Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 2, 2026

OPTIMIZING RECOMMENDATION SYSTEMS USING GRAPH-BASED METHODS AND MACHINE LEARNING ALGORITHMS

[1]Huzaifah Bin Khawar, [2]Talha Nadeem, [3]Muhammad Zarar, [4] Hood Khizer

[1] AI Researcher, Research and Development Department, Trouve Labs, AHOY DMCC and The Superior University Lahore, Pakistan huzaifahkhawar.ieee@gmail.com

[2] Systems Architect AI/ML, Research and Development Department, Trouve Labs, AHOY DMCC and The Superior University Lahore, Pakistan talhanadeem.ieee@gmail.com

[3] AI Researcher, Research and Development Department, Trouve Labs, AHOY DMCC and The Superior University Lahore, Pakistan muhammadzarar.ieee@gmail.com

[4] CEO Trouve Labs, CTO AHOY DMCC, Research and Development Department, Trouve Labs, AHOY DMCC, Dubai hoodkhizer@gmail.com

Keywords
Recommendation systems, graph-based methods, graph neural networks, machine learning, reinforcement learning, top-K ranking, data sparsity, cold start.

Article History
Received: 02 Feb, 2026
Accepted: 11 Feb, 2026
Published: 25 Feb, 2026

Copyright @Author

Corresponding Author: * Talha Nadeem*

Abstract

Recommendation systems must rank relevant items from sparse, dynamic and policy-shaped interaction data. This paper presents an evidence-bounded framework that combines a bipartite user-item graph, a predictive machine-learning score and an adaptive policy value within one top-K ranking interface. Graph propagation captures higher-order connectivity; supervised or deep models incorporate interaction and side information; and reinforcement learning represents delayed utility when reliable feedback is available. The revision formalizes graph construction, message passing, score fusion, policy updating, pairwise ranking loss and top-K evaluation through ten editable equations, a notation table and an IEEE/ACM-style algorithm. Four requested figures and the wider visual set are rebuilt without watermarks. Because the source manuscript provides no raw run-level logs or reproducible benchmark outputs, the paper makes no new empirical-superiority claim. Instead, it specifies a leakage-aware evaluation protocol, computational complexity, deployment controls, and ethical requirements for privacy, fairness, transparency and human oversight.

1. INTRODUCTION

A system of recommendations helps to eliminate information overload because it predicts what content is most relevant to a user. Collaborative filtering makes use of interaction patterns, content-based methods of utilizing the properties of the user or item, and hybrid methods are a mix of the two. The item based collaborative filtering and matrix factorisation are still useful bases, but with sparse interactions, new items, and evolving user

preferences, solely relying on past co-occurrence is not enough to provide useful recommendations [1] [3].

Graph-based recommendation applies observed interactions/relationships as a relationship/edge between a user node and an item node. Multi-hop evidence and structured side information can be then aggregated using message passing or link prediction [5]-[9]. The linear prediction can be extended to nonlinear prediction, provided by machine-learning models; and delayed outcomes can be considered for adaptive policies, when recommendations affect future observed patterns [10]-[15].

Things that are useful for a hybrid design are actually only when the components are assessed on similar split, candidate set, label budget and serving purpose. No claim with regard to an improvement in the accuracy or user satisfaction for combining modules.

This revision thus focuses on the framework as a specification for reproducing/reducing a design and evaluating it, instead of as a report on the unobserved experiments.

1.1 Background and Motivation

In industrial recommending, typically, the job of retrieval is split up from the task of the ranking. Rich graph, contextual, and policy features can be used for ranking while high recall at low latency is required for retrieval. In the web scale, neighbourhood sampling and graph convolution are previously applied to maintain the efficiency in generating embedding for recommendations [7] and simplified propagation is proposed to decrease unnecessary transformation in graph collaborative filtering [9].

