Trend Analytics at Scale: Real-time Review Sentiment and Topic Analysis Pipeline

Team ***Spark Syndicate***

Kashinath Alias Kapil Subhash Naik
kashinathn@iisc.ac.in

Sujith Shetty
sujith1@iisc.ac.in

Atreyee Mondal
atreyeem@iisc.ac.in

Pranav N
pranavn@iisc.ac.in

1. Problem Definition

E-commerce platforms like Flipkart receive millions of customer reviews daily, with valuable insights buried in large-scale unstructured data. Manual monitoring is ineffective in identifying emerging problems such as product defects, delivery issues, or sudden sentiment shifts. Detecting these issues in real time empowers businesses to rapidly respond, improve customer satisfaction, and reduce negative impacts.

1.1 Motivation

Large volumes of reviews, diverse product categories, and constantly changing customer sentiments make it difficult to detect negative product trends and recurring issues in real time. Current manual or batch-only review analysis methods create significant delays, preventing teams from identifying and resolving critical problems quickly. This lag is costly because 75% of consumers read reviews before purchasing, making timely review analysis essential for business success. The challenge is further amplified by scale major platforms can receive over 10,000 reviews per minute, demanding highly distributed and fault-tolerant processing. As product issues and sentiment shifts can occur instantly, organizations need real-time detection capabilities to prevent revenue loss and maintain brand trust. Moreover, raw review text is unstructured and overwhelming, so it must be transformed into categorized, clustered, and actionable insights that product, support, and business teams can immediately use.

1.2 Design Goals and Features

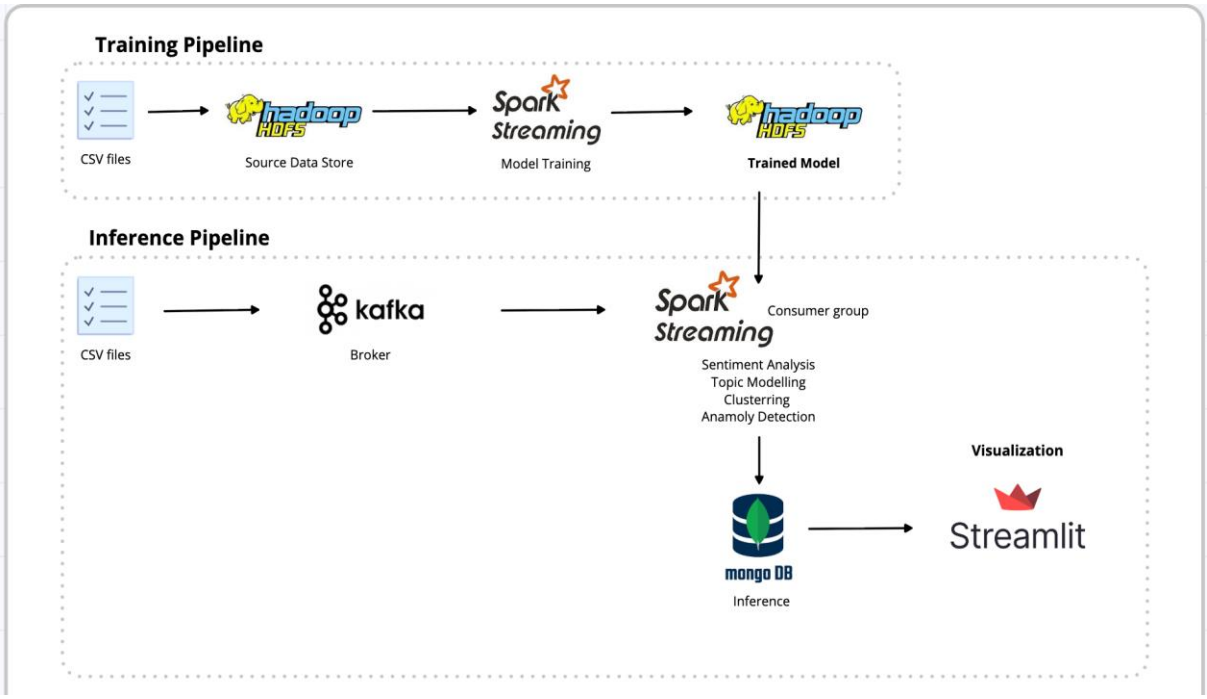
Real-time processing must ensure sub-minute latency from review of ingestion to dashboard visualization, enabling immediate detection of emerging trends and issues. High throughput is essential, with the platform capable of handling multiple reviews per minute and scaling horizontally to manage increasing workloads. To generate deeper insights, the system must support multi-dimensional analysis, performing sentiment analysis, topic modeling, and customer segmentation concurrently across incoming data streams. Fault tolerance is equally critical, allowing the system to recover automatically from failures while maintaining data consistency and uninterrupted processing. Finally, strong business intelligence capabilities must transform complex analytical outputs into clear, actionable insights, empowering product and business teams to make informed decisions quickly and effectively.

