Hybrid Cascade Model for Personalized Recommendations

Vladyslav Isariev[1], Iryna Hurklis[1] Kateryna Shulakova[1,2] and Oksana Vasylenko[3]
[1]*Department of Information Technologies and Cybersecurity, State University of Intelligent Technologies and Telecommunications, Kuznechna Str. 1, Odesa, Ukraine*
[2]*Anhalt University of Applied Sciences, Bernburger Str. 57, Köthen, Germany*
[3]*Preparatory College of Saxony-Anhalt, Anhalt University of Applied Sciences, Lohmann Str. 23, 06366 Köthen, Germany*
vladsw764@icloud.com, i.v_hurklis@suitt.edu.ua, katejojo29@gmail.com, oksana.vasylenko@hs-anhalt.de

Keywords: Recommendation Systems, Collaborative Filtering, Personalized Pagerank, Hybrid Model, Information Retrieval.

Abstract: In today's digital ecosystem, users are overwhelmed by heterogeneous streams of information from social networks, e-commerce platforms, and streaming services, and the ability to filter this flow and deliver personalized content has become a key differentiator for digital platforms. Recommendation systems form the core of modern personalization technologies, shaping how users interact with digital platforms and influencing engagement, retention, and overall growth. Yet older approaches still face persistent obstacles. Collaborative filtering often performs well once a solid history of interactions exists, but its accuracy quickly drops when data are sparse or when new users and items appear. Graph-based methods such as Personalized PageRank capture the structure linking users and items, yet they demand more computation and scale less easily. This paper presents a hybrid cascade that combines collaborative filtering with a Personalized PageRank re-ranker to balance accuracy, coverage, and scalability. The system is built in Java with Spring Boot, PostgreSQL, and Redis in a modular design that supports real-time operation. Collaborative filtering first proposes candidate recommendations, then a graph-diffusion pass on a bipartite user–item graph orders them. On a social media analytics dataset, the hybrid outperforms either method alone, achieving a precision of 0.75, a recall of 0.68, and a mean absolute error of 0.10. These results confirm the effectiveness of combining similarity-driven and structure-aware techniques to overcome data sparsity and improve robustness. The contribution of this study lies in the design of a hybrid architecture validated on real-world data with measurable performance gains and in pointing toward future extensions with deep learning models and applications at an industrial scale.

1 INTRODUCTION

Digital content keeps expanding across the platforms that now shape most aspects of daily life, and this growth shows no sign of slowing down. People move through unending streams of mixed data-text, audio, images, and video-ranging from entertainment catalogs to news feeds and educational resources. Faced with such abundance, few attempt to search for everything on their own. Most depend on systems that can quickly narrow the field, pick out what matters, and return results with minimal delay.

This reliance has made recommender systems a core part of how people interact with digital media. These systems filter and rank information under both statistical and operational limits. They must learn from implicit feedback that is often incomplete or noisy, balance lasting preferences with short-term intent, and adapt to entirely new users and items. They also have to run efficiently at scale, sometimes in real time, while keeping privacy, fairness, and business goals in balance. When they succeed, they ease the user's mental load, speed up discovery, and strengthen engagement. Personalisation no longer complements the interface – it defines it.

Beyond their technical dimension, recommender systems now carry economic and social weight. By presenting the right content at the right moment, they help platforms maintain attention, build loyalty, and improve revenue.

Industry reports consistently show that a large share of user activity on major media platforms is driven by recommender outputs, often exceeding 70% on services such as Netflix, Spotify, and YouTube. In retail settings the effect is equally

concrete. Amazon and comparable marketplaces use ranking and re-ranking pipelines to raise purchase likelihood, grow basket size, and stabilize retention. Personalisation now functions as an economic lever that shapes discovery, inventory exposure, and even supplier dynamics across sectors.

Despite this maturity, core technical challenges persist. Collaborative filtering, including matrix factorisation and neighbourhood models, performs well once interaction histories are dense and stable. It struggles under sparse implicit feedback, cold-start users and items, non-stationary preference drift, and long-tail catalogs. Training-time shortcuts often help little at serving time, where latency budgets, cache locality, and A/B guardrails impose hard limits on model complexity and candidate set size.

