Real-Time Event Detection in Social Media

Bonda Naveen Kumar
Data Science and Artificial Intelligence
IIIT Naya Raipur, INDIA
bonda22102@iiitnr.edu.in

Pediredla Suman
Data Science and Artificial Intelligence
IIIT Naya Raipur, INDIA
pediredla22102@iiitnr.edu.in

Sambangi Chaitanya
Computer Science
IIIT Naya Raipur,INDIA
sambangi22100@iiitnr.edu.in

Mallikharjuna Rao K
Assistant Professor, DSAI
IIIT Naya Raipur, INDIA
mallikharjuna@iiitnr.edu.in

***Abstract*—The rapid development of social media sites has generated huge volumes of unstructured, high-speed data, making real-time event detection a challenging but essential task. Conventional systems are not scalable with such data and tend to miss emerging trends. This project suggests a scalable and fault-tolerant architecture for real-time event detection with Apache Kafka, Hadoop, and Apache Mahout.A Kafka producer outputs social media streams, and a Kafka consumer consumes and writes information to the Hadoop Distributed File System (HDFS). Apache Mahout is used to carry out topic modeling with methods such as Latent Dirichlet Allocation (LDA), from which recent events are obtained from stored information The system effectively extracts events such as natural disasters, social movements, and public announcements by recognizing keyword bursts and thematic clusters.Supported by a multi-node Hadoop cluster, the architecture supports efficient processing of high-throughput data and showcases the practical use of big data technologies in social analytics, trend identification, and crisis management.**

***Index Terms*—Big Data, Apache Kafka, Hadoop, Apache Mahout, Real-time Processing, Social Media Analytics, Event Detection, Topic Modeling, HDFS**

I. Introduction

In today's networked, digital society, social media streams have become important sources of up-to-the-minute information, voicing shifting public opinions, emerging news stories, and breaking events. Social media streams generate massive quantities of user-created content by the second and thus generate streams of unstructured data that pose gigantic challenges for event detection systems. Detection of meaningful patterns or real-world occurrences from this data in a timely and scalable manner is of utmost importance in use cases such as crisis management, market surveillance, political trend forecasting, and public opinion monitoring.

However, traditional data processing architectures do not keep up with the speed, size, and heterogeneity of social media data. Their inability to scale horizontally and process high-throughput, unstructured text data in real-time restricts their use in modern-day analytics applications. This requirement demands deploying robust big data architectures that manage streaming consumption of data, distributed storage, and machine learning-driven analysis efficiently.

TIn support of these requirements, this project introduces a real-time event discovery pipeline that relies on Apache Kafka for continuous data ingestion, Hadoop Distributed File System (HDFS) for fault-tolerant distributed storage, and Apache Mahout for bulk topic modeling. Using simulated real-time streams constructed from a Kafka producer and in conjunction with consumer-side ingestion into HDFS, the system offers persistent high-rate data streams. This cached information is then analyzed by applying topic model algorithms such as Latent Dirichlet Allocation (LDA) to find topical trends and evolution in enormous sets of social media messages.

The architecture design is not only aimed at enhancing detection efficiency and topic relevance but also at enabling the system to be implemented in real-world settings with real-time insight needs. The utility of such a system is brought out in high-stakes domains—natural disaster alerts, early outbreak detection, and socio-political unrest—where the capacity to distill timely actionable information from extensive social chatter can enable faster decision-making and response.

II. Literature Review

The emergence of social media sites has led to an immediate epidemic of real-time, user-generated data, which necessitates the creation of scalable and efficient event detection mechanisms. Traditional event detection methods cannot handle such high velocity, volume, and unstructured data.New work has indicated the promise provided by Big Data technologies and machine learning in addressing these challenges. Li et al. (2022) presented a comprehensive survey of event detection methods using Twitter data sets. The data sets are diverse and easily accessible but differ in rendering it difficult to perform reasonable comparative tests between models This again advocates for the need for standardization in benchmarking real-time detection systems.

