Event-Based Data Pipelines in Recommender Systems: The Data Engineering Perspective

Deexith Reddy[1](✉), Urjoshi Sinha[2], and Rohan Singh Rajput[3]

[1] University of Connecticut, Storrs, Connecticut, USA
deexith.reddy@uconn.edu
[2] Iowa State University, Ames, Iowa, USA
urjoshi@iastate.edu
[3] Headspace, Los Angeles, California, USA
rohan.rajput@headspace.com

Abstract. Recommender Systems (RS) are information retrieval systems that can be used for serving personalized content to online users. Most industrial recommendation systems utilize a large amount of online data to generate personalized recommendations for users. The quality of the data plays an important role in the performance of the RS. The majority of the RS data is generated from event data that are stored in data lakes through multiple data pipelines. Event-based data pipelines have emerged as a popular approach to handle the massive amount of data generated by modern applications. In this paper, we explore the impact of event-based data pipelines on recommendation systems. We discuss how these pipelines enable efficient data ingestion, real-time processing, and low-latency recommendations.

Keywords: Information Retrival · Data Engineering · Data Pipeline · Recommender Systems

1 Introduction

Recommendation Systems have become a crucial component of modern applications, providing users with personalized content and product suggestions based on their preferences, behavior, and context. However, as the volume and velocity of data continue to grow, traditional data processing methods struggle to keep up with the demand for real-time recommendations. Event-based data pipelines, which rely on asynchronous data streaming and processing, have emerged as a promising solution to address these challenges.

In this section, we provide a complete background of our study. First, we briefly describe recommendation systems and event-based data pipelines. Next, we present the importance of incorporating event-based data pipelines in RS. We also provide a real-world example of RS, demonstrating the use of event-based pipelines. Finally, we present the key contributions of our study.

D. Reddy, U. Sinha, and R. S. Rajput—Contributed equally to this work.

M. H. Miraz et al. (Eds.): iCETiC 2023, LNICST 538, pp. 29–43, 2024.
https://doi.org/10.1007/978-3-031-50215-6_3

Recommendation Systems: RS analyze user data to provide personalized suggestions and enhance user experience. They can be classified into three major categories: collaborative filtering, content-based filtering, and hybrid systems. Collaborative filtering leverages user-item interactions, content-based filtering considers item features, whereas hybrid systems combine both approaches.

Event-Based Data Pipelines: They are designed to handle high-velocity and high-volume data streams. They are built using event-driven architectures, where events represent changes in the state of an application. The main components of event-based data pipelines include event producers, event brokers, and event consumers. Producers generate events, brokers manage and route events, and finally consumers process and react to events.

Advantages of Event-Based Data Pipelines on Recommendation Systems: An event-based data pipeline can play a pivotal role in the performance of a recommendation system as compared to a traditional data processing pipeline. Below we discuss some of the advantages of event-based data pipelines.

- **Efficient Data Ingestion:** Event-based data pipelines enable efficient ingestion of user activity and content metadata, providing a scalable and fault- tolerant mechanism for handling large-scale data streams. This capability is critical for recommendation systems, as the volume and variety of data can directly impact the quality and relevance of recommendations [21].
- **Real-Time Processing:** By enabling real-time data processing, event-based data pipelines allow recommendation systems to react to user behavior and preferences more rapidly. Real-time recommendations can significantly enhance user experience, thereby increasing engagement, satisfaction, and conversion rates [21].
- **Low-Latency Recommendations:** Event-based data pipelines facilitate low-latency recommendations by minimizing the time between data ingestion and recommendation generation. This feature is especially important for applications with highly dynamic content, such as news or e-commerce platforms, where outdated recommendations can lead to reduced user satisfaction and missed opportunities.

Motivating Real-World Example: In the context of the rapid growth of digital platforms, YouTube, a prominent online video-sharing platform, faced difficulties in providing real-time, relevant recommendations to its users due to the high volume and variety of data generated by user interactions and content updates [9]. This issue resulted in decreased user satisfaction as the users did not receive accurate video recommendations and missed opportunities for both YouTube and its content creators. The platform had to deal with an extensive catalog of videos, constantly updated content, and a myriad of user interactions that included views, likes, comments, and shares. The complexity and scale of this data overwhelmed their traditional data processing pipelines.

This real-world example highlights the need for a more robust and efficient event-based data pipeline to address the challenges faced by recommendation systems in similar scenarios. YouTube's struggles and solutions underscore the importance of efficient data engineering in improving personalized recommendations and enhancing user experiences in today's large-scale digital platforms.

