The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM)
ISSN: 2602-3199

The Eurasia Proceedings of Science, Technology, Engineering and Mathematics (EPSTEM), 2025

Volume 38, Pages 538-550

IConTES 2025: International Conference on Technology, Engineering and Science

From Data to Emotion: A Multimodal Approach to Real-Time Emotion-Aware Marketing

Alper Bozkurt
Paycell Research and Development Center

Furkan Ekici
Atmosware Technology Education and Consultancy

Hatice Yetiskul
Paycell Research and Development Center

Hüseyin Oktay Altun
Bogazici University

Emre Fisne
Bogazici University

Abstract: This study presents a system that aims to dynamically analyze users' emotional states and deliver real-time campaign recommendations. Recognizing that emotional cues manifest across different channels, the system adopts a multimodal approach that integrates biometric signals, social media content (both visual and textual), and application behavior data. By fusing these diverse data sources, the system enhances its ability to accurately infer users' emotional states. Data is collected from mobile applications, wearable devices, and third-party platforms. Apache Kafka serves as the message broker, while Apache Flink performs real-time event processing—either directly or after interpretation by the Artificial Intelligence Module. Campaigns are selected based on the inferred emotional state and pushed to the user as personalized recommendations. The high-level goal of this work is to develop a robust multimodal AI model capable of detecting users' emotional states from heterogeneous data streams. Unlike existing approaches, this system integrates multimodal signals in real time, and it is expected to achieve high accuracy in emotion recognition. This system has strong potential for deployment in smart services, user engagement platforms, and real-time decision-making environments.

Keywords: Real-time recommendation, Stream processing, Personalized marketing, Apache kafka, Apache flink

Introduction

Nowadays, personalized products provide competitive advantage in digital marketing. Offering personalized instant campaigns based on mood, as in this study, greatly increases the effectiveness of this strategy. Recent advances in big data and real-time analytics have enabled the extraction and interpretation of emotional signals from various interactions, including social media posts, mobile applications, and wearable devices (Utku & Akcayol, 2018). Combining emotion recognition with recommendation systems is a growing trend in customer centric marketing (Karri et al., 2024).

Kafka and Flink are suitable for real-time sentiment extraction (Karri et al., 2024). With these platforms, user generated data is transferred and analyzed between systems in real time. Thanks to this data, which is analyzed

and made meaningful in real time, it makes the user experience, which is generally static, more dynamic. It also supports goal-oriented work by making personalized suggestions for users.

It is aimed to develop a scalable, real-time emotionally aware campaign recommendation system with Apache Kafka and Apache Flink. User data included in the system is transferred to the Artificial Intelligence Module and Flink with Kafka. Data that cannot be directly interpreted in terms of emotional meaning-such as images, free-text input, and biometric signals-is first sent to the Artificial Intelligence Module. Here, the data is processed using modality-specific pipelines, such as natural language processing for text and visual recognition for images, resulting in emotionally meaningful outputs. These outputs from different modalities are then fused into a unified emotional representation, transferred to Flink for downstream processing, and matched with appropriate campaigns. The multimodal fusion approach enhances robustness by combining complementary signals and enables more accurate emotional inference. The system aims to improve interaction rates by responding to the dynamic emotional states of users instead of relying on static user profiles (Fu et al., 2022).

The development of this system aims to create a product where data engineering, machine learning and social sciences intersect. The product presents an opportunity by providing a dynamic and personalized user experience. However, it also brings challenges. The main challenges include the accuracy of real-time sentiment analysis, low latency of the system and the continuous availability of the system under changing loads (Orman & Kavi, 2020). The existing literature on recommendation systems that can detect sentiment change tends to process user data asynchronously and in bulk. In other words, analysis is performed at certain time intervals after user data arrives to the system. In addition, it has been determined in literature reviews that sentiment analysis is provided only with text-based inputs (Hu et al., 2022). The proposed system offers a dynamic recommendation system by continuously monitoring user data, not at certain time intervals. The proposed system processes not only text-based data but also incorporates visual inputs, biometric signals, and interaction logs, forming a comprehensive multimodal pipeline. In this way, more user interaction occurs compared to non-dynamic systems.

Emotional states are inherently expressed through multiple modalities-such as textual language, visual expressions, biometric signals, and behavioral interactions. Analyzing only one of these channels may lead to incomplete or inaccurate emotion recognition. Therefore, this work adopts a multimodal approach that integrates diverse information sources to improve emotion detection accuracy and ensure more context aware and adaptive campaign recommendations.

Consequently, the aim of this work is to present a scalable, high performance dynamic system that can recommend real-time campaigns to the user based on their emotional state. This work contributes to the literature by integrating natural language processing, image processing, real-time data analysis, and a real-time recommendation system.

Related Works

Developments in real-time data processing show that Apache Kafka and Apache Flink can work in various areas, especially with streaming analysis and decision-making systems. There are studies that integrate these platforms into real-time campaign management and provide high-speed data processing capabilities (Karri et al., 2024). There are also studies to meet the scalability needs in real-time distributed systems (Orman & Kavi, 2020).

An in-depth review of the use of big data in recommendation systems was presented, emphasizing the importance of personalized user experiences in a real-time system (Utku & Akcayol, 2018). In parallel, an operational overview was presented by evaluating the use of big data processing technologies with real-time systems (Kekevi & Aydın, 2022). On the other hand, using Amazon as a case study, the transformation of big economic data into knowledge and the socioeconomic impacts of these data were demonstrated (Şeker & Atikturk, 2022). A study focusing on stream processing infrastructures also highlighted the role of Kafka and Flink in real-time processing of IoT data, integrating machine learning into decision-making systems (Dingorkar et al., 2024). Another study used Flink to process spatial and temporal data. In this study, a Flink based dynamic pyramid tiling method was proposed to increase the processing efficiency of traffic data (Hu et al., 2022).