Karaman et al. (2017) also demonstrated, in a further study, the utility of using a 5W1H analytical framework (Who, What, When, Where, Why, How) to glean structured information from unstructured Twitter streams. The method was faster and more relevant than conventional news sources, highlighting the potential of social media as an in-real-time sensor.

TABLE I
LITERATURE REVIEW

Ref No	Title	Dataset Used	Remarks
[6]	Event Detection from Social Media Stream: Methods	Twitter event datasets	Publicly available and diverse datasets, but lack consistency, making it difficult to compare performance fairly across different event detection approaches and studies.
[2]	Event detection from social media: 5W1H analysis on big data	Twitter stream data	Real-time tweets are used as input, enabling faster event detection and 5W1H analysis, outperforming conventional news sources in speed.
[8]	A Hybrid Hadoop-Based Sentiment Analysis Classifier for Tweets	COVID-19_Sentiments dataset	The hybrid model combines the CNN and fuzzy C4.5 on Hadoop, achieving great scalability and scalability for sentiment analysis of tweets, especially during the COVID-19 data surges.
[4]	Social Media Analytics through Big Data Using Hadoop Framework	Social media dataset	integrating Hadoop with data warehouses efficient social media analytics and not structured data within academic institutions.

To enhance scalability and efficiency, hybrid architectures fusing deep learning and distributed processing have been proposed. Es-Sabery et al. (2024), for example, proposed a Hadoop-based classifier that used CNN and fuzzy C4.5 algorithms to categorize tweets about the COVID-19 pandemic as either positive or negative. The hybrid model was highly efficient with scalability on large data, showing the ability to combine Big Data frameworks and deep learning. Additionally, Almajed et al. (2023) made a contribution towards coupling Hadoop with organizational data warehouses for decision-making purposes using social media analysis.Their system handled unstructured text rather efficiently, corroborating the contribution of Big Data ecosystems in aiding big-size analysis in educational and organizational settings.

In total, these works highlight the importance of distributed and scalable systems in processing social media streams in real-time. Use of frameworks such as Apache Kafka, Hadoop HDFS, and Apache Mahout can significantly contribute to the speed and precision of event discovery. This body of work provides a good foundation for the proposed real-time event detection system for our study.

III. METHODOLOGY

This is the step-by-step procedure adopted in designing a Big Data-driven real-time event detection system technologies. There are five important phases of the system: dataset preparation, real-time data streaming using Apache Kafka, distributed storage using Hadoop HDFS, topic modeling using Apache Mahout, and final event extraction and trend detection.

A. Dataset and Preprocessing

The information utilized herein within this project is from https://www.kaggle.com/competitions/si650winter11/dataKaggle SI650 Winter 2011 competition, collected first from Opinmind.com. It replicates user-generated content like those who are on social media websites.

- **Training Set:** 7,086 textual messages.
- **Test Set:** 33,052 messages, each consisting of a single sentence.

Every message is like normal short-form social media content, i.e., "Earthquake in City A" or "Concert tonight" at Central Park.

Preprocessing steps included:

- Text normalization (lowercasing, punctuation removal)
- Tokenization (splitting sentences into individual words)
- Stop-word removal (eliminating common, non-informative words)
- Stemming (reducing words to their base forms)

These preprocessing steps ensured cleaner data input for later stages, particularly for topic modeling using Mahout.

B. Real-Time Data Streaming Using Kafka Producer

In the system proposed, Apache Kafka is the actual time data ingestion layer. Kafka is a distributed event streaming platform for low-latency, high-throughput in- streaming data management. In the project, there is a Kafka Producer component was employed to simulate a constant flow of text messages, acting as a substitute for actual real-time tweet streams.