Contributions: Below we list the key contributions of this work which includes-

- A detailed study and presentation of event-based pipeline issues: The paper provides in-depth knowledge of problems that can arise in event-based pipelines, a crucial component for recommender systems. By elaborating on challenges like event duplication, missing events, and event corruption, the paper provides a framework for practitioners and researchers to analyze their own systems.
- Performance improvement of recommender systems: By addressing the identified issues, the recommender system's performance can be enhanced, leading to better resource utilization, accurate recommendations, and improved end-user experience.

In the next section, we discuss the gaps in relevant literature focusing on event-based pipelines. Next, we present in detail all the possible challenges associated with event-based pipelines. We further present a case study that helps us understand these existing issues better. Finally, we discuss our observations and some of the possible solutions to the existing problems. We then conclude with future directions of our work.

2 Related Work

In several existing literature, such as the survey by Bobadilla et al. [4], various challenges and data quality issues are outlined that affect recommender systems. However, these work do not exclusively focus on event-based pipelines. They provide some general insights that can be applied to this context.

Several studies have highlighted the importance of data pipelines in the context of recommender systems. Chen and Gao [7] proposed a comprehensive pipeline for a hotel recommendation system, which involved pre-processing raw data and training prediction models. Their work emphasized the importance of data organization and analysis in building effective recommender systems.

Deng et al. [10] provided a comprehensive review of recommender systems based on graph embedding techniques. The study proposed a general design pipeline and compared graph embedding-based recommendation models with conventional models, suggesting a trade-off between the two approaches in different tasks. Mazaheri et al. [17] explored the feasibility of a collaborative filtering system to recommend pipelines and datasets based on provenance records from previous executions. Their work underscored the potential of data pipelines in enhancing the performance of recommender systems. Vrijenhoek et al. Vrijenhoek et al. [27] conducted an analysis of the MIND dataset [31] for their research focused on diverse news recommendations. They further discussed the effects that various stages of the recommendation pipeline have on the distribution of different article categories. Tagliabue et al. [25] have argued that immature data pipelines are preventing a large portion of industry practitioners from leveraging the latest research on recommender systems. The study proposed a template data stack for machine learning at a reasonable scale, demonstrating how many challenges can be addressed by embracing a serverless paradigm.

While these studies implicitly acknowledge the importance of data pipelines in recommender systems, they do not explicitly discuss the role of event-based data pipelines.

This observation underscores the motivation for our study, which aims to explore the potential challenges, impact, and solutions of event-based data pipelines in recommender systems. By doing so, we hope to fill this gap in the literature and provide valuable insights for other researchers and practitioners in the field.

3 Challenges with Event-Based Pipelines

In this section, we first present an overview of the different components of the recommendation system architecture. Next, we identify and discuss some common issues associated with event-based pipelines in RS.

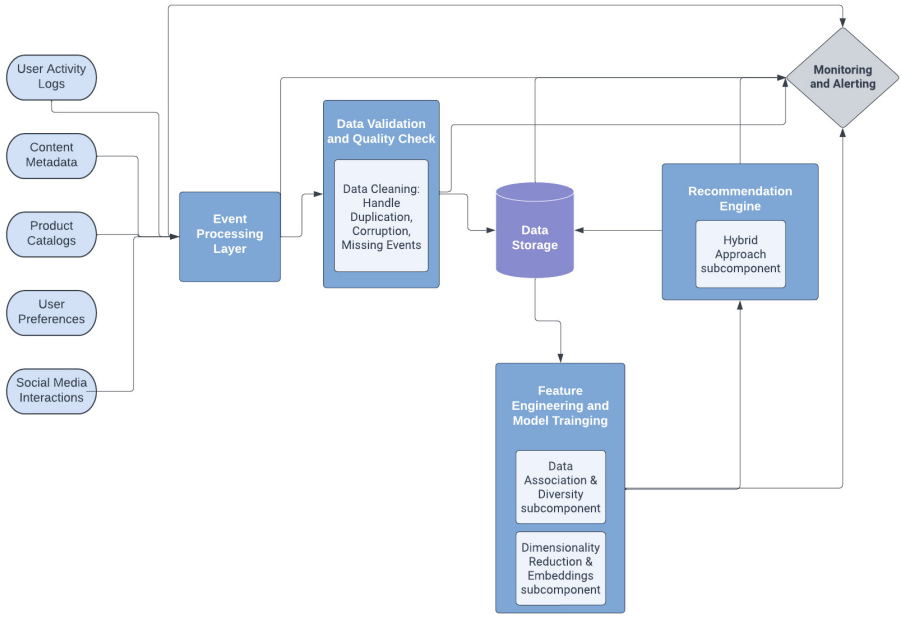

Fig. 1. Architecture of data pipelines for Recommendation Systems

Figure 1 delineates the various stages involved in the development and operation of the recommender system. It commences with feature engineering and model training, which are crucial for the recommendation engine to generate relevant suggestions.