Graph-based models explore another side of the recommendation problem. They map users, items, and contextual features into a heterogeneous network and, in doing so, capture higher-order proximity and multi-hop relations that collaborative filtering usually misses. Common methods include Personalized PageRank, random walks with restart, and message passing over knowledge or session graphs. These approaches enable path-aware reasoning and can provide interpretable evidence through subgraphs, which matters in regulated or high-stakes environments.

The advantages are clear, but they come with practical costs. Building features and sampling negatives take careful tuning, memory use rises quickly as graphs grow, and distributed training often slows down because of communication limits. In production, the model must also keep its graph updated as new edges appear, nodes vanish, or user behaviour shifts over time.

In practice, strong pipelines mix the two families. Collaborative filtering often handles candidate generation, while graph-based ranking sharpens local context and improves coverage of rarely seen items. Together, they align model capacity with system limits and highlight where progress is still needed – especially fresher data flows, faithful interpretability, and speed under tight latency budgets.

This study investigates the theoretical underpinnings of personalised recommendation and surveys contemporary modeling strategies in the field. We design and implement a hybrid recommender that integrates collaborative filtering with graph-based algorithms, and we evaluate the approach on real-world interaction data to test both accuracy and robustness.

The research pursues several objectives. First, it analyses how recommendation systems are classified and how catalogs are organised, with attention to task formulations and deployment settings. Second, it examines collaborative filtering and graph-based methods in depth, including Personalised PageRank and random-walk formulations that capture higher-order proximity. Third, it specifies a modular system architecture built with Java and Spring Boot, with PostgreSQL for persistent storage and Redis for fast access paths. Fourth, it develops a cascade pipeline that separates candidate generation from context-aware re-ranking and allows end-to-end optimisation under serving constraints. Finally, it conducts an experimental evaluation using established metrics, namely Precision, Recall, and the Mean Absolute Error, abbreviated MAE.

The contribution is twofold. Methodologically, the work advances scalable and accurate recommendation techniques that transfer across heterogeneous platforms. Practically, the hybridisation of collaborative and graph-based signals yields a robust path toward production-ready systems that deliver high-quality personalisation while respecting latency, resource, and maintainability constraints.

2 RELATED WORK

Collaborative filtering has long been a core technique in recommender systems, setting reliable baselines across media, retail, and social platforms. Yet classical forms still hit the same old limits: sparse user–item matrices, cold-start users and items, and the heavy computation that comes with large catalogs [1], [2], [3]. More recent work moves past these issues through representation-learning extensions such as neural CF and deep matrix factorization. These methods refine latent embeddings and often give better results in both top-k ranking and rating prediction [2]. The gains are real, yet they continue to depend on sufficiently dense interaction signals, careful regularisation, and access to side information when available.

Graph-based recommenders emerged as an alternative that represents users, items, and contextual entities as nodes connected by typed edges, enabling multi-hop reasoning and improved explainability. Personalised diffusion methods, notably Personalised PageRank and random walk with restart, diffuse a user-specific preference distribution over the interaction graph and retrieve items that are near the source in a stationary or expected-hitting-time sense [4], [5]. They capture higher-order and cross-type relations in heterogeneous or multiplex graphs and

often improve recall on the long tail. However, these gains come with larger compute and memory budgets.

Representation learning on graphs pushed the area forward. Node2Vec derives context-preserving embeddings through biased second-order walks. At the same time, LightGCN removes heavy feature transforms and retains only neighbourhood aggregation, which preserves higher-order connectivity yet remains tractable at production scale [6], [7], [8]. Graph neural networks (GNNs) unify neighborhood aggregation with collaborative learning, and recent surveys have emphasized their growing role in recommendation tasks [7], [8].

Hybrid approaches that combine collaborative filtering (CF) with graph-based ranking have gained wide attention over the past few years. In practical systems, CF usually handles the first step, producing a list of candidate items for each user. The list is then re-ordered through graph propagation or neural re-ranking that takes into account how users and items are linked. This combination helps reduce cold-start effects and brings rare, long-tail items into view while keeping the process fast enough for real-time use. Optimization approaches, including genetic algorithms applied in network design, have also been explored to improve system efficiency and performance [9].

