Research: Graph-based recommender systems with explicit negative feedback encoding

Made by Sidorova Anna, Tishin Roman

Sections

The articles under study:

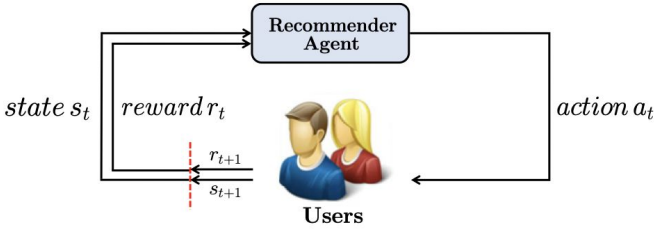
1. Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning
2. NFARec: A Negative Feedback-Aware Recommender Model
3. DFGNN: Dual-frequency Graph Neural Network for Sign-aware Feedback
4. PANE-GNN: Unifying Positive and Negative Edges in Graph Neural Networks for Recommendation
5. Learning and Optimization of Implicit Negative Feedback for Industrial Short-video Recommender System
6. Multi-Behavior Sequential Recommendation with Temporal Graph Transformer
7. KGUF: Simple Knowledge-aware Graph-based Recommender with User-based Semantic Features Filtering
8. Explicit Behavior Interaction with Heterogeneous Graph for Multi-behavior Recommendation
9. Pareto-based Multi-Objective Recommender System with Forgetting Curve
10. Multi-behavior Session-based Recommendation via Graph Reinforcement Learning
11. A Metric Learning Perspective on the Implicit Feedback-Based Recommendation Data Imbalance Problem
12. Negative Can Be Positive: Signed Graph Neural Networks for Recommendation
13. Reinforced Negative Sampling over Knowledge Graph for Recommendation
14. RevGNN: Negative Sampling Enhanced Contrastive Graph Learning for Academic Reviewer Recommendation
15. Positive, Negative and Neutral: Modeling Implicit Feedback in Session-based News Recommendation

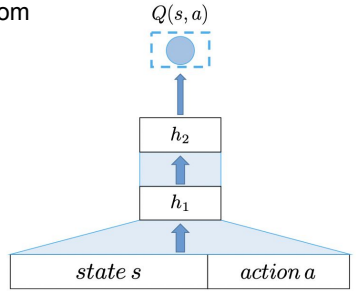
The articles under study:

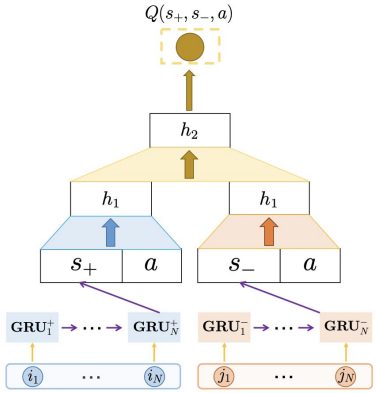
16. SIGformer: Sign-aware Graph Transformer for Recommendation
17. Amplify Graph Learning for Recommendation via Sparsity Completion
18. Graph Contrastive Learning with Multi-Objective for Personalized Product Retrieval in Taobao Search
19. Challenging the Myth of Graph Collaborative Filtering: a Reasoned and Reproducibility-driven Analysis
20. Multi-behavior Self-supervised Learning for Recommendation
21. Denoising Neural Network for News Recommendation with Positive and Negative Implicit Feedback
22. Beyond Positive History: Re-ranking with List-level Hybrid Feedback
23. TempGNN: Temporal Graph Neural Networks for Dynamic Session-Based Recommendations
24. Negative Sampling in Recommendation: A Survey and Future Directions
25. Towards Automated Negative Sampling in Implicit Recommendation
26. Learning from Negative User Feedback and Measuring Responsiveness for Sequential Recommenders
27. Enhancing Sequential Music Recommendation with Negative Feedback-informed Contrastive Learning
28. Negative Feedback for Music Personalization
29. Comprehensive Analysis of Negative Sampling in Knowledge Graph Representation Learning

Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning - 2018: Architecture

- The optimal strategy is made by maximizing the expected long-term cumulative reward from users.
- A novel recommender system with the capability of continuously improving its strategies during the interactions with users.
- Model the sequential interactions between users and a recommender system as a Markov Decision Process (MDP) and leverage Reinforcement Learning (RL) to automatically learn the optimal strategies via recommending trial-and-error items and receiving reinforcements of these items from users' feedback.
- Leverage Deep Q-Network (DQN), an (adapted) artificial neural network, as a non-linear approximator to estimate the action-value function in RL (this model-free reinforcement learning method does not estimate the transition probability and not store the Q-value table -> flexible to support huge amount of items in recommender systems).
- A deep reinforcement learning based framework DEERS.







