:21 INFO SparkContext: Java version 11.0.29
26/05/27 01:49:21 INFO ResourceUtils: ==============================================================
26/05/27 01:49:21 INFO ResourceUtils: No custom resources configured for spark.driver.
26/05/27 01:49:21 INFO ResourceUtils: ==============================================================
```

```
26/05/27 01:49:22 INFO Executor: Fetching spark://pop-os:33503/jars/org.lz4_lz4-java-1.8.0.jar with timestamp 177984296
1885
26/05/27 01:49:22 INFO Utils: Fetching spark://pop-os:33503/jars/org.lz4_lz4-java-1.8.0.jar to /tmp/spark-be8044f9-efde
-4dfd-ad6c-9e114ef4be8b/userFiles-77ceeea5-d156-4c9a-9997-fecae680dadc/fetchFileTemp17709818648729816244.tmp
26/05/27 01:49:22 INFO Utils: /tmp/spark-be8044f9-efde-4dfd-ad6c-9e114ef4be8b/userFiles-77ceeea5-d156-4c9a-9997-fecae68
0dadc/fetchFileTemp17709818648729816244.tmp has been previously copied to /tmp/spark-be8044f9-efde-4dfd-ad6c-9e114ef4be
8b/userFiles-77ceeea5-d156-4c9a-9997-fecae680dadc/org.lz4_lz4-java-1.8.0.jar
26/05/27 01:49:22 INFO Executor: Adding file:/tmp/spark-be8044f9-efde-4dfd-ad6c-9e114ef4be8b/userFiles-77ceeea5-d156-4c
9a-9997-fecae680dadc/org.lz4_lz4-java-1.8.0.jar to class loader default
26/05/27 01:49:22 INFO Executor: Fetching spark://pop-os:33503/jars/org.apache.kafka_kafka-clients-3.4.1.jar with times
tamp 1779842961885
26/05/27 01:49:22 INFO Utils: Fetching spark://pop-os:33503/jars/org.apache.kafka_kafka-clients-3.4.1.jar to /tmp/spark
-be8044f9-efde-4dfd-ad6c-9e114ef4be8b/userFiles-77ceeea5-d156-4c9a-9997-fecae680dadc/fetchFileTemp9931442194000266310.t
mp
26/05/27 01:49:22 INFO Utils: /tmp/spark-be8044f9-efde-4dfd-ad6c-9e114ef4be8b/userFiles-77ceeea5-d156-4c9a-9997-fecae68
0dadc/fetchFileTemp9931442194000266310.tmp has been previously copied to /tmp/spark-be8044f9-efde-4dfd-ad6c-9e114ef4be8
b/userFiles-77ceeea5-d156-4c9a-9997-fecae680dadc/org.apache.kafka_kafka-clients-3.4.1.jar
26/05/27 01:49:22 INFO Executor: Adding file:/tmp/spark-be8044f9-efde-4dfd-ad6c-9e114ef4be8b/userFiles-77ceeea5-d156-4c
9a-9997-fecae680dadc/org.apache.kafka_kafka-clients-3.4.1.jar to class loader default
26/05/27 01:49:23 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' o
n port 39899.
26/05/27 01:49:23 INFO NettyBlockTransferService: Server created on pop-os:39899
26/05/27 01:49:23 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication
policy
26/05/27 01:49:23 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, pop-os, 39899, None)
26/05/27 01:49:23 INFO BlockManagerMasterEndpoint: Registering block manager pop-os:39899 with 434.4 MiB RAM, BlockMana
gerId(driver, pop-os, 39899, None)
26/05/27 01:49:23 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, pop-os, 39899, None)
26/05/27 01:49:23 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, pop-os, 39899, None)
```

Figure III.21: SparkContext 3.5.8 initialization for streaming: Java 11.0.29, BlockManager registered with 434.4 MiB RAM

Streaming Experiments

A total of five streaming experiments were conducted, each combining a specific pre-trained model with a specific input dataset. The experiments are designed to evaluate the generalization capability of the trained models when applied to data streams that differ from their original training distribution.

Experiment 1 — 1.6M Model on Sentiment140 Stream

Model: The best-performing model trained on the Sentiment140 dataset (Baseline Logistic Regression, numFeatures=10000, Accuracy 77.44%, AUC 0.847), loaded from /models/twitter_lr_model.
Data stream: Sentiment140 dataset (1.6 million tweets), streamed via Producer 1.
Preprocessing: Tokenization and HashingTF (numFeatures=10000) applied inline within the

streaming pipeline, consistent with the baseline training pipeline.
Output path: /Model_1.6_million_and_dataset_1.6_million/results
Objective: Validate in-distribution streaming performance — the model processes data drawn from the same distribution as its training set.

Experiment 2 — 1.6M Model on 3M Binary Stream

Model: Same 1.6M-trained Baseline Logistic Regression model (/models/twitter_lr_model).
Data stream: Twitter 3M binary dataset (Positive and Negative tweets only), streamed via Producer 2.
Preprocessing: Same inline tokenization and HashingTF (numFeatures=10000).