Recent hybrid graph-based models report improvements in standard evaluation metrics, including precision, recall, and mean absolute error [10], [11], [12]. Collaborative filtering is still appreciated for its simplicity and interpretability, though it struggles with sparse data and new users or items. Graph-based methods can model multi-hop and structural relations, but need more computation. Using both together in a hybrid cascade has proven effective for reaching accuracy and scalability in personalized
recommendation [1]-[12].

3 PROPOSED APPROACH

The approach uses a two-stage design that pairs collaborative filtering with graph-based ranking to unite fast retrieval with structure-aware re-scoring. Collaborative filtering mines interaction logs for latent neighborhoods and returns a compact candidate slate with strong recall at serving time. A subsequent diffusion step, instantiated as Personalized PageRank, operates on the user–item graph and related context to adjust scores by structural proximity. In combination, the stages widen coverage of infrequent items, dampen cold-start sensitivity, and stay within typical latency budgets.

3.1 Collaborative Filtering Layer

In the first stage, collaborative filtering generates a target user's initial set of candidate items. We adopt a user-item interaction matrix $R \in Rm \times n$, where m is the number of users and n is the number of items. Each entry r_{ui} represents the rating or implicit interaction between user u and item i.

User-based CF predicts the preference score $\widehat{r_{ui}}$ as:

$$\widehat{r_{ui}} = \bar{r_u} + \frac{\sum_{v \in N(u)} w(u,v) \cdot (r_{vi} - \bar{r_v})}{\sum_{v \in N(u)} |w(u,v)|}, \quad (1)$$

where $N(u)$ denotes the neighborhood of similar users, $\bar{r_u}$ is the average rating of user u, and $w(u, v)$ is the similarity weight between users u and v, often computed via cosine similarity or Pearson correlation.

This layer yields a ranked candidate list based on collaborative patterns but remains limited when the dataset is sparse or when a user/item suffers from the cold-start problem.

3.2 Graph Construction

To enrich the model with structural information, we construct a user-item bipartite graph $G = (V, E)$, where $V = U \cup I$ consists of users U and items I, and edges E record observed interactions. The adjacency matrix A shows binary or weighted links between these entities. This structure makes it possible to follow multi-hop connections and uncover indirect relations that collaborative filtering on its own cannot reach.

3.3 Personalized PageRank Diffusion

On top of the candidate list from CF, we apply Personalized PageRank (PPR) to refine recommendations. Given a starting distribution vector p_0 that emphasizes the target user's known interactions, PPR iteratively computes a stationary probability distribution:

$$p = \alpha p_0 + (1 - \alpha) W^T p, \quad (2)$$

where W is the normalized adjacency matrix of the graph, and α is the restart probability (typically 0.15). This mechanism balances local preference reinforcement (via restart to p_0) with global exploration of the graph structure.

PPR effectively highlights items that are structurally proximate to the user's history, even if

they lack direct interactions. Compared with pure CF, this allows surfacing of long-tail items and mitigates sparsity issues.

3.4 Hybrid Cascade Pipeline

The overall pipeline operates in two main phases. In the first phase, candidate generation is performed with collaborative filtering, which identifies a top-k set of items based on similarity patterns. This ensures that highly correlated items are included and not overlooked. In the second phase, re-ranking is carried out with Personalized PageRank, which propagates signals across the user–item graph and adjusts candidate rankings so that items with stronger graph connectivity to the user's neighborhood receive higher scores.

The final recommendation list is obtained by combining the CF prediction score $\widehat{r_{ui}}$ and the PPR stationary probability p_i. A weighted linear fusion is applied:

$$s_{ui} = \beta \cdot \widehat{r_{ui}} + (1 - \beta) \cdot p_i, \quad (3)$$

where $\beta \in [0,1]$ controls the trade-off between collaborative and graph-based contributions.

3.5 Conceptual Architecture

The system is organized as a modular architecture. At the first stage, it relies on the data layer where user-item interactions are stored in PostgreSQL and cached in Redis to ensure fast retrieval. The second stage is the collaborative filtering module, which computes similarity values and generates the initial scores. The third stage is the graph module, which constructs a bipartite user-item graph and applies the Personalized PageRank algorithm. The final stage, a hybrid fusion block, merges the scores from both modules into a single list that is presented to the user.

