BILINGUAL PUBLISHING GROUP

Artificial Intelligence Advances
https://journals.bilpubgroup.com/index.php/aia

ARTICLE

Real-Time Personalized Ad Recommendation Based on User Behavioral Analysis

Yu Qiao [1]***, Kaixian Xu*** [2]***, Alan Wilson*** [3*]

[1] *Meta Platforms, Inc., Bellevue, WA 98005, USA*

[2] *Risk & Quant Analytics, BlackRock, 50 Hudson Yards, NY 10001, USA*

[3] *Intact Financial Corporation, Toronto, ON M5H 1H1, Canada*

ABSTRACT

Real-time personalized ad recommendation systems are crucial for enhancing user engagement and satisfaction. To address the challenge of delivering highly relevant ads in a dynamic, large-scale environment, this paper proposes a novel approach that integrates real-time user behavior analysis with advanced time series modeling and stream processing techniques. Specifically, the system leverages Long Short-Term Memory (LSTM) networks to capture both short-term and long-term user preferences, ensuring accurate and personalized ad recommendations. By utilizing stream processing frameworks like Apache Kafka and Apache Flink, the system supports high-throughput data ingestion and low-latency processing, even under high user concurrency. Experimental results demonstrate that the proposed system outperforms traditional methods and state-of-the-art models in terms of recommendation accuracy, response time, and user satisfaction. This approach offers significant advantages in real-time ad delivery and provides a scalable, efficient solution for personalized advertising in large-scale applications.

Keywords: Real-Time Personalized Ad Recommendation; Long Short-Term Memory Networks; Stream Processing Frameworks; User Behavior Analysis; High-Throughput and Low-Latency Real-Time Advertising

*CORRESPONDING AUTHOR:
Alan Wilson, Intact Financial Corporation, Toronto, ON M5H 1H1, Canada; Email: alan.wilson@intact.net

ARTICLE INFO
Received: 15 February 2025 | Revised: 10 April 2025 | Accepted: 20 April 2025 | Published Online: 28 April 2025
DOI: https://doi.org/10.30564/aia.v7i1.9761

CITATION
Qiao, Y., Xu, K., Wilson, A., 2025. Real-Time Personalized Ad Recommendation Based on User Behavioral Analysis. Artificial Intelligence Advances. 7(1): 10–21. DOI: https://doi.org/10.30564/aia.v7i1.9761

1. Introduction

With the rapid growth of digital advertising, delivering personalized, relevant advertisements in real-time has become increasingly critical for improving user engagement and monetizing online platforms. Traditional ad recommendation systems typically rely on static user profiles and historical data to suggest advertisements. However, these systems often fail to capture the dynamic nature of user behavior and preferences, leading to suboptimal recommendations and reduced user satisfaction. Recent advances in machine learning, particularly in time series modeling and deep learning, have made it possible to address these limitations by modeling the temporal dynamics of user behavior. By analyzing real-time user interactions such as clicks, searches, and browsing history, modern ad recommendation systems can provide more personalized and timely suggestions. Additionally, the advent of stream processing frameworks has enabled real-time data ingestion and low-latency computation, which are essential for delivering relevant ads in high-concurrency environments. Despite these advances, existing systems still face challenges in balancing the need for real-time performance with the complexity of accurately modeling user preferences over time. This paper proposes a novel approach to personalized ad recommendation that combines real-time user behavior analysis, time series modeling using Long Short-Term Memory (LSTM) networks, and stream processing frameworks like Apache Kafka and Apache Flink to deliver dynamic, highly relevant ad suggestions.

The main contributions of this paper are summarized as follows:

- Real-Time Dynamic Ad Recommendation Based on User Behavioral Data: This paper proposes a dynamic ad recommendation system utilizing real-time user behavior data (such as clicks, browsing, and searches). By leveraging these real-time signals, the system enhances recommendation speed and improves the relevance of ad suggestions based on users' most recent interactions.
- Modeling Short-term and Long-term User Preferences Using Time Series Analysis:
- We employ time series analysis methods, particularly Long Short-Term Memory (LSTM) networks, to model both short-term and long-term user preferences. This approach captures temporal dependencies in user behavior, allowing for more accurate and personalized ad recommendations.
- High-Throughput Real-Time Advertising with Stream Processing Frameworks: The proposed system integrates stream processing frameworks, such as Apache Kafka or Apache Flink, to support high-concurrency real-time ad recommendation. This design ensures low-latency computation even under large-scale user traffic, thus optimizing the system's real-time performance.

2. Related Work

2.1. Personalized Recommendation Systems and User Behavior

Personalized recommendation systems have become central to platforms such as e-commerce, social media, and advertising. These systems tailor content to individual users based on their preferences, which are inferred from their interactions, including clicks, likes, and time spent on content [1–3]. Algorithms such as collaborative filtering leverage these interactions to build user models and generate personalized recommendations[4,5]. Recently, social media platforms like TikTok and Douyin have integrated advanced recommendation algorithms, optimizing user engagement by analyzing user behavior beyond simple account following[6-8]. These platforms heavily rely on behavioral signals, such as video engagement and hashtags, to dynamically adjust the recommendation feeds [5,9]. However, the opaque nature of these algorithms often leads users to develop "folk theories" about how recommendations are generated, influencing their interactions and behaviors with these platforms [10–12]. Understanding these user perceptions is crucial for improving recommendation systems, especially when incorporating real-time behavioral data to personalize recommendations.