Output path: /Model_1.6_million_and_dataset_3_million_binary/results
Objective: Evaluate cross-dataset generalization — the 1.6M model is tested on a stylistically different dataset to assess its robustness to domain shift.

Experiment 3 — 3M Balanced Model on 3M tweets Stream

Model: Best balanced multiclass model trained on the 3M dataset (Balanced V1, Logistic Regression, Accuracy 67.90%, F1-Score 68.23%), loaded as a full PipelineModel from /models/model_twitter3m_v1_balanced.
Data stream: Full Twitter 3M dataset, streamed via Producer 3.
Preprocessing: URL removal and non-alphabetic character stripping via regexp_replace, applied to the clean_text column, consistent with the balanced model's training preprocessing.
Output path: /Model_3_million_balanced_and_dataset_3_million_full/results
Objective: Test the balanced multiclass model in a streaming context on the full three-class dataset, including Neutral tweets.

Experiment 4 — 3M Unbalanced Model on 3M tweets Stream

Model: Best unbalanced multiclass model trained on the 3M dataset with class weights and combined unigram-bigram features (Unbalanced V2, Accuracy 79.15%, F1-Score 79.24%), loaded from /models/twitter3m_model_v2.
Data stream: Full Twitter 3M dataset, streamed via Producer 4.
Preprocessing: Complete text normalization pipeline (lowercase, URL/mention/hashtag removal, non-alphabetic filtering, trim) applied to the tweet column, identical to the training preprocessing of Unbalanced V2.
Output path: /Model_3_million_unbalanced_and_dataset_3_million_full/results
Objective: Evaluate the strongest multiclass model in a real-time setting on the same data distribution it was trained on.

Experiment 5 — 3M Binary Model on Multiple Streams

The best binary model trained on the Twitter 3M dataset (Binary V1, Logistic Regression, num-Features $=2^{18}$, Accuracy 78.39%, AUC 0.850), saved at /models/twitter_binary_model, was deployed in two sub-experiments:

- Experiment 5a — 3M binary stream: The model was applied to the Twitter 3M binary dataset (Positive and Negative tweets only), streamed via Producer 6. The preprocessing pipeline (lowercase, URL/mention/hashtag removal, non-alphabetic filtering, trim, empty tweet filtering) was applied consistently with the model's training conditions. Output saved to /Model_3_million_binary_and_dataset_3_million_binary/results.

- Experiment 5b — Sentiment140 stream: The same 3M binary model was applied to the Sentiment140 stream, streamed via Producer 5. Since Sentiment140 is also a binary dataset (Positive/Negative only), this experiment tests the cross-dataset generalization of the 3M binary model on tweets from a different time period and stylistic origin. The same full preprocessing pipeline was applied. Output saved to /Model_3_million_binary_and_dataset_1.6_million/results.

For both sub-experiments, the same full preprocessing pipeline was applied: lowercase conversion, URL/mention/hashtag removal, non-alphabetic filtering, trimming, and empty tweet filtering.

Summary of Streaming Experiments

Table III.4 summarizes all streaming experiments conducted, detailing the model used, the input data stream, and the HDFS output path.

Table III.4: Summary of streaming experiments

Exp.	Model	Input stream	Output path (HDFS)
1	1.6M model (LR)	Sentiment140 (1.6M)	/Model_1.6M_and_1.6M/
2	1.6M model (LR)	3M binary dataset	/Model_1.6M_and_3M_binary/
3	3M balanced (LR)	3M full dataset	/Model_3M_balanced_3M_full/
4	3M unbalanced (LR)	3M full dataset	/Model_3M_unbalanced_3M_full/
5a	3M binary (LR)	3M binary dataset	/Model_3M_binary_3M_binary/
5b	3M binary (LR)	Sentiment140 (1.6M)	/Model_3M_binary_1.6M/

All streaming results were written to HDFS in CSV format with append output mode and dedicated checkpoint directories to ensure fault tolerance and exactly-once processing semantics. Each output record contains the processed tweet text, the predicted sentiment label (0 = Negative, 1 = Positive, or 2 = Neutral for multiclass models), and the processing_time timestamp indicating when the prediction was made.

III.5 Performance Evaluation

III.5.1 Evaluation Metrics

The following metrics were used to assess model performance:

Accuracy measures the proportion of correctly classified instances over the total number of instances:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{III.1}$$

F1-Score is the harmonic mean of Precision and Recall, providing a balanced measure particularly useful under class imbalance. For multiclass problems, the macro-averaged F1-Score is used:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{III.2}$$

Precision measures the proportion of positive predictions that are actually correct:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{III.3}$$

AUC (Area Under the ROC Curve) measures the ability of a binary classifier to distinguish between classes across all classification thresholds, independently of any specific decision boundary. A value of 1.0 indicates a perfect classifier, while 0.5 corresponds to random prediction.

III.5.2 Evaluation on Sentiment140 Dataset

Five models were evaluated on the Sentiment140 binary classification task (Positive vs. Negative). The dataset was split into 80% training and 20% test sets (approximately 1,280,000 and 320,000 samples respectively) with a fixed random seed of 42. Table III.5 reports the performance of all five models.