The data validation and quality check stage ensures the integrity and accuracy of the data being processed. This stage is critical for maintaining the reliability of the system and ensuring that the recommendations generated are based on valid and high-quality data.

The monitoring and alerting stage is responsible for overseeing the system's performance and raising alerts in case of any anomalies or issues. This stage is essential for maintaining the system's robustness and swiftly addressing any problems that could impact its performance or the quality of its recommendations. The recommendation

engine, which is prominently featured in the diagram, is the core of the system. It utilizes processed and validated data to generate recommendations.

Having outlined the architecture and operational flow of the recommender system from a data engineering perspective, it is essential to address the inherent challenges that come with designing, implementing, and maintaining such complex data-driven recommender systems. In the following section, we delve into these challenges, their impact on the data engineering landscape, and the potential solutions to mitigate them.

3.1 Event Duplication

Event duplication is a common problem in event-based pipelines in recommendation systems. The event duplication results from inadequate event auditing and a lack of a de-duplication pipeline. The system contains multiple components that trigger events regularly, causing some events to be logged repeatedly. Such an event may also include additional logic to collect logs from various system sections. For instance, in a music streaming platform like Spotify, user interactions such as playing a song, liking a song, adding a song to a playlist, and browsing through song recommendations are collected [3]. These events are crucial for generating personalized recommendations for users. However, when not instrumented properly, these events may generate many duplicate entries in the database.

The duplicate entries can lead to several negative effects, such as increased storage consumption, reduced query performance, and data inconsistencies. Increased storage consumption in turn may result in higher infrastructure costs and can become a bottleneck for systems with limited resources. Duplicate entries could also slow down query performance, as the database has to process redundant data, leading to increased latency for real-time analytics and user-facing applications. Moreover, data inconsistencies can arise when multiple duplicate records are updated with different values, causing confusion and errors in the downstream processes [3, 6].

Impact: Event duplication can cause multiple problems in the performance of the recommendation system. In sequential recommendation, a duplicate event might create a popularity or repeatability bias. For example, an online streaming service can create false data about users' repeated play of content, such as a song being played multiple times due to duplicate events. Such behavior can incorrectly infuse a popularity bias into the system. It can also lead to overfitting of the model.

Potential Solution: In order to avoid event duplication, performing a proper audit of the event is necessary. It also involves performing rigorous testing of the system's event instrumentation to ensure accurate tracking and logging of user interactions, thus eliminating duplicate entries and improving the recommendation system's performance.

3.2 Missing Events

Missing events can occur when an event doesn't fire correctly or is implemented incorrectly. For example, in an e-commerce website, the system may not capture user behavior

entirely, such as failing to log product views or add-to-cart actions. A real-world example of the impact of missing events can be found in a study by Zhang and Jansen [32], which investigated the effects of missing click-through data on click-through rate (CTR) estimation in sponsored search advertising.

Impact: Missing data in a machine learning model can lead to several disadvantages, including reduced accuracy, biased predictions, poor performance, and difficulty in model training. Algorithms may struggle to handle incomplete data, causing longer training times, which is detrimental when time is a critical factor.

Potential Solution: To tackle the issue of missing events, it is crucial to establish a well-designed and reliable event tracking system, such as using a combination of client-side libraries and server-side event collectors. For example, implementing a solution such as a custom-built event tracking system can provide comprehensive event capture and logging. This solution should encompass meticulous testing methods for client event instrumentation, ensuring accurate capture of all relevant user events, minimizing the risk of data loss, and seamless integration with the existing data pipeline. In addition, incorporating resilient monitoring systems, such as Prometheus or ELK stack, and error reporting techniques, like Sentry or Rollbar, can help detect and resolve missing event problems promptly. These tools enable real-time tracking and alerting on potential issues, allowing developers to address them proactively. Adopting this strategy guarantees that the recommendation system functions with comprehensive and precise data, resulting in enhanced model performance and dependable recommendations [11, 12, 22, 30].

3.3 Event Corruption

The primary reason for event corruption is the faulty implementation of events on the client or server side. If an event is not correctly captured, it adds noise to the machine-learning model. One such example of event corruption is capturing an incorrect input of a click event from the user. Many e-commerce websites add multiple buttons to capture various user actions. However, due to faulty implementation, it may capture false clicks of events and thus add noise to the data.