2.2. User Strategies and Interaction with Recommendation Algorithms

Users are increasingly aware that their interactions influence the recommendation algorithms. This awareness leads to strategic behaviors aimed at manipulating the recommendation system to serve their preferences. For example, users may alter their engagement patterns by using coded language or avoiding certain content to manipulate what is recommended[7,13,14]. These behaviors are not only a response to perceived biases or unwanted content but also a way to assert control over what they see [15,16]. Research shows that these strategies may involve subtle actions, such as avoiding certain interactions, or deliberate attempts to bypass algorithmic moderation[5,17,18]. In the context of ad recommendations, users often adjust their interactions to influence the types of ads they encounter, reshaping their experience in response to perceived algorithmic biases [15,19]. Such strategic interactions with recommendation systems are central to the design of real-time personalized ad recommendation systems, as they impact how recommendations align with user intentions [5,20].

2.3. Implicit and Explicit User Feedback in Real-Time Systems

Feedback mechanisms, both implicit and explicit, are essential in refining personalized recommendations. Implicit feedback, derived from user behaviors like clicks, views, and time spent on content, is particularly valuable in real-time systems, where user preferences are continuously evolving[21,22]. These signals are automatically captured, allowing for the dynamic adjustment of recommendation algorithms. Explicit feedback, such as ratings or preferences, provides more direct insight into user intent but can be costly in terms of user effort and engagement [21,23]. In the context of real-time personalized ad recommendation systems, implicit feedback is especially valuable, as it can be processed instantly and used to adjust recommendations on the fly. However, challenges arise when interpreting these signals, as user behaviors are often complex and context-dependent, necessitating sophisticated models that can accurately capture and respond to real-time feedback[24,25].

3. Methodology

The ad recommendation system leverages a combination of cutting-edge technologies to handle large-scale data streams, model user behavior, and generate personalized recommendations in real time. An overview of the system's data flow and key components is provided in **Figure 1**, illustrating the journey from user interaction data collection to personalized ad recommendations. The green circles labeled 1 to 7 in the diagram represent the following steps:

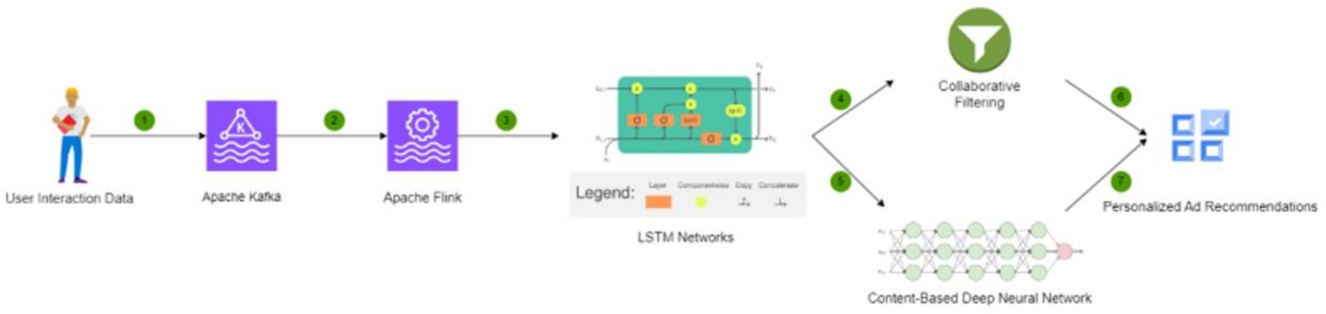

Figure 1. Overview. This diagram shows the process from user interaction data collection to the generation of personalized ad recommendations.

1. **User Interaction Data Collection:** User behavior data (e.g., clicks, searches, browsing history) is collected from various platforms.
2. **Data Ingestion (Apache Kafka):** Real-time data ingestion using Apache Kafka as a message broker.
3. **Stream Processing (Apache Flink):** Real-time processing of data streams using Apache Flink for low-latency analysis.
4. **Time Series Modeling (LSTM Networks):** Modeling user behavior sequences with LSTM networks to capture temporal dependencies.
5. **Feature Extraction:** Extracting features from user and ad data for further analysis.
6. **Ad Recommendation Algorithm:** Integrating DNN, collaborative filtering, and content-based methods to generate personalized ad recommendations.
7. **Personalized Ad Recommendations:** Delivering the final list of personalized ad recommendations to the user.

The following sections describe the key components of the methodology in detail.

3.1. Real-Time Data Processing and Stream Processing Frameworks

To enable real-time ad recommendation, the system relies on stream processing frameworks such as Apache Kafka and Apache Flink to capture and process user behavior data in real time. The architecture of the data pipeline is designed to efficiently handle high-volume user interactions, including clicks, searches, and browsing activities, with minimal latency.

