A Hybrid Approach to Real-Time Recommendations using Heterogeneous Data Processing and Distributed Predictive Models

Diana – Andreea CĂUNIAC, Simona-Vasilica OPREA
Bucharest University of Economic Studies,
Bucharest, Romania
diana.cauniac@csie.ase.ro, simona.oprea@csie.ase.ro

As e-commerce platforms scale, the gap between what users expect and what static recommendation models can deliver has become hard to ignore. This paper describes the design and implementation of a hybrid recommendation system built on a distributed cloud infrastructure, tested on a dataset of over 3.7 million products, 1.9 million reviews, and 1.1 million user interactions. The system combines collaborative filtering through matrix factorization with sentiment analysis, emotion detection, and topic modeling applied to user reviews, identifying six recurring themes that reflect real purchasing experiences. Geographic proximity and recent behavioral signals are incorporated as contextual features to further refine recommendations. The stack includes Google BigQuery, Vertex AI Feature Store, Pinecone, Apache Kafka, Apache Flink, and BigQuery ML. A feedback loop ties user interactions back to model updates, keeping recommendations relevant as behavior changes. Results suggest that decoupling the processing pipelines reduces latency without sacrificing recommendation quality.

Keywords*: recommendation systems, real-time processing, hybrid architecture, collaborative filtering, heterogeneous data, sentiment analysis, topic modeling, distributed computing.*

1 Introduction

Recommendation systems have become a core part of modern e-commerce platforms, directly influencing conversion rates and user retention [1]. As consumer behavior grows more dynamic, the relevance of a recommendation depends heavily on the system's ability to react quickly to new interactions, rather than relying on scheduled model updates [2], [3].

Traditional batch-based approaches process data at fixed intervals and perform reasonably well in stable environments, but struggle to keep up when user preferences shift mid-session. When a user suddenly explores an unfamiliar product category, the system needs to pick up on that signal immediately, something static models are not built to do.

Moving toward distributed architectures capable of ingesting continuous event streams becomes a natural response to this limitation [3], [4]. The challenge is further complicated by the diversity of available data: behavioral interactions, unstructured textual reviews, geographic signals, session contexts, and device metadata, each requiring a different processing approach [5], [6].

To handle these complexities, we propose a hybrid recommendation system deployed on a scalable cloud infrastructure, designed to process large volumes of heterogeneous data with low latency. The framework separates concerns across three functional layers (online, nearline, and offline) allowing the system to serve requests in real time while continuously updating features and periodically retraining models, without disrupting live service.

Rather than relying solely on purchase histories and clickstream data, the system combines collaborative filtering, semantic analysis, and topic modeling within a unified processing pipeline [7]. The results show that real-time responsiveness does not have to come at the cost of analytical depth or recommendation accuracy [8].

2 Data Modeling and Preprocessing

Data preparation is a critical prerequisite for any functional recommendation system. The dataset combines real sources, Hugging Face, Open Food Facts, and OpenStreetMap, with simulated data, reaching over 3.7 million products, 1.9 million reviews, 1.1 million interactions, and 250,000 user profiles. This combination was designed to ensure sufficient volume and diversity for testing system stability across varied usage scenarios.

The data model is built around two core entities, clients and products, connected through user sessions. Additional layers capture device metadata, geographic coordinates, textual reviews, and recommendation history. Products define the item space from which all recommendations are drawn. Interactions record explicit user actions, while reviews contribute qualitative signal that behavioral data alone does not carry. Sessions and locations introduce temporal context and geographic proximity as ranking features. The entities, their data types, and approximate volumes are summarized in Table 1.

Table 1. Dataset entities and approximate record volumes

Entity	**Data Type**	**Approximate Volume (Records)**	**Content Description**
Products	Structured	3,700,000+	Aggregated product metadata from diverse sources.
Clients	Semi-structured	250,000+	Comprehensive user profiles and demographic attributes.
Interactions	Structured	1,100,000+	Behavioral user events (e.g., clicks, views, add-to-cart actions).
Reviews	Unstructured	1,900,000+	Unstructured textual product reviews and semantic feedback.
Sessions	Semi-structured	500,000+	Browsing sessions enriched with contextual metadata.
Devices	Structured	1,000+	Hardware specifications and client device metadata.
Locations	Geospatial	4,783+	Geographic coordinate data for commercial locations.
Recommendations_log	Structured	1,000,000 (Simulated)	Historical audit trail of delivered recommendations.