Impact: Noisy data can have a negative impact on the performance of a machine learning model. If the data used to train the model is noisy, meaning it contains a lot of irrelevant or incorrect information, it can cause the model to make inaccurate predictions. This is because the model is being trained on data that doesn't accurately capture the real-world phenomenon it is trying to model. So the model does not learn the correct patterns and relationships. As a result, the model is unable to generalize well to new data and might produce poor results when used in practice. It is therefore important to clean and preprocess the data before using it to train a machine learning model, to ensure that it is as accurate and relevant as possible.

Potential Solution: To mitigate event corruption, it is essential to adopt a comprehensive approach encompassing thorough event examination and validation processes. By crafting tailored test cases that align with the specific client event instrumentation, data integrity can be maintained. Furthermore, integrating advanced real-time tracking

systems, along with machine learning based anomaly identification techniques, allows for the swift detection and resolution of event corruption issues. One notable example is the use of machine learning for anomaly detection in e-commerce platforms, where event data can be monitored and analyzed for inconsistencies, such as unusual patterns in user clicks or browsing behavior [30]. By following this proactive strategy, the recommendation system can work with high-quality data, resulting in improved model performance.

3.4 Data Association

Data association refers to the process of linking and correlating various data points, often originating from different sources, to create a comprehensive and cohesive understanding of user behavior and preferences. In recommendation systems, data association plays a vital role in generating accurate and personalized recommendations by leveraging the relationships between different data elements.

Impact: Inadequate or incorrect data association can lead to an incomplete or distorted view of user preferences, resulting in sub-optimal recommendations being generated. Therefore, it is essential to ensure accurate and robust data association in recommendation systems to optimize the quality of generated recommendations.

Potential Solution: To tackle data association challenges in recommendation systems, it is crucial to implement a robust data integration and preprocessing pipeline. This pipeline should involve merging data from multiple sources, cleaning and transforming data, and identifying meaningful relationships among various data elements. Employing advanced techniques such as entity resolution, record linkage, and feature engineering can help improve data association and create a more accurate representation of user preferences.

3.5 Data Evolution

Data evolution refers to the continuous changes in data patterns and distributions over time, which can result from shifts in user preferences, market trends, or system updates. In the context of recommendation systems, data evolution can lead to challenges when adapting to these changes to maintain the accuracy and effectiveness of the recommendations. Understanding these evolution patterns is key to understanding the structures of input data for recommender systems, highlighting the importance of considering data evolution in the design and implementation of recommender systems [28].

Impact: As data evolves, the underlying relationships and patterns in the data may shift, which can render a previously effective machine learning model less accurate or even obsolete. Failing to adapt the model to these changes can result in recommendations that are less relevant to users. Consequently, it is crucial to monitor data evolution in recommendation systems and update the models to reflect the changing data landscape.

Potential Solution: To address data evolution in recommendation systems, it is essential to establish a dynamic and adaptive modeling process. This process should include continuous monitoring of data trends, timely model retraining with up-to-date data, and fine-tuning model parameters as necessary. Leveraging techniques such as online learning, transfer learning, and active learning can also help in adapting to data evolution effectively. Employing these methods would help the recommendation system in staying attuned to the ever-changing data environment, thus ensuring the provision of relevant and accurate recommendations to users.

3.6 Cold Start

Recommender systems, particularly those employed in e-commerce platforms, rely heavily on user interactions such as browsing, searching, adding items to a cart, making purchases, leaving reviews, and providing feedback. However, a significant challenge arises when a new user visits the platform for the first time. In such instances, the system lacks any prior information about the user, leading to what is known as the 'cold-start' problem. This issue hampers the system's ability to provide recommendations with the same confidence as it does for existing users. Furthermore, each time a user visits the website, their interests might differ, adding another layer of complexity to the problem.

Impact: The cold-start problem in recommender systems can significantly impact their performance and user experience. As per Bouneffouf et. al [5], this issue can lead to inefficiency, as the system may struggle to provide accurate recommendations due to a lack of data, leading to potential resource wastage. It can further result in poor user experience. Furthermore, this problem can lead to a loss of revenue, as users may not engage with poorly targeted recommendations, leading to missed sales opportunities. Finally, the cold-start problem can make it difficult for the system to evaluate new items accurately, affecting the overall quality of recommendations. Thus, addressing the cold-start problem is crucial for the effectiveness of recommender systems [16, 28].

