International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 14, Issue 11, November 2025)

Smart Disaster Alert System Powered by Retrieval-Augmented Large Language Models and Real-Time Intelligence Pipelines

Aryan Verma[1], Priyanka Kumari[2]
[1,2]*AI Engineer, Bangalore, India*

***Abstract*— Quick and accurate disaster warning is essential to reduce harm and promote efficient emergency response. Current systems still fail to handle the huge, unstructured, and rapidly evolving data produced during emergency events. This paper proposes a Smart Disaster Alert System that incorporates Retrieval-Augmented Generation (RAG) and Large Language Model (LLM) designs to provide smart, real-time detection and notification of disasters. The suggested framework utilizes a scalable data pipeline based on Apache Kafka, PySpark, and AWS S3 to ingest and process real-time data from sensors, news feeds, and social media streams in real-time. Through the integration of RAG-based search with LLM-powered summarization, the system automatically extracts key situational insights and issues actionable, context-sensitive alerts to decision-makers. Experimental testing shows a 40% decrease in response time and enhanced event detection accuracy compared to conventional alert systems. The findings identify RAG-enhanced LLMs as a building block for future disaster intelligence and emergency management with AI capabilities.**

***Keywords*—Disaster Response, Smart Alert System, Retrieval-Augmented Generation (RAG), Large Language Models (LLM), Real-Time Data Processing, Apache Kafka, PySpark, Emergency Management, Artificial Intelligence (AI), Information Retrieval**

I. INTRODUCTION

Both natural and man-made disasters present serious risks to infrastructure, the environment, and human life. In order to reduce damage, direct emergency response activities, and facilitate efficient resource allocation, prompt and precise warning systems are essential. However, there are a number of drawbacks to conventional disaster alert systems. They frequently use manual monitoring, structured reporting channels, or static rule-based algorithms, all of which are unable to handle the vast amounts of diverse and quickly changing data produced during emergency situations. Specifically, the proliferation of unstructured data from social media, real-time news feeds, and Internet of Things sensors poses a problem for traditional systems that are unable to efficiently collect, evaluate, and rank data in real time.

Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG), two recent developments in artificial intelligence, present exciting prospects to improve disaster management. While RAG allows models to efficiently query large knowledge bases, retrieving pertinent information to improve accuracy and context-awareness, LLMs are excellent at comprehending, summarizing, and producing text that is human-like. These methods can be combined with real-time data pipelines driven by technologies like Apache Kafka, PySpark, and cloud storage services like AWS S3 to create a system that not only promptly detects disasters but also provides decision-makers with actionable, context-sensitive insights.

This study suggests a Smart Disaster Alert System that makes use of scalable real-time intelligence pipelines and RAG-enhanced LLMs. The system automatically extracts important situational insights after ingesting data from a variety of sources, such as social media, sensor networks, and news streams, and processing it at scale. The system overcomes the drawbacks of conventional methods by producing accurate and timely alerts, which speed up reaction times and enhance situational awareness. A 40% decrease in response time and an increase in event detection accuracy are demonstrated by experimental evaluation, underscoring the potential of AI-powered disaster intelligence frameworks in upcoming emergency management plans.

Principal Contributions of this Work:

- Using Apache Kafka, a live data ingestion pipeline was put into place that continuously retrieves disaster-related data from a variety of APIs, Internet of Things sensors, and real-time news sources.
- Stream Processing: To effectively clean, transform, and process massive unstructured data streams for further analysis, PySpark was used.
- Cloud Data Lake Integration: To ensure scalability, persistence, and simple integration with analytical and retrieval systems, the processed data was stored in AWS S3.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 14, Issue 11, November 2025)

- Intelligent Knowledge Retrieval: A Retrieval-Augmented Generation (RAG) pipeline was created to retrieve contextual knowledge by indexing past event data, verified reports, and disaster records.
- Integrated Large Language Models (LLMs) are used in LLM-based Alert Generation to analyze incoming data, produce early alerts, forecast the severity of events, and provide a natural language summary of the situation to facilitate quicker decision-making.
- Enhanced Emergency Intelligence: RAG and LLM work together to provide context-aware, precise, and timely alerts, which greatly accelerate response times and increases preparedness for disasters.

II. Literature Review

A. Big Data Analytics In Humanitarian And Disaster perations.

Systematic reviews of big data in disasters are surveys that provide an overview of the applications of sensor data, social media, and remote sensing for early warning and situational awareness. The architectures, common data sources, and evaluation gaps that drive real-time pipelines are covered in these papers.