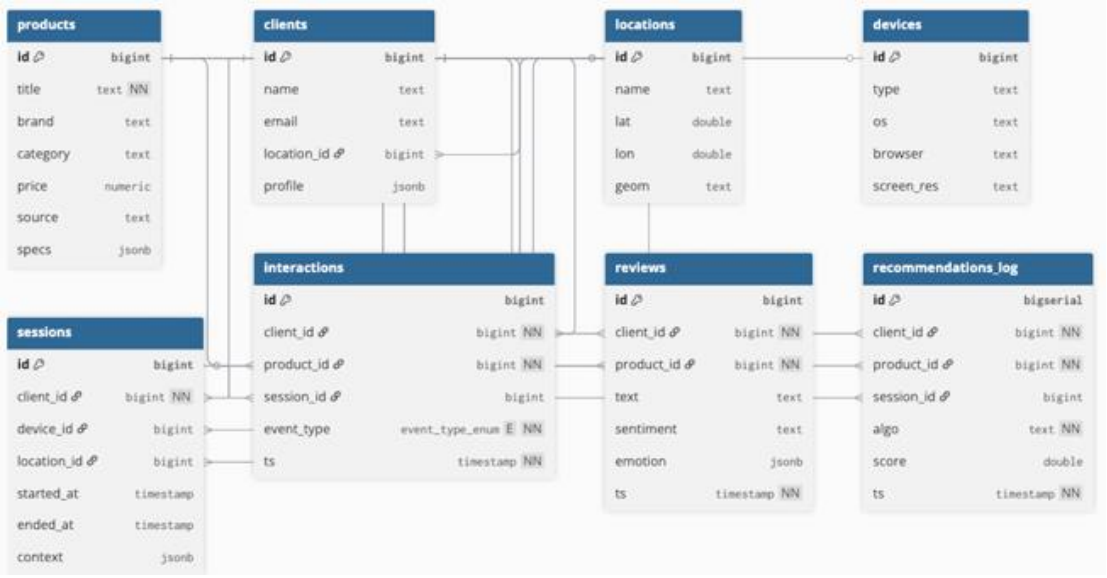

Fig. 1. Database schema and entity relationships

All data flows through Google BigQuery, which handles storage and computation within the same environment, allowing machine learning models to be called directly through SQL without moving data between systems. Data is organized into three zones: raw ingestion, curated transformations, and a semantic layer built from views used in both analysis and model training. The relational structure connecting these entities is illustrated in Figure 1.

Preprocessing is applied separately to each entity. In the products table, records missing both name and category are excluded, as these fields are required for semantic placement within the recommendation space. Missing prices are estimated using the category median: $pc = \text{median}(pi \mid i \in c)$, values deviating significantly from category norms are identified via z-scores and either adjusted or removed. Interactions falling outside their corresponding session window are discarded [9]. Incomplete client profiles are reconstructed from distributions observed among users with similar attributes. Numerical variables are normalized using min-max scaling. Session durations are standardized through z-score scaling, given their skewed distribution.

Textual reviews are cleaned and encoded into numerical representations using NLP models in BigQuery ML. Geographic records are filtered to Romanian territory, and locations in close proximity are merged to prevent duplicate entries from affecting distance calculations.

The preprocessed data is collected into a curated layer, with events ordered chronologically and semantic views built to support consistent feature extraction across all models used in later stages.

3 Sentiment Analysis and Topic Modeling in Product Evaluation

User reviews contain information that behavioral data alone does not capture [10]. A product frequently viewed but consistently criticized carries a different signal than one with similar interaction patterns and positive feedback [11]. The system addresses this through sentiment analysis and topic modeling applied directly to review text [12].