2. Approach and Methods

The system is divided into two major pipelines: a **Training Pipeline** and an **Inference Pipeline**, each serving a distinct purpose in the overall analytics workflow. In the Training Pipeline, historical CSV datasets are first ingested into **Hadoop HDFS**, which acts as the scalable source data store. Spark Streaming then processes this stored data to perform model training for tasks such as sentiment analysis, topic modelling, clustering, and anomaly detection. Once training is complete, the resulting machine-learning models are saved back into HDFS, making them accessible for downstream inference workloads.

The **Inference Pipeline** focuses on real-time analytics using the trained models. CSV review inputs are streamed into a **Kafka broker**, which decouples data producers from consumers and enables scalable ingestion. Spark Streaming operates as a consumer group, pulling data from Kafka and applying the trained models for live sentiment analysis, topic modelling, clustering, and anomaly detection. The processed results are then written to **MongoDB**, which serves as a fast, flexible inference data store. Finally, a **Streamlit dashboard** reads from MongoDB to generate real-time visualizations, enabling insights such as sentiment trends and cluster distributions to be viewed interactively.

System Architecture



2.1 Data Acquisition and Structure

The system ingests customer review data through two coordinated pathways: batch ingestion for historical data and streaming ingestion for real-time updates. Batch ingestion loads large volumes of Flipkart review data over 150,000 records from CSV files into the distributed storage layer, automatically inferring data types and preparing the dataset for downstream processing. This dataset is cached in memory to accelerate repeated access during model training, significantly reducing disk I/O and improving pipeline performance.

In parallel, real-time ingestion is handled through a Kafka-based streaming path. A producer continuously reads new review records from a source file and streams them into a dedicated Kafka topic at a controlled rate, with minor randomized delays to simulate real-world customer activity. Each message includes structured attributes such as a unique review identifier, timestamp, product details, full review text, summary, rating, and price. The streaming consumer processes only newly arriving messages, parsing each Kafka payload into structured records using a predefined schema and flattening them into a clean tabular format suitable for immediate inference.

Both ingestion paths operate on standardized structured logs. Batch ingestion handles records containing six core attributes product_name, product_price, Rate, Review, Summary, and Sentiment forming a historical dataset of 170,677 reviews. Streaming ingestion introduces two additional fields, review_id and timestamp, enabling temporal analytics and product-level aggregation. A strict StructType schema enforces field presence and type correctness across the pipeline, ensuring uniformity and reliability.

Once ingested, the structured logs undergo preprocessing to compute derived features such as review_length, summary_length, and combined_text. As data flows through the machine learning pipeline, additional analytical fields are appended, including sentiment predictions, confidence scores, dominant topics, topic distributions, cluster assignments, human-readable cluster names, and anomaly flags. The resulting enriched documents—containing original fields, preprocessing features, and model outputs persisted in MongoDB for querying, dashboards, and business intelligence use cases.

This unified ingestion and data structure design enables the system to leverage both historical review patterns and evolving real-time signals, supporting continuous learning, monitoring, and analytics across the entire review ecosystem.