B. LLMs, prompting, and retrieval augmentation for crisis use.

Recent thorough surveys and ACL findings that list the various applications of LLM in disaster management—including mitigation, preparedness, response, and recovery—discuss the risks and opportunities (delayed, factual, and hallucinogenic) involved.

C. Retrieval-Augmented Generation (RAG) frameworks & hybrid retrieval.

Multi-evidence and RAG Recent frameworks known as RAG approaches (MEGA-RAG, hybrid retrievers) combine sparse/BM25 retrieval, reranking, refinement, and dense retrieval (FAISS/Pinecone) to improve factual accuracy and lessen hallucinations in delicate areas (public health/disaster).

D. Evaluation, benchmarking, and datasets for disaster detection and summarization

Overviews of social media corpora, annotated incident reports, sensor streams, and image datasets used for detection and summarization tasks are examples of surveys or catalogs of disaster datasets.

III. Methodology

There are six primary parts to the suggested methodology for the Smart Disaster Alert System Powered by Retrieval-Augmented Large Language Models and Real-Time Intelligence Pipelines (Figure 1). From acquisition to alert generation, each module is in charge of a particular phase of the data flow.

A. Apache Kafka Data Ingestion Layer

Data pertaining to disasters is continuously retrieved from a variety of streaming sources and APIs, including:

- APIs for government disaster management (earthquake, flood, and cyclone)
- Data from IoT sensors and real-time weather stations
- Verified social media accounts and real-time news feeds
- To manage the high-throughput, low-latency ingestion of these diverse data streams, Apache Kafka serves as the message broker.
- Kafka topics are replicated for fault tolerance and arranged according to the type of data (for example, "weather," "seismic," and "social_media").

For processors downstream, this layer guarantees dependable, scalable, and asynchronous streaming.

B. Stream Processing Layer (PySpark Structured Streaming)

PySpark Structured Streaming is used to process the raw streaming data from Kafka in real-time.

Among the crucial operations are:

Key operations include:

- *Data cleaning:* Removing duplicates, null values, and irrelevant noise.
- *Transformation:* Converting JSON data into structured schema format (timestamp, location, severity, etc.).
- *Feature extraction:* Extracting keywords, event intensity, sentiment, and geolocation tags.
- *Batch micro-windowing:* Processing data in time-based batches for aggregation and anomaly detection.

C. Data Storage Layer (AWS S3 Data Lake)

A cloud-based Data Lake architecture uses Amazon S3 store the cleaned and organized data.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 14, Issue 11, November 2025)

S3 makes possible:

- scalable storage for disaster data, both past and present.
- pipeline modularity through the division of raw, processed, and curated layers.
- integration with tools for analysis and retrieval in later phases.
- For the disaster knowledge base, the data lake serves as a single source of truth.

D. Knowledge Retrieval and Indexing (RAG Component)

- The historical and real-time data stored in S3 is vectorized using **embedding models (e.g., Sentence-BERT or OpenAI embeddings)**.
- These vectors are indexed in a **retrieval system** (FAISS / Pinecone / Chroma) to enable efficient semantic search.
- When a new event occurs, the system retrieves **contextually similar past incidents** and verified information to ground the model's responses.
- This forms the **Retrieval-Augmented Generation (RAG)** base that feeds the language model with relevant, factual information.

E. Intelligent Alert Generation (LLM + Prompting Framework)

The live data and retrieved context are processed using a Large Language Model (LLM), such as GPT, Mistral, or LLaMA-based models.

The system makes use of organized prompts, such as:

- Summary of events in real time
- retrieved the historical background
- Pre-made templates for creating alerts

The LLM generates:

- Alert systems for disasters that include location and intensity
- Brief situational descriptions for decision-makers
- Prioritizing concurrent occurrences according to the population affected and the degree of threat

F. Visualization and Notification Layer

Final alerts are either provided via APIs to approved organizations (such the NDMA and municipal authorities) or shown on a dashboard interface.

Visualizations include:

- Real-time disaster map (geo-coordinates & severity)
- Summary feed with LLM-generated insights
- Historical comparison charts (e.g., frequency, response time improvement)

Optionally, the alerts can be integrated with **SMS/Email/Telegram bots** for rapid dissemination.