Table III.5: Model performance on Sentiment140 dataset (binary classification)

Model	Pipeline	Accuracy	F1-Score	AUC
Baseline	LR · HashingTF (10K)	77.44%	77.44%	0.847
V1	LR · HashingTF (20K) + IDF	76.51%	76.50%	0.839
V2	LR · Bigrams (30K) + IDF	65.24%	65.20%	0.718
V3	Naïve Bayes · HashingTF (20K) + IDF	74.50%	74.50%	0.499
V4	LinearSVC · HashingTF (25K) + IDF	76.56%	76.54%	0.839

The confusion matrix of the best-performing Baseline model on the Sentiment140 test set is shown in Table III.6, illustrating the distribution of correct and incorrect predictions across both sentiment classes.

Table III.6: Confusion matrix — Baseline model on Sentiment140 test set

	Predicted Negative	Predicted Positive
Actual Negative	121,434	38,799
Actual Positive	33,339	126,219

The Baseline model — a minimal pipeline of Tokenizer, HashingTF, and Logistic Regression without stop-word removal — achieved the highest performance on this dataset (Accuracy 77.44%, AUC 0.847). This counterintuitive result suggests that on the Sentiment140 corpus, stop-words carry discriminative sentiment signals (e.g., negations, intensifiers) that are lost when removed. Model V1, which adds stop-word removal and IDF weighting, drops slightly to 76.51%, confirming this observation. Model V2 suffers the sharpest degradation (65.24%),

as relying solely on bigram features without unigrams dramatically increases feature sparsity on a dataset of this size. Model V3 (Naïve Bayes) achieves reasonable accuracy (74.50%) but produces a near-random AUC (0.499), revealing that its posterior probability estimates are unreliable despite producing correct hard labels in many cases. Models V1 and V4 (LinearSVC) achieve comparable results (76.51% vs. 76.56%), confirming that linear classifiers perform equivalently on this task when given similar feature representations.

III.5.3 Evaluation on Twitter 3M Dataset

Scenario 1 — Balanced Multiclass Classification

Three models were evaluated on a balanced subset of the Twitter 3M dataset, where Negative and Positive classes were downsampled to match the Neutral class size (∼10,000 tweets per class, ∼30,000 total). Results are presented in Table III.7.

Table III.7: Model performance — balanced multiclass classification (Twitter 3M)

Model	Pipeline	Accuracy	F1-Score	Precision
B-V1	LR · HashingTF (30K)	67.90%	68.23%	68.68%
B-V2	LR · Bigrams (100K) + IDF	50.22%	50.65%	51.91%
B-V3	Word2Vec (200d) + Random Forest	67.82%	67.71%	67.83%

The moderate accuracy range of 67–68% achieved by the two best models (B-V1 and B-V3) reflects the combined effect of two compounding difficulties: the inherent ambiguity of three-class sentiment discrimination, and the severe data starvation induced by downsampling. Reducing the training set to approximately 24,000 samples — a 99% reduction from the original 3 million — fundamentally limits the statistical reliability of TF-IDF term weights, as rare but informative n-grams receive unstable IDF scores when document frequency counts are based on such a small corpus. This phenomenon explains why B-V1 (Logistic Regression with HashingTF) marginally outperforms B-V3 (Word2Vec + Random Forest): in a low-data regime, the compact and statistically stable unigram vocabulary of TF-IDF is more data-efficient than the dense 200-dimensional Word2Vec embeddings, which require sufficient co-occurrence statistics to produce meaningful representations. B-V2 collapsed to near-random performance (50.22%), as the exponential growth of the bigram feature space combined with the small training set produced a severely sparse and statistically unreliable feature matrix, causing the optimizer to converge to a near-constant prediction. This result directly illustrates the data starvation effect: the larger the feature space, the more training samples are required to estimate reliable weights — a constraint that downsampling systematically violates.

Scenario 2 — Unbalanced Multiclass Classification with Class Weights

Two models were trained on the full Twitter 3M dataset using class weights computed as $w_c = \frac{N}{k \cdot n_c}$, where N is the total sample count, $k = 3$ is the number of classes, and n_c is the count for class c. Results are presented in Table III.8.

Table III.8: Model performance — unbalanced multiclass classification with class weights (Twitter 3M)

Model	Pipeline	Accuracy	F1-Score	Precision
U-V1	LR · CV (30K) + IDF + weights	77.17%	77.29%	77.62%
U-V2	LR · Unigrams + Bigrams + weights	79.15%	79.24%	79.49%

The class-weighted training strategy on the full dataset significantly outperforms the balanced downsampling approach, with U-V2 achieving 79.15% accuracy and 79.24% F1-Score — an improvement of over 11 percentage points compared to the best balanced model (B-V1: 67.90%). This gap is not attributable to architectural differences, since both models use Logistic Regression with TF-IDF features. The improvement stems exclusively from the training strategy: rather than discarding majority-class samples, the class-weighted approach preserves the full 3 million samples while compensating for imbalance through the loss function. The weight assigned to the Neutral class ($w_c = \frac{N}{k \cdot n_c}$, where N is the total sample count, $k = 3$, and n_c is the class count) amplifies its gradient contribution during optimization proportionally to its underrepresentation, allowing the model to learn meaningful Neutral-class boundaries despite the 150:1 class ratio. This confirms that preserving training data volume through loss reweighting is systematically more effective than reducing it through downsampling in large-scale imbalanced classification. U-V2 further improves over U-V1 by approximately 2 percentage points through the addition of bigram features via VectorAssembler, confirming that short-range syntactic co-occurrences provide complementary discriminative signal when sufficient training data ensures stable bigram frequency estimates. U-V2 is the best-performing model across all multiclass scenarios.