1.2 Problem Statement

The following four assumptions drive the design of this framework:the interaction matrix is sparse, new users and new items have short memory, graph and slate size will limit the latency and memory, and logged feedback will be biased according to the selection policy used for prior recommendations. These should not be glossed over with a single number for accuracy and should be discussed in more detail.

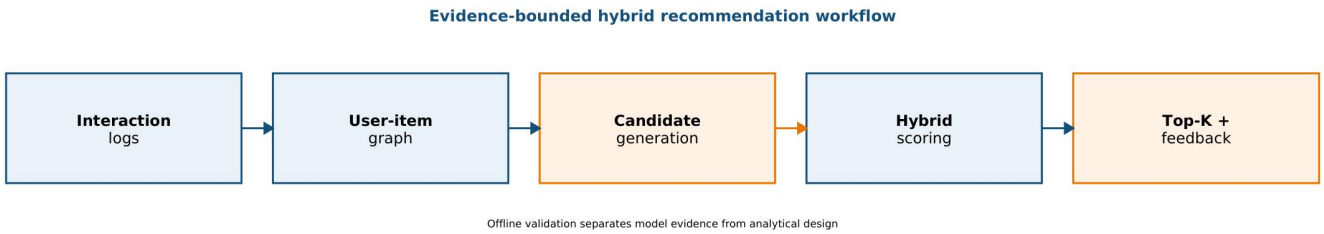


Figure 1. End-to-end workflow for graph-based candidate generation, hybrid scoring and feedback.

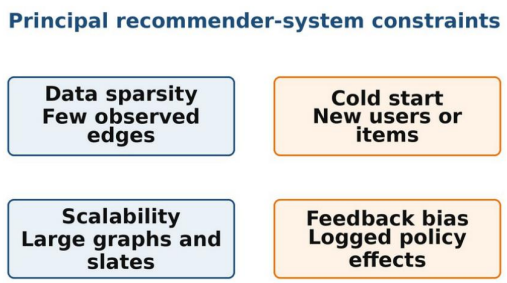


Figure 2. Principal technical constraints in recommendation-system optimisation.

1.3 Objectives of the Study

The research has five objectives as stated.

1.3.1 Formalise graph-based recommendation

Model a bipartite graph of interaction between users and items, and define what domain knowledge embeddings for users/items are and how they are captured by neighbourhood evidence.

1.3.2 Integrate predictive and adaptive signals

Fuse together relevance of the graph, an ML predictor and a policy value using transparent fusion weights.

1.3.3 Address sparsity and cold start

Discuss the use of side features, IGE, and CCG without assuming that all cold-start configurations are addressed.

1.3.4 Define reproducible evaluation

List splits with leakage, set of comparables, candidate set reporting and top-K metrics.

1.3.5 Document operational and ethical controls

Report latency, memory requirement, drift requirement, privacy requirement, fairness, explainability, and including human-oversight requirement.

2. LITERATURE REVIEW

2.1 Traditional and Latent-Factor Methods

A neighbourhood collaborative filtering derives preference information from a set of similar users or items [1]. Matrix factorisation is a powerful baseline approach to modelling users and items in a common latent factor space [3] and is a robust and scalable approach. Bayesian personalised ranking targetting pairwise-order instead of missing-key as negative feedback [4] These methods are attractive in terms of computation, however, these can only represent low order graph structure, and also side information is limited.

2.2 Graph-Based Recommendation

Rating prediction via graph convolutional matrix completion: The problem of rating prediction is regarded as a bipartite graph link prediction task [5]. Inductive neighbourhood aggregation is an approach that enables unseen nodes to be supported when meaningful node features exist [6]. The method PinSage uses random-walk sampling and graph convolution for recommendation in web-scale scenarios [7]; the method NGCF can share the collaborative signals via user-item graphs [8], [9].