Kafka Producer Architecture and Operation: label=**0)**, left-margin=2em

1) **Record Creation:** A test message (e.g., "Earthquake in City A") is enclosed within a ProducerRecord object. The record contains:
   - **Topic Name:** social-media-stream
   - **Key (Optional):** Not used, allowing Kafka to perform automatic partitioning.
   - **Value:** The actual social media message as a string.
2) **Serialization:** Key and value are serialized by StringSerializer, which transforms them into byte arrays ready for network transfer.
3) **Partitioning:** Kafka employs round-robin partitioning strategy since no master key has been provided. This ensures that messages are spread evenly among the partitions in the topic.

4) **Buffering and Batching:** Buffering and Batching: For a short period Serial- ized messages are teminally internally buffered. Kafka sizes these messages by message size (message.size) or linger.ms time elapsed to enhance network utilization.
5) **Asynchronous Sending:**One thread to send. sends batched the messages to be sent to the Kafka broker It has high throughput and low latency.
6) **Configuration Parameters:** These are the most important param- eters that control the operation of Kafka producer while creating messages. parameters such as linger.ms and batch.size To achieve maximum throughput, define how messages are batched into groups, either by delay or by size. Retries specifies how often the producer will attempt to resend in the event of a failure, while buffer.memory sets the space to hold unsent records. Besides essential properties are timeout.ms (network timeout settings), client.id (for source tracing), compression (for minimize message size), and acks=1 (leader-only acknowledgement for improved performance).
7) **Broker Acknowledgment:** The producer is set with acks=1, i.e., wait for acknowledgement is per- established exclusively by the leader replica of the Kafka broker. This is a compromise between performance and reliabil- abilities.
8) **Properties:**The properties are used to represent high-level operating features of the producer. Round- robin partitioning ensures messages are balanced dis- tributed among subject partitions, while enabling load balancing. The design is high throughput (through batch- ing and async sending), low latency (being non-blocking I/O), and fault tolerance (with retry and acknowledge-) ment parameters). The enhancements these components add to the Kafka producer component most appropriate to real-time data streaming operations like social media event detection

This configuration enabled the system to simulate a bal-next-gen and scalable social media stream, ready for the future real-time analysis..

C. Distributed Data Storage Using Hadoop HDFS

The Kafka Consumer Component (Fig 2) constantly subscribes to the social-media-stream topic and consumes real-time messages from the Kafka producer. The extremely scalable and fault-tolerant Hadoop Distributed File System (HDFS), built for massive volumes of dispersed data, stores these messages. HDFS consists, as shown in Fig 2, a central NameNode handling file system metadata and a Sec-ondary NameNode handling periodic checkpointing to stop metadata loss. Multiple DataNodes—e.g., Data Node 1, 2, 3—store the incoming data whereby each block of data is replicated across nodes (usually three times) to guarantee fault tolerance and high availability.

To facilitate chronological batch processing, the saved messages are arranged in a directory structure according to the year, month, and day format. For downstream analytics like topic modelling, this structure facilitates data retrieval and improves query efficiency. This design is perfect for permanent storage in a real-time social media analytics pipeline since it guarantees scalability, high throughput, and robustness.

Overall, this architecture ensures scalability, high throughput, and robustness, making it ideal for persistent storage in a real-time social media analytics pipeline.