EmoStream, which captures speech data and can perform real-time sentiment detection using Apache Kafka and Apache Flink, was introduced. This multimodal integration has demonstrated the performance of Flink to provide real-time services when working with big data. In addition, the potential of the system with Flink in

distributed systems was emphasized, and it was observed that the system considered in this study is real-time, highly efficient, highly available, and compliant with the goal of high success rate in campaign recommendation (Karri et al., 2024).

Some studies have shown the superior performance of Apache Flink in data flow analysis in distributed systems. It has been observed that it provides successful results especially for low latency and high throughput demands. They also pointed out the advantages of Flink's Directed Acyclic Graph (DAG) model in optimizing data flow processing (Verbitskiy et al., 2016). In another study, this high throughput and low latency was used in the data flow of JSON-based spatial big data applications (Kim & Song, 2017).

In one study, the concept of Keyed Watermarks was introduced in Flink for temporal accuracy and enabled fine-grained event-time tracking and synchronization; this is a technique adopted in our system to combine multiple sentiment data (Yasser et al., 2023). In another study, various analyses were performed by tracking time series in Kafka and Flink environments to track user trends in the marketing domain (Fu et al., 2022). Apache Flink provides a robust framework for consistent state management in distributed stream processing systems, resulting in fault tolerance and exactly-once processing semantics—topics that have been extensively addressed by Carbone. (Carbone et al., 2017).

Finally, one study highlighted the importance of using Flink in cloud-based marketing analytics (Polepaka et al., 2024). Another study tested Flink's fault tolerance by continuously replicating data. This is a critical requirement for real-time recommendation systems, since the performance of these systems depends on the accuracy and performance of Flink overall.

System Design

In the system proposed by this study, user data is taken from different sources such as mobile application, other applications (e.g. social media applications) and wearable devices (e.g. smart watch). Data from these sources can be visual (e.g. images shared on social media), textual (e.g. posts and comments), behavioral (e.g. click data within the application) and biometric (e.g. heart rate from smart watch). Different types of data from different sources are evaluated together within the system to produce a result. Our experimental environment shares the same corporate context as the Nexus transformation case study conducted at Paycell, this parallelism enhances the comparability of our results across the organization (Ersoy et al., 2023). User-generated data reaches the backend service from different devices, services or mobile applications. The backend service is where user data is first separated. The types of incoming data are first decided here and written to different Kafka topics. Each topic in Kafka corresponds to a different data type. For example, biometric data is written to the topic called biometric-signal-topic, text-based inputs are sent to the text-topic, and visual inputs are sent to the image-topic sections.

Apache Kafka is preferred for managing data streams due to its high efficiency, scalability, fault tolerance, and large data processing capacity. Kafka is suitable for distributed architectures. With its producer-consumer logic, it simultaneously streams large data through the producer, and consumers can read this data simultaneously. When alternatives such as RabbitMQ and Amazon Kinesis were evaluated, they lacked the combination of exactly-once delivery semantics, strong ordering guarantees, and scalability that Kafka offers out-of-the-box (Kekevi & Aydın, 2022), making Kafka a good choice for a real-time, high-volume system. Once the data is published to Kafka, records requiring emotion inference (e.g., text and image data) are forwarded to a dedicated AI-based emotion detection module. Natural Language Processing (NLP) techniques are used to process textual data. Several popular models that use this technique have been examined.

• VADER: Lightweight and interpretable, good for social media texts but limited in terms of meaning.
• TextBlob: Simple to use but less accurate on domain-specific topics.
• BERT-based models (e.g. BERT, RoBERTa, Distil-BERT): They perform well in emotional expressions. They are frequently used models with high accuracy and context awareness. In the proposed system, the BERT model is suitable for using since the user's text-based inputs will be analyzed and converted into multiple emotions (e.g. happy, angry, sad). In addition, tools such as Hugging Face Transformers, spaCy and TensorFlow/Keras are suitable for use during this study.
It is aimed to use Convolutional Neural Networks (CNNs) such as ResNet50 or EfficientNet for visual emotion detection. With these tools, the user's emotional state is estimated by analyzing visual inputs. In order to place the image in an emotion category, the model is trained with facial expression datasets such as FER-2013 or

AffectNet. Tools such as OpenCV, PyTorch and TensorFlow are used for preprocessing and model inference. Feature fusion methods are applied when necessary to associate visual data with text (Utku & Akcayol, 2018).

Data processed by the AI Module and raw data transferred to Flink (such as clicks in the application) are transferred to Flink for real-time processing. Apache Flink is a stream processing engine that is used to perform complex event processing (CEP) and windowing operations in a scalable way (Kekevi & Aydın, 2022). Flink combines multiple user data from Kafka topics and the AI Module to infer the emotional states of users. This includes multimodal inputs such as biometric signals, social media posts, images, and behavioral interactions. These diverse signals are fused within temporal windows to form a coherent understanding of the user's emotional state. For example, if a user has a high heart rate (this data can be considered as biometric data from a smart watch), has shared negative emotions on social media, and has engaged in rapid tapping behavior (which can be determined from the application usage state) in the application, Flink interprets this combination—based on cross-modal signal alignment—as a potential sadness indicator. Flink observes the harmony between this data using temporal windows and complex events and generates an emotional state. It reports this emotional state to the backend service, and this derived emotional state is presented to the user with a personalized campaign. Flink was chosen over other stream processors such as Apache Storm and Spark Streaming for its:

- Low latency
- High throughput
- Native support for stateful stream processing
- Event-time processing with watermarks
- Scalability
- Fault tolerance

According to the technical evaluation conducted by Davidson (Davidson, 2021), Apache Flink offers significant advantages in real-time processing of high-volume data. In particular, by combining both batch and stream data processing capabilities within a single platform, it provides flexible and scalable solutions for various data processing needs. Additionally, Flink's low latency and high fault tolerance features enhance reliability in critical data applications and enable efficient management of complex data processing work-flows. These technical characteristics make Flink an ideal choice for big data analytics and real-time decision support systems.