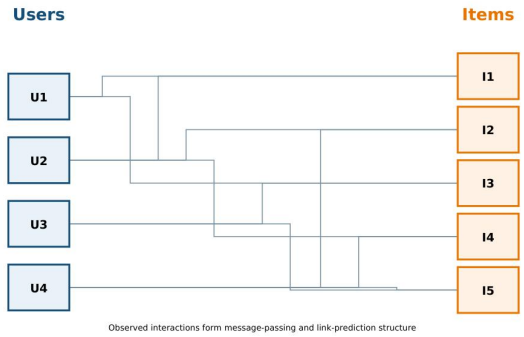


Figure 3. Rebuilt bipartite user-item relationship network without watermark artefacts.

The graph view reveals interactions that are not in the flat interaction matrix, but also has over-smoothing, edge propagation noise, and popularity amplification. This is because it is important to validate the graph depth, sampling strategy, and edge construction rather than simply picking them off the training loss.

2.3 Machine Learning and Deep Ranking

For instance, neural collaborative filtering has been developed for learning nonlinear interaction functions, DeepFM uses low and high order feature interactions, and two-stage deep recommenders split the problem into retrieval and ranking stages. Sequential self-attention models are able to model order dependent preferences [13]. They only have positive side-effects on volume of data, regularisation, negative sampling and deployment latency.

2.4 Reinforcement Learning

In fact, recommendation is a problem of sequential decision making, which can be modelled with reinforcement learning. There are value-based methods, that estimate long-term returns [14] and there are policy gradient systems which can optimize a slate directly. When training or evaluating on a set of historical data, logged recommendations bring exposure and selection bias to the problem, thus necessitating off-policy correction [15].

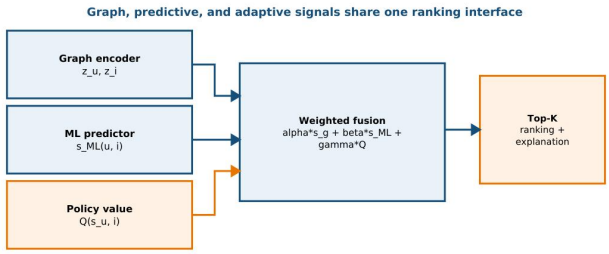


Figure 4. Hybrid architecture combining graph, predictive and adaptive recommendation signals.

3. PROPOSED HYBRID FRAMEWORK

3.1 Data and Graph Construction

An interaction log D is cleaned prior to the construction of features. Duplicate events, invalid identifiers and post-outcome features removed. Positive (or weighted) interactions are then used to create a user-item graph G =(U union I, E). The feature matrix X could consist of item contents, and user context can be stored as edge attributes, while type of events and confidence can be retained as edge attributes along with time.

3.2 Candidate Generation and Ranking

Candidate generation: An appropriate set C_u is obtained from the neighbourhoods of a graph, from the nearest neighbour search (approximate), or from a business rule. The ranking stage calculates a structural score, a ML relevance score and a relevant policy value (if appropriate). To avoid detracting from a ranking that may be too high because of failure to retrieve the candidate, the recall is reported separately.

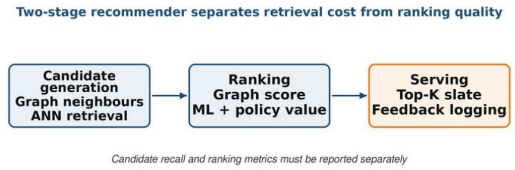


Figure 5. Two-stage candidate-generation, ranking and serving workflow.

3.3 Graph Representation Learning

Every graph layer is a pooling of messages from a node neighbourhood and mixing with its own representation. The final embeddings are for either dot-product or learned decoding. The sampling, normalization and the combination of layers must be assigned in the experiment protocol as that is a very significant factor that affects the accuracy and the costs of the experiment [6]-[9].

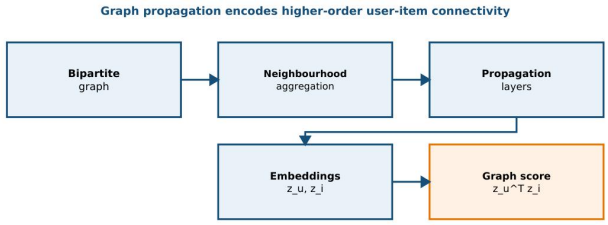


Figure 6. Graph message-passing process for user and item embedding generation.