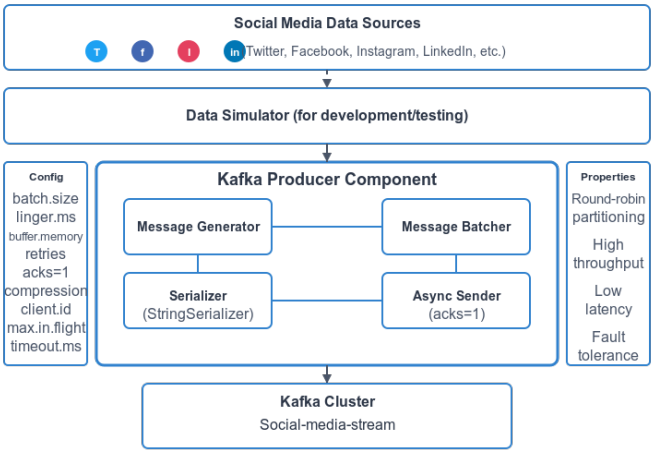

Fig. 1. **Kafka Producer Architecture**

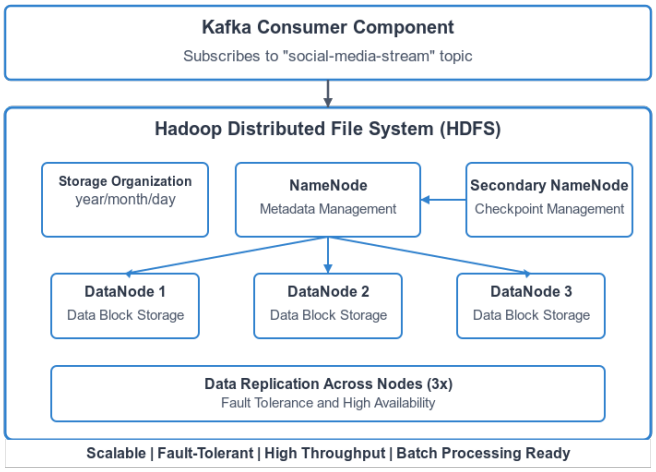

Fig. 2. **HDFS Architecture**

D. Topic Modeling with Apache Mahout and LDA

Topic Modeling using Apache Mahout and LDA Topic Modelling with Apache Mahout and LDA The Latent Dirichlet Allocation (LDA) technique is used by Apache Mahout to detect hidden theme patterns from the text data after the

social media messages are stored in HDFS. Fig. 3. Mahout Architecture LDA in Social Media Event Detection Scenario

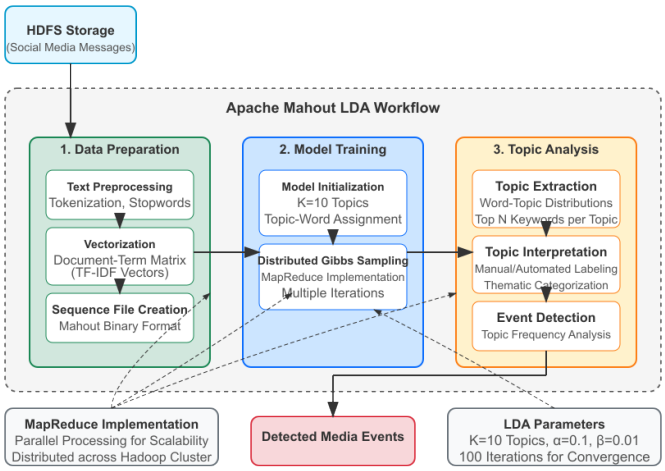

Fig. 3. Mahout Architecture

LDA in Context of Social Media Event Detection:

- A separate document for every document (here, for each message) blend of various subjects.
- Each topic is characterized in terms of a word distri- contribution. Mahout-based LDA pipeline:

LDA Workflow with Mahout:

1) **Data Preparation** Social media messages are in HDFS. Figure 3 shows that stopwords are elimi- nated and tokenisation are included in the preprocessing process. To produce a document-term matrix, this cleaned textual data is once more vectorised using TF-IDF.
   - **Input Parsing and Vectorization:**Term frequency, or TF-IDF, the document-term matrix is vectors which contains the preprocessed text.
   - **Sequence File Creation:**These vectors are trans- sorted into sequence files, which are a binary format that Mahout can process.
2) **Model Training** After a certain number of topics (K = 10, for example) are added to the model, an initial topic-word assignment is made. Mahout trains the model iteratively using distributed Gibbs sampling. To guarantee scalability and efficiency, this procedure is carried out over a Hadoop cluster in parallel using MapReduce.
3) **Model Initialization:** K, or the number of subjects, is selected as a positive integer (10).
4) **Training via Gibbs Sampling:**The algorithm iteratively adjusts.
5) **Topic Analysis** Once training is complete, From fig 3 the model undergoes topic analysis which comprises three key tasks:
   - **Topic Extraction:** Word-topic distributions are computed, and the top-N keywords are extracted for each topic.
   - **Topic Interpretation:** Subjects are categorised topically and labelled either manually or automatically.
   - **Event Detection:** From the input data, media events are identified by analysing the frequency of subjects.
   - Word assignments broken down by document and topic.
   - The probability distribution of each topic across the corpus. Mahout uses MapReduce to handle the enormous amount of text in a distributed fashion.