Potential Solution: To address the cold-start problem, it is crucial to monitor the data fed into the recommender systems meticulously. One potential solution proposed in the literature involves the use of a new algorithm named DotMat. The DotMat algorithm, designed to address the cold-start problem in recommender systems, leverages Zipf's Distribution and the RankMat algorithm. Zipf's Distribution, a statistical law observed in many natural phenomena, states that a few items occur very frequently while many items occur rarely. For example, in a large text, a few words like "the", "and", "of" are used very often, while most words are used infrequently. This principle is also applied in the RankMat algorithm, which remodels the probabilistic framework of matrix factorization using power law distribution, a pattern often observed in user behavior. We could consider a music streaming platform where a few songs are played very frequently while many songs are played rarely. RankMat leverages this pattern to predict user preferences, enabling accurate recommendations even with limited user history. Thus, by combining the principles of Zipf's Distribution and RankMat, DotMat offers a promising solution to the cold-start problem, ensuring accurate and high-quality recommendations [16, 28].

3.7 Sparsity

Sparsity in recommender systems refers to situations where there is insufficient transaction and feedback data available, making the prediction process biased and less optimal. This problem can arise when the end-user interacts with only a small portion of items in a particular application domain. For instance, in datasets containing ratings of movies, there may exist rating matrices that are not fully populated, say only about 10% of that matrix bears ratings. Another common issue is the lack of data about new entities such as a new item or a new user. Whenever a new user is added to a system, where they have not yet had an opportunity to rate any items, the system cannot compute similarity to other users. Similarly, for a new product, the system can't recommend the product before a rating has actually been received for it [8, 28].

Impact: The sparsity problem can significantly affect the quality of recommendations. With sparse data, it becomes difficult for the system to make high-quality recommendations. This can lead to a sub-optimal user experience, as the recommendations may not align with the user's preferences or needs. Furthermore, the lack of data about new entities can hinder the system's ability to provide relevant recommendations for new users or to promote new products.

Potential Solution: A proposed solution to the sparsity problem is the DotMat algorithm [28]. This algorithm, which does not require any additional input data, can be used as a preprocessing step for recommender systems to alleviate sparsity problems. This process allows us to fill in the missing values in the user-item rating table by using the features of users and items, before generating any recommendations.

3.8 Maintaining the Integrity of the Specifications

When designing recommender systems, it is crucial to consider various design aspects relevant to structuring and handling the specific design complexities from the application's context. However, most of the existing literature does not present models for easing the capture and use of requirement specifications when designing recommender systems. There is a scarcity of models that provide an overview and explanation of the design considerations engineers face when developing recommender systems, particularly for domain-specific complex recommender system applications.

Impact: The lack of models and methods for maintaining the integrity of specifications in recommender systems can lead to inefficiencies in the design process and potential inaccuracies in the resulting recommendations.

Potential Solution: To support the designer in this task of making a wellconsidered recommender system design, it is necessary to develop and incorporate methods to maintain the integrity of requirements specifications. Following good practices as described for software requirements engineering would be helpful in this case [2].

3.9 Event Corruption

Event corruption refers to the distortion or degradation of data as it moves through the data pipelines. This typically occurs when there is a loss of data due to missing information, discrepancies between expected and actual values, or intentional or unintentional tampering with the data pipelines. Event corruption can also be caused by mismatches in database schemas. A lack of error handling and validation safeguards can further amplify the occurrence of event corruption incidents.

Impact: Wrong data about user interactions or user interests might make their way to the recommender system loop, which can cause the system to make incorrect recommendations and in turn lead to a loss of revenue. For example, if a user's preferences are recorded incorrectly, the recommendations may not align with their actual preferences. Or, if a popular item is mistakenly labeled as having high user ratings, it may be recommended more often, even if it is not a good match for a particular user.

Potential Solution: One could ensure data quality checks where returning user data is compared to the user's previous visits to a site. These could include a data-completeness check where the matrix that is provided as input to the recommender system is complete. An alternate method may include data consistency checks. If user information is stored in multiple databases, then it must be consistent across all of them. While data duplication can occur if a user has rated the product twice, the system must take only a single record when considering that particular user. Lastly, data timeliness checks are also necessary. For such a check, only the latest interaction data for the user is considered and provided as input to the recommender system. Finally, it is also necessary to establish appropriate ranges on the data that is being provided to the recommender system.

3.10 Data Volume

The event-based data pipeline processes a large amount of data. This substantial bulk of event-based data primarily originates from transaction records. Many recommender system algorithms, such as collaborative filtering, utilize vast amounts of user-item data. High volume of such data may contribute to several issues in event-based data pipelines.

Impact: As data volume increases, the scalability of the recommender system must be increased. If the data volume is large and the system is not scaled enough, it may lead to slow response times, high resource utilization, and difficulty in adapting to changing data. These factors can yet again lead to low user engagement for any business.