4. MACHINE LEARNING AND ADAPTIVE RECOMMENDATION

4.1 Machine-Learning Model Families

The recommendation problem is addressed through different pieces, such as predictive, graph and sequential models. There are also factors such as latent factor and neural predictors that have been adopted to estimate relevance, relational propagation with graph models, sequential models that account for recency and order, and reward optimization for a defined reward using an adaptive policy. The selection of the model should be based on the amount of data available, the latency with which it is required, and not the novelty of the model alone as the user task.

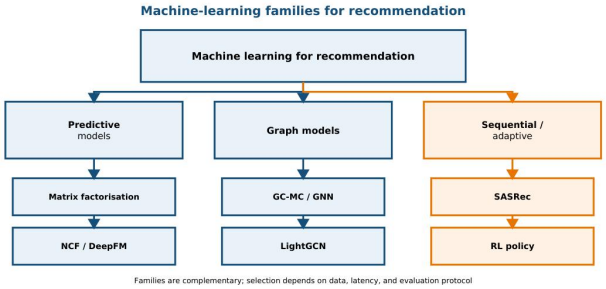


Figure 7. Rebuilt taxonomy of machine-learning methods used in recommendation systems.

4.2 Reinforcement-Learning Interaction Loop

An adaptive recommender sees a state, chooses a slate, gets back some feedback which it updates in value or policy model. The reward needs to be specified in advance and not be a black box "engagement score" that aggregates many stakeholder interests. Achieving short term clicks with long term satisfaction and safety can be conflicting business needs.

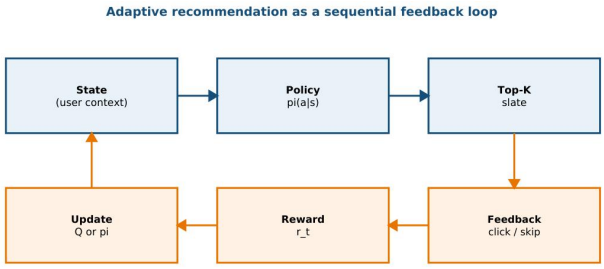


Figure 8. Rebuilt reinforcement-learning loop for adaptive recommendation.

Different methods, such as value-based, policy-gradient, actor-critic and contextual-bandit, are based on different assumptions regarding the different parts of the task (action space, horizon and counterfactual feedback). The list of principal families is summarized in figure 9, but without ranking these to compare them in general as being better.

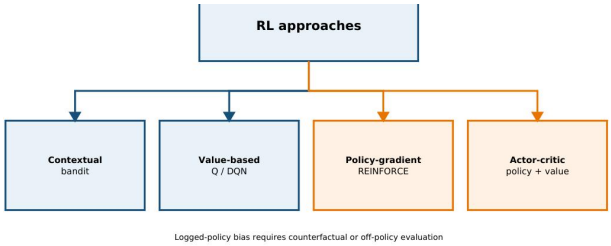


Figure 9. Taxonomy of adaptive recommendation approaches.

4.3 Online Service Architecture

Production deployment breaks down the ingestion from the events, stores the features, updates the graph, trains offline, retrieves candidates, ranks them, serves and monitors. The separation lets the graph encoder remain stable while it is being updated with the predictive model and the adaptive policy.

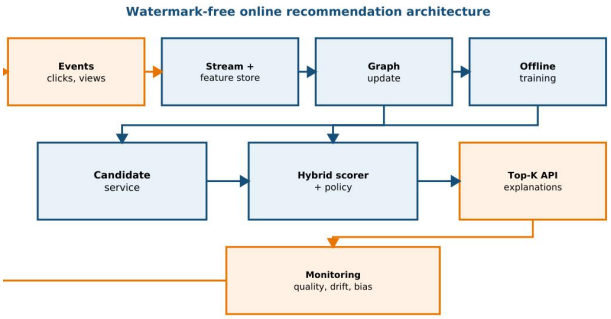


Figure 10. Rebuilt watermark-free service architecture for online recommendation and monitoring.