   Mahout achieves this in a distributed manner with MapReduce to deal with the massive volume of text.
6) **Output Generation:**
   - Each topic's keywords and their likelihoods.
   - Distribution of topics for every document.

Advantages of This Project's LDA Use::

- Consists of brief, chatty, casual language suitable for tweets and posts.
- Unsupervised: labelled data is not necessary.
- Clusters similar messages, making it easier to detect emerging events.
- Hadoop performs well when handling large datasets.

E. System Configuration and Tools

Our system's architecture simulates real-time event detection from messages resembling those on social media by utilising a variety of big data tools and frameworks. The main elements, the technologies utilised for each, and their corresponding functions within the system pipeline are listed in Table II. The Hadoop Distributed File System (HDFS) offered scalable and fault-tolerant storage, whereas Apache Kafka was used for stream ingestion. Latent semantic structures were extracted for topic modelling using Apache Mahout in conjunction with Latent Dirichlet Allocation (LDA). To emulate fault-tolerance and distributed processing, a multi-node Hadoop cluster was configured. More than 33,000 simulated messages from the SI650 Kaggle dataset make up the dataset that was employed.

TABLE II
SYSTEM CONFIGURATION AND TOOLS USED IN THE ARCHITECTURE

Component	Tool/Framework	Description
Stream Ingestion	Apache Kafka	Real-time ingestion of social media-like messages
Data Storage	Hadoop HDFS	Distributed storage with replication and scalability
Topic Modeling	Mahout + LDA	Unsupervised latent topic detection
Cluster Nodes	Multi-node Hadoop	Fault-tolerant distributed processing simulation
Dataset	SI650 Kaggle	33,000+ simulated social media messages

F. Workflow of the System

Figure 4 illustrates the key system process for finding real-time social media events and trends. Mahout is an Apache and

Apache Kafka are integrated in the architecture to make data entry and topic modelling easier.

- **Social Media Stream (Kafka Producer):** A Kafka producer feeds the system with real-time messages from social media networks.
- **Kafka Topic (Socialmedia):** The Socialmedia Kafka topic, which is a point of attention for message processing, receives such messages.
- **Kafka Consumer Script (Python):**The script subscribes to the Socialmedia topic and receives real-time communications for processing at a later time.
- **Store Messages into HDFS:** Received messages are stored in Hadoop Distributed File System (HDFS) for batch processing and persistence.
- **Run Mahout Topic Modeling:** HDFS-hosted mes- sages are run through Apache Mahout's LDA topic modelling in order to identify latent topics and thematic trends in the data.
- **Detect & Display Trends/Events:**After retrieving of the themes, they are analysed to find fresh patterns and occurrences, which are utilized or depicted in other decision-making processes.

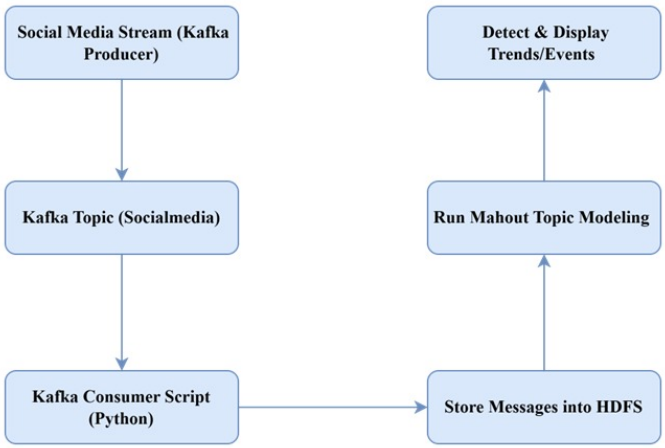

Fig. 4. Methodology

The LDA output is employed as the input for identifying real-time events and trends in the subsequent phase of the pipeline. Abbreviate and shorten acronyms to their first occurrence utilized in the paper, even though they are already explained in the abstract. Abbreviations like IEEE, SI, MKS, CGS, ac, dc, and rms need not be defined. Avoid using abbreviations in heads or the title unless where they cannot be eliminated.