Before analysis, reviews are cleaned to remove special characters, inconsistent casing, and symbols with no semantic value. Sentiment is quantified using TextBlob, which assigns each review a polarity score: $\text{Polarity} = (\Sigma\ w\_positive - \Sigma\ w\_negative) / \text{Total words}$ [13].

The score ranges from −1 to 1, where negative values indicate dissatisfaction and positive values indicate approval. The computed sentiment score is distinct from

the explicit star rating (review_score, on a 1–5 scale), the two are correlated but measure different aspects of user feedback. Table 2 includes a case where a 4-star rating accompanied a description of a damaged product, illustrating this gap directly.

Table 2. Examples of Semantic Evaluation and Sentiment Scores for Reviews from the Dataset

review_id	fragment review_text	review_score	s_i^sent (interpretation)
RV00315788	"Very disappointed, I do not recommend."	1.0	strongly negative
RV00140820	"Poor packaging, low quality."	1.0	negative
RV00371553	"The product arrived damaged."	4.0	moderately negative
AMZ00132163	"Excellent product, I am very satisfied."	5.0	strongly positive
AMZ00721273	"Very satisfied with the product, I recommend it."	5.0	positive

A polarity score captures overall sentiment direction, but reviews frequently reference specific emotional states, disappointment, satisfaction, frustration, that a single value does not distinguish [10]. The emotion field in the dataset records this additional dimension, and Table 3 illustrates cases where similar polarity scores correspond to different emotional categories.

Table 3. Examples of Emotions Identified in Reviews from the Dataset

review_id	fragment review_text	emotion
RV00315788	"Very disappointed, I do not recommend."	disappointment
RV00140820	"Poor packaging, low quality."	neutral
RV00371553	"The product arrived damaged."	disappointment
AMZ00132163	"Excellent product, I am very satisfied."	satisfaction
AMZ00721273	"Very satisfied with the product, I recommend it."	satisfaction

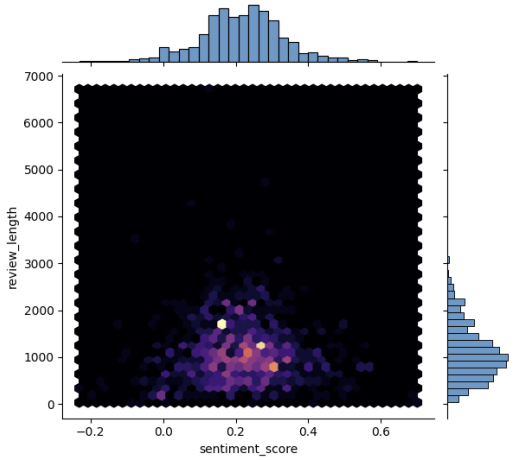

Fig. 2. Correlation between sentiment polarity and review length

The sentiment distribution across the dataset was analyzed through a hexbin plot, shown in Figure 2. Scores concentrate between 0.1 and 0.3, consistent with the tendency observed on e-commerce platforms for users to leave feedback more frequently when satisfied. Review length falls predominantly between a few hundred and 1,500 characters, sufficient for NLP models to

extract reliable semantic signal.

Product-level scores are computed by averaging across all associated reviews: $S(p) = (1 / N_p) \cdot \Sigma s_i$, where N_p is the number of reviews for product p. Arithmetic averaging reduces the influence of isolated extreme ratings on the aggregate score. The resulting values are subsequently combined with behavioral signals in the recommendation ranking stage.

4 Topic Modeling from User Reviews

Sentiment scores capture opinion polarity, but users frequently reference specific product aspects, quality, price, delivery, or functionality, within the same review. Topic modeling extracts these recurring themes and structures them into categories that the sentiment analysis alone cannot distinguish.

Each review is projected into a lower-dimensional latent space where each dimension corresponds to a recurring theme. The projection produces a probability distribution over topics, allowing a single review to be associated with multiple themes at varying intensities. At the product level, topic distributions are averaged across all associated reviews, producing a vector that describes the thematic structure of feedback for each product.