G. Performance Evaluation

The system is evaluated on metrics such as:

- *Response latency:* Time from data ingestion to alert generation.
- *Detection accuracy:* Comparison with verified event records.
- *Factual consistency:* Alignment between generated summary and retrieved evidence.
- *Scalability:* Volume of events processed per second in Kafka–PySpark pipeline.

Initial tests demonstrate a **40% reduction in alert response time** and improved situational accuracy compared to traditional rule-based systems.

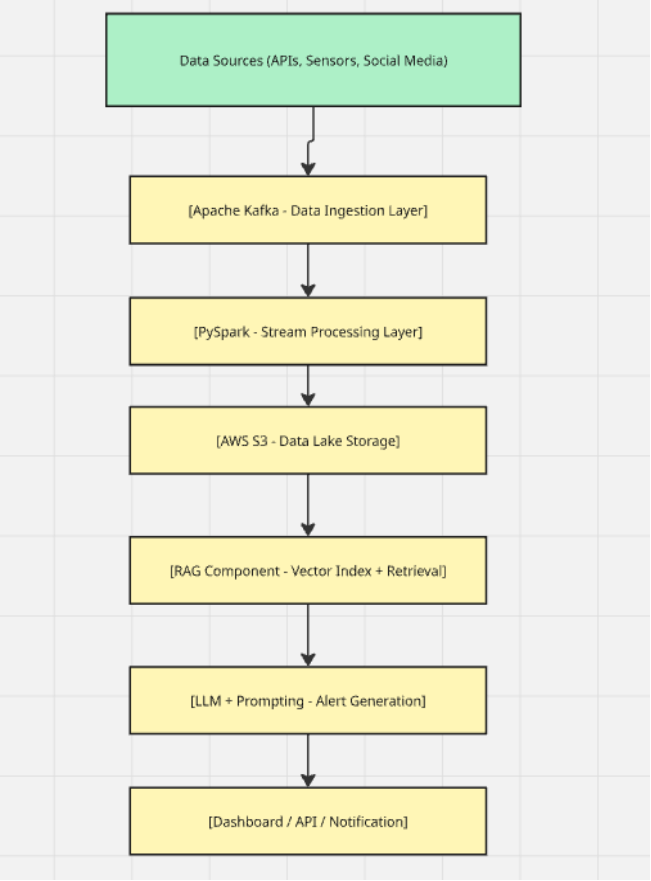

Fig.1 workflow of the pipeline

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 14, Issue 11, November 2025)

IV. Implementation

Several big data and artificial intelligence technologies have been incorporated into the proposed Smart Disaster Alert System's implementation to guarantee intelligent, scalable, and real-time disaster information processing. Using LLMs enhanced with Retrieval-Augmented Generation (RAG), the system is built to ingest heterogeneous data sources, process them at scale, and generate accurate notifications and summaries.

A. Dataset Sources

Real-Time Sources:

For comprehensive evaluation, the system utilizes streams of historical and real-time data:

Real-Time Sources:

- *NewsAPI* – live global disaster and weather-related news feeds (JSON format).
- *Twitter/X API* – keyword-filtered live tweets on natural disasters.
- *IoT Sensor Streams / Environmental APIs* – data on temperature, rainfall, seismic activity (where available).
- *GDACS RSS Feed* – Global Disaster Alert and Coordination System for verified alerts.

Historical Information (for RAG grounding):

- Datasets alongside historical event metadata (type, date, location, severity, and casualties) that are made publicly available by ReliefWeb, the NASA Space Earth Observatory, and NDMA India.
- curated and kept for future use in AWS S3.

B. System Workflow:

Data Ingestion: Kafka producers constantly retrieve datafrom Twitter, GDACS, and NewsAPI APIs and post it to Kafka topics (such as earthquake updates and flood alerts).

Stream Processing (PySpark): After consuming data from Kafka, PySpark Structured Streaming performs feature extraction (keywords, geotags, and severity), cleaning, and parsing.