Scenario 3 — Binary Classification (Positive vs. Negative)

Four models were evaluated on the Twitter 3M binary dataset after removing Neutral tweets. Both Accuracy and AUC are reported, as AUC provides a threshold-independent measure of discriminative power for binary tasks. Results are presented in Table III.9.

Table III.9: Model performance — binary classification on Twitter 3M dataset (Positive vs. Negative)

Model	Pipeline	Accuracy	F1-Score	AUC
Bin-V1	LR · HashingTF (2^{18}) + IDF	78.39%	78.39%	0.850
Bin-V2	LR + L2 · HashingTF (2^{17}) + IDF	78.06%	78.06%	0.854
Bin-V3	Naïve Bayes · HashingTF (2^{17}) + IDF	75.75%	75.74%	0.522
Bin-V4	LR + ElasticNet · HashingTF (2^{17}) + IDF	77.07%	77.04%	0.847

Bin-V1 achieves the highest accuracy (78.39%) while Bin-V2 achieves the highest AUC (0.854), indicating that L2 regularization improves probability calibration and ranking ability even when it marginally reduces raw accuracy. Bin-V4 with ElasticNet regularization (regParam=0.01, elasticNetParam=0.1) performs comparably (77.07%, AUC 0.847), with the combined L1-L2 penalty yielding slightly lower performance than pure L2. As consistently observed across all tasks, Bin-V3 (Naïve Bayes) produces a near-random AUC (0.522) despite reasonable accuracy (75.75%), confirming that its posterior probabilities are unreliable for ranking purposes regardless of the dataset or task setting.

III.5.4 Comparative Analysis

Table III.10 consolidates the performance of all trained models across both datasets and all scenarios, enabling a global comparison.

Table III.10: Consolidated performance comparison of all trained models

Dataset	Scenario	Model	Pipeline	Accuracy	F1	AUC / Prec.
Sentiment140 (1.6M)	Binary	Base.	LR · HashingTF (10K)	77.44%	77.44%	0.847
		V1	LR · HashingTF (20K)	76.51%	76.50%	0.839
		V2	LR · Bigrams (30K)	65.24%	65.20%	0.718
		V3	Naïve Bayes (20K)	74.50%	74.50%	0.499
		V4	LinearSVC (25K)	76.56%	76.54%	0.839
Twitter 3M	Balanced (3-class)	B-V1	LR · HashingTF (30K)	67.90%	68.23%	Prec: 68.68%
		B-V2	LR · Bigrams (100K)	50.22%	50.65%	Prec: 51.91%
		B-V3	Word2Vec + RF	67.82%	67.71%	Prec: 67.83%
	Unbalanced (3-class)	U-V1	LR · CV (30K) + weights	77.17%	77.29%	Prec: 77.62%
		U-V2	LR · Uni+Bi + weights	79.15%	79.24%	Prec: 79.49%
	Binary	Bin-V1	LR · HashingTF (2^{18})	78.39%	78.39%	0.850
		Bin-V2	LR + L2 · HTF (2^{17})	78.06%	78.06%	0.854
		Bin-V3	Naïve Bayes (2^{17})	75.75%	75.74%	0.522
		Bin-V4	LR + ElasticNet (2^{17})	77.07%	77.04%	0.847

Four key observations emerge from the consolidated results:

Training strategy dominates over feature engineering. The transition from balanced downsampling (∼68% accuracy) to class-weighted training on the full dataset (∼79% accuracy)

yields an improvement of over 11 percentage points on the Twitter 3M multiclass task — far exceeding any gain attributable to architectural choices. Since both B-V1 and U-V1 use identical Logistic Regression pipelines with TF-IDF features, the performance gap is entirely explained by the training strategy: class weighting preserves the full data volume while correcting the distributional skew through the loss function, whereas downsampling eliminates 99% of available training signal to achieve artificial balance.

Stop-word removal is detrimental for sentiment-bearing short text. The Baseline model, which omits stop-word removal, outperforms all other models on Sentiment140 (77.44% vs. 76.51% for V1). This result is explained by the linguistic structure of sentiment in tweets: function words such as not, never, very, and so are critical carriers of polarity. The phrase "I do not like this" becomes "like" after standard stop-word removal, inverting its sentiment signal entirely. In tweet-level text averaging 11–15 words, such function words represent a disproportionately large fraction of the total lexical signal compared to longer documents where they are genuinely redundant. This finding aligns with observations in the literature [26] cautioning against applying standard stop-word lists to short social media text without domain-specific adaptation.