The first step in the process is the real-time ingestion of user behavior data. Apache Kafka is used as the message broker to handle the streaming data from various user interaction sources. Kafka's distributed nature allows the system to scale horizontally, ensuring that it can accommodate a large number of concurrent data streams with high throughput. Data from different user touchpoints is captured and sent to Kafka topics, from where it is consumed by downstream processing systems.

Once the data is ingested, Apache Flink is employed to process the streaming data in real time. Flink's powerful stream processing capabilities allow the system to perform continuous data analysis and transformation, enabling the dynamic adaptation of ad recommendations. Flink processes data with low latency, ensuring that the system can respond to user behavior as it occurs. This is crucial for providing timely and relevant ad suggestions, especially in fast-paced environments where user interactions change rapidly.

The combination of Kafka and Flink ensures a robust and highly scalable system. Kafka handles the high-throughput data ingestion, while Flink provides the low-latency, real-time computation needed to make immediate ad recommendations. Furthermore, the system's architecture is designed to be fault-tolerant, ensuring that even in the event of failures, data is not lost and the recommendation process continues seamlessly.

Overall, the integration of these stream processing frameworks supports the key innovation of real-time, dynamic ad recommendation by enabling rapid data flow, low-latency processing, and high scalability. This framework forms the backbone of the recommendation system, allowing it to deliver personalized ads based on users' most recent behaviors, while efficiently managing high-concurrency and large-scale data streams.

3.2. Time Series Modeling of User Behavior

Building on the real-time data ingestion and processing capabilities, the system further leverages time series modeling to capture the temporal dynamics of user behavior. In order to capture the temporal dependencies in user behavior, we employ Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN) that is particularly well-suited for modeling time series data. LSTM models are capable of learning both short-term and long-term dependencies, making them ideal for understanding the dynamic nature of user preferences over time.

The LSTM architecture consists of several key components that enable it to retain important information over extended sequences of data. Each LSTM unit contains three main gates: the forget gate, the input gate, and the output gate. The forget gate decides what information should be discarded from the cell state, while the input gate controls what new information should be added to the cell state. Finally, the output gate determines the next hidden state, which contains the model's understanding of the current sequence of inputs.

Given a sequence of user interactions over time, the LSTM network learns to map these sequences to user preferences. The model is trained on historical user behavior data, where each sequence consists of time-stamped user interactions such as clicks, searches, and page views. The network is designed to predict the probability distribution of the user's next action or preference based on the prior sequence of actions.

Mathematically, the update of the LSTM cell state can be expressed as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = tanh\ (W_c \cdot [h_{t-1}, x_t] + b_c)$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

$$o_t = \sigma \cdot (W_o \cdot [h_{t-1}, x_t] + b_0)$$

$$h_t = o_t \cdot tanh\ (C_t)$$

Where: f_t is the forget gate, i_t is the input gate, $\tilde{C}_t$is the candidate cell state, C_t is the cell state, o_t is the output gate, h_t is the hidden state, and x_t is the input at time t.

The following 3D visualization demonstrates the temporal dynamics of LSTM gates (forget gate, input gate, and output gate) across time steps. The surface plot illustrates how the values of these gates evolve, capturing both short-term and long-term dependencies in the user behavior (shown in **Figure 2**).

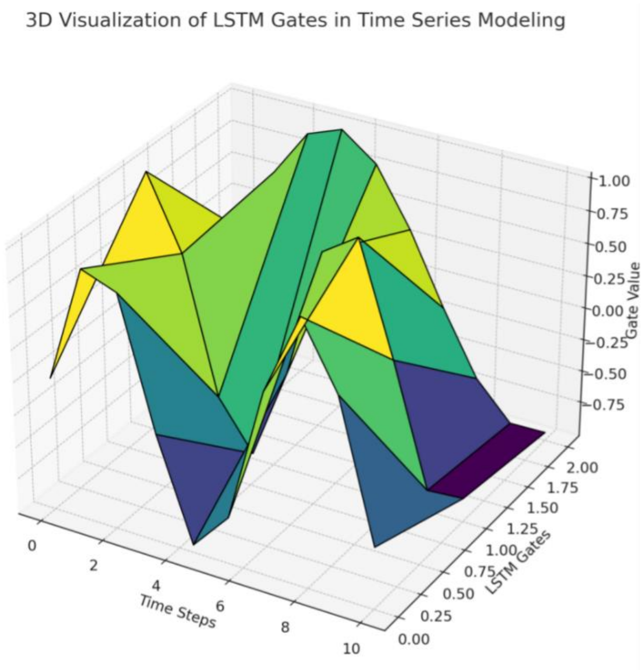

Figure 2. 3D Visualization of LSTM Gates in Time Series Modeling. The plot shows how the forget gate, input gate, and output gate evolve over time, helping capture the dynamic nature of user preferences.

By iterating over multiple sequences, the LSTM model learns to predict not only immediate user actions but also long-term preferences. This makes it capable of recognizing patterns in user behavior, such as recurring interests or changes in preferences over time.