2.2 Stream Processing

Stream processing consumes incoming review data from Kafka in micro-batches, typically at 10-second intervals, enabling low-latency real-time analytics. The streaming pipeline applies the same preprocessing logic used during batch training, including text tokenization, bigram generation, stopword removal, and high-dimensional vectorization using a 50,000-word vocabulary. In parallel, numeric features such as star rating, review length, and summary length are extracted and standardized. All processed features are combined into a single vector representation that merges text embeddings with scaled numerical attributes.

Pre-trained machine learning models are then applied to each micro-batch. The sentiment classifier assigns each review a label (negative, neutral, or positive) along with a confidence score. The topic modelling component generates a probability distribution across five learned topics, and the dominant topic is selected based on the highest probability. The clustering model groups incoming reviews into previously learned behavioural or issue-based clusters. Topic names and cluster categories are translated into human-readable labels for interpretability.

At the micro-batch level, the pipeline computes the proportion of negative reviews and raises an alert when the negative-rate threshold exceeds 70%, providing early detection of emerging problems. All enriched streaming outputs including sentiment predictions, topic distributions, cluster assignments, confidence scores, and anomaly indicators are continuously written to a database for incremental accumulation and downstream analytics. The streaming engine maintains checkpointed state to ensure fault tolerance; if processing is interrupted, it automatically resumes from the last committed offset without data loss.

2.3 Batch Processing

Batch processing trains machine learning models using historical review data stored in HDFS. The system loads 153,609 records, applies comprehensive preprocessing including tokenization, stopword removal, bigram generation, and CountVectorization with a 50,000-word vocabulary, and splits the dataset into training, validation, and test sets in a 70/15/15 ratio. For sentiment classification, a multi-stage Spark ML pipeline applies text tokenization, n-gram expansion, stopword filtering, vectorization, TF-IDF weighting, feature scaling, and vector assembly before training a Logistic Regression classifier. Hyperparameters are optimized through cross-validation across a grid of regularization and elastic net values, evaluating nine total combinations. The best model is selected based on validation accuracy, and final metrics including accuracy, F1-score, weighted precision, and weighted recall are computed on the test set.

For topic modeling, the pipeline utilizes Latent Dirichlet Allocation (LDA) to automatically discover hidden thematic structures without manual categorization. This approach operates on the assumption that each document is a mixture of topics, and each topic is a mixture of words, allowing the algorithm to learn topic-word distributions iteratively. A second vectorization step with a 5,000-word vocabulary feeds into the LDA model, which is configured with five topics and twenty iterations. Model quality is assessed using log perplexity to measure how well the model predicts held-out words; in our implementation, this value is negated so that higher scores indicate better topic coherence.

Once trained, the *ldaModel.describeTopics()* method extracts top keywords which are mapped back to the vocabulary to interpret specific themes. The model successfully identified distinct categories including:

- **Product Quality** (Topic 0): Defined by keywords such as “quality”, “defective”, and “durable”.
- **Delivery & Shipping** (Topic 1): Characterized by terms like “delivery”, “fast”, “slow”, “delayed” and “package”.
- **Price & Value** (Topic 2): Identified by words including “price”, “expensive”, “cheap” and “worth”.

Clustering analysis evaluates multiple values of k (2-5) using K-Means on a combined feature space consisting of sentiment scores, star ratings, text-length features, and topic distributions. The Silhouette Score determines the optimal number of clusters. To ensure interpretability, the system computes descriptive statistics such as average sentiment, rating, and review length for each cluster to generate human-readable labels. For example, clusters

exhibiting negative sentiment and low ratings are automatically labeled **"Critical Issues"** while others are categorized as **"Satisfied Customers", "Mixed Feedback",** or **"Service Concerns"**.