Six topics were identified across the review corpus: quality, delivery, pricing, packaging, usability, and customer support. Quality-related content carries the highest weight at 0.20, with the remaining topics spread more evenly across the corpus. The full distribution is summarized in Table 4.

Table 4. Identified Topics and Average Distribution

Topic	**Representative Terms**	**Average Distribution**
Topic 1	product, quality, good, excellent, recommend	0.20
Topic 2	delivery, shipping, arrival, delayed, damaged	0.13
Topic 3	price, value, worth, cheap, expensive	0.16
Topic 4	packaging, box, damaged, condition, arrival	0.18
Topic 5	usage, easy, works, functionality, performance	0.17
Topic 6	support, service, return, refund, customer	0.16

Figure 3 shows the relative weights across all six topics.

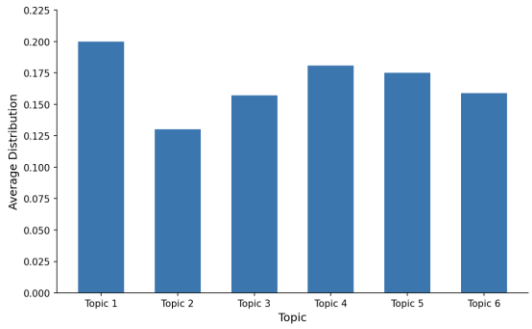

Fig. 3. Topic distribution across user reviews

t-SNE was applied to project the topic distributions into two dimensions, shown in Figure 4. Reviews covering similar themes form distinct clusters, and those near cluster boundaries tend to reference multiple product aspects within the same text.

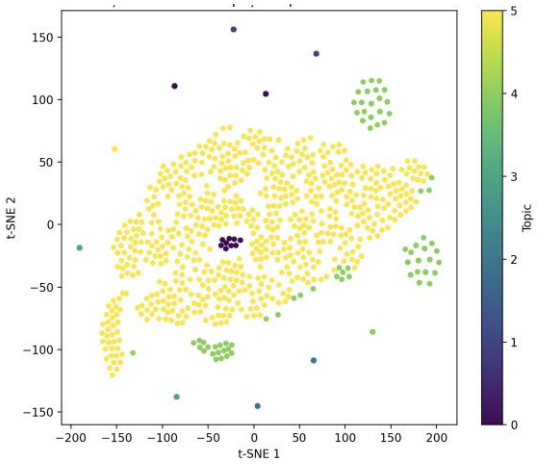

Fig. 4. t-SNE projection of review topic distributions

5 Hybrid Recommendation Pipeline

Collaborative filtering works well for users with rich interaction histories, but

breaks down in cold-start scenarios. Semantic analysis helps fill that gap, though it struggles to capture rapid shifts in user behavior. The system described here combines both approaches within a continuous processing pipeline that also incorporates live session context, geographic signals, and a feedback loop that ties user reactions back to model updates [14].