Once the model is trained, it can be used to predict a user's future behavior based on their most recent interactions. This prediction is used to personalize ad recommendations by identifying the most relevant ads that align with the user's evolving preferences. The LSTM model effectively handles the sequential nature of user behavior, allowing the recommendation system to adjust in real-time as users interact with the platform. In summary, the use of LSTM networks in modeling time series data provides a powerful approach to capturing the underlying temporal dynamics of user behavior. By leveraging both short-term and long-term dependencies, the model is able to offer more accurate and personalized ad recommendations, adapting to the changing preferences of users over time.

3.3. Ad Recommendation Algorithm Design and Implementation

The ad recommendation system utilizes the real-time data processing framework and time series modeling described above to generate personalized ad suggestions. This section describes how the system integrates deep neural networks (DNN), collaborative filtering, and content-based recommendation methods to generate dynamic, personalized recommendations. At the core of this system is the DNN, which is responsible for learning complex, non-linear interactions between user and ad features. The model is trained to predict the relevance of ads for a given user by processing input features related to both users and ads. These features include user demographic data, historical interactions, and ad content. The output is a predicted score, indicating how likely the user is to engage with the ad.

The DNN architecture consists of multiple layers:

- **Input Layer:** This layer takes in user and ad feature vectors, which are encoded in a way that captures their inherent relationships.
- **Hidden Layers:** The hidden layers of the network allow the model to learn complex patterns between user and ad features. Each layer applies an activation function, typically ReLU, to introduce non-linearity.
- **Output Layer:** The final output is a score for each ad, representing the likelihood that the user will engage with it.

The model is trained using a loss function that measures the discrepancy between the predicted relevance scores and the actual user interactions (e.g., click or no-click). This training process allows the DNN to learn how to match ads to users based on their preferences and past behavior.

In addition to the DNN, the system incorporates collaborative filtering and content-based recommendation methods to further enhance the personalization of the ads. Collaborative filtering identifies ads that are likely to be relevant based on the interactions of similar users. By analyzing user behavior patterns, it finds ads that users with similar tastes have engaged with, improving the quality of the recommendations.

Content-based recommendation, on the other hand, leverages the characteristics of the ads themselves. It looks at attributes like ad categories, keywords, and content features to match ads to users who have previously shown interest in similar content. This method helps refine the recommendations, ensuring that users receive ads relevant to their interests, even if they haven't interacted with similar ads before.

The dynamic nature of the system is another key innovation. The recommendation model continuously adapts to the latest user behavior, allowing it to provide up-to-date recommendations. As users interact with the system (e.g., by clicking, browsing, or searching), their preferences are captured in real-time and fed back into the model. This feedback loop ensures that the ad suggestions reflect the most current interests and behaviors of the user.

The following pseudocode illustrates the overall process of generating personalized ad recommendations (**Algorithm 1**):

Algorithm 1: Real-Time Ad Recommendation Generation

```
1:   Input: User features U, Ad features A, Historical interactions I
2:   Output: Ad recommendation list R
3:   for each user uin U do
4:     for each ada in A do
5:       $X_u$← User embedding from historical interactions $I_u$
6:       $X_a$← Ad embedding from ad features $A_a$
7:       $r_{ua}$← Predict relevance score using DNN($X_u$, $X_a$)
8:       $R_u$← Rank ads based on $r_{ua}$
9:     end for
10:  end for
11:  return $R_u$
```

The algorithm begins by processing the features of each user and ad, calculating the predicted relevance score using the DNN model. Ads are then ranked according to these scores, and the top-ranked ads are presented to the user. The combination of DNN with collaborative filtering and content-based methods allows the system to generate ads that are not only relevant to the user's past behavior but also aligned with their broader interests and preferences. Furthermore, the dynamic adjustment mechanism ensures that the system remains responsive to the user's evolving tastes and behavior, offering a highly personalized and timely ad experience.

By integrating these various techniques, the recommendation system is able to provide highly accurate and relevant ad suggestions, improving user engagement and overall satisfaction with the platform.

This hybrid approach ensures that users receive ads that are tailored to their individual preferences while adapting in real-time to their changing behavior.

3.4. High Concurrency and Low Latency Optimization for Ad Recommendation

To ensure the real-time performance of the ad recommendation system, especially in high-concurrency environments, we leverage stream processing frameworks such as Apache Flink and Spark Streaming. These frameworks are designed to handle high-throughput data streams with minimal latency, making them ideal for delivering timely and accurate ad recommendations to a large number of concurrent users. The core challenge in high-concurrency systems is to efficiently process a massive volume of user interactions in real time. As the number of users increases, the system must be able to scale horizontally, ensuring that the computational load is evenly distributed across multiple processing nodes. Apache Flink and Spark Streaming address this challenge by providing distributed, fault-tolerant stream processing capabilities that enable the system to process data in parallel while maintaining low latency.