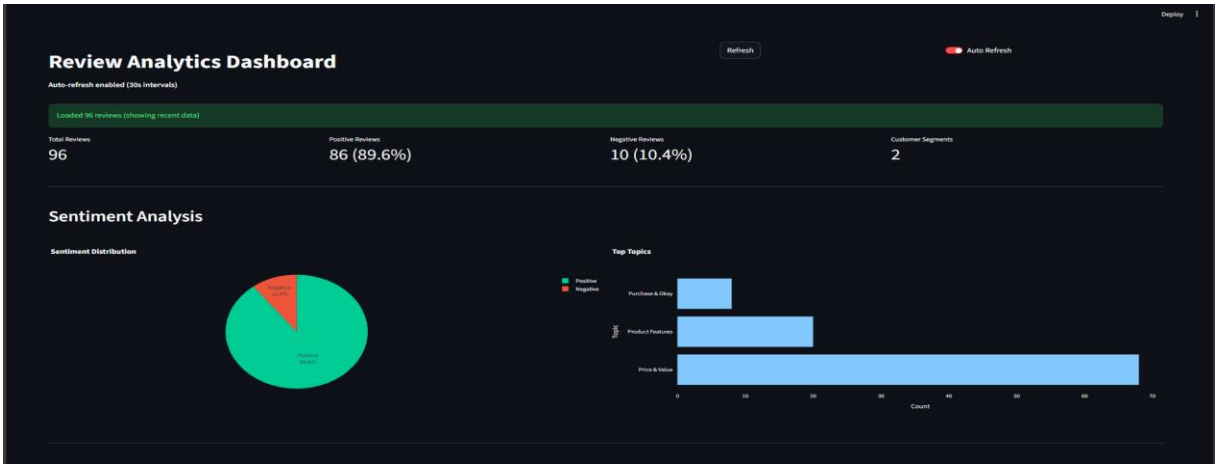
All trained models are stored in HDFS, while model metadata such as identifiers, storage paths, validation performance, timestamps, and algorithm type along with evaluation results persisted in MongoDB for tracking, auditing, and future comparisons.