5. MATHEMATICAL FORMULATION AND IEEE/ACM-STYLE ALGORITHM

The framework is formalized using other components such as a bipartite graph, graph propagation, score fusion, adaptive updating, pairwise ranking and top-K evaluation. All the symbols used in the equations 1-10 are defined in table 1.

Table 1. Mathematical notation for the hybrid graph-ML-RL framework.

Symbol	Meaning	Dimension or range
G=(V,E)	User-item interaction graph	V=U union I
U, I	User and candidate-item sets	Finite entity sets
R	Observed interaction matrix	R^(|U| x |I|)
N(v)	Neighbourhood of node v	Subset of V
h_v^(l)	Node representation at layer l	R^d
z_u, z_i	Final user and item embeddings	R^d
s_g, s_ML	Graph and predictive relevance scores	Normalized real values
Q(s,a)	Long-term action value	Real scalar
alpha, beta, gamma	Fusion weights	Non-negative; sum to 1
eta, delta	Learning rate and discount factor	(0,1] and [0,1]
C_u, K	Candidate set and returned-list size	C_u subset of I; K positive
Theta, lambda	Trainable parameters and regularization	Model dependent; lambda >= 0
Rel_u, Rec_u^K	Relevant and returned item sets	Subsets of I

5.1 Graph and Ranking Relations

$$G = (V, E), \quad V = U \cup I \tag{1}$$

5.2 Policy, Loss and Evaluation Relations

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) \tag{7}$$

Symbol	Meaning	Dimension or range
$E \subseteq U \times I$		
$R = [r_{ui}] \in \mathbb{R}^{\lvert U\rvert \times \lvert I\rvert}$ $r_{ui} \in \{0,1\}$		(2)
$\tilde{h}_v^{(\ell)} = \mathrm{AGG}\left(\{h_n^{(\ell)} : n \in N(v)\}\right)$		(3)
$h_v^{(\ell+1)} = \sigma\left(W_\ell \tilde{h}_v^{(\ell)} + B_\ell h_v^{(\ell)}\right)$		(4)
$s_g(u,i) = z_u^{\mathrm{T}} z_i$		(5)
$s_h(u,i) = \alpha s_g + \beta s_{\mathrm{ML}}$ $+\gamma Q(s_u, i), \quad \alpha + \beta + \gamma = 1$		(6)
$+\eta r_t + \eta\delta \max_a Q(s_{t+1}, a)$ $-\eta Q(s_t, a_t)$		
$L_{\mathrm{BPR}} = -\sum_{(u,i,j)} \ell_{uij} + \lambda \Theta^{\mathrm{T}} \Theta$ $\ell_{uij} = \log\sigma\left(s_{ui} - s_{uj}\right)$		(8)
$\mathrm{Prec@}K = \frac{\lvert \mathrm{Rel}_u \cap \mathrm{Rec}_u^K \rvert}{K}$ $\mathrm{Rec@}K = \frac{\lvert \mathrm{Rel}_u \cap \mathrm{Rec}_u^K \rvert}{\lvert \mathrm{Rel}_u \rvert}$		(9)
$\mathrm{NDCG@}K = \frac{\mathrm{DCG@}K}{\mathrm{IDCG@}K}$ $\mathrm{DCG@}K = \sum_{r=1}^{K} \frac{2^{\mathrm{rel}_r} - 1}{\log_2(r+1)}$		(10)

Equations (1)-(2) define the bipartite graph and observed interaction matrix. Equation (3) defines neighbourhood aggregation, and Equation (4) updates each node representation. Equation (5) gives the graph relevance score. Equation (6) fuses structural, predictive and long-term utility signals; its weights must be selected on validation data and reported.