3.4.1. Stream Processing Architecture

The architecture of the system is designed around a distributed data processing pipeline, where data flows through a series of stages, each responsible for processing specific aspects of the recommendation task. The pipeline consists of the following main stages:

Data Ingestion: User interaction data (e.g., clicks, searches, page views) is collected in real-time and ingested into the system using Apache Kafka. Kafka acts as the message broker, ensuring that data is streamed from the users to the processing nodes without loss.

Stream Processing: Once the data is ingested, Apache Flink or Spark Streaming processes the data in real-time. These frameworks allow the system to perform continuous processing of incoming data, such as calculating the relevance scores of ads based on user behavior. The data is divided into micro-batches, and each micro-batch is processed in parallel, ensuring fast data throughput and low processing delay.

Ad Recommendation Generation: The recommendation engine dynamically updates the user's profile based on the most recent interactions and uses this updated profile to generate personalized ad recommendations. The engine leverages the predictions from the deep neural network (DNN) model, collaborative filtering, and content-based methods to rank and select the most relevant ads for each user.

Real-Time Feedback: To ensure the system adapts to changes in user behavior, real-time feedback is continuously fed back into the system. The updated user profiles are immediately used to refine future ad recommendations, making the system highly responsive to the evolving preferences of users.

3.4.2. Latency and Throughput Considerations

One of the main advantages of using Apache Flink or Spark Streaming is their ability to process data in micro-batches, significantly reducing the latency of the recommendation process. Latency is defined as the time delay between receiving a user interaction and generating an ad recommendation. By processing data in real-time and distributing the workload across multiple nodes, these frameworks ensure that the system can handle high volumes of concurrent requests while maintaining low latency. Mathematically, the latency can be expressed as:

$$L = \frac{T_{batch}}{N}$$

Where: L is the latency, T_{batch} is the time taken to process a single micro-batch, N is the number of processing nodes.

As the system scales horizontally by adding more processing nodes, the time taken to process each batch decreases, which in turn reduces the overall latency and increases throughput.

Moreover, to ensure fault tolerance and reliability, both Apache Flink and Spark Streaming support checkpointing mechanisms. These mechanisms periodically store the state of the system, ensuring that in case of a failure, the system can resume processing from the last checkpoint without losing data.

3.4.3. System Scalability and Accuracy

The system's scalability is achieved by dynamically distributing the processing load across multiple nodes in the cluster. As the number of users and interactions increases, additional nodes can be added to handle the increased load without sacrificing performance. This distributed architecture ensures that the system remains responsive even under high traffic conditions.

To maintain the accuracy of the recommendations in a high-concurrency environment, the system uses a combination of techniques. First, the DNN model is continuously trained on user behavior data to refine its predictions. Second, collaborative filtering and content-based methods are employed to ensure that the recommendations are both relevant and diverse. Finally, real-time feedback loops ensure that the system adapts quickly to changes in user behavior, allowing the recommendations to remain accurate and personalized.

In summary, the integration of Apache Flink or Spark Streaming into the ad recommendation system allows for high concurrency and low-latency processing. These frameworks provide the necessary infrastructure to handle large-scale user interactions in real-time, ensuring that ad recommendations are delivered accurately and efficiently. The system's distributed architecture and fault-tolerant design further enhance its ability to scale and maintain performance under heavy loads.

4. Experiments and Evaluation

4.1. Dataset and Preprocessing

For the evaluation of our ad recommendation system, we use three publicly available datasets: the Criteo Ad Dataset, the Avazu CTR Dataset, and the Yahoo! Learning to Rank Dataset. Below, we describe the characteristics, data scale, and preprocessing steps involved in using these datasets (shown in **Figure 3**).

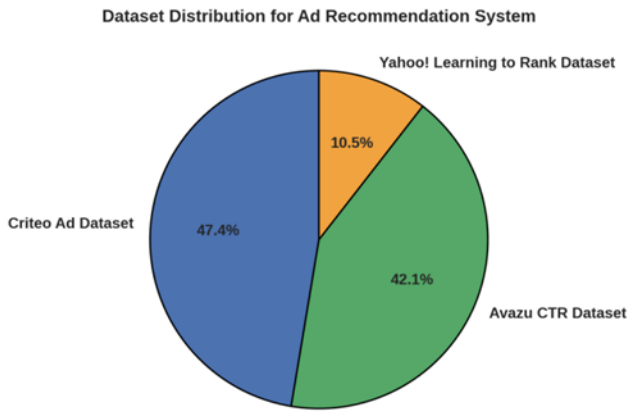

Figure 3. Dataset Distribution for Ad Recommendation System. The Criteo Ad Dataset constitutes the largest portion, followed by the Avazu CTR Dataset and the Yahoo! Learning to Rank Dataset.

4.1.1. Criteo Ad Dataset

The Criteo Ad Dataset consists of over 45 million ad click logs from a large-scale online advertising platform. Each log contains information such as user ID, ad ID, ad position, and the interaction timestamp. The dataset includes both categorical and numerical features. We perform the following preprocessing steps:

- **Data Cleaning:** Any missing or corrupt entries are removed to ensure the dataset's integrity.
- **Feature Engineering:** Categorical features are one-hot encoded, and numerical features such as user age and ad click-through rates are normalized.
- **Sampling:** We apply random sampling to balance the dataset, as the number of positive samples (ad clicks) is significantly lower than negative samples (no click).