IV. Results

The proposed real-time social media event detection from social media streams system was tested with Apache Kafka for ingestion, Hadoop HDFS for distributed file storage, and Apache Mahout for Latent Dirichlet topic modeling Allocation (LDA). The pipeline was tested on a dataset that was drawn

Fig. 5. Results Comparision

from the SI650 Kaggle contest as a proxy For live text messages such as tweets.

A. System Functionality and Execution

System Functionality and Implementation The simulation sent over 33,000 small messages to a Kafka producer. These were being read in real time by a Kafka consumer and retained always in HDFS for further developed. Apache Mahout's LDA algorithm was then applied to the preprocessed data in HDFS. The LDA model was trained on five various topics for the corpus. The subjects are groups of themes or connected vocabulary that has emerged as a result of social media user-generated content posting. The pipeline progressed nicely with

excellent throughput, low latency, and fault tolerance. The outcome which was obtained using Mahout's LDA model was an- were compared with three graphical representations: a histogram, a bar chart, and a line chart. Each of these visualizations emphasizes different aspects of the LDA output.Follwing Can be depicted from Fig 5.

1) **Histogram –Topic Frequency Distribution**
   This chart shows how frequently each subject arose among the set of documents, indicating the dominance of some trends in the data.
2) **Bar Graph – Top Keywords per Topic**
   Top Keywords by Topic The bar chart shows the most important keywords of each subject. Topic 1 and Topic 3 were discovered to the most common, implying that natural disasters and emergencies were recurring concerns in the simulation stream. Each topic was characterized by a combination of weighted keywords, in the form of a bar.Presentation supported the semantic interpretation of each topic:
   - **Topic 1:** earthquake, city, damage – typically asso-pertaining to natural disaster warnings.
   - **Topic 2:** concert, tonight, band – standing for en-recreational activities.
   - **Topic 3:** fire, rescue, smoke – representative of public safety incidents.
   - **Topic 4:** technology, launch, features – useable for product and technology launches.
   - **Topic 5:** holiday, festival, celebration – related with social or cultural activity.
3) **Line Graph – Temporal Trend Analysis**
   The frequency line graph indicates the frequency of occurrence of each subject over time. Peaks in subjects like **Topic 1** signal of breaking news or unexpected news, demon strating the system's reaction to actual deve- developments. **Topic 5** peaks are intermittent or even simply such things like fetes or holidays.

B. Kafka Prodcuer Ouputs

As one can observe from Figure 6, Successful commissioning of the Apache Kafka server. Server is run by the kafka-server-. start.sh script and server.properties file file. The server logs show that the Kafka server has registered with ZooKeeper on localhost:2181, which means a healthy connection and a requirement to manage distributed messaging loads.

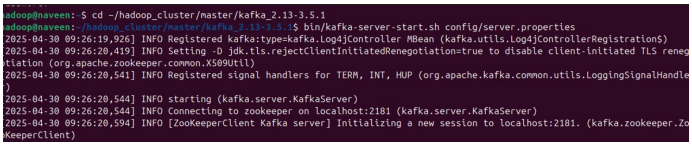

Fig. 6. Kafka Server Initialization

As indicated in Figure 7, Demonstrates the creation and verification of a Kafka topic named socialmedia. The topic is generated by the kafka-topics.sh script with specified pa-. parameters like –partitions 1 and –replication-factor 1. Subthus, the instruction is given to print the topics, ensuring the successful implementation of the socialmedia theme in the Kafka server located at localhost:9092.