- **State** s: $s = (s_+, s_-) \in S$ is a state, where $s_+ = \{i_1, \cdots, i_N\}$ is defined as the previous N items that a user clicked or ordered recently, and $s_- = \{j_1, \cdots, j_N\}$ is the previous N items that the user skipped recently. The items in s_+ and s_- are processed in chronological order.
- **Transition from** s **to** s': When RA recommends item a at state $s = (s_+, s_-)$ to a user, if users skip the recommended item, we keep $s'_+ = s_+$ and update $s'_- = \{j_2, \cdots, j_N, a\}$. If users click or order the recommended item, update $s'_+ = \{i_2, \cdots, i_N, a\}$, while keeping $s'_- = s_-$. Then set $s' = (s'_+, s'_-)$.

Optimal action-value function $Q^*(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s',a')|s,a\right]$.

Loss function $L(\theta) = \mathbb{E}_{s,a,r,s'}\Big[\Big(y - Q(s_+, s_-, a; \theta)\Big)^2 - \alpha\Big(Q(s_+, s_-, a; \theta) - Q(s_+, s_-, a^C; \theta)\Big)^2\Big]$, $y = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q(s'_+, s'_-, a'; \theta^p)|s_+, s_-, a]$

Derivatives of loss function $\nabla_\theta L(\theta) = \mathbb{E}_{s,a,r,s'}\Big[\Big(y - Q(s_+, s_-, a; \theta)\Big)\nabla_\theta Q(s_+, s_-, a; \theta) - \alpha\Big(Q(s_+, s_-, a; \theta) - Q(s_+, s_-, a^C; \theta)\Big)\Big(\nabla_\theta Q(s_+, s_-, a; \theta) - \nabla_\theta Q(s_+, s_-, a^C; \theta)\Big)\Big]$.

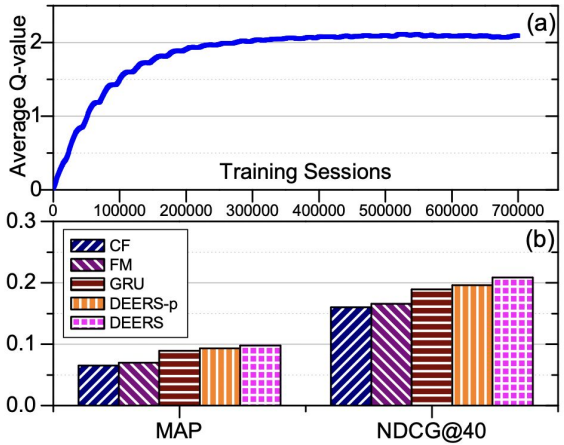
Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning - 2018: Dataset & Results

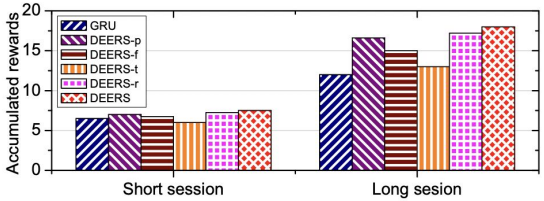
“We test the simulator on users' logs(not the data for training the DEERS framework and simulator), and experimental results demonstrate that the simulated online environment has overall 90% precision for immediate feedback prediction task.”

“We evaluate our method on a dataset of September, 2017 from JD.com . We collect 1,000,000 recommendation sessions (9,136,976 items) in temporal order, and use first 70% as training set and other 30% as test set. For a given session, the initial state is collected from the previous sessions of the user. In this paper, we leverage N = 10 previously clicked/ordered items as positive state and N = 10 previously skipped items as negative state.” ***Authors not provide link to dataset,** but i find another version (2014) - JD.com E-Commerce Data (download with 3ru2 passwd)

Metrics: MAP, NDCG@40;
Surveys models:

- **CF** (Collaborative filtering)**,**
- **FM** (Factorization Machines),
- **GRU** (Gated Recurrent Units),
- **DEERS-p** (DQN with s+ embeddings)
- **DEERS-f** (DQN where all layers are fully connected, state - GRU)
- **DEERS-t** (remove GRU and concatenate 10 s+ state and 10 s-)
- **DEERS-r** a = 0





Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning - 2018: Limitations & Ideas to improve

Limitations:

- Cold-start problem: The system relies on historical data to learn user preferences, meaning it struggles with new users or items that lack sufficient interaction data.

- Scalability: As the system needs to process pairwise comparisons between all candidate items, it can become computationally expensive for large-scale applications with millions of items and users.

- Exploration-exploitation tradeoff: The system must balance between exploiting known preferences (based on past interactions) and exploring new items. This tradeoff is particularly challenging when incorporating negative feedback, as too much exploration may result in lower short-term user satisfaction.

- Setup: you need to adjust the alpha and N parameters for the algorithm to work correctly

Ideas to improve:

- Hierarchical Reinforcement Learning: Introducing hierarchical reinforcement learning could help decompose the recommendation task into sub-tasks (e.g., recommending a genre first, then specific items), which might reduce the computational complexity of pairwise comparisons.