Potential Solution: With improvements in cloud computing, recommender systems can be scaled to handle larger volumes of data. Also, parallel processing and distributed computing can be leveraged to distribute the computation workload across multiple processors or machines, allowing the system to handle large volumes of data more efficiently. Another important remedy for large data volumes is incremental learning, which involves updating the recommendation algorithm in real-time as new data becomes available. This can help the system adapt to changing data and improve accuracy over time, while also reducing the volume of data that needs to be processed at once.

3.11 Data Diversity

As the number of users and items in a recommender system increases, the diversity of the data also escalates. This escalating diversity poses a challenge in generating accurate recommendations. Particularly, if the data is skewed towards a specific demographic or genre, the system may struggle to provide precise recommendations for a broader audience [15, 29].

Impact: The increasing data diversity in recommender systems can lead to biased recommendations, especially when the data is skewed towards a particular demographic or genre. This bias can result in less accurate recommendations for a broader audience, thereby reducing the effectiveness of the recommender system. The issue becomes more pronounced as the number of users and items in the system grows, making it more challenging to create accurate recommendations [15, 29].

Potential Solution: To address the problem of data diversity, it is beneficial to incorporate contextual data such as user demographics, location, and browsing history. This data can be used to personalize recommendations and make them more relevant to the user. By incorporating contextual data, the system can better understand the user's preferences and provide recommendations that are tailored to their interests. Additionally, ensemble methods can be used, which involve combining multiple recommendation algorithms to produce more accurate and diverse recommendations. By using different algorithms, the system can account for the diversity of user preferences and item characteristics, leading to more accurate and diverse recommendations. Diversity constraints can also be added to the recommendation algorithm to ensure that the system generates a diverse set of recommendations. For instance, the system can be designed to recommend items from different categories or genres to provide a broader range of options [15, 29].

4 Comparative Analysis of Research Studies

In this section, we present an analysis by studying several work which highlight the issues related to event-based data pipelines for recommender systems. We identify the issues discussed earlier in these studies and try to correlate the dependency of such issues on each other. We further present the areas where certain issues are more likely to occur than others.

Table 1 provides a brief survey of different Recommendation System papers that covers various related problems commonly observed for recommendation systems. We linked each of those problems with the associated event-based problems.

In our comparative analysis of research papers, we identified key challenges and related problems in the context of event-based data pipelines for recommender systems. The table above presents a succinct overview of these issues. The challenges were categorized into nine distinct problems: Gathering Known Ratings for Matrix, Cold Start, Sparsity, Scalability, Over Specialization, Lack of Data, Changing Data, Changing User Preferences, and Unpredictable Items [1, 2, 13, 14, 18–20, 23, 24, 26]. Each of these problems has associated issues related to event-based data, such as Missing Events,

Table 1. Different problems in data pipelines along with related event-based issues

Problem	Related Problems	Event Based Issues
Gathering Known Ratings for Matrix	Sparsity, Cold Start	Missing Events
Cold Start	Gathering Known Ratings for Matrix, Sparsity	Missing Events, Duplicate Events
Sparsity	Gathering Known Ratings for Matrix, Cold Start	Missing Events
Scalability	Lack of Data	Missing Events
Over Specialization	Lack of Diversity	Missing Events
Lack of Data	Scalability	Duplicate Events, Event Corruption
Changing Data	Unpredictable Items, Changing User Preferences	Duplicate Events, Event Corruption
Changing User Preferences	Changing Data, Unpredictable Items	Duplicate Events
Unpredictable Items	Changing Data, Changing User Preferences	Duplicate Events, Missing Events

Duplicate Events, and Event Corruption. Moreover, each problem is intricately connected to other challenges, thereby forming a complex web of interrelated issues. For instance, the "Gathering Known Ratings for Matrix" issue is closely related to the sparsity and the cold start problem, and this issue is further exacerbated in an event-based context due to the occurrence of missing events. Similarly, "Cold Start", "Sparsity", and "Scalability" issues all grapple with the challenge of missing events.

The "Over Specialization" and "Lack of Data" issues share the common event-based issue of Missing Events, with the latter also facing challenges related to Duplicate Events and Event Corruption. Furthermore, the problems of "Changing Data", "Changing User Preferences", and "Unpredictable Items" are intertwined, with all three facing difficulties related to Duplicate Events and Event Corruption. The "Unpredictable Items" problem also involves the additional challenge of Missing Events.

This summary underscores the complexity and inter-connectedness of the complexities in developing and maintaining event-based data pipelines for recommender systems. Addressing these issues requires a holistic approach that takes into account the multifaceted nature of these problems and the intricate ways in which they interact within an event-based environment.