Bigrams complement but cannot replace unigrams. On Sentiment140, bigram-only features (V2) degraded accuracy by 12 percentage points relative to the Baseline, while on the Twitter 3M unbalanced task, combining unigrams and bigrams (U-V2) improved accuracy by 2 percentage points over unigrams alone (U-V1). This asymmetry is explained by feature sparsity: with a vocabulary of V unigrams, the theoretical bigram space grows as $O(V^2)$, and each bigram occurs far less frequently than its constituent unigrams. On datasets below several hundred thousand samples, most bigrams appear fewer than 5 times, making their TF-IDF weights statistically unreliable. On 3 million samples, sufficient bigram frequency counts exist to produce stable weights, and bigrams then contribute complementary syntactic context. Bigram features are therefore beneficial as a supplement to unigrams when training data is abundant, but harmful as a replacement when data is scarce.

Naïve Bayes produces unreliable probability estimates despite acceptable accuracy. Across all binary tasks, Naïve Bayes achieved near-random AUC values (0.499 on Sentiment140, 0.522 on Twitter 3M binary) despite moderate accuracy (74.50% and 75.75% respectively). This dissociation between accuracy and AUC is a known failure mode of Naïve Bayes on text data [31]: the conditional independence assumption is systematically violated in natural language, where word co-occurrences are highly correlated. This violation causes posterior probability estimates $P(y \mid x)$ to be pushed toward extreme values (near 0 or 1) regardless of the true uncertainty, producing correct hard labels through frequency-dominated boundaries while generating miscalibrated confidence scores unsuitable for ranking or threshold-sensitive applications.

HashingTF is the principled choice for distributed Big Data sentiment analysis. The selection of HashingTF over Word2Vec or contextual embeddings reflects a deliberate architectural decision aligned with the scalability requirements of this work. HashingTF maps tokens to a fixed-size feature vector in $O(1)$ per token with no vocabulary construction pass, producing sparse vectors natively supported by Spark MLlib's distributed linear algebra. Word2Vec requires a full corpus scan and stores dense vectors consuming significantly more memory, while providing no measurable accuracy gain in the low-data balanced regime (B-V3: 67.82% vs. B-V1: 67.90%). Contextual embeddings such as BERT [32] introduce quadratic attention complexity $O(n^2)$ and require GPU infrastructure incompatible with the distributed CPU-based Spark environment. The experimental results confirm that HashingTF-based Logistic Regression achieves competitive accuracy at 3-million-tweet scale while maintaining the horizontal scalability that defines the Big Data paradigm.

III.5.5 Summary

The performance evaluation demonstrates that the distributed Spark MLlib pipeline successfully handles datasets of up to 3 million tweets while maintaining competitive classification performance. The best binary model on Sentiment140 is the Baseline Logistic Regression (Accuracy 77.44%, AUC 0.847). The best binary model on Twitter 3M is Binary V1 (Accuracy 78.39%, AUC 0.850). The best multiclass model overall is the Unbalanced V2 pipeline combining unigram and bigram TF-IDF features with class-weighted Logistic Regression (Accuracy 79.15%, F1-Score 79.24%, Precision 79.49%). These results confirm that linear classifiers combined with TF-IDF-based sparse representations remain strong and computationally efficient baselines for large-scale social media sentiment analysis, particularly when class imbalance is addressed through weighting rather than data reduction.

III.5.6 Comparison with State-of-the-Art Models

While the models implemented in this work are designed for scalable distributed processing of large-scale Twitter data, it is important to contextualize their performance against state-of-the-art approaches reported in the literature, particularly those based on the Transformer architecture [33].

The Transformer Architecture in NLP

The Transformer architecture, introduced by Vaswani et al. (2017), revolutionized natural language processing through its self-attention mechanism, which enables models to capture long-range contextual dependencies without relying on recurrent structures. Built upon this foundation, encoder-based models such as BERT [32] and its variants have consistently achieved state-of-the-art results across a wide range of NLP benchmarks, including sentiment analysis on social media data. BERTweet [34], a BERT-based model pre-trained specifically on English tweets, further demonstrated the effectiveness of domain-specific pre-training for Twitter analytics. More recently, hybrid approaches combining Transformer-based representations with classical classifiers such as XGBoost have been explored, with [35] reporting an overall accuracy of 90% on social media sentiment classification, underscoring the growing performance ceiling of Transformer-based pipelines.

Comparative Results

Table III.11 presents a direct comparison between the best-performing models from this work and representative Transformer-based results reported in the literature on binary sentiment classification tasks using datasets of comparable nature.

Table III.11: Comparison with state-of-the-art Transformer-based models on binary sentiment classification

Model	Type	Accuracy	F1-Score	Infrastructure
Our Baseline LR (1.6M)	TF-IDF + LR	77.44%	77.44%	Distributed Spark, no GPU
Our Best Binary (3M)	TF-IDF + LR	78.39%	78.39%	Distributed Spark, no GPU
BERT-base (1.6M) [36]	Transformer	83.7%	83.4%	Single node, GPU required
DistilBERT (1.6M) [36]	Transformer	85.0%	84.4%	Single node, GPU required
BERT + XGBoost (1.6M) [35]	Transformer + Ensemble	90.0%	90.0%	Single node, GPU required

Analysis

The results in Table III.11 reveal a performance gap of approximately 6–7 percentage points between our best TF-IDF Logistic Regression pipeline and DistilBERT, and approximately 11–12 percentage points relative to the BERT + XGBoost hybrid approach. This gap is expected and can be attributed to three fundamental differences.