- Accounting for clicks on purchases

- Using product information

NFARec: A Negative Feedback-Aware Recommender Model - 2024: Architecture

"The main contributions of our work can be summarized as follows:

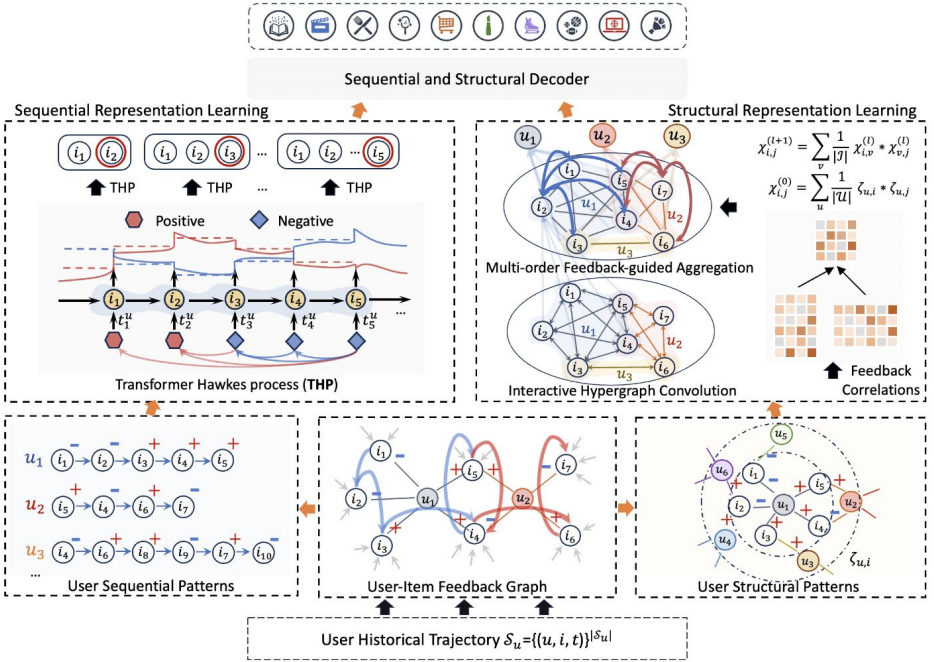
- We propose a negative feedback-aware recommender model that maximizes the leverage of negative feedback in sequential and structural patterns.
- We adopt an auxiliary task to learn the sentiment relatedness between the user's interacted items based on THP. We empirically find that the task promotes the NFARec to identify items the user has interacted with but may not prefer.
- We propose a two-phase HGC approach that leverages feedback relations between users and items to construct an optimal path for message-passing in HGC.
- Extensive experiments on five public real-world datasets demonstrate that our NFARec model outperforms SOTA recommender methods."

Loss function for RecSys task: $\mathcal{L}_{main} = \sum_{(u \in \mathcal{U})} \sum_{(i \in \mathcal{I})} c_{u,i} * \gamma_i^{(u)} \log(\alpha_4(\hat{\mathbf{r}}_{u,i})),$

Loss for predicting the feedback sentiment polarity of the next interaction (in Sequential Representation Learning):

$$\mathcal{L}_{auxi} = \sum_{(u \in \mathcal{U})} (\underbrace{\sum_{j=1}^{|\mathcal{S}_u|} \log \lambda(t_j \mid \mathcal{T}_t)}_{\text{interactive log-likelihood}} - \underbrace{\int_{t_1}^{t_{|\mathcal{S}_u|}} \lambda(t \mid \mathcal{T}_t)\, dt}_{\text{continuous log-likelihood}}).$$

$$\mathcal{L}_{final} = \mathcal{L}_{main} + \delta_2 \mathcal{L}_{auxi},$$



NFARec: A Negative Feedback-Aware Recommender Model - 2024: Dataset & Results

Authors collected five public datasets with rating scores, i.e., Yelp 2023, Beauty, Books, Recipes, and MovieLens.
Authors regard interactions with ratings below 4 as negative feedback; others are positive. They exclude users and items with fewer than n interactions, where n is 15, 10, 25, and 5 for Yelp 2023, Recipes, Books, and Beauty, respectively.