Figure 2 presents the occurrence of a specific issue as indicated in related literature. The y-axis represents the issues observed and the x-axis denotes the percentage of time a specific issue is observed out of the total papers reviewed. We see that Cold-Start, Scalability and Sparsity issues are the most frequently occurring problems as noted by other users. On the other hand, issues such as ones related to Latency and Changing User Preferences are fairly common but less frequent than some other problems.

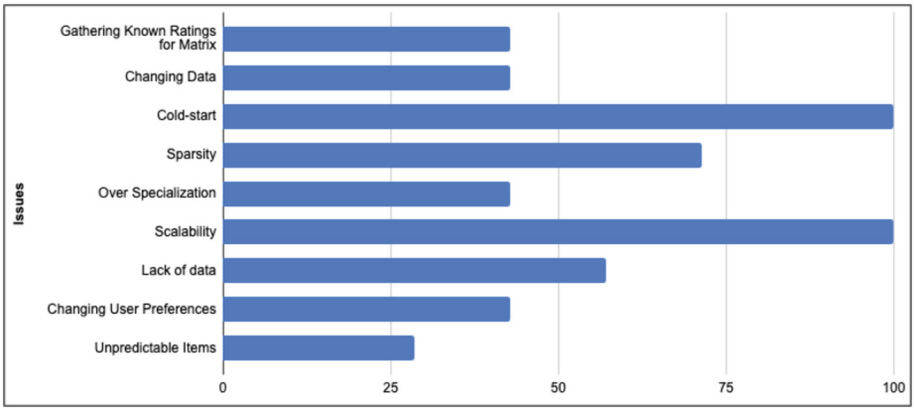

Fig. 2. Distribution of occurrence (in %) of problems observed in event based Recommender Systems

5 Conclusion

Event-based data pipelines can facilitate the creation of more personalized and relevant recommendations, leading to improved user engagement and satisfaction. Event-based data pipelines are likely to play an increasingly important role in the development of next-generation recommendation systems.

This work includes a comprehensive review of the challenges associated with event-based data pipelines in recommendation systems, such as event duplication, missing events, event corruption, data association, cold start, data evolution, sparsity, integrity, event corruption, data volume, and data diversity issues. We analyzed the impact of these challenges on the performance and effectiveness of recommendation systems. We further explored potential solutions for mitigating the issues listed. In this study, we presented the key takeaways that would be helpful for engineers and other stakeholders building recommender systems.

In conclusion, event-based data pipelines can have a significant impact on the performance of recommendation systems. By enabling efficient data ingestion, real-time processing, and low-latency recommendations, event-based data pipelines can provide a more scalable and effective solution for handling high-velocity and high-volume data streams.

6 Limitations and Future Work

There are several impediments in the sharing of information by companies that build and use event-based recommendation system architectures. This may include security, data privacy, and other constraints. The lack of such information makes it difficult for researchers in this area to understand or reproduce similar issues and challenges.

In order to alleviate these problems, we need a standardized framework that can help us catalog multiple types of architectures that can be used for different types of systems.

For instance, in some cases, a monolithic architecture may provide better performance over micro-service-based architectures. Future research could explore the impact of event-based data pipelines on different types of recommendation systems, including collaborative filtering, content-based filtering, and hybrid systems.

Acknowledgements. While the original ideas, study, findings, and interpretations expressed in this paper are our own, the clarity of the presentation in specific sub-sections was achieved with the assistance of ChatGPT which helped us in enhancing the readability of the paper.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005). https://doi.org/10.1109/TKDE.2005.99
2. Afchar, A., Epure, E.V., Mille, A., Moussallam, M.: Explainability in music recommender systems. arXiv preprint arXiv:2201.10528 (2022)
3. Ahmad, S., Lavin, A., Purdy, S., Agha, Z.: Unsupervised real-time anomaly detection for streaming data. Neurocomputing **262**, 134–147 (2017)
4. Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A.: Recommender systems survey. Knowl.-Based Syst. **46**, 109–132 (2013)
5. Bouneffouf, D., Rish, I., Aggarwal, C.: Survey on applications of multi-armed and contextual bandits. IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (2020). https://doi.org/10.1109/CEC48606.2020.9185782
6. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., Tzoumas, K.: Apache flink: Stream and batch processing in a single engine. Bull. Tech. Committee Data Eng. **38**(4), 13–26 (2015)
7. Chen, J., Gao, Z.: A comprehensive pipeline for hotel recommendation system. arXiv preprint arXiv:2009.01860 (2020)
8. Choi, S.M., Lee, D., Jang, K., Park, C., Lee, S.: Improving data sparsity in recommender systems using matrix regeneration with item features. Mathematics **11**(2), 292 (2023). https://doi.org/10.3390/math11020292
9. Covington, P., Adams, J., Sargin, E.: Deep neural networks for youtube recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 191–198 (2016)
10. Deng, Y.: Recommender systems based on graph embedding techniques: a comprehensive review. arXiv preprint arXiv:2109.09587 (2021)
11. Gautham, S., Bakirtzis, G., Will, A., Jayakumar, A.V., Elks, C.R.: Stpadriven multilevel runtime monitoring for in-time hazard detection. arXiv preprint arXiv:2204.08999 (2022)
12. Gomes-Ju´nior, A.R., Santana, T., Winter, O.C., Sfair, R.: The main perturbing objects on the orbits of (616) prometheus and (617) pandora. arXiv preprint arXiv:2202.01617 (2022)
13. Kohar, M., Rana, C.: Survey paper on recommendation system. Int. J. Comput. Sci. Inf. Technol. **3**(2), 3460–3462 (2012)
14. Kumar, B., Sharma, N.: Approaches, issues and challenges in recommender systems: a systematic review. Indian J. Sci. Tech. **9**(47), 94892 (2016). https://doi.org/10.17485/ijst/2016/v9i47/94892
15. Kunaver, M., Požrl, T.: Diversity in recommender systems a survey. Knowl.-Based Syst. 123, 154–162 (2017). https://doi.org/10.1016/j.knosys.2017.02.009, https://www.sciencedirect.com/science/article/pii/S0950705117300680