The workflow can be detailed sequentially. The first is that the system gathers raw interaction data from consumers. The second is that collaborative filtering produces a candidate set of recommendations. The third is that the graph-based re-ranking process fine-tunes the candidates by spreading the signals throughout the graph. The fourth is that the hybrid scoring module combines the outcome into a final-ranked list. In this way, the pipeline transforms raw data into a personalized recommendation output that is both accurate and scalable.

4 IMPLEMENTATION AND EXPERIMENTATION

The prototype is built in Java 17 with Spring Boot and follows a layered controller–service–repository organization, instrumented with Micrometer for timing and operational metrics. A dedicated scheduled job, `CascadeRecommendationJob`, orchestrates the end-to-end pipeline; it is annotated with a one-minute cron trigger and processes users in cursored batches to avoid load spikes and to resume seamlessly between runs. The job keeps an idempotency window, advances a Redis-backed cursor, and records latency via timers, which enables safe re-execution and precise monitoring under production traffic.

During execution, the job builds a cascade out of pluggable modules. A list of `CandidateGenerators` produces raw candidates first. Then, `RecommendationFilters` will rule out candidate items on the list. Next, the elements are scored by the `Rankers`. After that, the `Diversifiers` are responsible for honoring presentation rules, e.g., capping repetition and encouraging diversity. Last but not least, the `TopKSelector` finishes the pipeline by creating the final feed. This design facilitates modular and composable code by separating algorithmic responsibility clearly and allowing for replacing particular stages with alternate implementations while keeping the process stable overall.

Persistence combines PostgreSQL for durable state with Redis for low-latency access. The system maintains a Redis store for "recently viewed" posts (service `RedisViewedPostStore`), which is read on every request to exclude already-seen content and to personalize subsequent ranking. Separately, it tracks post popularity (table `PostPopularity`) for popularity-aware signals. The recommendation result itself is cached in Redis together with metadata such as the generation timestamp, the active algorithm, and the per-source distribution; a freshness flag marks whether the snapshot is still valid for serving.

After filtering, the pipeline applies presentation-level diversification. It restricts the posts per author, inserts variability across topics with tags, and inserts advertisements in prechosen locations of the list. Last, the system takes the top fifty entries and outputs them to the cache along with their related metadata. The above solution maintains the feed as a personal variable with stable latency at retrieval time.

The service provides a lean REST interface that allows clients to consume this. A GET request to

`/api/v1/recommendations` returns the list at hand along with diagnostic fields (active algorithm, time to generate, candidate count, and source mix) to support observability and quality monitoring. User interactions are captured explicitly: the client posts to `/{postId}/interaction?type=LIKE` (or `type=VIEW`) to log feedback that will be folded into the next refresh cycle, gradually tightening personalization.

For offline evaluation, the project constructs the standard artefacts required by the hybrid approach. The first artefact is a user–item matrix that supports collaborative filtering. The second artefact is a bipartite interaction graph that enables graph-based diffusion. Candidate generation and re-ranking are then assessed using common information-retrieval metrics. Precision@K estimates the share of relevant items in the top-K. Recall@K measures the coverage of relevant items retrieved. Mean Absolute Error (MAE) reports the absolute error of predicted ratings against the ground truth of user interactions. This protocol is applied to a social-media analytics dataset that represents real-world consumption patterns.

To serve production-like volumes, the job processes users in fixed-size pages (1 000 IDs per batch) using a Redis cursor to continue from the last processed identifier. This rolling window lets the system refresh recommendations continuously without full re-scans and provides predictable background load characteristics.

The implementation has been described together with its current performance limits and scaling options. Currently, the system sustains operation for roughly one hundred thousand users without visible degradation. Reaching higher loads will probably require partial or incremental graph updates and, eventually, a distributed deployment. Contextual factors of personalisation – such as time of day or device class—are not yet represented in the model. Planned extensions include incorporating richer content features into the ranking layer, enabling streaming or incremental graph updates through Kafka, and improving Redis control via more precise TTL configurations. Future evaluation will move beyond precision and recall to include engagement-oriented metrics, for example, click-through rate, retention, and audience coverage, supported by an internal monitoring dashboard that tracks data sources, graph consistency, and cache utilization.