Dataset	Metric	CF-based		GCN-based			GCL-based			Dif-based	S&G-based			Ours	Improv.	p-value
		DirectAU	InvCF	LightGCN	SGL	HCCF	NCL	LightGCL	XSimGCL	DiffRec	SHT	AutoCF	EEDN			
Yelp 2023	**Recall@5**	**0.0316**	**0.0321**	**0.0344**	**0.0366**	**0.0299**	**0.0360**	**0.0341**	**0.0376**	**0.0323**	**0.0340**	**0.0391**	**0.0426**	**0.0449**	**+5.6% ***	**1.4e-07**
	NDCG@5	0.0497	0.0607	0.0540	0.0551	0.0426	0.0535	0.0479	0.0566	0.0487	0.0619	0.0614	0.0668	**0.0741**	+10.9% *	2.4e-09
	Recall@10	**0.0517**	**0.0464**	**0.0564**	**0.0581**	**0.0403**	**0.0572**	**0.0502**	**0.0590**	**0.0529**	**0.0476**	**0.0535**	**0.0672**	**0.0707**	**+5.2% ***	**2.2e-06**
	NDCG@10	0.0507	0.0644	0.0587	0.0603	0.0457	0.0571	0.0498	0.0595	0.0548	0.0673	0.0704	0.0733	**0.0778**	+6.1% *	4.9e-07
	Recall@20	**0.0722**	**0.0798**	**0.0752**	**0.0721**	**0.0518**	**0.0795**	**0.0675**	**0.0857**	**0.0809**	**0.0796**	**0.0939**	**0.1031**	**0.1074**	**+4.2% ***	**6.1e-06**
	NDCG@20	0.0714	0.0795	0.0715	0.0740	0.0526	0.0743	0.0661	0.0756	0.0713	0.0785	0.0810	0.0858	**0.0894**	+4.1% *	2.2e-06
MovieLens	**Recall@5**	**0.0726**	**0.0728**	**0.0603**	**0.0665**	**0.0603**	**0.0673**	**0.0690**	**0.0726**	**0.0642**	**0.0698**	**0.0691**	**0.0737**	**0.0789**	**+7.1% ***	**1.8e-08**
	NDCG@5	0.2969	0.3144	0.2438	0.2767	0.2490	0.2579	0.2556	0.3165	0.2710	0.2822	0.3084	0.3222	**0.3716**	+15.3% *	1.9e-12
	Recall@10	**0.1044**	**0.1085**	**0.0863**	**0.1069**	**0.0967**	**0.1029**	**0.1046**	**0.1037**	**0.0923**	**0.1072**	**0.0991**	**0.1136**	**0.1213**	**+6.7% ***	**8.3e-06**
	NDCG@10	0.2819	0.2831	0.2476	0.2829	0.2539	0.2497	0.2494	0.2754	0.2670	0.2739	0.2729	0.2878	**0.3314**	+15.1% *	3.03-13
	Recall@20	**0.1504**	**0.1481**	**0.1107**	**0.1432**	**0.1287**	**0.1298**	**0.1313**	**0.1426**	**0.1204**	**0.1509**	**0.1470**	**0.1628**	**0.1736**	**+6.6% ***	**3.5e-07**
	NDCG@20	0.2592	0.2537	0.2047	0.2541	0.2312	0.2479	0.2540	0.2481	0.2284	0.2400	0.2452	0.2631	**0.3007**	+14.2% *	5.6e-10
Recipes	**Recall@5**	**0.0235**	**0.0167**	**0.0208**	**0.0265**	**0.0224**	**0.0207**	**0.0164**	**0.0253**	**0.0193**	**0.0249**	**0.0162**	**0.0269**	**0.0292**	**+8.6% ***	**1.6e-09**
	NDCG@5	0.0359	0.0352	0.0283	0.0365	0.0330	0.0267	0.0257	0.0389	0.0283	0.0377	0.0264	0.0395	**0.0418**	+9.4% *	1.5e-08
	Recall@10	**0.0426**	**0.0266**	**0.0333**	**0.0406**	**0.0340**	**0.0330**	**0.0296**	**0.0419**	**0.0324**	**0.0405**	**0.0273**	**0.0401**	**0.0442**	**+5.5% ***	**3.6e-06**
	NDCG@10	0.0429	0.0254	0.0304	0.0382	0.0332	0.0293	0.0275	0.0435	0.0305	0.0412	0.0237	0.0429	**0.0459**	+5.5% *	1.4e-06
	Recall@20	**0.0595**	**0.0376**	**0.0399**	**0.0566**	**0.0386**	**0.0402**	**0.0360**	**0.0595**	**0.0416**	**0.0560**	**0.0331**	**0.0592**	**0.0609**	**+2.4% ***	**1.4e-03**
	NDCG@20	0.0476	0.0227	0.0329	0.0465	0.0346	0.0319	0.0299	0.0481	0.0392	0.0406	0.0186	0.0475	**0.0492**	+2.3% *	2.3e-04
Books	**Recall@5**	**0.0392**	**0.0301**	**0.0329**	**0.0338**	**0.0252**	**0.0359**	**0.0275**	**0.0440**	**0.0254**	**0.0317**	**0.0329**	**0.0473**	**0.0489**	**+3.4% ***	**1.7e-04**
	NDCG@5	0.0973	0.0827	0.0905	0.0874	0.0524	0.0912	0.0723	0.1029	0.0654	0.0834	0.0890	0.1082	**0.1139**	+5.3% *	1.0e-06
	Recall@10	**0.0696**	**0.0590**	**0.0655**	**0.0693**	**0.0474**	**0.0667**	**0.0513**	**0.0712**	**0.0484**	**0.0680**	**0.0516**	**0.0729**	**0.0753**	**+3.3% ***	**1.1e-05**
	NDCG@10	0.0988	0.0804	0.0956	0.0896	0.0635	0.0969	0.0715	0.0990	0.0632	0.0913	0.0727	0.1011	**0.1065**	+5.3% *	3.1e-07
	Recall@20	**0.1029**	**0.0854**	**0.0994**	**0.0981**	**0.0670**	**0.0971**	**0.0672**	**0.1037**	**0.0647**	**0.0828**	**0.0762**	**0.1075**	**0.1112**	**+3.4% ***	**7.8e-05**
	NDCG@20	0.1024	0.0792	0.1007	0.0992	0.0812	0.0943	0.0645	0.1033	0.0560	0.0799	0.0730	0.1099	**0.1141**	+3.9% *	1.1e-08
Beauty	**Recall@5**	**0.0518**	**0.0530**	**0.0489**	**0.0522**	**0.0367**	**0.0521**	**0.0468**	**0.0548**	**0.0479**	**0.0520**	**0.0490**	**0.0548**	**0.0585**	**+6.8% ***	**5.7e-07**
	NDCG@5	0.0447	0.0476	0.0414	0.0467	0.0344	0.0453	0.0417	0.0486	0.0402	0.0415	0.0437	0.0476	**0.0514**	+8.0% *	7.8e-07
	Recall@10	**0.0725**	**0.0710**	**0.0704**	**0.0737**	**0.0517**	**0.0688**	**0.0615**	**0.0733**	**0.0657**	**0.0719**	**0.0759**	**0.0779**	**0.0811**	**+4.1% ***	**1.2e-06**
	NDCG@10	0.0527	0.0522	0.0499	0.0530	0.0387	0.0502	0.0488	0.0557	0.0476	0.0513	0.0542	0.0558	**0.0596**	+6.8% *	8.1e-09
	Recall@20	**0.0908**	**0.1022**	**0.0879**	**0.0939**	**0.0672**	**0.0839**	**0.0803**	**0.1007**	**0.0876**	**0.0901**	**0.1034**	**0.1054**	**0.1103**	**+4.6% ***	**3.0e-05**
	NDCG@20	0.0614	0.0613	0.0570	0.0607	0.0413	0.0529	0.0499	0.0640	0.0517	0.0587	0.0625	0.0648	**0.0683**	+5.4% *	4.1e-05