First, contextual representation: Transformer models learn deep bidirectional contextual embeddings through pre-training on large corpora, enabling them to capture semantic nuances, negations, and sarcasm that TF-IDF sparse representations inherently cannot encode. TF-IDF treats each token independently, losing word order and contextual meaning.

Second, computational requirements: BERT-based models require GPU-accelerated fine-tuning and operate on fixed-length sequences (typically 512 tokens). They are not designed for distributed horizontal scaling. In contrast, our Spark MLlib pipelines processed datasets of up to 3 million tweets in a fully distributed environment on a single-node cluster without any GPU resources, maintaining competitive accuracy while ensuring scalability that Transformer models cannot natively provide.

Third, scalability vs. accuracy trade-off: The core contribution of this work lies not in maximizing accuracy on a fixed test set, but in demonstrating that meaningful sentiment analysis can be performed at Big Data scale using distributed computing frameworks. As noted by Goonetilleke et al. [24], scalability and real-time processing capability are primary requirements in large-scale Twitter analytics, where processing latency and infrastructure cost are as critical as predictive performance.

Bridging this performance gap while preserving scalability represents a concrete direction for future work, particularly through frameworks such as Spark NLP [37], which provides distributed Transformer inference natively compatible with Apache Spark, enabling BERT-level accuracy at Big Data scale.

III.6 Conclusion

This chapter presented the design and implementation of a scalable sentiment analysis system integrating Apache Hadoop HDFS, Apache Spark, Apache Kafka, and Spark MLlib into a unified pipeline supporting both batch and real-time stream processing of Twitter data.

Two large-scale datasets were used: the Sentiment140 corpus (1.6M tweets) and the Twitter 3M dataset, both preprocessed through rigorous PySpark pipelines and analyzed via Power BI dashboards. A total of nineteen machine learning models were trained across binary and multiclass classification scenarios using TF-IDF-based representations combined with Logistic Regression, Naïve Bayes, LinearSVC, and Word2Vec with Random Forest.

Key findings include: stop-word removal is detrimental for short tweet-level text; class-weighted training on the full dataset outperforms balanced downsampling by over 11 percentage points; combining unigrams and bigrams yields the best multiclass performance (Accuracy 79.15%, F1 79.24%); and Naïve Bayes consistently produces unreliable probability estimates despite moderate accuracy. The stream processing phase successfully operationalized all batch-trained models in real-time via Kafka and Spark Structured Streaming.

Overall, the results confirm that distributed TF-IDF pipelines with linear classifiers offer a strong, scalable baseline for large-scale social media sentiment analysis.

GENERAL CONCLUSION

The present thesis set out to investigate the role of Big Data analytics in understanding and analyzing social media data, with Twitter serving as the central case study. Driven by the rapid expansion of digital communication platforms and the unprecedented volume of user-generated content they produce, this work explored how modern distributed computing frameworks and advanced machine learning techniques can be combined to extract meaningful insights from massive, heterogeneous textual datasets.

The study was organized along three main axes. The first axis established the theoretical foundations of Big Data by reviewing existing definitions, classifying the core and complementary characteristics of Big Data systems, and surveying the principal tools and frameworks that constitute the Big Data analytics ecosystem. This conceptual groundwork demonstrated that Big Data is not merely a question of volume: it encompasses velocity, variety, veracity, and value, all of which impose distinct challenges that cannot be addressed by conventional database systems.

The second axis examined Twitter as both a microblogging platform and a large-scale data source. The analysis covered the structural properties of tweets, the role of Twitter APIs in enabling systematic data acquisition, and the spectrum of analytical approaches that have been applied to Twitter data in the literature. Two broad categories of methods were identified: content-oriented approaches, including topic modeling, sentiment analysis, and similarity-based trend detection, and network-oriented approaches, which model information diffusion and community structure through graph-theoretic representations of user interactions. The chapter established that the richness of Twitter data lies precisely in its dual nature: the textual content of tweets and the relational structure of the interactions they generate are complementary dimensions that, when analyzed jointly, yield a deeper and more nuanced understanding of digital behavior.

The third axis constituted the practical and experimental core of the thesis. A scalable end-to-end sentiment analysis system was designed and implemented using Apache Hadoop HDFS for distributed storage, Apache Spark and PySpark for batch and streaming computation, Apache Kafka for real-time data ingestion, and Spark MLlib for machine learning. Two large-scale datasets were used: the Sentiment140 corpus containing 1.6 million binary-labeled tweets, and a Twitter dataset comprising 3 million tweets annotated across three sentiment classes. Across both datasets, a total of fourteen machine learning pipelines were trained and evaluated, covering binary and multiclass classification scenarios using Logistic Regression, Naïve Bayes, Linear SVM, Word2Vec with Random Forest, and various TF-IDF feature configurations.