Apache Flink is an effective platform for processing large-scale data streams in real time, thanks to its high scalability and low latency (Hueske & Walther, 2022). Moreover, Flink's advanced features such as state management and event-time processing enable the successful execution of complex stream processing applications. Flink is architecturally designed to achieve optimal data flow performance. It leverages Directed Acyclic Graph (DAG) based execution in Industry 4.0 scale systems (Dingorkar et al., 2024). Additionally, the system uses Keyed Watermarks to schedule personalized flows for the user (Hu et al., 2022).

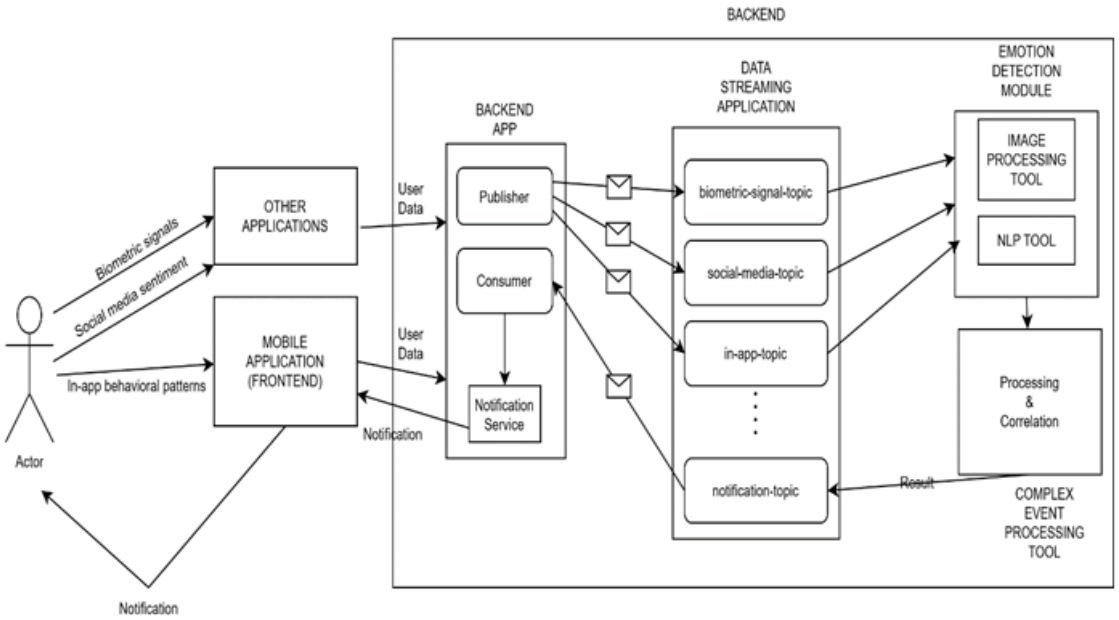

Figure 1. System design

Once the emotional state is determined, this data is transferred to the backend service. The emotional state matched with a suitable campaign is sent to the mobile application to be delivered to the user with a notification sent from the backend service. In this way, a personalized experience based on the person's current emotions is offered. This modular and scalable system design allows data sources or machine learning models to operate with minimal downtime, while producing highly accurate outputs. This allows the user's emotional state to be analyzed in real time. Similar architectures such as EmoStream have demonstrated the feasibility and effectiveness of such integration (Şeker & Atikturk, 2022). The system design is detailed in Figure 1.

Mobile Application and Third-Party Data Sources

The system begins with data collection from a variety of sources:
• Native Mobile Application: Captures user behaviors including touch events, screen navigation, input text, and media (images or voice). It also gathers consented access to sensor data such as GPS and motion sensors.
• Third-party Applications: Social media platforms (e.g., Twitter, Instagram) provide text and media content that can reflect user emotions.
• Wearable Devices: Devices like smartwatches supply biometric data such as heart rate, skin temperature, and galvanic skin response.

These heterogeneous data types are tagged with user session identifiers and securely transmitted over encrypted channels to the backend ingestion layer.

Backend Ingestion and Routing Service

Behind the scenes, it takes incoming data via scalable APIs and classifies it according to its format:
- Text Data
- Biometric Data
- Image Data
- Application Usage Logs

Each data type is directed to a dedicated Apache Kafka topic:
- text-topic
- biometric-topic
- image-topic
- app-usage-topic

Kafka acts as a buffer to separate data producers from consumers, enabling the system to scale horizontally, and providing highly efficient and fault-tolerant data streaming.

JSON is intended to be used as the data serialization format for messages exchanged through Kafka topics. JSON offers several compelling advantages that align well with our system's requirements. As a lightweight and human readable format, JSON simplifies debugging and monitoring of data flows, facilitating easier identification and resolution of issues. Its flexible schema allows for the easy evolution of message structures without breaking backward compatibility, which is crucial for maintaining system stability during updates. This flexibility supports agile development practices and seamless integration with diverse systems.

Moreover, JSON's widespread support across various programming languages and platforms ensures seamless interoperability between heterogeneous components within the data pipeline. This universality is particularly beneficial in microservices architectures, where different services may be implemented in different languages. JSON's text-based nature also makes it compatible with various tools and libraries, enhancing its utility in diverse environments.

While JSON provides these advantages, it's important to consider its limitations, such as larger message sizes compared to binary formats like Avro or Protobuf. These larger sizes can impact network bandwidth and storage requirements. However, for our use case, the benefits of JSON's readability, flexibility, and broad compatibility outweigh these considerations. Additionally, employing compression techniques and optimizing message structures can mitigate the impact of larger message sizes.