NFARec: A Negative Feedback-Aware Recommender Model - 2024: Limitations & Ideas to improve

Limitations:

- Dependence on Explicit Feedback: NFARec focuses on explicit feedback like ratings, overlooking implicit signals (e.g., skipped items).
- High Computational Complexity: NFARec's structural representation learning $O(|I|^2)$ can be computationally expensive, especially for large datasets.

Ideas to improve:

- Authors mention:
  - explore the impact of implicit negative feedback to improve performance, such as skipping content
  - incorporate large language models to make more interpretable recommendations.
- Optimize Efficiency: Explore optimization techniques

DFGNN: Dual-frequency Graph Neural Network for Sign-aware Feedback - 2024:

Architecture

Scheme is not provide by authors, I will give you an approximate scheme, which I took from the methods:

(1) dual frequency graph filter: a dual-frequency graph filter (DGF), which uses a low-passing graph filter (LGF) to capture the low-frequency signal in positive feedback and adopts a high-passing graph filter (HGF) to capture the high-frequency signal in negative feedback.

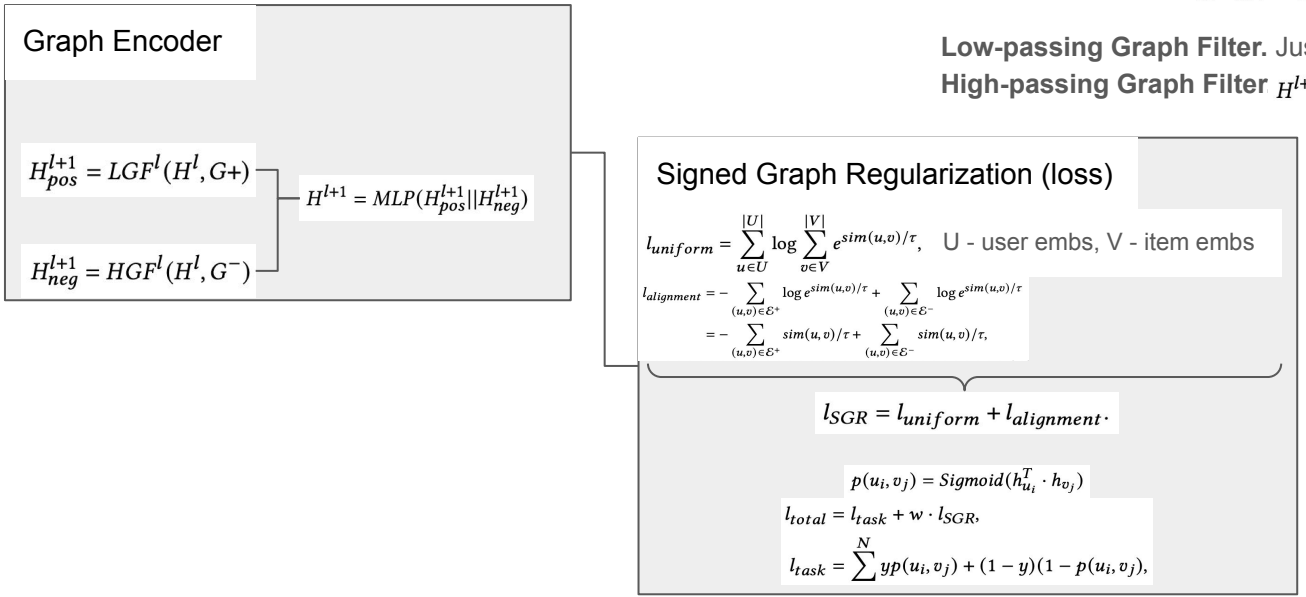
(2) graph regularization: module is adopted to alleviate the representation degeneration problem.