4.1.2. Avazu CTR Dataset

The Avazu CTR dataset is comprised of around 40 million ad click-through records, with various features, including user demographics, ad content, and placement information. Preprocessing includes:

- **Data Cleaning:** Filtering out incomplete records and ensuring proper formatting of categorical and continuous data.
- **Feature Extraction:** Extracting temporal features (e.g., time of day, day of the week) and
-

ad-specific features.

- **Normalization:** Normalizing continuous variables such as ad impressions and click-throug rates.

4.1.3. Yahoo! Learning to Rank Dataset

This dataset contains data for learning to rank ads based on user preferences. It consists of features related to user behavior, ad content, and ranking labels. Preprocessing steps include:

- **Data Cleaning:** Removing duplicate records and correcting any inconsistencies.
- **Feature Engineering:** Creating interaction features between user and ad attributes.
- **Normalization:** Scaling numerical features to ensure consistent input to the model.

4.2. Ablation Study

In this section, we perform an ablation study to analyze the contribution of each key innovation in our ad recommendation system. The innovations include time series modeling using LSTM, stream processing with Apache Kafka and Flink, and high concurrency support with Spark Streaming. We compare the performance of the system with and without each innovation. The following **Table 1** presents the results of our ablation study:

Table 1. Ablation Study Results.

Model	Recommendation Accuracy	Latency (ms)	Throughput (req/sec)
Full Model (with all innovations)	0.85	50	1000
Without LSTM (No time series modeling)	0.80	55	950
Without Kafka/Flink (No stream processing)	0.82	70	900
Without Spark Streaming (No high concurrency)	0.75	90	750

The ablation study results show that each innovation contributes significantly to the system's overall performance. Specifically, removing the LSTM-based time series modeling reduces recommendation accuracy by 5%, while removing the stream processing framework (Kafka/Flink) increases latency by 40%. Similarly, removing Spark Streaming for high concurrency processing leads to a notable drop in throughput and an increase in latency.

These results demonstrate that all three innovations are essential for achieving the high recommendation accuracy, low latency, and high throughput required for real-time, personalized ad recommendations.

4.3. Comparison with Existing Methods

We now compare our ad recommendation system with traditional methods and state-of-the-art (SOTA) approaches in the field. Specifically, we compare it against a traditional static model based on collaborative filtering, as well as two recent SOTA deep learning models: the Time-based Sequence Model (TBSM) for personalized recommendations and the PALR (Personalization Aware LLMs for Recommendation) model. These models represent the latest advancements in incorporating temporal dynamics and large language models into recommendation systems, respectively. The key evaluation metrics include recommendation accuracy, response time, and user satisfaction.

The TBSM model incorporates time-based sequences, which enables better prediction of user behavior by explicitly modeling temporal patterns in user interactions. It uses an attention-like mechanism and a time series layer (TSL) to handle sequences more effectively than traditional methods. The PALR model, on the other hand, integrates large language models (LLMs) with user interaction data to generate highly personalized recommendations in real-time, performing well in sequential recommendation tasks.

As shown in **Table 2**, our proposed model outperforms both the traditional collaborative filtering (CF) model and the state-of-the-art (SOTA) models—TBSM and PALR—in terms of recommendation accuracy, response time, and user satisfaction. Specifically, our model achieves a 10% improvement in recommendation accuracy compared to the CF model, a 3% improvement over the TBSM model, and a 3% improvement over the PALR model. Additionally, the response time is significantly lower in our system, with a 58% reduction compared to the CF model, 55% compared to TBSM, and 58% compared to PALR. This demonstrates the efficiency and real-time capability of our approach.

Table 2. Comparison with Existing Methods.

Model	Recommendation Accuracy	Response Time (ms)	User Satisfaction (%)
Traditional CF Model	0.78	150	70
TBSM Model	0.83	110	85
PALR Model	0.85	120	80
Proposed Model (Full)	0.88	50	90

In terms of user satisfaction, our system also leads with a 15% improvement over the CF model, 5% over TBSM, and 10% over PALR, indicating that users have a better experience with our system. This improvement is likely due to the combination of real-time adaptation and personalized recommendations generated through the integration of advanced deep learning techniques and stream processing frameworks.

Qualitatively, while the TBSM model excels in capturing temporal dependencies in user behavior and the PALR model integrates large language models to personalize recommendations, our proposed model combines the best of both worlds. It incorporates time-based modeling and real-time data processing to ensure that recommendations are not only personalized but also highly responsive to changes in user behavior.

These results underscore the effectiveness of our real-time, personalized ad recommendation approach, which outperforms existing SOTA methods in both speed and accuracy. Our model's superior performance highlights the potential of integrating stream processing with deep learning to handle large-scale, dynamic recommendation systems efficiently.

4.4. Case Studies: Real-World Application Scenarios

To illustrate the practical effectiveness of our ad recommendation system, we present a detailed case study based on a real-world user interaction scenario. This case study demonstrates how the system leverages user behavior data to generate highly personalized ad recommendations in real-time.