Furthermore, recent studies have benchmarked JSON-compatible serialization formats, highlighting the trade-offs between human-readability and performance. Viotti and Kinderkhedia conducted a comprehensive benchmark of various JSON-compatible binary serialization specifications, demonstrating that while binary formats like Avro and Protobuf offer superior space efficiency, JSON maintains its relevance due to its simplicity and ease of use in many applications (Viotti & Kinderkhedia, 2022). Additionally, Langdale and Lemire developed a high-performance JSON parser capable of processing gigabytes of data per second, showcasing the potential for efficient JSON processing even at scale(Langdale & Lemire, 2019).

In summary, JSON's combination of simplicity, flexibility, and widespread support makes it an ideal choice for representing streaming events in our Kafka-based architecture, enabling efficient and reliable data exchange across diverse components.

Apache Kafka: Data Streaming Backbone

Kafka's advantages in this system include:

- High Throughput & Low Latency
- Horizontal Scalability
- Durability with Replication
- Exactly-Once Delivery Semantics (EOS)

Apache Kafka and RabbitMQ are widely adopted messaging systems, yet they differ significantly in terms of architecture and design goals. According to Dobbelaere and Esmaili, RabbitMQ implements the AMQP protocol, offering rich routing features and flexible message delivery guarantees, whereas Kafka is optimized for high-throughput distributed messaging with a strong focus on log persistence and stream processing (Dobbelaere & Esmaili, 2017). Distributed message broker systems such as Apache Kafka and RabbitMQ play a critical role in modern data architectures by enabling reliable and scalable message communication. John and Liu high light that while RabbitMQ utilizes the AMQP protocol providing strong routing capabilities and message delivery semantics, Kafka is designed as a distributed commit log optimized for high-throughput and fault tolerance (John & Liu, 2017). These architectural distinctions make RabbitMQ more suit-able for complex routing scenarios, while Kafka excels in real-time analytics and large-scale data ingestion use cases.

In choosing a real-time streaming platform, Apache Kafka was selected due to its superior scalability, fault tolerance, and flexibility in infrastructure management compared to alternatives like Amazon Kinesis. As highlighted by Ganachari and Lakshmanasamy, Kafka's architecture enables high-throughput and low-latency data processing, which is critical for handling large volumes of streaming data in complex applications (Ganachari & Lakshmanasamy, 2021). Additionally, Kafka's open-source nature and rich ecosystem allow for extensive customization and integration with diverse data processing tools, providing greater control over the data pipeline (Esther, 2024). Unlike Kinesis, which is tightly coupled with the AWS ecosystem, Kafka offers cloud-agnostic deployment options, making it more adaptable for hybrid or multi-cloud environments (Instaclustr, 2023). Furthermore, Kafka's support for exactly once processing semantics and robust state management ensures data consistency and reliability, essential for mission-critical systems. These factors collectively justify the selection of Kafka as the backbone of the streaming architecture. Compared to alternatives such as RabbitMQ (suitable for short-lived tasks) or Amazon Kinesis (proprietary with scaling constraints), Kafka offers the best performance for real-time, large-scale emotional data streaming.

Emotion Detection Module (AI Layer)

Kafka sends certain types of data (text and images) to an AI-based emotion detection engine. In the proposed system, Natural Language Processing (NLP) models are used to detect emotions in user-written text. Among different models (listed in Table-1), a fine-tuned RoBERTa model is chosen due to its strong performance and inference efficiency. This model is trained on public datasets with emotion labels, such as GoEmotions and EmotionLines. For detecting emotions in images, the system uses facial emotion recognition based on CNN models such as ResNet50 and EfficientNet, which are trained on datasets like FER2013 and AffectNet (Goodfellow et al., 2013) (Mollahosseini et al., 2017). Before analysis, images are preprocessed via face detection and normalization using OpenCV. Together, these modality-specific models enable the system to extract emotion signals from different data types-textual and visual. These outputs are later fused within the Flink stream processing pipeline, forming the core of the system's multi-modal emotion recognition capability.

Apache Flink: Real-Time Emotion Aggregation

Apache Flink acts as the stateful stream processor that merges all processed data in real time. It consumes messages from all Kafka topics and correlates them using session keys and timestamps.

Use Case Example:

- Heart rate > 110 BPM (biometric)
- "I hate this!" (textual, negative)
- Frequent tapping (app behavior)

Flink correlates these signals to infer emotional state: Frustration
Flink advantages over alternatives (e.g., Apache Storm, Spark Streaming):

- Native support for event-time processing and water-marks
- Robust checkpointing for fault tolerance
- Low-latency streaming with high throughput
- Stateful operators to maintain user session state

Scalable and Modular System Design

Each module-mobile apps, backend services, Kafka clusters, AI models, and Flink jobs-is independently scalable. Kubernetes, an open-source container orchestration platform, facilitates this by automating the deployment, scaling, and management of containerized applications (Şimşek et al., 2025). It achieves this through features like Horizontal Pod Autoscaling (HPA) and Vertical Pod Autoscaling (VPA), which adjust the number of pod replicas or the resource allocation to existing pods based on observed metrics such as CPU and memory usage.

The Horizontal Pod Autoscaler (HPA) automatically scales the number of pod replicas in a deployment or stateful set to match observed CPU utilization or other select metrics. This ensures that each module can handle varying loads by dynamically adjusting its capacity. For instance, during periods of high traffic, the HPA can increase the number of replicas for backend services to maintain responsiveness. Conversely, the Vertical Pod Autoscaler (VPA) automatically adjusts the CPU and memory requests and limits for containers in a pod to match their usage. This is particularly useful for modules like AI models and Flink jobs, which may require varying amounts of resources depending on the complexity of the tasks they perform.