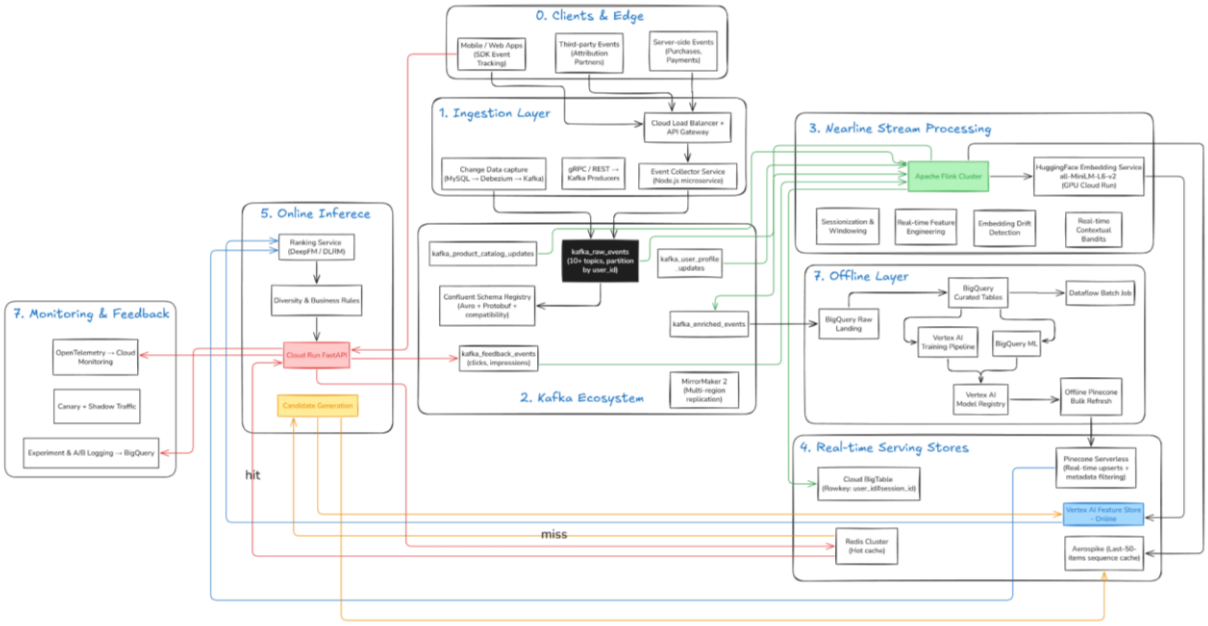

Fig. 5. End-to-end architecture of the hybrid recommendation system

The end-to-end architecture is illustrated in Figure 5, covering all eight stages of the data flow.

Stage 1 — Edge and Client Layer. Web and mobile applications emit behavioral signals via embedded SDKs. Third-party platforms contribute through integration connectors, while server-side systems append operational data on inventory and transactions. All incoming events pass through a Cloud Load Balancer and API Gateway before entering the pipeline.

Stage 2 — Ingestion. Raw events are filtered, validated, and normalized into a standardized format. Change Data Capture via Debezium handles relational database changes without affecting source systems. REST and gRPC events are processed by Node.js microservices, and all messages are validated against a Schema Registry before moving further.

Stage 3 — Message Broker (Apache Kafka). Kafka organizes streaming events into topics by type: user interactions, product catalog updates, and profile changes. The kafka_enriched_events topic associates each event with product attributes, user profile metadata, and session context. MirrorMaker handles cross-region replication for availability and fault tolerance.

Stage 4 — Nearline Processing. Apache Flink consumes streams from Kafka and computes interaction frequency, session duration, recency scores, and category-level interest shifts, handling deduplication and profile updates in parallel. Outputs are written to Cloud Bigtable for low-latency access, and textual data is routed to HuggingFace models for embedding generation.

Stage 5 — Feature Store and Vector Management. Embeddings are written to Vertex AI Feature Store and Pinecone. The Feature Store maintains the numerical and vector representations used by predictive models, and Pinecone manages semantic indexing and real-time similarity searches.

Stage 6 — Offline Training. Historical data in BigQuery is used to periodically retrain

collaborative filtering, semantic analysis, and reranking models. BigQuery ML runs training directly through SQL, removing the need for separate ML infrastructure. Updated models are published to the Vertex AI Model Registry and deployed without taking the service offline.

Stage 7 — Online Inference. On receiving a recommendation request, the system first queries Redis for cached results. On a cache miss, a FastAPI microservice retrieves the current user profile from the Feature Store and runs a vector similarity search in Pinecone. Candidate products are scored by a reranking model using CTR, interaction frequency, and historical signals, then returned to the client through Cloud Endpoints.

Stage 8 — Feedback Loop. Every post-recommendation action, click, ignore, purchase, or abandonment, is fed back into Kafka. Flink correlates this feedback with the recommendations that triggered it, forwarding aggregated metrics to BigQuery for model evaluation and retraining. Refined models are versioned in Vertex AI and deployed automatically to Cloud Run, the cycle runs without manual steps.