The experimental results yielded several important findings. On the Sentiment140 dataset, the

best-performing model was a minimal Logistic Regression pipeline without stop-word removal, achieving an accuracy of 77.44% and an AUC of 0.847. This result reveals a dataset-specific characteristic: in short tweet-level text, common words such as negations and intensifiers carry discriminative sentiment signals that are lost when standard stop-word removal is applied indiscriminately. On the Twitter 3M dataset, the class-weighted unbalanced training strategy consistently and substantially outperformed balanced downsampling, with the best model combining unigram and bigram TF-IDF features under a class-weighted Logistic Regression achieving an accuracy of 79.15% and an F1-Score of 79.24%. This result confirmed that preserving the full training data volume while compensating for class imbalance through weighting is more effective than discarding majority-class samples. Naïve Bayes consistently produced near-random AUC values across all tasks despite achieving moderate accuracy scores, revealing that its posterior probability estimates are poorly calibrated for tweet-level text regardless of dataset or classification setting.

The real-time component of the system, implemented through a Kafka producer and Spark Structured Streaming consumer pipeline, successfully demonstrated the operationalization of batch-trained models in a live inference scenario. Six streaming experiments were conducted, combining different pre-trained models with different input data streams to evaluate both in-distribution and cross-dataset generalization performance. The architecture ensured preprocessing consistency between training and inference phases, a critical requirement for maintaining the integrity of the feature space seen by the deployed models.

Taken together, the results of this study confirm the central hypothesis that motivated the research: Big Data analytics techniques are capable of extracting valuable and actionable knowledge from large-scale social media data. The integration of content-based analysis with distributed computing infrastructure enables the identification of sentiment trends, behavioral patterns, and interaction dynamics at a scale and speed that would be entirely infeasible using conventional analytical methods. At the same time, the study highlighted important challenges that remain relevant for practitioners and researchers in this field: class imbalance in real-world datasets requires careful handling rather than naive downsampling; feature engineering choices must be adapted to both the nature of the text and the volume of available data; and the quality of preprocessing pipelines has a direct and measurable impact on classification performance.

Several limitations of the present work should be acknowledged. The unavailability of live Twitter API access required the use of pre-collected Kaggle datasets, which restricts the temporal scope and topical diversity of the data. The sentiment labels in both datasets were assigned automatically through distant supervision, introducing label noise that may partially account for the performance ceiling observed across all models. Furthermore, the present study focused exclusively on textual content and did not incorporate multimedia data or user profile metadata, which could substantially enrich the analytical framework.

These limitations point toward several promising directions for future research. First, the integration of transformer-based language models such as BERT or its Twitter-specific variant BERTweet could significantly improve classification performance, particularly for nuanced sentiment categories and informal language structures. Second, extending the current framework to incorporate network-based analysis — modeling retweet graphs, follower communities, and information diffusion cascades — would enable a more holistic understanding of how opinions form and spread across the platform. Third, the application of the proposed methodology to other social media platforms such as Facebook, Reddit, or LinkedIn would allow researchers to assess the generalizability of the analytical framework beyond Twitter . Finally, addressing data quality challenges through improved annotation strategies, active learning, or crowdsourced labeling could yield higher-quality training corpora and ultimately more reliable analytical results.

In conclusion, this thesis demonstrates that Big Data analytics represents a powerful and effective framework for studying social media platforms at scale. By combining distributed storage, stream processing, and machine learning within a unified and reproducible pipeline, it is possible to move from raw, noisy social media data to structured, interpretable knowledge about user sentiment, behavioral trends, and interaction patterns. The findings of this work contribute to the broader field of computational social science and provide a concrete methodological foundation for researchers and practitioners seeking to leverage Big Data technologies for the analysis of digital communication in an increasingly connected world.

BIBLIOGRAPHY

[1] E. T. K. Sang and A. Van den Bosch, "Dealing with big data: The case of twitter," Computational Linguistics in the Netherlands Journal, vol. 3, pp. 121–134, 2013.

[2] M. Felt, "Social media and the social sciences: How researchers employ big data analytics," Big data & society, vol. 3, no. 1, p. 2053951716645828, 2016.

[3] S. Biffignandi, A. Bianchi, and C. Salvatore, "Can big data provide good quality statistics? a case study on sentiment analysis on twitter data," in Proceedings of the International Total Survey Error Workshop Duke Initiative Survey Methodol (ITSEW-DISM), Durham, NC, USA, vol. 3, 2018.

[4] H. Htet and Y. Y. Myint, Social media (Twitter) Data analysis using maximum entropy classifier on big data processing framework (Case study: Analysis of health condition, education status, states of business). PhD thesis, MERAL Portal, 2018.

[5] A. A. Sayed, M. M. Abdallah, A. M. Zaki, and A. A. Ahmed, "Big data analysis using a metaheuristic algorithm: Twitter as case study," in 2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE), pp. 20–26, IEEE, 2020.

[6] S. B. Abkenar, M. H. Kashani, E. Mahdipour, and S. M. Jameii, "Big data analytics meets social media: A systematic review of techniques, open issues, and future directions," Telematics and informatics, vol. 57, p. 101517, 2021.

[7] Y. Qin, "Analysis and application of big data in social media marketing," International Journal of Global Economics and Management, vol. 2, no. 2, pp. 266–272, 2024.