Graph Filter. The convolution operation is generally adopted as a filter to extract useful signals using Fourier transform.

$$x * g = \mathcal{F}^{-1}(\mathcal{F}(x) * \mathcal{F}(g)) = \mathcal{F}^{-1}(f(\lambda)g(\lambda))$$
$$= E((E^T g)(E^T x)) = E\, g(\lambda)\, E^T x,$$

Low-passing Graph Filter. Just GCN

High-passing Graph Filter $H^{l+1} = \sigma(LHW) = \sigma(D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}HW), H^0 = X,$



DFGNN: Dual-frequency Graph Neural Network for Sign-aware Feedback - 2024: Dataset & Results

Datasets: MovieLens, different categories of Amazon Review datasets, and Yelp 2023.

Table 1: Results on recommendation task. All improvements are significant over baselines (t-test with $p < 0.05$).

Dataset	Model	GCN	GAT	SGCN	SBGNN	SBGCL	SIGRec	DFGNN	Improve
ML1M	MRR	0.5085	0.5096	0.4866	0.5177	0.4707	0.5259	**0.5433**	3.31%
	HIT@10	0.8132	0.8234	0.8041	0.8279	0.7932	0.8381	**0.8576**	2.33%
	NDCG@10	0.2850	0.2882	0.2696	0.2972	0.2645	0.2948	**0.3118**	4.91%
	HIT@50	0.9524	0.9592	0.9558	0.9587	0.9527	0.9608	**0.9628**	0.21%
	NDCG@50	0.3013	0.3060	0.2912	0.3153	0.2877	0.3184	**0.3311**	3.99%
Arts	MRR	0.0291	0.0296	0.0324	0.0441	0.0301	0.0296	**0.0643**	45.80%
	HIT@10	0.0406	0.0427	0.0482	0.0746	0.0432	0.0409	**0.1107**	48.39%
	NDCG@10	0.0230	0.0234	0.0254	0.0350	0.0236	0.0231	**0.0521**	48.86%
	HIT@50	0.1038	0.1067	0.1093	0.1799	0.1101	0.1076	**0.2284**	26.96%
	NDCG@50	0.0323	0.0326	0.0342	0.0516	0.0334	0.0328	**0.0713**	38.18%
GFood	MRR	0.0421	0.0400	0.0409	0.0447	0.0372	0.0383	**0.0475**	6.26%
	HIT@10	0.0642	0.0599	0.0615	0.0718	0.0597	0.0585	**0.0756**	5.29%
	NDCG@10	0.0325	0.0314	0.0328	0.0363	0.0289	0.0306	**0.0375**	3.31%
	HIT@50	0.1434	0.1273	0.1248	0.1510	0.1258	0.1189	**0.1528**	1.19%
	NDCG@50	0.0444	0.0417	0.0423	0.0488	0.0390	0.0394	**0.0493**	1.02%
Kindle	MRR	0.0381	0.0436	0.0264	0.0476	0.0299	0.0316	**0.0703**	47.69%
	HIT@10	0.0851	0.0930	0.0548	0.0980	0.0621	0.0682	**0.1569**	60.10%
	NDCG@10	0.0221	0.0263	0.0135	0.0315	0.0160	0.0170	**0.0502**	59.37%
	HIT@50	0.2285	0.2299	0.1555	0.2452	0.1740	0.1928	**0.3380**	37.85%
	NDCG@50	0.0416	0.0455	0.0253	0.0531	0.0303	0.0326	**0.0796**	49.91%
Yelp	MRR	0.0378	0.0420	0.0448	0.0460	0.0422	0.0313	**0.0511**	11.09%
	HIT@10	0.0819	0.0893	0.0925	0.0975	0.0902	0.0649	**0.1112**	14.05%
	NDCG@10	0.0204	0.0231	0.0250	0.0257	0.0230	0.0164	**0.0298**	15.95%
	HIT@50	0.2189	0.2382	0.2370	0.2590	0.2353	0.1917	**0.2875**	11.00%
	NDCG@50	0.0377	0.0432	0.0442	0.0476	0.0423	0.0317	**0.0543**	14.08%