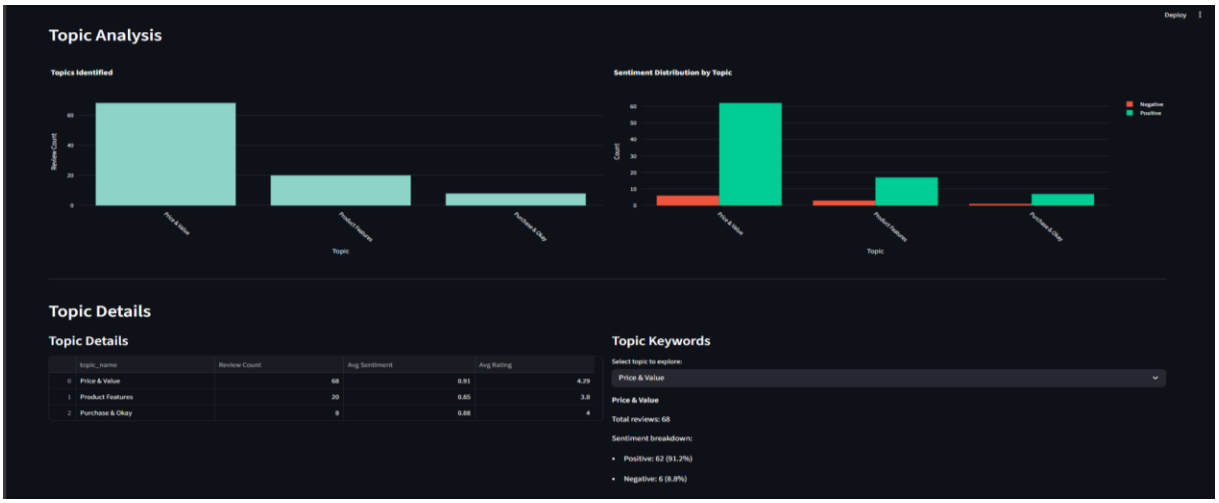
2.4 Anomaly Detection and Alerting

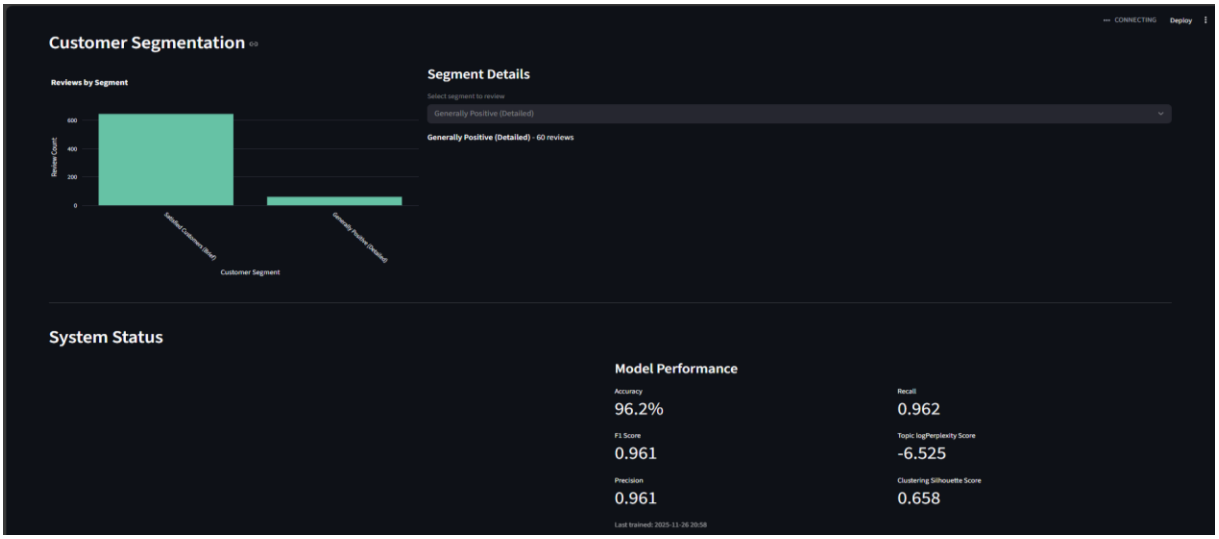
The streaming pipeline incorporates real-time anomaly detection and alerting by analyzing every incoming micro-batch of reviews at 10-second intervals. For each batch, the system calculates the proportion of negative reviews by identifying entries classified with a negative sentiment label and dividing their count by the total batch size. When this percentage exceeds a predefined threshold set at 70%, the system flags the batch as anomalous, indicating a sudden spike in negative customer feedback. In response, it generates a structured alert record containing details such as the alert type, batch identifier, timestamp, number of negative and total reviews, the computed negative percentage, and a human-readable description summarizing the issue. This alert is then stored in the MongoDB alerts collection, creating a persistent log of historical anomalies. On the visualization side, the Streamlit dashboard continuously monitors this collection and prominently displays active alerts in a highlighted banner at the top of the interface, including key metrics and suggested actions. This real-time alerting mechanism ensures that stakeholders can quickly identify emerging issues such as product defects, delivery delays, or service disruptions and take timely corrective action based on actionable insights.

2.5 Visualization

Visualization is delivered through an interactive Streamlit dashboard that continuously queries MongoDB and renders real-time analytics based on the latest streaming and batch-processed review data. The dashboard presents sentiment trends using time-series charts that illustrate how positive, neutral, and negative sentiments evolve over selectable periods such as the last hour, day, or week. Cluster insights are provided through donut charts that display the proportion of reviews belonging to each customer segment such as "Critical Issues", "Satisfied Customers," "Mixed Feedback," and "Service Concerns" with each segment acting as a drill-down filter to explore corresponding reviews for deeper analysis. Topic modeling results are visualized with horizontal bar charts ranking the five extracted topics by frequency, accompanied by tooltips that reveal top keywords to help users understand the underlying themes. A dedicated alert section highlights active anomalies with prominent red banners, listing details such as the affected product, the percentage of negative reviews, and the alert timestamp to ensure critical issues are immediately visible. Additional visual components include rating distribution histograms, enabling comparison of 1-5 star patterns across products, as well as KPI summary cards that present key metrics such as total reviews processed, average sentiment trends, top-reviewed products, and the most common complaint clusters. Interactive filters allow users to segment results by product, sentiment class, date range, or cluster assignment, with all charts updating in real time. To maintain low-latency updates, MongoDB queries are optimized using indexes on identifiers, product fields, and timestamps, allowing the dashboard to refresh every few seconds and provide immediate visibility into emerging customer sentiment patterns and product performance issues.