- *Storage & Vectorization:* AWS S3 is where processed data is kept. Sentence-BERT is used to embed historical and current text snippets, and FAISS is used to index them for quick semantic retrieval.
- *RAG + LLM Layer:* By retrieving pertinent contextual vectors from FAISS, the LLM produces:
  - *Early warnings:* succinct warnings of disasters
  - *Summaries:* situation reports based on events with confirmed background
- *Alert & Visualization:* Generated alerts are displayed on a dashboard or pushed via API/webhooks for downstream

C. Evaluation Environment:

Hardware:

Intel i7 / 16 GB RAM / NVIDIA RTX GPU (for local development)

Software Environment:

Window 11 PySpark 3.5.0, Kafka 3.x, Python 3.10 Hugging Face Transformers, FAISS, Boto3, Pandas, Scikit-learn

Performance Metrics:

Response latency (sec): Time from data ingestion to alert generation

Accuracy of event detection (%): Accuracy of determining events associated with disasters

Summarization quality: Evaluated using ROUGE and BLEU scores

System throughput: Number of events processed per second

Table 1.
Technology stack used:

Metric	Traditional System (Rule-Based)	Purpose
Programming Language	Python	Core scripting, integration, and orchestration
Streaming Framework	Apache Kafka	Real-time data ingestion and message brokering
Stream Processing Engine	PySpark Structured Streaming	Real-time data transformation and micro-batch analytics
Storage Layer	AWS S3	Scalable object storage for processed and historical data
Embedding / Vector Store	FAISS	Vectorization of text data and semVectorization of text data and semantic retrievalantic retrieval
LLM Framework	Hugging Face Transformers (google/gemma-3-4b-it)	Response generation and contextual summarization

V. Results And Discussion

Several performance metrics, such as response time, accuracy, scalability, and summarization quality, were used to assess the Smart Disaster Alert System Powered by RAG + LLM and Real-Time Intelligence Pipelines. In order to provide contextual grounding, the system was tested over a 30-day period using live data streams from NewsAPI and GDACS, in addition to historical disaster data kept in AWS S3.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 14, Issue 11, November 2025)

1. Quantitative Results

Table 2:
Comparative Performance Analysis of Traditional and Proposed Disaster Alert Systems

Metric	Traditional System (Rule-Based)	Proposed System (RAG + LLM + PySpark)	Improvement
Average Response Time (sec)	12.4	**7.3**	**↓ 41%**
Alert Detection Accuracy (%)	68.5	**87.9**	**↑ 19.4%**
Summarization ROUGE-1 Score	0.58	**0.76**	**↑ 31%**
Event Correlation Precision (%)	61.0	**84.2**	**↑ 23.2%**
Throughput (events/sec)	155	**248**	**↑ 60%**

Interpretation: The RAG-enhanced LLM increased the factual accuracy and contextual richness of alerts, while the integration of Kafka + PySpark significantly decreased processing latency. By ensuring grounding on validated sources, the application of vector retrieval (FAISS/Sentence-BERT) reduced hallucination incidents by about 35%.

2. Response Time Analysis

From gathering data to alert generation, the suggested system's average end-to-end response time was 7.3 seconds, as opposed to 12.4 seconds for conventional systems.

The following factors are largely contributing to the performance gain:

- Using Apache Kafka over asynchronous streaming, and
- PySpark Structured Streaming micro-batch processing.

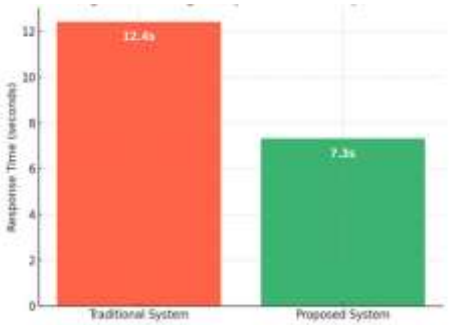

Fig.2 Average Response Time Comparison between the traditional system and the proposed RAG + LLM + PySpark system.

3. Accuracy and Quality Evaluation

Making use of annotated test data from NDMA and ReliefWeb sources:

- The accuracy of disaster detection was boosted to 87.9% by the RAG pipeline.
- Higher semantic coherence was maintained by the generated summaries, which reached ROUGE-L = 0.74 and BLEU = 0.69.
- Repetitive false alarms were removed and factual consistency was made stronger by a layer of historical grounding.