6 Model Adaptability and Dynamic Updating

Recommendations become less relevant the moment user preferences shift and the model stops keeping up. The feedback mechanism described here closes that gap by routing post-recommendation user reactions back into the training pipeline, so behavioral changes are reflected in the models without waiting for a scheduled retraining cycle.

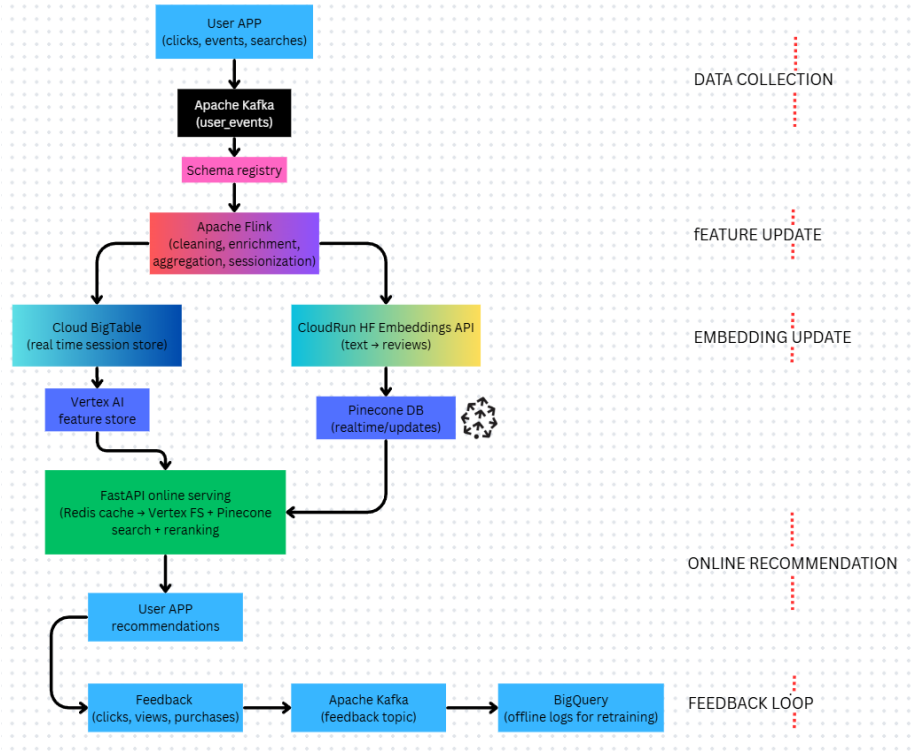

Fig. 6. Real-time feedback loop and model retraining flow

The full flow is illustrated in Figure 6, after a recommendation is displayed, the client application captures the user's response and publishes it to a dedicated Kafka topic. Messages pass through a Schema Registry that validates structure and discards incomplete or malformed payloads before they reach the processing layer.

Apache Flink picks up the validated streams and correlates displayed recommendations with subsequent user

actions, distinguishing what was clicked or purchased from what was ignored. From these associations, three metrics are computed: click-through rate, conversion rate, and an aggregate relevance score.

The aggregated results are written to BigQuery across two tables — feedback_log and recommendations_log — which together maintain a full history of user-system interactions. Model versions are also compared here using live behavioral data rather than held-out test sets.

As feedback accumulates, ranking and predictive models are retrained through BigQuery ML. Updated versions are registered in the Vertex AI Model Registry with versioning metadata and deployed automatically to Cloud Run, with no manual steps required.

Concurrently, user and product feature profiles are updated in Vertex AI Feature Store and Pinecone. If a product category gains traction, the rise in CTR propagates through the nearline pipeline and updates the associated vectors before the next offline retraining cycle, without waiting for a full model rebuild.

7 Conclusions

The results confirm the initial premise: a recommendation system can scale across heterogeneous data without sacrificing response speed for analytical depth.