16. Lika, B., Kolomvatsos, K., Hadjiefthymiades, S.: Facing the cold start problem in recommender systems. Expert Syst. Appl.: Int. J. **41**, 2065–2073 (2014). https://doi.org/10.1016/j.eswa.2013.09.005
17. Mazaheri, M., Kiar, G., Glatard, T.: A recommender system for scientific datasets and analysis pipelines. arXiv preprint arXiv:2108.09275 (2021)
18. Mishra, N., et al.: Research problems in recommender systems. J. Phys.: Conf. Ser. **1717**(1), 012002 (2021). https://doi.org/10.1088/1742-6596/1717/1/012002
19. Mohamed, M.H., Khafagy, M.H., Ibrahim, M.H.: Recommender systems challenges and solutions. In: 2019 International Conference on Innovative Trends in Computer Engineering (ITCE'2019), pp. 1–6 (2019)
20. Ngoc, T.V., Thi, H.T.: Techniques, benefits, and challenges of recommendation system in e-commerce: a literature review. In: Proceedings of the International Conference on Industrial Engineering and Operations Management, pp. 107–114 (2021)
21. Ponnuswami, G., Kailasam, S., Dinesh, D.A.: Event-driven data pipeline for network management systems. In: 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1–6 (2020). https://doi.org/10.1109/ICCCNT49239.2020.9225344
22. Roa, J., Farnocchia, D., Chesley, S.R.: A novel approach to asteroid impact monitoring and hazard assessment. arXiv preprint arXiv:2108.03201 (2021)
23. Roy, D., Dutta, M.: A systematic review and research perspective on recommender systems. J. Big Data **9**(1), 59 (2022). https://doi.org/10.1186/s40537-022-00414-2
24. Sharma, L., Gera, A.: A survey of recommendation system: research challenges. Int. J. Eng. Trends Technol. **4**(5), 1989–1995 (2013)
25. Tagliabue, J.: You do not need a bigger boat recommendations at reasonable scale in a (mostly) serverless and openstack. In: Woodstock '18: ACM Symposium on Neural Gaze Detection, p. 6. ACM (2018)
26. Tiwalola, A.B., Asafe, Y.N.: A comprehensive study of recommender systems: prospects and challenges. Int. J. Adv. Res. Comput. Sci. Softw. Eng. **9**(5), 224–230 (2019)
27. Vrijenhoek, S.: Do you mind? Reflections on the mind dataset for research on diversity in new recommendations. arXiv preprint arXiv:2304.08253 (2023)
28. Wang, H.: Dotmat: solving cold-start problem and alleviating sparsity problem for recommender systems. arXiv preprint arXiv:2303.14419 (2023)
29. Wang, H.: Evolution of the online rating platform data structures and its implications for recommender systems. arXiv preprint arXiv:2303.14419 (2023)
30. Wang, Y., et al.: Experimental comparison of various techniques for spot size measurement of high-energy x-ray source. arXiv preprint arXiv:1511.07668 (2015)
31. Wu, F., et al.: Mind: a large-scale dataset for news recommendation. In: Proceedings of the Association for Computational Linguistics (ACL) (2020)
32. Zhang, M., Jansen, B.J.: The effect of missing click-through data on click through rate estimation in sponsored search. Inf. Process. Manage. **47**(4), 671–688 (2011)