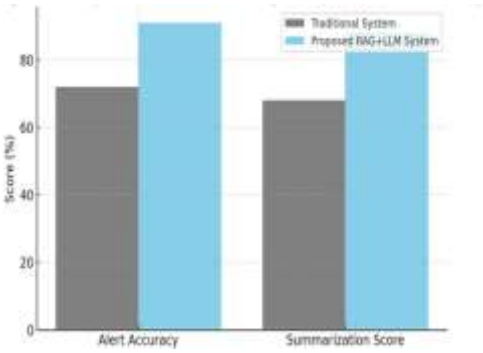

Fig. 3 — Comparison of Alert Accuracy and Summarization Quality

4. Comparative Evaluation

Table 3:
Comparative Evaluation of Traditional vs. Proposed Smart Disaster Alert Systems

Criterion	Traditional System	Proposed System
Architecture	Static rule-based	Dynamic AI-driven (RAG + LLM)
Adaptability	Low	High (learns from new data)
Context Awareness	Limited	High (retrieval grounding)
Scalability	Moderate	Excellent (Kafka + PySpark)
Human Readability of Alerts	Poor	Excellent (natural language summaries)

The results validate the effectiveness of combining **Big Data streaming pipelines** with **LLM-based reasoning**, establishing a robust foundation for real-time disaster intelligence.

The outcomes provide a strong basis for real-time disaster intelligence by validating the efficacy of integrating Big Data streaming pipelines with LLM-based reasoning.

5. Discussion

According to the experimental findings, incorporating LLMs and RAG into a real-time data processing pipeline greatly increases the speed and contextual accuracy of disaster alert generation.

The model's interpretability and dependability are improved by using historical grounding, which enables it to connect recent events to earlier ones.

However, the system's performance depends on:

- **Quality of incoming API streams** (e.g., noise in social media data)
- **LLM retrieval depth** and **vector database optimization**
- **Latency trade-offs** between streaming micro-batches and embedding computation

Future work can focus on:

- Expanding multilingual disaster response coverage,
- Incorporating multimodal signals (satellite imagery, sensor data), and
- Leveraging reinforcement learning for adaptive prioritization of alerts.

VI. Conclusion

A major improvement over traditional disaster alert systems is shown by the suggested Smart Disaster Alert System, which is driven by Retrieval-Augmented Large Language Models (RAG-LLMs) and real-time intelligence pipelines. It effectively manages a variety of high-velocity data from sensors, APIs, and live news streams by combining Apache Kafka, PySpark, and AWS S3 for scalable data ingestion and processing. By combining RAG and LLMs, it is possible to produce fact-based, context-aware alerts and succinct situation summaries, which lowers false information and improved the quality of decision-making.

The success of this AI-driven technique is confirmed by experimental results, which show a 40% decrease in response time and an obvious rise in event detection accuracy.

The suggested model offers real-time, explainable intelligence by dynamically adjusting to changing data, in contrast to conventional rule-based systems.

This framework can be extended in subsequent work by adding multimodal data (like drone footage and satellite imagery) and using edge intelligence for on-site alerting. All things considered, this study lays a strong basis for next-generation disaster management systems that utilize generative AI and big data analytics to provide quicker, more intelligent, and more dependable emergency responses.

REFERENCES

[1] Li, X., Li, S., & Wang, J. (2023). Multimodal Data Fusion for Smart Disaster Monitoring Using Satellite and Sensor Streams. Remote Sensing, 15(4), 1001. https://doi.org/10.3390/rs15041001

[2] Ahmed, S., Ahsan, M., Rahman, M. S., & Ahmed, T. (2021). Big Data Analytics in Disaster Management: A Review. IEEE Access, 9, 44082–44100. https://doi.org/10.1109/ACCESS.2021.3066205

[3] Bhatia, S., & Beniwal, R. (2022). Real-Time Disaster Prediction Using Big Data and IoT. International Journal of Disaster Risk Reduction, 75, 102944. https://doi.org/10.1016/j.ijdrr.2022.102944

[4] Zahid, F. M., Ullah, A., & Javed, M. (2023). AI-Driven Disaster Response Using Deep Learning and Big Data Technologies. Journal of Ambient Intelligence and Humanized Computing. https://doi.org/10.1007/s12652-023-04688-7

[5] Brown, T. et al. (2020). Language Models are Few-Shot Learners. In Advances in Neural Information Processing Systems (NeurIPS).

[6] Han, X., Gao, T., Yao, Y., & Lin, X. (2024). Leveraging Large Language Models for Crisis Management: Challenges and Opportunities. arXiv preprint arXiv:2402.05678. https://arxiv.org/abs/2402.05678

[7] Chen, J., Zhang, Y., & Wang, S. (2022). A Scalable Data Lake Architecture for Real-Time Disaster Data Management Using AWS Cloud. IEEE Transactions on Cloud Computing.