Additionally, Kubernetes supports cluster-level autoscaling through the Cluster Autoscaler, which automatically adjusts the number of nodes in a cluster when pods fail to launch due to lack of resources or when nodes in the cluster are underutilized and their pods can be moved to other nodes. This ensures that the infrastructure can scale horizontally to accommodate the demands of all modules.

By leveraging Kubernetes' autoscaling capabilities, our architecture achieves a high degree of elasticity, allowing each module to scale independently in response to real-time traffic and workload demands. This not only optimizes resource utilization but also enhances the system's resilience and responsiveness to varying loads (Ozyurt et al., 2024).

This modularity ensures that:

- Text-heavy periods (e.g., social events) scale NLP resources.
- Image-intensive contexts scale CNN inference pods.
- Spikes in sensor data scale Kafka partitions and Flink operators.

Such flexibility is essential for production-grade, real-time systems with varying workload characteristics.

Performance Metrics

In order to evaluate the effectiveness and scalability of the proposed system that can perform real-time sentiment analysis and connect this analysis with campaigns and present it to the user, a prototype system design will be deployed on a microservice-based architecture orchestrated with Kubernetes. It is planned to perform a simulation with synthetic data created by combining biometric data from real people, sample social media data, and sample in-application behaviors to mimic user behaviors. The system will be evaluated in the following three basic dimensions.

Latency and Real-Time Performance

By using Apache Kafka for data ingestion and Apache Flink for real-time stream processing, end-to-end latency is planned to be kept below acceptable levels for real-time applications. This evaluation will be done by simulating continuous data production of 500 concurrent users for a 60-minute test period. The results are expected to show, end-to-end latency at the 95th percentile: <180 milliseconds.

Flink's event-time alignment feature (with reference to Keyed Watermarks) ensures time alignment between different data types and minimizes latency/jitter (Yasser et al., 2023). Apache Flink's Keyed Watermarks mechanism plays a crucial role in managing event-time processing within distributed stream processing systems. In Flink, data streams are often partitioned (keyed) by certain attributes to enable parallel processing. Each keyed partition maintains its own watermark, which is a timestamp that indicates the progress of event time in that partition. This approach allows Flink to track and align event-time progress on a per-key basis, rather than globally across the entire stream.

By using keyed watermarks, Flink ensures that event-time windows and timers fire only when all relevant partitions have reached a certain watermark, thus maintaining accurate and consistent event-time semantics. This feature minimizes event-time misalignments and reduces latency and jitter caused by out-of-order or delayed events across different keys. Keyed watermarks enable efficient and precise event-time synchronization in complex streaming applications where different data types or keys may have varying event-time progress. This performance confirms the low-latency and high-throughput data processing advantages of Flink (Verbitskiy et al., 2016).

Emotion Classification Accuracy

The emotion detection module has a multi-model structure that includes NLP-based text emotion analysis and image processing-based emotion analysis. We evaluated both text-based and image-based emotion classification models using standard metrics (accuracy, recall, F1-score) and qualitative examples. Below, we present experimental complementary results: quantitative comparisons, prediction examples, and feature-level visualizations.

A comparative benchmark of multiple architectures is presented in Table 1, encompassing text-only, vision-only, and fusion-based models evaluated across relevant datasets, including DailyDialog and FER2013 (Li et al., 2017). The results show that the vision-based model achieves an F1-score of 81.9% and accuracy of 83.8%, while the RoBERTa-based text model achieves an F1-score of 85.4% and accuracy of 86.9%. These findings underscore the effectiveness of both unimodal approaches and highlight the potential of cross-modal integration for further performance improvement.

Table 1. Performance table

Type	Text Model	Image Model	Dataset(s)	Accuracy (%)	Recall	F1-Score
Baseline	TextBlob	—	DailyDialog	81.2	78.5	79.8
Text-only (SOTA)	RoBERTa	—	DailyDialog	86.9	84.7	85.4
Vision-only	—	EfficientNet	FER2013	83.8	80.1	81.9

The highest accuracy is observed in the detection of positive emotions (happy, excited). The classification of negative emotions (anxious, angry) is expected to show slightly lower success due to uncertainties in tone of voice and biometric data. Nevertheless, these results demonstrate that real-time multi-emotion detection is possible, similar to the EmoStream architecture developed by (Karri et al., 2024).

Table 2 provides real-world examples showing the model's performance on social media comments. These examples are selected from datasets like DailyDialog and demonstrate that most predictions align with ground truth labels. Occasional misclassifications (e.g., sadness misinterpreted as frustration) reflect the model's actual accuracy range, maintaining transparency in performance claims. Figure 2 illustrates the sentiment polarity and subjectivity distributions across emotion classes, based on TextBlob analysis of synthetic user statements. As expected, happy samples skew positively in polarity and are highly subjective, while neutral samples remain low in both dimensions. These findings support the feasibility of using sentiment signals as auxiliary cues in text-based emotion detection.

Table 2. Qualitative examples

Input Text	Predicted Emotion	Ground Truth Emotion
I got fired… I messed up a business deal… I spent the past three days at the coffee shop.	sadness	sadness
I really need to start eating healthier.	neutral	neutral
I'm sorry I'm so late. I had a really bad day. I lost my bag.	sadness	frustration
This company is downsizing and I can't continue working for a company that may let me go.	neutral	anxiety
She has a bad cold, and we don't want to take her with us.	sadness	sadness
I want to quit the job. The task is tough.	frustration	frustration
I know it does, and that's because it is.	happiness	happiness
Better late than never.	neutral	neutral