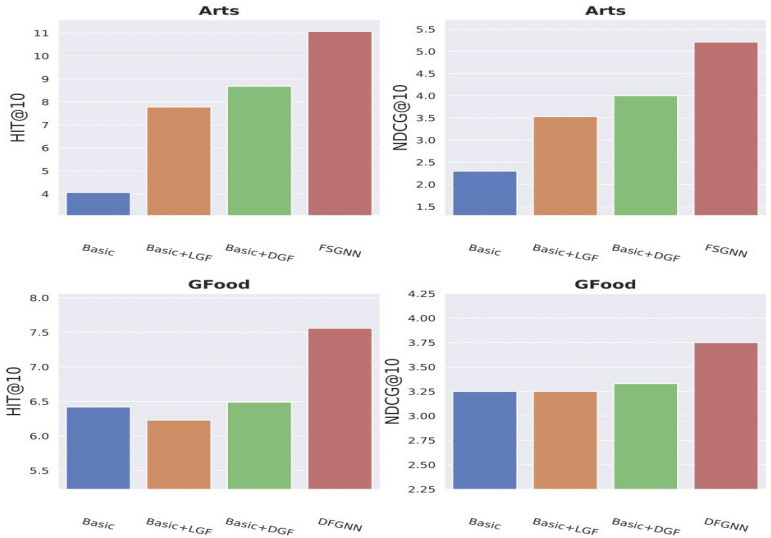


Figure 6: Ablation study of recommendation task on Arts and GFood dataset

(1) Basic is the basic version of DFGNN, which only contains the positive edge and low-passing GNN (2) Basic + LGF is the version that applies low-passing graph filter on negative edges. (3) Basic + DGF, which adopts our DGF to encode the signed graph (4) DFGNN is the full version of our model, which both adopts our SGR and DGF to the signed graph.

DFGNN: Dual-frequency Graph Neural Network for Sign-aware Feedback - 2024:

Limitations & Ideas to improve

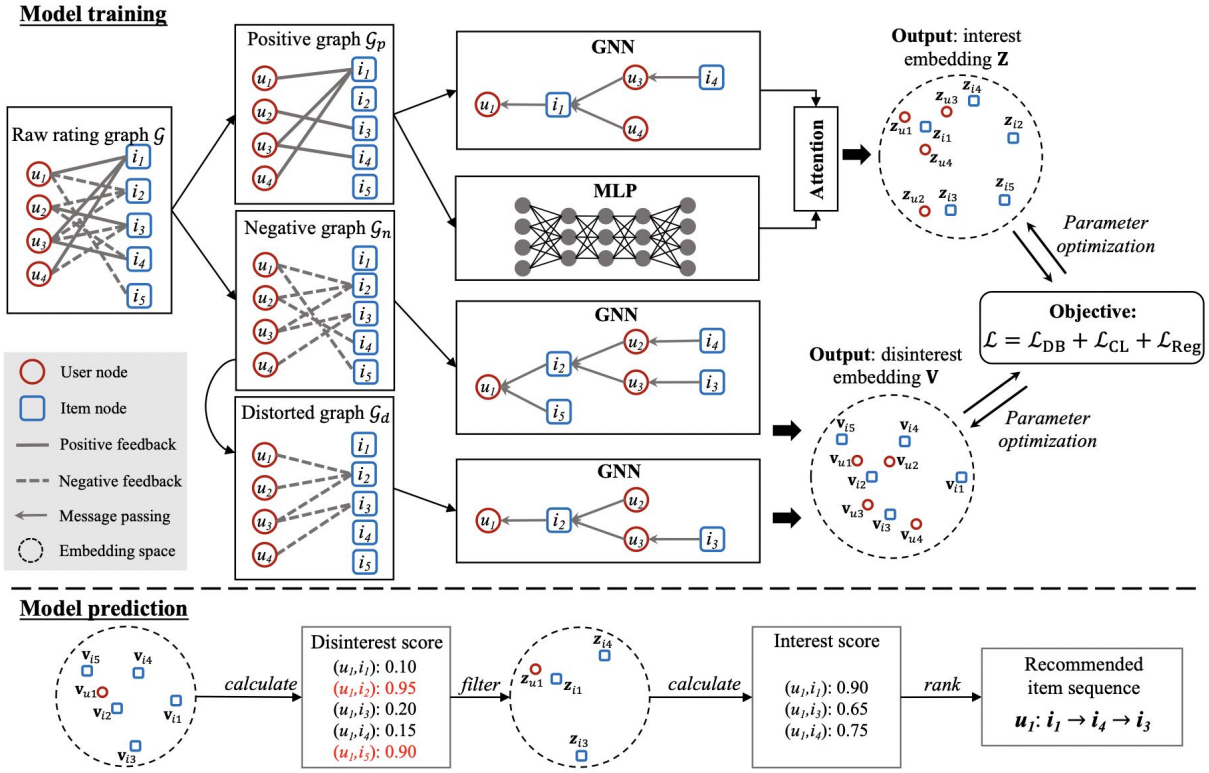
Limitations:

- GNNs, particularly deeper ones, tend to suffer from over-smoothing, where the node representations become indistinguishable. This can reduce the expressiveness of the learned embeddings, a problem acknowledged in the paper. Although the proposed graph regularization mitigates this to some extent, deeper analysis is needed to explore how the model performs on much larger datasets, where over-smoothing can be more pronounced.
- Like many GNN-based approaches, scalability remains a concern when applied to extremely large graphs (e.g., millions of nodes). The paper does not delve into how computationally efficient the model is for large-scale industrial applications.