3. Evaluation

We evaluated the system on two primary dimensions: the predictive quality of the machine learning components and the computational performance of the distributed streaming pipeline under varying load conditions.

3.1 Machine Learning Model Performance

Experiment: We trained a Logistic Regression model for sentiment classification using a historical dataset split into training (70%), validation (15%), and testing (15%) sets. The model utilized a vocabulary of 50,000 words and was optimized using a grid search over regularization parameters.

Result: The model achieved a **96.2% overall accuracy** on the test set.

- **Positive Sentiment:** Achieved a **96.1% F1-score**, indicating high reliability in identifying satisfied customers.
- **Negative Sentiment:** Maintained balanced precision and recall (~90%). This confirms the system is effective at its primary design goal: detecting negative product trends and emerging defects.

3.2 System Scalability and Throughput

Experiment: To validate horizontal scalability, we tested the system's processing efficiency (Actual RPM vs. Target RPM) by increasing the compute resources from 1 to 3 Spark worker nodes while maintaining a constant input load of 60 Reviews Per Minute (RPM).

Result: The system demonstrated clear horizontal scaling benefits.

- **Baseline (1 Worker):** Achieved **101.0% efficiency** (60.6 RPM actual).
- **Optimal Configuration (3 Workers):** Achieved peak performance with **107.0% efficiency** (64.2 RPM actual).
- **Resource Usage:** Memory usage scaled linearly (~3-9% increase per worker), confirming that the system can safely accommodate additional nodes without resource exhaustion.

3.3 Real-Time Latency Analysis

Experiment: We measured the end-to-end latency defined as the time from review ingestion in Kafka to data availability in MongoDB across all operational scenarios to determine if the sub-minute latency requirement.

Result: The average latency remained consistent at **~1300ms** (1.3 seconds) regardless of the worker count.

- **Latency Breakdown:**
  - **Spark Processing:** 87.5% of total time (Primary Bottleneck).
  - **MongoDB Storage:** 12.1%.
  - **Kafka Ingestion:** 0.4%. While Spark processing is the dominant factor, the total latency is well within the real-time design goals.

3.4 Robustness and Stress Testing

Experiment A: Multi-Producer Support - We simulated a complex, high-volume environment by connecting **3 concurrent data producers** generating a combined load of **240 RPM**.

Result: The system handled the increased complexity effortlessly, achieving **100.2% efficiency**. This validates the robustness of the Kafka ingestion layer in decoupling multiple data sources.

Experiment B: Burst Traffic Handling - We subjected the system to variable traffic spikes, oscillating the load between **60 RPM and 300 RPM** to simulate sudden viral events.

Result: This scenario identified a system limitation. The processing efficiency dropped to **43.7%** during peak bursts. While the system did not crash, the throughput gap indicates that message queues build up during extreme spikes, highlighting the need for future auto-scaling mechanisms.

4. Summary

The architecture presented provides a comprehensive solution for real-time product issue detection and trend analytics on a large scale. It combines stream and batch processing using Apache Kafka and Apache Spark, enabling both immediate sentiment classification and historical topic modeling. The use of MongoDB and Streamlit for storage and visualization ensures that critical insights such as sudden spikes in negative feedback or recurring defects are readily available for immediate action. By leveraging these technologies, the system offers a scalable and fault-tolerant platform that transforms unstructured review data into timely, actionable business intelligence.

5. References

- Manias, G., Mavrogiorgou, A., Kiourtis, A., Kakomitas, D., & Kyriazis, D. (2021, December). Real-time kafka-based topic modeling and identification of tweets. In *2021 IEEE International Conference on Progress in Informatics and Computing (PIC)* (pp. 212-218). IEEE.
- https://priscillajosping.medium.com/running-hadoop-on-docker-step-by-step-setup-af06f703196a
- https://www.demandsage.com/online-review-statistics/
- https://spark.apache.org/docs/latest/api/python/reference/pyspark.ml.html