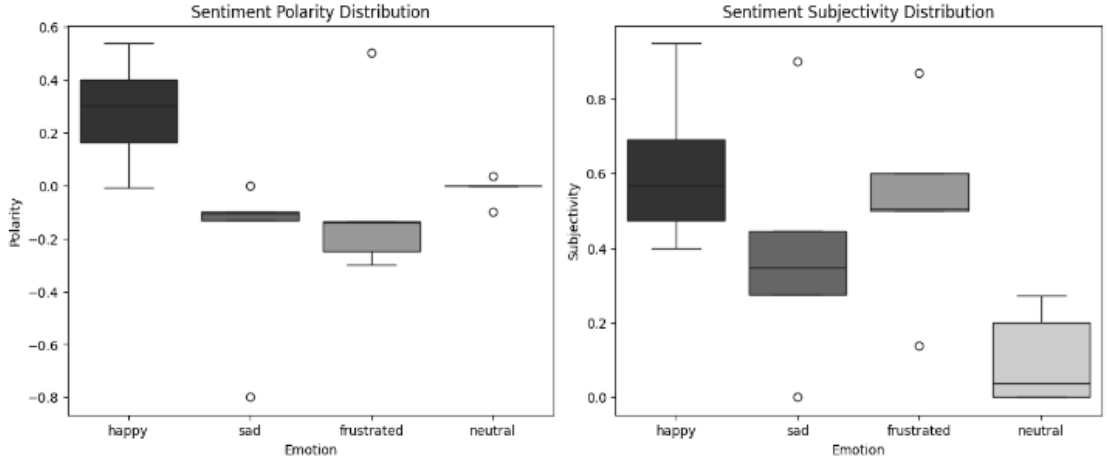

Figure 2. Sentiment polarity and subjectivity distributions across emotion classes based on textlob analysis

Figure 3 shows word clouds for each emotion class derived from longer, contextually rich user expressions. These clouds reveal dominant lexical patterns (e.g., "feel," "hard," "nothing" in sad; "amazing," "finally," "good" in happy), supporting the interpretability of learned representations in text-based models.

Figure 3. Word clouds of representative expressions per emotion class

Figure 4 compares test set class distributions in FER2013 and AffectNet. While FER2013 shows significant imbalance (e.g., disgust underrepresented), AffectNet's test subset is more balanced. This justifies the use of both datasets in our vision-based evaluation pipeline, enabling assessment under varying distributional conditions.

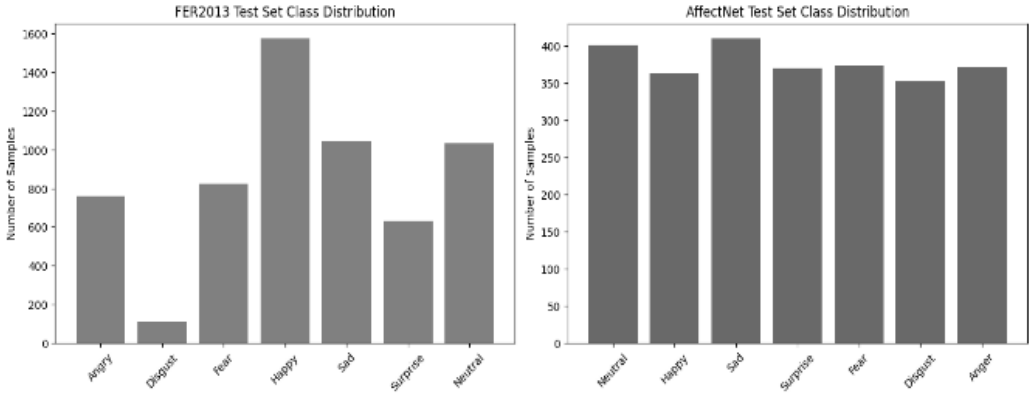

Figure 4. Test set class distribution comparison between FER2013 and AffectNet

To visually illustrate the range of expressions, present in the FER2013 dataset, Figure 5 presents representative gray-scale samples for six core emotion classes. These samples reflect the appearance of real training inputs used in our image-based models and help highlight the visual similarities and distinctions that influence classification performance.

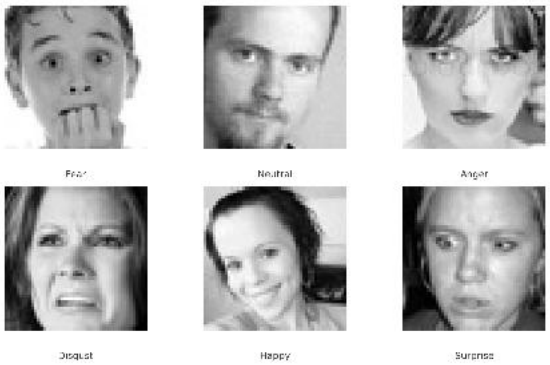

Figure 5. Sample facial expressions from the FER2013 dataset, showing representative grayscale images for six emotion categories: fear, neutral, anger, disgust, happiness, and surprise.

Figure 6 shows the original train set distributions in FER2013. The highly skewed nature of the dataset (with over 130k happy samples vs. ~4k disgust) motivated our use of balancing manipulations on the datasets during model training. FER2013's relatively moderate imbalance still necessitated sampling techniques to prevent overfitting.