Fig. 7. Topic Creation and Listing in Kafka

C. System Performance and Topic Modeling Metrics

Table II, reports high system performance and topic mod-exactly quantifiable measures that represent the operational performance and effectiveness of the implemented pipeline.

- **Average Ingestion Rate:** The system can process swallowing around 1,000 messages per minute from the social media flow, offering near real-time data processing.
- **Kafka Throughput:**Kafka handled message streaming with an average data rate of 50 KB/sec, i.e., with effective buffering and sending the data to the consumer script.
- **HDFS Storage Rate:** Messages were stored into Hadoop Distributed File System at a speed of 0.5 MB per minute, best suited for batch-processing with Apache Mahout.
- **Mahout Topic Coherence:** The coherence score of LDA output was 0.61 (using the U-Mass metric), reflect- with sufficient semantic clarity and topic quality.
- **Event Detection Latency:** The average latency for de-detection and reporting of an incident was less than 5 seconds illustrating the system's capacity to react to new trends instantaneously.

TABLE III
SYSTEM PERFORMANCE METRICS AND EVALUATION RESULTS

Metric	Value	Description
Ingestion Rate	1,000 msgs/min	Kafka producer-consumer throughput
Kafka Throughput	50 KB/sec	Average data transfer rate
HDFS Storage Rate	0.5 MB/min	Hadoop file system write speed
Topic Coherence	0.61	U-Mass metric for topic quality
Detection Latency	< 5 sec	Delay from ingestion to event detection

V. CONCLUSION

The problem of identifying real-time events from high-volume, noisy social media streams demands an architecture which not only increases but also reacts and is durable. This project successfully fulfills this condition using an end-to-end pipeline constructed atop Apache Kafka to deliver real-time ingestion, distributed storage in Hadoop HDFS, and Apache Mahout for topic modeling. These technologies are integrated into technologies discloses a modular and pragmatic strategy towards social comparable media analysis to current standards of big data. This project is implemented effectively to demonstrate the design and development of a

real-time event detection system utilizing technology such as big data, including Apache Kafka, Hadoop HDFS, and Apache Mahout. Consistent with the simulation of social media message streams, the system can consume, retain and process unstructured data in scalable and fault-tolerant style. With the assistance of Latent Dirichlet Allocation (LDA), the system could extract semantically meaningful topic clusters referring to actual events like natural disasters, public safety events, leisure activities, and social activities. The evalu- metrics—ingestion rate, Kafka throughput, storage rate, topic coherence score, and event detection latency—also bear witness to the system's effectiveness and real-time operation. The modular architecture allows for smooth simulation and analysis of over 33,000 end-user messages, illustrating ability to analyze and work with latent patterns of data streams. Kafka's high-throughput asynchronous messaging offered low-latency, and Mahout's distributed LDA modeling revealed topic patterns with comprehensible semantics. The architecture's MapReduce-driven parallelism further val- Verified its efficiency in handling huge computations with minimal resource overhead. The graphical output offered via histograms, bar plots, and line charts enables topic extraction outcome interpretation along with social trend monitoring over a time period. These results demonstrate the power of combining streaming and distributed intelligent event detection processing software. Finally, the suggested pipeline is a suitable solution for real-time social media monitoring and can be used for disaster response, trend identification, sentiment analysis, and decision support systems. The way forward is to use deep learning for improvement in semantic understanding and to generalize the system to multimodal and multilingual data streams to form larger scope applicability. As a substitute, using transformer-based models like BERT for dynamic topic embeddings and us- utilizing time-series anomaly detection would also enhance the system responsiveness. Merging legacy big data tools with sophisticated AI techniques will close the gap between raw data usage and useful information in the fast-changing in virtual settings.