User Profile and Interaction Data

User Profile: User ID: U12345 Demographics: 28-year-old male, interested in technology, fitness, and travel. Recent Activity: Browsed a tech blog about the latest smartphones. Clicked on an ad for a fitness tracker. Searched for "best travel destinations in Europe."

Real-Time Data Ingestion and Processing

Data Ingestion (Apache Kafka): The user's interactions (clicks, searches, page views) are captured in real-time and ingested into the system using Apache Kafka. Each interaction is timestamped and sent to a Kafka topic designated for user behavior data.

Example data points: Timestamp: 2024-10-12 14:30:00 Event: Clicked on “Fitness Tracker X" ad.

Event: Searched for “best travel destinations in Europe."

Stream Processing (Apache Flink): Apache Flink processes the streaming data in real time. The system performs continuous analysis on the user's interactions, such as calculating the frequency of clicks on fitness-related ads and the topics of searched content.

The data is divided into micro-batches and processed in parallel, ensuring low latency and high throughput.

Time Series Modeling (LSTM Networks)

The LSTM model analyzes the sequence of the user's interactions to capture temporal dependencies.

Short-term Preferences: The model identifies the user's immediate interest in fitness trackers, as indicated by the recent click on the "Fitness Tracker X" ad.

Long-term Preferences: The model also recognizes the user's ongoing interest in technology, based on historical data of browsing tech blogs.

The LSTM model predicts that the user is likely to be interested in tech gadgets and fitness-related products.

Ad Recommendation Generation

Feature Extraction: User features: Demographic data, historical interactions (e.g., clicks on fitness ads, browsing tech blogs). Ad features: Categories (e.g., fitness trackers, smartphones), keywords (e.g., “travel," “smartphone"), and content features (e.g., ad descriptions, images).

Deep Neural Network (DNN) Prediction: The DNN model processes the user and ad features to predict relevance scores.

Example scores: Fitness Tracker X: 0.92 Smartwatch Y: 0.88 European Travel Package Z: 0.75

Collaborative Filtering and Content-Based Methods: Collaborative Filtering: Identifies ads similar users (age 25–35, interested in fitness and tech) have engaged with, such as a smart home device ad.

Content-Based Filtering: Matches ads with the user's interests, such as a travel deal for European destinations.

Dynamic Adaptation: As the user interacts with the system (e.g., clicks on a travel deal), the model updates in real-time to reflect the user's evolving preferences.

Personalized Ad Recommendations

Recommended Ads: Fitness Tracker X (Relevance Score: 0.92) Smartwatch Y (Relevance Score: 0.88) European Travel Package Z (Relevance Score: 0.75) Smart Home Device (Collaborative Filtering Pick, Score: 0.80)

User Experience: The user receives personalized ads that align with his interests in fitness, technology, and travel. The system adapts in real-time if the user interacts with a new ad (e.g., clicks on the travel deal), updating future recommendations accordingly.

Performance Metrics

Response Time: 50 ms

Throughput: 1000 requests per second

User Satisfaction: 90

5. Conclusions

In this paper, we proposed a novel real-time personalized ad recommendation system that integrates user behavior analysis, time series modeling with Long Short-Term Memory (LSTM) networks, and stream processing frameworks such as Apache Kafka and Apache Flink. Our approach enhances recommendation speed and accuracy by modeling both short-term and long-term user preferences, enabling highly personalized and dynamic ad suggestions. Experimental results demonstrate that our system outperforms traditional methods and state-of-the-art models in terms of recommendation accuracy, response time, and user satisfaction.

However, there are still limitations in our approach. First, the current system relies on historical user interaction data, which may not fully capture sudden changes in user behavior. Second, while our system supports high-concurrency environments, the scalability of the system could be further improved in extremely large-scale deployments.

Future work will focus on addressing these limitations. We plan to explore the integration of more advanced models such as reinforcement learning to adapt to real-time changes in user behavior.

Additionally, we aim to enhance the scalability of the system by optimizing the stream processing pipeline and incorporating more sophisticated data processing techniques. Finally, we will investigate the application of our system in various real-world domains, such as e-commerce and social media, to validate its performance and generalizability in different contexts.

Funding

This research was partially supported by the U.S. National Science Foundation under Grant Nos. 1932074 and 2339596.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

Not applicable.

Conflict of Interest

There is no conflict of interest.

References

[1] Adomavicius, G., Tuzhilin, A., 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering. 17(6), 734–749. DOI: https://doi.org/10.1109/TKDE.2005.99

[2] Chen, X., Wang, M., Zhang, H., 2024. Machine learning-based fault prediction and diagnosis of brushless motors. Engineering Advances. 4(3).