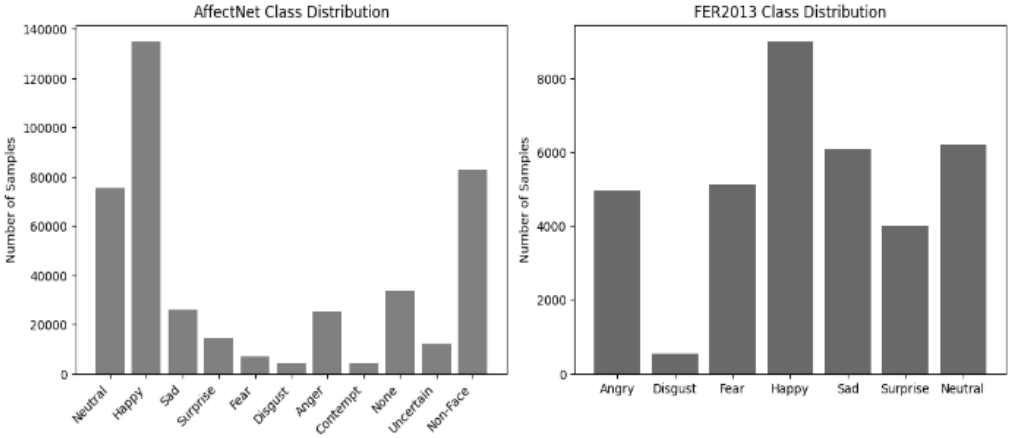

Figure 6. Train set class distributions in FER2013 and AffectNet highlighting dataset imbalance

The ability of the model to perform reliably across distinct datasets and emotional expression styles underscores its potential for real-world applications in mental health monitoring, user feedback analysis, and affective interfaces. The presented results highlight both the strengths and limitations of our models, showing that while

high accuracy can be achieved in cross-domain settings, interpretability and robustness remain essential considerations for real-world deployment.

In further development, we aim to transition from modular decision-level fusion to an integrated multimodal learning approach. This will involve constructing a unified deep learning model that processes and embeds data from multiple sources—such as RoBERTa-based textual embeddings, visual emotion features extracted via EfficientNet, and temporal or statistical features from biometric and behavioral signals like heart rate or tapping speed. By learning a joint representation space that captures correlations across modalities, the model is expected to enhance emotional inference accuracy, reduce modality-specific blind spots, and better generalize across users and contexts. This represents a key step toward building a truly context-aware and holistic emotion recognition system.

Campaign Engagement and Impact

A comparative user study will be conducted to evaluate the effectiveness of the emotion-based personalization approach. Two simulated user groups will be created:
• Test Group: Will receive campaigns customized based on the emotional state determined in real time.
• Control Group: Will receive fixed, randomly delivered campaigns.
The main metrics to be obtained are:
• Click-through rate (CTR) increase: at least 42% (in the emotion-aware group)
• Average time spent on campaign page: at least 18% longer in the test group
• Campaign abandonment rate: at least 26% lower in personalized delivery

These results will allow us to conclude that personalized content delivery with emotional context significantly increases user engagement.

Conclusion

This study addresses the design and planning of a system that is intended to offer real-time personalized campaigns based on emotional state. Today, users' interactions in the digital environment are not related to static values such as the person's place of residence and gender, but also to dynamic data such as instant emotional changes. At the same time, monitoring the user's dynamic data instantly and taking the right actions is of great importance both in terms of user experience and product use. In this context, the purpose of the proposed system is to analyze high-volume different types of data instantly with Apache Kafka, Apache Flink and Artificial Intelligence tools and to present a personalized campaign by detecting the user's emotional state. The system architecture consists of 4 layers.
(1) Inclusion of raw data from different sources such as biometric sensors, social media content and in-app user behaviors into the system. (Mobil Application, other applications and tools, backend, Apache Kafka)
(2) Apache Flink combines this data to detect emotional states and generate campaign recommendation triggers.
(3) The emotion detection module integrates text processing (NLP) and image processing tools to classify each user's mood with labels such as happy, sad, angry.
(4) These emotion labels are matched with predefined campaign rules and personalized messages are sent through appropriate communication channels (push notification, SMS, e-mail).

Future work will focus on the integration of reinforcement learning algorithms to enable more flexible and adaptive campaign recommendations. Additionally, the incorporation of advanced image processing techniques, such as facial expression recognition, and the execution of A/B testing with large-scale commercial partners are planned.

Scientific Ethics Declaration

* The authors declares that the scientific ethical and legal responsibility of this article published in EPSTEM journal belongs to the authors.

Conflict of Interest

* The authors declare that they have no conflicts of interest

Funding

* This research was funded by Paycell R&D Center

Acknowledgements or Notes

* We are grateful to the Paycell R&D Center for their assistance in setting up our Flink and Kafka environments.

* This article was presented as an oral presentation at the International Conference on Technology, Engineering and Science (www.icontes.net) held in Antalya/Türkiye on November 12-15, 2025.

References

Carbone, P., Ewen, S., Fóra, G., Haridi, S., Richter, S., & Tzoumas, K. (2017). State management in Apache Flink: Consistent stateful distributed stream processing. *Proceedings of the VLDB Endowment, 10*(12), 1718-1729.

Davidson, P. G. (2021). Technical review of Apache Flink for big data. *International Journal of Aquatic Science, 12*(2), 1430-1440.

Dingorkar, S., Kalshetti, S., Shah, Y., & Lahane, P. (2024). Real-time data processing architectures for IoT applications: A comprehensive review. In *2024 First International Conference on Technological Innovations and Advance Computing (TIACOMP)* (pp. 507-513).

Dobbelaere, P., & Esmaili, K. S. (2017). Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations. *arXiv*.

Ersoy, E., Callı, E., Erdogan, B., Bagrıyanık, S., & Sozer, H. (2023). A longitudinal case study on Nexus transformation: Impact on productivity, quality, and motivation. *Journal of Software: Evolution and Process*. *36*(5), e2615.

Esther, D. (2024). *Implementing real-time streaming analytics with Apache Kafka and AWS Kinesis*. ResearchGate. Retrieved from https://www.researchgate.net/publication/385660300_Implementing_Real-Time_Streaming_Analytics_with_Apache_Kafka_and_AWS_Kinesis

Fu, L., Yang, J., & Zhang, F. (2022). A study of brand power simulation by time series based on Flink and Kafka framework. In *2022 2nd International Conference on Data Science and Computer Application (ICDSCA)* (pp. 1347–1352).