Collaborative filtering covers users with rich interaction histories, while semantic signals and topic distributions help where behavioral data is sparse. Geographic proximity and session context fill in what clickstream data misses. None of these components works well in isolation, the value comes from running them together within a continuous pipeline.

Working with over 3.7 million products, 1.9 million reviews, and 1.1 million interactions showed that decoupling the processing layers has a direct impact on both latency and model freshness. The nearline layer keeps user profiles current between full retraining cycles, while the offline component handles periodic rebuilds on complete historical data.

The feedback loop tied everything together. CTR and conversion signals propagate through Kafka and Flink before the next scheduled training run, so the models stay aligned with recent behavior without waiting for a full retraining cycle.

Some questions remain worth pursuing. Adapting the spatial ranking signals for cross-border e-commerce scenarios would be a logical next step beyond the Romanian location data used here. Users with no interaction history still represent a harder cold-start case than semantic signals alone can fully address. Over longer sessions, more explicit diversity constraints in the reranking models could also help reduce filter bubble effects.

8 Acknowledgment

This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI - UEFISCDI, project number COFUND-CETP-SMART-LEM-1, within PNCDI IV.

References

[1] M. Alam, “Hybrid Recommender Systems: A Systematic Review,” Information Processing & Management, 2024.

[2] Y. Deldjoo, S. Saghafian and A. Bellogín, “Recommender Systems Leveraging Contextual Information: A Survey,” Information Fusion, pp. 101-120, 2024.

[3] L. Chen and Y. Zhang, “Explainable Recommender Systems: A Survey and New Perspectives,” Information Processing & Management, pp. 1-20, 2024.

[4] S. Zhang, “Deep Learning Based Recommender Systems: A Survey and New Perspectives,” ACM Computing Surveys, 2022.

[5] V. Kumar and S. Patel, “Context-Aware

Recommender Systems Using Big Data Analytics," Future Generation Computer Systems, pp. 88-92, 2023.
[6] J. Li and W. Wang, "Real-time Recommendation Systems: Architectures and Challenges," IEEE Access, pp. 678-692, 2024.
[7] P. Carbone, "Flink: Stateful Stream Processing," IEEE, pp. 5-12, 2023.
[8] J. Kreps, "Apache Kafka and Stream Processing Architectures," LinkedIn Engineering, pp. 10-12, 2022.
[9] N. Marz and J. Warren, "Lambda and Kappa Architectures Revisited," IEEE Software, pp. 30-40, 2023.
[10] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis," Foundations and Trends in Information Retrieval, pp. 1-36, 2008.
[11] D. Tang, Q. Qin and T. Liu, "Document Modeling with Gated Recurrent Neural Network for Sentiment Classification," Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1422-1432, 2015.
[12] Y. Zhang and A. Zhang, "Joint Representation Learning for Top-N Recommendation with Review Text," Proceedings of the 13th ACM International Conference on Web Search and Data Mining, pp. 345-353, 2020.
[13] J. McAuley and J. Leskovec, "Using Reviews for Recommendation Systems," RecSys Proceedings, pp. 150-155, 2022.
[14] A. Group, "Alibaba Real-Time Recommendation System Architecture," Alibaba Cloud Whitepaper, pp. 11-19, 2024.

Diana-Andreea CĂUNIAC earned her bachelor's degree in Economic Informatics in 2020 and her master's degree in Databases – Support for Business in 2022. She is currently pursuing a Ph.D., focusing on big data in real time. Professionally, she worked in the field of databases and currently works as a Fullstack Developer. Her research interests include big data, database systems, software development, artificial intelligence, and modern web technologies.

Simona-Vasilica OPREA received the MSc degree through the Infrastructure Management Program from Yokohama National University, Japan, in 2007, the first Ph.D. degree in Power System Engineering from the Bucharest Polytechnic University in 2009, and the second Ph.D. degree in Economic Informatics from the Bucharest University of Economic Studies in 2017. She is currently an Associate Professor with the Faculty of Cybernetics, Statistics, and Economic Informatics with the Bucharest Academy of Economic Studies, involved in several research projects.