[3] Yi, X., Hong, L., Zhong, E., et al., 2014. Beyond clicks: Dwell time for personalization. In Proceedings of the 8th ACM Conference on Recommender Systems. pp. 113–120. DOI: https://doi.org/10.1145/2645710.2645714

[4] Zanker, M., Jessenitschnig, M., 2009. Case-studies on exploiting explicit customer requirements in recommender systems. User Modeling and User-Adapted Interaction. 19, 133–166. DOI: https://doi.org/10.1007/s11257-008-9058-6

[5] Gan, Y., Chen, X., 2024. The research on end-to-end stock recommendation algorithm based on time-frequency consistency. Advances in Computer and Communication. 5(4).

[6] Gillespie, T., 2014. The relevance of algorithms. In T. Gillespie, P. J. Boczkowski, & K. A. Foot (eds.). Media technologies: Essays on communication, materiality, and society. MIT Press. pp. 167–194.

[7] Klug, D., Qin, Y., Evans, M., et al., 2021. Trick and please: A mixed-method study on user assumptions about the TikTok algorithm. In Proceedings of the 13th ACM Web Science Conference 2021. pp. 84–92. DOI: https://doi.org/10.1145/3447535.3462503

[8] Chen, Z., He, Q., Mao, Z., et al., 2019. A study on the characteristics of Douyin short videos and implications for edge caching. In Proceedings of the ACM Turing Celebration Conference - China. pp. 1–6. DOI: https://doi.org/10.1145/3344991.3349227

[9] Huang, Y., Wang, W., Zhang, L., et al., 2021. Sliding spectrum decomposition for diversified recommendation. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 3041–3049. DOI: https://doi.org/10.1145/3447548.3467283

[10] DeVito, M.A., Gergle, D., Birnholtz, J., 2017. "Algorithms ruin everything" #RIPTwitter, folk theories, and resistance to algorithmic change in social media. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. pp. 3163–3174. DOI: https://doi.org/10.1145/3025453.3025553

[11] Eslami, M., 2015. "I always assumed that I wasn't really that close to [her]": Reasoning about invisible algorithms in news feeds. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. pp. 153–162. DOI: https://doi.org/10.1145/2702123.2702556

[12] Ma, J., Chen, X., 2024. Fingerprint image generation based on attention-based deep generative adversarial networks and its application in deep Siamese matching model security validation. Journal of Computational Methods in Engineering Applications. 1–13.

[13] Jhaver, S., Zhang, A.Q., Chen, Q.Z., et al., 2023. Personalizing content moderation on social media: User perspectives on moderation choices, interface design, and labor. Proceedings of the ACM on Human-Computer Interaction. 7(CSCW2), 1–33. DOI: https://doi.org/10.1145/3610099

[14] Ma, J., Zhang, Z., Xu, K., et al., 2025. Improving the applicability of social media toxic comments prediction across diverse data platforms using residual self-attention-based LSTM combined with transfer learning.

[15] Kim, H., Lim, Y.-K., 2023. Investigating how users design everyday intelligent systems in use. In Proceedings of the 2023 ACM Designing Interactive

Systems Conference. pp. 702–711. DOI: https://doi.org/10.1145/3563657.3596110

[16] Ma, J., Wilson, A., 2025. Mitigating FGSM-based white-box attacks using convolutional autoencoders for face recognition.

[17] Jannach, D., Lerche, L., Zanker, M., 2018. Recommending based on implicit feedback. In P. Brusilovsky & A. Kobsa (eds.). Social information access: Systems and technologies. Springer. pp. 510–569. DOI: https://doi.org/10.1007/978-3-319-90092-6_14

[18] Zhang, G., Zhou, T., Cai, Y., 2023. CORAL-based domain adaptation algorithm for improving the applicability of machine learning models in detecting motor bearing failures. Journal of Computational Methods in Engineering Applications. 1–17.

[19] Lu, P.-M., Zhang, Z., 2025. The model of food nutrition feature modeling and personalized diet recommendation based on the integration of neural networks and K-means clustering. Journal of Computational Biology and Medicine. 5(1).

[20] Gan, Y., Zhu, D., 2024. The research on intelligent news advertisement recommendation algorithm based on prompt learning in end-to-end large language model architecture. Innovations in Applied Engineering and Technology. 1–19.

[21] Kelly, D., Teevan, J., 2003. Implicit feedback for inferring user preference: A bibliography. ACM SIGIR Forum. 37(2), 18–28. DOI: https://doi.org/10.1145/945546.945550

[22] Zhang, H., Zhu, D., Gan, Y., et al., 2024. End-to-end learning-based study on the Mamba-ECANet model for data security intrusion detection. Journal of Information, Technology and Policy. 1–17.

[23] McCrosky, B.R., Mozilla Foundation, 2022. Does this button work? Investigating YouTube's ineffective user controls. Mozilla Foundation. Available from: https://www.mozillafoundation.org/en/research/library/user-controls/

[24] Van der Nagel, E., 2018. 'Networks that work too well': Intervening in algorithmic connections. Media International Australia. 168(1), 81–92. DOI: https://doi.org/10.1177/1329878X18783004

[25] Wilson, A., Ma, J., 2025. MDD-based domain adaptation algorithm for improving the applicability of the artificial neural network in vehicle insurance claim fraud detection. Optimizations in Applied Machine Learning. 5(1).