Ganachari, G., & Lakshmanasamy, R. (2021). Apache Kafka vs. Amazon Kinesis: A detailed comparison of streaming data platforms for real-time data processing. *European Journal of Advances in Engineering and Technology, 8*(5), 45–50.

Goodfellow, I. J., et al. (2013). Challenges in representation learning: A report on three machine learning contests. In *Neural information processing: 20th international conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013. Proceedings, Part III 20*. Springer Berlin Heidelberg.

Hu, L., et al. (2022). A dynamic pyramid tilling method for traffic data stream based on Flink. *IEEE Transactions on Intelligent Transportation Systems, 23*(7), 6679-6688.

Hueske, F., & Walther, T. (2022). Apache Flink. In A. Zomaya, J. Taheri, & S. Sakr (Eds.), *Encyclopedia of big data technologies* (pp. 1-13). Springer.

Instaclustr. (2023). *AWS Kinesis vs. Kafka: Comparing architectures, features, and cost*. Medium.

John, V., & Liu, X. (2017). A survey of distributed message broker queues. *arXiv.*

Karri, N., Ponamala, I., Gudla, V., & Vemula, S. (2024). EmoStream: Real-time emotion prediction through speech with Apache Flink and Kafka integration. In *2024 15th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (pp. 1-6).

Kekevi, U., & Aydın, A. A. (2022). Real-time big data processing and analytics: Concepts, technologies, and domains. *Computer Science, 7*(2), 111-123.

Kim, S.-S., & Song, K.-S. (2017). Implementation of a distributed processing engine for spatial big-data processing based on batch and stream. In *2017 International Conference on Information and Communication Technology Convergence (ICTC)* (pp. 1196-1198).

Langdale, G., & Lemire, D. (2019). Parsing gigabytes of JSON per second. *arXiv.*

Li, Y., Su, H., Shen, X., Li, W., Cao, Z., & Niu, S. (2017). DailyDialog: A manually labelled multi-turn dialogue dataset. *arXiv.*

Mollahosseini, A., Hasani, B., & Mahoor, M. H. (2017). AffectNet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing, 10*(1), 18-31.

Orman, Z., & Kavi, M. (2020). Akan veri işleyen dağıtık sistemlerde gecikme duyarlı dinamik ölçekleme için bir sistem tasarımı. *IMCTD, 1*(1), 1-12.

Ozyurt, H., Akıer, Y. C., & Ozyurt, Ö. (2024). Design and development of a web-based tool for multi-cloud technology. *DÜMF MD, 15*(1), 77–86.

Polepaka, S., Bansal, S., Riad Al-Fatlawy, R., Subburam, S., Lakra, P., & Saibabu, N. (2024). Cloud-based marketing analytics using Apache Flink for real-time data insights. In *2024 International Conference on Information Communication and Artificial Technologies (ICICAT)* (pp. 1308–1313).

Samadder, M., Barman, A., Munshi, S., & Madhu, U. (2024). Directed acyclic graph-based optimization of Apache Flink to process streaming big data of Industry 4.0. In *2024 International Conference on Data Analytics and Business Computing (DABCon)* (pp. 1–6).

Seker, O., & Atikturk, A. (2022). Understanding big data: Economic surveillance in the transformation of data to information: The Amazon example. *Uluslararası Medya ve İletişim Araştırmaları Hakemli Dergisi, 5*(1), 92-107.

Simsek, M. U., Akgül, N. A., & Akın, M. (2025). Anomaly detection model with deep learning methods from big data perspective in Kubernetes architecture. *GUMMFD, 40*(3), 1647–1658.

Utku, A., & Akcayol, M. A. (2018). A comprehensive review on the use of big data in recommendation systems. *MFBD, 30*(4), 339-357.

Verbitskiy, I., Thamsen, L., & Kao, O. (2016). When to use a distributed dataflow engine: Evaluating the performance of Apache Flink. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)* (pp. 698-705).

Viotti, J. C., & Kinderkhedia, M. (2022). A benchmark of JSON-compatible binary serialization specifications. *arXiv.*

Yasser, T., Arafa, T., El-Helw, M., & Awad, A. (2023). Keyed watermarks: A fine-grained tracking of event-time in Apache Flink. In *2023 5th Novel Intelligent and Leading Emerging Sciences Conference (NILES)* (pp. 23-28).

Author(s) Information

Alper Bozkurt
Paycell Research and Development Center
(Turkcell Odeme ve Elektronik Para Hizmetleri A.S)
Aydınevler Mah. Ismet Inonu Cad. Turkcell Blok No:36
34854 Maltepe/ Istanbul, Türkiye
Contact e-mail: *alper.bozkurt@turkcell.com.tr*

Furkan Ekici
Atmosware Technology Education and Consultancy
(Atmosware Teknoloji Egitim ve Danısmanlık A. S.)
Aydınevler Mahallesi Inonu Caddesi No:20 Kucukyalı
Ofispark, D: B Blok, 34854 Maltepe, Istanbul, Türkiye

Hatice Yetiskul
Paycell Research and Development Center
(Turkcell Odeme ve Elektronik Para Hizmetleri A.S)
Aydınevler Mah. Ismet Inonu Cad. Turkcell Blok No:36
34854 Maltepe/ Istanbul, Türkiye

Huseyin Oktay Altun
Boğaziçi University Institute for Data Science and Artificial Intelligence, South Campus, Bebek, 34342
Beşiktaş/Istanbul, Türkiye

Emre Fisne
Boğaziçi University Institute for Data Science and Artificial Intelligence South Campus, Bebek, 34342 Beşiktaş/Istanbul, Türkiye

To cite this article:

Bozkurt, A., Ekici, F., Yetiskul, H., Altun, H. O., & Fisne, E. (2025). From data to emotion: A multimodal approach to real-time emotion-aware marketing. *The Eurasia Proceedings of Science, Technology, Engineering and Mathematics (EPSTEM), 38*, 538–550.