Equation (7) is a temporal-difference update, while Equation (8) is a pairwise ranking objective for implicit feedback. Equations (9)-(10) evaluate top-K relevance and ordering. Zero-relevant-item cases, negative sampling and the candidate universe must be defined explicitly.

Table 2. Dominant computational terms of the reference workflow.

Module	Principal operation	Time complexity	Memory
Preprocessing	Clean D and construct G	O(|D|)	O(|V|+|E|)
Graph encoder	L message-passing layers	O(L|E|d)	O(|V|d)
Candidate scoring	Graph, ML and policy fusion	O(Cd)	O(C)
Top-K ranking	Maintain a heap	O(C log K)	O(K)
Policy update	Mini-batch update	O(Bd)	O(Bd)

The above are Asymptotic Reference terms, not hardware timings. The actual cost varies with the above factors, sampling, sparsity, caching, numerical precision, and accelerator utilization and serving architecture.

Algorithm 1. Hybrid Graph-ML-RL Recommendation Optimisation

Require: interaction log D; features X; top-K K; fusion weights alpha, beta, gamma; learning rate eta; discount delta

Ensure: ranked list Rec_u^K for each target user and an updated adaptive policy

1 Clean D, remove duplicates, encode attributes, and record the data split before feature construction.

2 Construct G=(U union I, E), R, and the normalized adjacency representation.

3 Initialize node features and train L graph-propagation layers using Equations (3)-(4).

4 Obtain z_u and z_i; generate candidate set C_u for each target user u.

5 For each candidate i in C_u, compute s_g(u,i), s_ML(u,i), and Q(s_u,i).

6 Fuse the three signals using Equation (6), applying validation-derived weights only.

7 Rank C_u by s_h(u,i) and retain the first K items as Rec_u^K.

8 Log the displayed slate, propensity or serving policy, user response, and reward r_t.

9 Update the policy using Equation (7) or a validated off-policy alternative.

10 Optimize representation parameters with Equation (8) on training interactions.

11 Evaluate candidate recall, Precision@K, Recall@K, and NDCG@K on held-out users or time.

12 Return Rec_u^K, model artifacts, evaluation logs, and monitoring thresholds.

6. EVALUATION AND VALIDATION PROTOCOL

6.1 Data, Splits and Baselines

There are existing publicly available benchmarks in movies for using a controlled studies from MovieLens [16]. The specific data set version, filtering rules, feedback definition and split for a final experiment should be indicated. For dynamic recommendation, randomly split interaction may be leaking future behavior, chronological or user-held-out is often more realistic. Statistics must be used for training only to drive and build the graph, and for normalising and negative sampling.

Popularity, item-based collaborative filtering, matrix factorisation BPR, a neural predictor and a graph model [1], [3], [4], [9], [10]. The hybrid model needs to be examined by ablating the graph, the ML or the policy, and still providing the same amount of candidate sets and the same optimization budget.

6.2 Metrics and Reproducibility

Precision@K, Recall@K and NDCG@K measure complementary aspects of top-K ranking [2]. Candidate recall, coverage, novelty or diversity, latency, memory and calibration may also be needed depending on the user task. Repeated runs, uncertainty intervals, seeds and hardware should accompany any performance claim.

Table 3. Minimum reporting protocol for a controlled recommendation study.

Component	Required specification	Control
Data preparation	Version, filters, feedback threshold, duplicates	No test information enters features or graph
Split	Chronological or user-held-out	Document cold-start and warm-start subsets
Candidates	Same candidate universe for all rankers	Report candidate recall separately
Ranking	Precision@K, Recall@K, NDCG@K	Report K, negatives and relevance rule
Operations	Latency, throughput, peak memory	Use same hardware and batch policy
Robustness	Seeds, intervals, drift and ablations	Release configuration and raw logs

7. DISCUSSION

7.1 Expected Strengths and Boundaries

The framework can integrate the relational, predictive and adaptive information in a modular fashion. Higher order connectivity is supported by graph embeddings, a policy component represents delayed utility, and side features are added to supervised models. These are design capabilities and not an indication that the integrated system will be better than all baselines.