Overall, the implementation translates the hybrid cascade into a production-shaped service. It includes a minute-granularity scheduler, modular pipeline stages, fast caches for both inputs and outputs, transparent REST endpoints, and an evaluation loop grounded in Precision@K, Recall@K, and MAE. This architecture provides measurable quality gains while keeping latency low and operations observable. It also leaves clear levers for scaling and for richer modeling in subsequent iterations.

5 RESULTS

The performance of the hybrid cascade model was tested against two reference methods, Collaborative Filtering (CF) and graph-based Personalised PageRank (PPR). All experiments used the Social Media Analytics dataset. Precision, Recall, and MAE were used as evaluation metrics, which showed how well the model balanced recommendation accuracy with the stability of predicted ratings.

5.1 Quantitative Evaluation

In addition to the numerical results, Figure 1 shows clear differences in Precision, Recall, and MAE among the three methods. The hybrid model achieves consistently better performance, with measurable gains in each metric.

Table 1 presents performance comparison of recommendation methods.

Table 1: Performance comparison of recommendation methods.

Method	Precision	Recall	MAE
Collaborative Filtering (CF)	0,65	0,52	0,2
Graph-based (PPR)	0,68	0,58	0,15
Proposed Hybrid Cascade	0,75	0,68	0,1

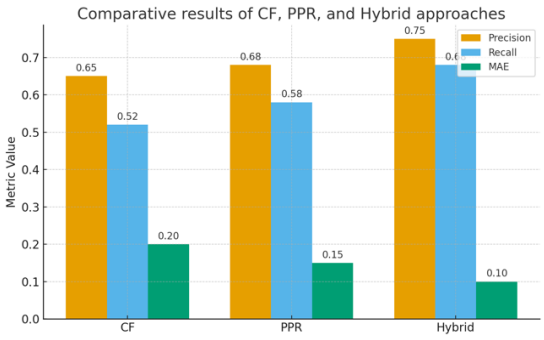

Figure 1: Comparative results of CF, PPR, and Hybrid approaches.

5.2 Discussion

The results demonstrate that the hybrid cascade model significantly outperforms individual methods. With a precision of 0.75, three out of four recommendations were relevant, compared to 0.65 for CF and 0.68 for PPR. The recall of 0.68 indicates the hybrid system retrieved more appropriate items, surpassing the coverage of baseline approaches. Additionally, an MAE of 0.10 shows very low prediction error, much lower than the usual ≥ 0.20 seen in collaborative filtering.

These findings support the approach of combining collaborative filtering with graph-based diffusion. While CF effectively captures local interaction patterns, it suffers from data sparsity. Graph-based PPR addresses this by utilizing multi-hop structural relationships. The cascade integration takes advantage of both strengths, leading to higher accuracy, wider coverage, and more robust recommendations.

6 CONCLUSIONS

This work introduced the design and analysis of a hybrid cascade recommenders that combine graph-based diffusion with collaborative filtering through Personalised PageRank. The framework effectively remedied serious drawbacks of classical CF, such as sparsity and cold-start problems, by taking advantage of structure information embedded in user–item graphs. Experimental analysis of the Social Media Analytics data set substantiated the success of the combined framework. The combining framework had higher precision and recall than CF and graph-based rivals, while at the same time keeping mean absolute error relatively small. These findings verify the potential benefit of combining similarity-guided and structure-conscious techniques to construct more robust recommenders.

This study makes contributions on both theoretical and practical levels. From a ranking that will, it demonstrates that a simple two-stage cascade can achieve state-of-the-art accuracy without the complexity of deep neural models. From a systems perspective, the architecture implemented in Java, Spring Boot, PostgreSQL, and Redis shows that research ideas can be translated into a production-ready service with low latency and high scalability. The hybrid pipeline could handle real-time recommendation refresh cycles, integrate user feedback dynamically, and support personalization with transparent monitoring.