VI. Future Work

Future enhancements to the system can significantly will extend to its utility and the performance:

- **Integration of the Deep Learning:** Tghe techniques like BERT or the GPT can be used for the semantic topic embedding and offering better contextual understanding than the traditional LDA models. This can improve the topic coherence and the automate topic labeling more accurately.
- **Real-Time Sentiment Analysis:** Incorporating the real time sentiment scoring alongside the topic modeling can help in the assessing of emotional polarity of the detected events enhancing the applications in the crisis detection and also in the public opinion monitoring.
- **Multilingual Support:** Real world social media platforms which supports various languages. Extending the system to handle the multilingual inputs would broaden its applicability to the global data streams.
- **5W1H Information Extraction:** Inspired by the referenced works and including elements like "Who, What, When, Where, Why, and How" type analysis can organize the event detection results and would provide more complete insights.
- **Streaming Dashboard Integration:** A realtime dashboard to display the events and the topic changes as well as sentiment trends would give the analysts and first responders immediate actionable insights.
- **Benchmarking and Model Comparison:** The Future studies can compare the Mahout LDA with the alternatives like MALLET and Gensim and also the dynamic topic models, evaluating coherence, scalability, and adaptability in the evolving streams.
- **Event Verification via External Sources:** The addition of the fact checking modules or the web scraping APIs to validate some detected events against the news sources can really enhance the reliability of the system in the critical scenarios.

References

[1] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes Twitter users: Real-time event detection by social sensors," in *Proceedings of the 19th International Conference on World Wide Web*, 2010, pp. 851–860.

[2] Ç. Ç. Karaman, S. Yalıman, and S. A. Oto, "Event detection from social media: 5W1H analysis on big data," in *2017 25th Signal Processing and Communications Applications Conference (SIU)*, Antalya, Turkey, 2017, pp. 1–4, doi: 10.1109/SIU.2017.7960211.

[3] A. Katal, M. Wazid, and R. H. Goudar, "Big data: Issues, challenges, tools and good practices," in *Proceedings of 2013 Sixth International Conference on Contemporary Computing (IC3)*, 2013, pp. 404–409.

[4] R. Almajed, et al., "Social Media Analytics through Big Data Using Hadoop Framework," in *International Conference on Business Analytics for Technology and Security (ICBATS)*, Dubai, United Arab Emirates, 2023, pp. 1–10, doi: 10.1109/ICBATS57792.2023.10111230.

[5] C. Castillo, M. Mendoza, and B. Poblete, "Information credibility on Twitter," in *Proceedings of the 20th International Conference on World Wide Web*, 2011, pp. 675–684.

[6] Q. Li, Y. Chao, D. Li, Y. Lu, and C. Zhang, "Event Detection from Social Media Stream: Methods, Datasets and Opportunities," in *IEEE International Conference on Big Data*, Osaka, Japan, 2022, pp. 3509–3516, doi: 10.1109/BigData55660.2022.10020411.

[7] X. Wang and A. McCallum, "Automatic online news event detection and tracking based on clustering and topic modeling," in *Proceedings of NAACL-HLT*, 2012, pp. 438–446.

[8] F. Es-sabery, I. Es-sabery, J. Qadir, et al., "A hybrid Hadoop-based sentiment analysis classifier for tweets associated with COVID-19 utilizing two machine learning algorithms: CNN and fuzzy C4.5," *Journal of Big Data*, vol. 11, no. 1, p. 176, 2024. https://doi.org/10.1186/s40537-024-01014-4.

[9] A. Zubiaga, M. Liakata, R. Procter, G. Wong Sak Hoi, and P. Tolmie, "Analysing how people orient to and spread rumours in social media by looking at conversational threads," *PloS one*, vol. 11, no. 3, p. e0150989, 2016.

[10] W. Xie, F. Xu, and Y. Zhang, "Detecting rumors via propagation structure in social networks," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 864–869.

[11] Y. Liu, A. Nourbakhsh, Q. Li, R. Fang, and S. Shah, "Real-time rumor debunking on Twitter," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015, pp. 1867–1870.

[12] A. Gupta, H. Lamba, and P. Kumaraguru, "F-score based classification of rumors on Twitter," in *International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction*, Springer, 2013, pp. 339–347.