Graph propagation can “noise up” or “popularize” noisy or popular nodes, deep rankers can overfit sparse data, and RL can exploit a missspecified reward. The more than one hyperparameter and the operating complexity of a hybrid fusion also are increased. The justification of the model should, therefore, be based on results of ablation, calibration, latency and robustness, and not on a single score for a set of top K.

7.2 Privacy, Fairness and Oversight

Sensitive preferences are apparent from the recommendation logs. Data minimization, purpose limitation, retention controls, access logging and handling secure features should be specified before implementing. The output from fairness analysis should be sets of tests applied to the exposure and error rates to determine lack of equality between relevant user/oriented groups and/or between relevant item/oriented groups, and explanations should provide details about the key evidence that must be ranked, without revealing private information or deceiving its users.

High impact domains, and changes to reward definitions, eligibility to any content or safety thresholds require human oversight. Some of the common elements monitored involve drift, popularity concentration, anomalous feedback, policy instability and harmful-item exposure. Recommendation system includes procedures for incident response and incident rollback, but does not include external administrative procedures.

7.3 Future Research

Future studies will involve evaluating inductive graph encoders for entities not previously seen, debiased offline learning, constrained multi-objective policies, and calibrated uncertainty as well as efficient graph updates. Staggered configurations, baseline differences, rigid evaluations and runling and evaluation protocols that mix scores from offline and online that are used as measures of quality, must be avoided to ensure that progress is reproducible.

8. CONCLUSION

This revision offers a coherent and mathematically rigorous guidelines for the optimization of the recommender systems with graph-based optimization techniques, machine learning and reinforcement learning, while avoiding watermarking and merging all the techniques into a single entity. Ten equations that can be modified denote how to build, propagate, integrate, update the policy, rank the loss and test the top-k in the graph; the workflow (Algorithm 1) is specified; and the reporting protocol excludes unsupported comparisons. The framework should be considered as a modular research design and requires to be evaluated with data splits that account for leakage, aligned sets of candidates, ablation studies, and uncertainty reporting and operational safeguards.

REFERENCES

1. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in Proc. 10th Int. Conf. World Wide Web, 2001, pp. 285-295.
2. J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," ACM Trans. Inf. Syst., vol. 22, no. 1, pp. 5-53, 2004.
3. Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, vol. 42, no. 8, pp. 30-37, 2009.
4. S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in Proc. 25th Conf. Uncertainty in Artificial Intelligence, 2009, pp. 452-461.

. R. van den Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," arXiv:1706.02263, 2017.

. W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in Adv. Neural Inf. Process. Syst., vol. 30, 2017, pp. 1024-1034.

. R. Ying et al., "Graph convolutional neural networks for web-scale recommender systems," in Proc. 24th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2018, pp. 974-983.

. X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in Proc. 42nd Int. ACM SIGIR Conf., 2019, pp. 165-174.

. X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in Proc. 43rd Int. ACM SIGIR Conf., 2020, pp. 639-648.

0. X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in Proc. 26th Int. Conf. World Wide Web, 2017, pp. 173-182.

1. H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A factorization-machine based neural network for CTR prediction," in Proc. 26th Int. Joint Conf. Artificial Intelligence, 2017, pp. 1725-1731.

2. P. Covington, J. Adams, and E. Sargin, "Deep neural networks for YouTube recommendations," in Proc. 10th ACM Conf. Recommender Systems, 2016, pp. 191-198.

3. W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in Proc. IEEE Int. Conf. Data Mining, 2018, pp. 197-206.

14. V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, pp. 529-533, 2015.
15. M. Chen et al., "Top-K off-policy correction for a REINFORCE recommender system," in Proc. 12th ACM Int. Conf. Web Search and Data Mining, 2019, pp. 456-464.
6. F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," ACM Trans. Interactive Intelligent Systems, vol. 5, no. 4, Art. 19, 2015.