At the same time, the research identified several directions for further work. Future efforts may focus on extending the cascade with graph neural networks such as LightGCN or NGCF in order to capture higher-order semantic relations. Another promising direction is domain adaptation and cross-platform recommendation, where user preferences can be transferred between different services. Practical implementation also necessitates incremental graph updates and streaming architecture with the aid of tools such as Kafka to achieve scalability at the industrial level. Last but not least, integration with higher-level contextual features, with the inclusion of temporal dynamics or device-specific interactions, would make the system more adaptive and user-oriented.

Finally, the hybrid cascade framework is an important advance in personalised recommendations. By uniting the efficiency and interpretability of collaborative filtering with graph-based methods' structural expressiveness, the resulting recommendations are enhanced in terms of accuracy, coverage, and robustness. By demonstrating both academic relevance and practical feasibility, this study contributes to the ongoing development of recommendation technologies and opens a path toward their application in large-scale real-world environments.

ACKNOWLEDGMENTS

We acknowledge support by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) and the Open Access Publishing Fund of Anhalt University of Applied Sciences.

REFERENCES

[1] H. Dong, S. Wen, L. Lv, H. Pei, L. Zhou, and B. Zhang, “A collaborative filtering recommender systems: Survey,” Neurocomputing, vol. 584, pp. 128718, 2025 (online 2024). doi: 10.1016/j.neucom.2024.128718.

[2] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep learning based recommender system: A survey and new perspectives,” ACM Computing Surveys, vol. 52, no. 1, article 5, pp. 1–38, 2019. doi: 10.1145/3285029.

[3] L. V. Bodnar, K. S. Shulakova, L. E. Hrizun. Visnyk NTU "KhPI". Series: System Analysis and Information Technologies, no. 2, pp. 110-115, 2021.doi: 10.20998/2079-0023.2021.02.16.

[4] X. He, H. Jin, B. Shi, X. Xie, W. Zhang, and J. Han, “Personalized diffusions for top-N recommendation,” in Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD’19), pp. 1520–1530, 2019. doi: 10.1145/3298689.3346985.

[5] A. Valdeolivas, L. Tichit, C. Navarro, S. Perrin, G. Odelin, N. Levy, and A. Baudot, "Random walk with restart on multiplex and heterogeneous biological networks," Bioinformatics, vol. 35, no. 3, pp. 497–505, 2019. doi: 10.1093/bioinformatics/bty637.

[6] X. He, K. Deng, X. Wang, F. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in Proc. 43rd Int. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR'20), pp. 639–648, 2020. doi: 10.1145/3397271.3401063.

[7] S. Wu, C. Tang, P. Cui, and M. Wang, "Graph neural networks in recommender systems: A survey," ACM Transactions on Recommender Systems, vol. 1, no. 1, article 1, pp. 1–34, 2022. doi: 10.1145/3535101.

[8] C. Gao, Y. Zheng, N. Li, Y. He, Y. Li, and Y. Yang, "A survey of graph neural networks for recommender systems: Challenges, methods, and directions," ACM Transactions on Recommender Systems, vol. 1, no. 1, article 3, pp. 1–38, 2023. doi: 10.1145/3568022.

[9] R. Tsarov, L. Nikityk, I. Tymchenko, V. Kumysh, K. Shulakova, S. Siden, and L. Bodnar, "Using a genetic algorithm for telemedicine network optimal topology synthesis," in Proceedings of the International Conference on Applied Innovation in IT, vol. 12, no. 1, pp. 19–24, 2024, doi: 10.25673/115637.

[10] Z. Z. Darban and M. H. Valipour, "GHRS: Graph-based hybrid recommendation system with application to movie recommendation," Expert Systems with Applications, vol. 198, article 116850, 2022. doi: 10.1016/j.eswa.2022.116850.

[11] M. Jia, F. Liu, X. Li, and X. Zhuang, "Hybrid graph neural network recommendation based on multi-behavior interaction and time sequence awareness," Electronics, vol. 12, no. 5, article 1223, 2023. doi: 10.3390/electronics12051223.

[12] A. Sami, W. El Adrousy, S. Sarhan, A. S. Khalil, and S. M. Abdelmaksoud, "A deep learning based hybrid recommendation model for Internet users," Scientific Reports, vol. 14, article 29390, 2024. doi: 10.1038/s41598-024-79011-z.