[8] S. Sagiroglu and D. Sinanc, "Big data: A review," in 2013 international conference on collaboration technologies and systems (CTS), pp. 42–47, IEEE, 2013.

[9] I. Emmanuel and C. Stanier, "Defining big data," in Proceedings of the International Conference on big data and advanced Wireless technologies, pp. 1–6, 2016.

[10] A. De Mauro, M. Greco, and M. Grimaldi, "A formal definition of big data based on its essential features," Library review, vol. 65, no. 3, pp. 122–135, 2016.

[11] P. Géczy, "Big data characteristics," The Macrotheme Review, vol. 3, no. 6, pp. 94–104, 2014.

[12] R. Kitchin and G. McArdle, "What makes big data, big data? exploring the ontological characteristics of 26 datasets," Big data & society, vol. 3, no. 1, p. 2053951716631130, 2016.

[13] G. Kapil, A. Agrawal, and R. Khan, "A study of big data characteristics," in 2016 international conference on communication and electronics systems (ICCES), pp. 1–4, IEEE, 2016.

[14] IBM, "What is big data analytics?," 04 2024. Accessed: 2026-02-12.

[15] AltexSoft Editorial Team, "Big data analytics: How it works, tools, and real-life applications," 05 2021. Accessed: 2026-02-12.

[16] B. Berisha, E. Mëziu, and I. Shabani, "Big data analytics in cloud computing: an overview," Journal of Cloud Computing, vol. 11, no. 1, p. 24, 2022.

[17] A. Bruns and S. Stieglitz, "Twitter data: what do they represent?," It-Information Technology, vol. 56, no. 5, pp. 240–245, 2014.

[18] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?," in Proceedings of the 19th international conference on World wide web, pp. 591–600, 2010.

[19] S. Kumar, F. Morstatter, and H. Liu, Twitter data analytics. Springer, 2014.

[20] T. Bucher, "Objects of intense feeling: The case of the twitter api," Computational Culture, no. 3, 2013.

[21] K. Makice, Twitter API: Up and running: Learn how to build applications with the Twitter API. " O'Reilly Media, Inc.", 2009.

[22] I. Dongo, Y. Cadinale, A. Aguilera, F. Martínez, Y. Quintero, and S. Barrios, "Web scraping versus twitter api: a comparison for a credibility analysis," in Proceedings of the 22nd International conference on information integration and web-based applications & services, pp. 263–273, 2020.

[23] J. v. Dijck, "Tracing twitter: The rise of a microblogging platform," International Journal of Media & Cultural Politics, vol. 7, no. 3, pp. 333–348, 2011.

[24] O. Goonetilleke, T. Sellis, X. Zhang, and S. Sathe, "Twitter analytics: a big data management perspective," ACM SIGKDD Explorations Newsletter, vol. 16, no. 1, pp. 11–20, 2014.

[25] H. Shirdastian, M. Laroche, and M.-O. Richard, "Using big data analytics to study brand authenticity sentiments: The case of starbucks on twitter," International Journal of Information Management, vol. 48, pp. 291–307, 2019.

[26] A. P. Rodrigues, R. Fernandes, A. Bhandary, A. C. Shenoy, A. Shetty, and M. Anisha, "Real-time twitter trend analysis using big data analytics and machine learning techniques," Wireless Communications and Mobile Computing, vol. 2021, no. 1, p. 3920325, 2021.

[27] European Parliament and Council of the European Union, "Regulation (EU) 2016/679 of the european parliament and of the council (general data protection regulation)," Tech. Rep. L 119, Official Journal of the European Union, 2016. OJ L 119, 4.5.2016.

[28] R. Abraham, J. Schneider, and J. vom Brocke, "Data governance: A conceptual framework, structured review, and research agenda," International Journal of Information Management, vol. 49, pp. 424–438, 2019.

[29] High-Level Expert Group on Artificial Intelligence, "Ethics guidelines for trustworthy AI," 2019. Brussels, Belgium.

[30] A. Olteanu et al., "Social data: Biases, methodological pitfalls, and ethical boundaries," Frontiers in Big Data, vol. 2, p. 13, 2019.

[31] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge University Press, 2008.

[32] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proceedings of NAACL-HLT, pp. 4171–4186, 2019.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in neural information processing systems, vol. 30, 2017.

[34] D. Q. Nguyen, T. Vu, and A. T. Nguyen, "BERTweet: A pre-trained language model for English tweets," in Proceedings of EMNLP: System Demonstrations, pp. 9–14, 2020.

[35] T. Batiuk and D. Dosyn, "Intellectual analysis of textual data in social networks using bert and xgboost," 2025.

[36] M. S. Asyaky, M. Al-Husaini, and H. H. Lukmana, "Sentiment analysis on short social media texts using distilbert," Journal of Computer Networks, Architecture and High Performance Computing, vol. 7, no. 2, pp. 524–533, 2025.

[37] V. Kocaman and D. Talby, "Spark NLP: Natural language understanding at scale," in Software Impacts, vol. 8, p. 100058, 2021.