Ideas to improve:

- The model could benefit from techniques like residual connections or dynamic graph pruning, which have been shown to alleviate over-smoothing in other GNN architectures. Additionally, experimenting with adaptive depth control could enhance flexibility in deeper networks (?).
- Applying sampling techniques like GraphSAGE or mini-batching strategies tailored for large-scale graphs could be integrated to improve computational efficiency.

PANE-GNN: Unifying Positive and Negative Edges in Graph Neural Networks for Recommendation - 2023: Architecture



Algorithm 1 PANE-GNN

Input: Positive graph $\mathcal{G}_p$, negative graph $\mathcal{G}_n$, trainable parameters $\Theta_{\text{Emb}} = \{\mathbf{Z}^{(0)}, \mathbf{V}^{(0)}\}$ and $\Theta_{\text{NN}} = \{\mathbf{W}^{(1)}_{\text{MLP}}, \mathbf{W}^{(2)}_{\text{MLP}}, \mathbf{W}^{(1)}_{\text{Att}}, \mathbf{W}^{(2)}_{\text{Att}}\}$, embedding size H, GNNs layer number K, hyperparameters $p, b, \delta, \lambda_1, \lambda_2, \tau$.
Output: Interest embedding matrix $\mathbf{Z}$, disinterest embedding matrix $\mathbf{V}$.

Initialize Θ_{Emb} and Θ_{NN} via the Glorot method;
Initialize embedding matrices: $\mathbf{Z} \leftarrow \mathbf{Z}^{(0)}$, $\mathbf{V} \leftarrow \mathbf{V}^{(0)}$, $\tilde{\mathbf{V}} \leftarrow \mathbf{V}^{(0)}$;
Distort $\mathcal{G}_n$ into $\mathcal{G}_d$ according to Eq. (9);
while not converged **do**
  Generate training set $\mathcal{D}_p$ from $\mathcal{G}_p$ based on Eq. (14);
  Generate training set $\mathcal{D}_n$ from $\mathcal{G}_n$ based on Eq. (15);
  for each mini-batch $\mathcal{B}_p \subset \mathcal{D}_p$ **do**
    Calculate $\mathbf{Z}'$ according to Eq. (5);
    Calculate $\mathbf{Z}''$ according to Eq. (7);
    Update $\mathbf{Z}$ according to Eq. (8);
  end for
  for each mini-batch $\mathcal{B}_n \subset \mathcal{D}_n$ **do**
    Update $\mathbf{V}$ according to Eq. (6);
    Update $\tilde{\mathbf{V}}$ according to Eq. (12);
  end for
  Calculate $\mathcal{L}_{\text{DB}}$ according to Eq. (17);
  Calculate $\mathcal{L}_{\text{CL}}$ according to Eq. (18);
  $\mathcal{L}_{\text{Reg}} \leftarrow \|\Theta_{\text{Emb}}\|^2$;
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{DB}} + \lambda_1 \cdot \mathcal{L}_{\text{CL}} + \lambda_2 \cdot \mathcal{L}_{\text{Reg}}$
  Update Θ_{Emb} and Θ_{NN} by taking one step of gradient descent on $\mathcal{L}$;
end while
return $\mathbf{Z}$, $\mathbf{V}$.

PANE-GNN: Unifying Positive and Negative Edges in Graph Neural Networks for Recommendation - 2023: Dataset & Results

Datasets: MovieLens, amazon-book, Yelp 2023, KuaiRec

Table 4: Results (%) of ablation studies on ML-1M and KuaiRec. Here "MP", "GCL", and "Filter" denote message passing, graph contrastive learning, and the disinterest-score filter, respectively.

Dataset	Variant	Description	$K=5$			$K=10$			$K=15$		
			Precision@K	*Recall@K*	*nDCG@K*	*Precision@K*	*Recall@K*	*nDCG@K*	*Precision@K*	*Recall@K*	*nDCG@K*
ML-1M	A	MP on $\mathcal{G}_n$	0.64±0.02	0.13±0.04	0.68±0.03	0.62±0.01	0.26±0.01	0.67±0.02	0.61±0.02	0.43±0.01	0.70±0.03
	B	MP on $\mathcal{G}_p$	31.51±0.15	12.11±0.10	34.49±0.20	26.35±0.22	19.23±0.11	32.59±0.17	23.32±0.20	24.47±0.19	32.34±0.15
	C	MP on $\mathcal{G}_p$ & $\mathcal{G}_n$	32.65±0.08	12.87±0.18	35.92±0.15	27.49±0.23	20.35±0.09	34.13±0.11	24.17±0.14	25.67±0.13	33.79±0.13
	D	Variant-C + GCL	33.46±0.11	13.05±0.15	36.56±0.21	27.77±0.20	20.40±0.11	34.45±0.09	24.36±0.12	25.70±0.17	34.05±0.04
	PANE-GNN	Variant-D + Filter	33.66±0.14	13.26±0.17	36.90±0.25	27.97±0.14	20.50±0.18	34.70±0.13	24.66±0.22	25.95±0.09	34.37±0.14
KuaiRec	A	MP on $\mathcal{G}_n$	5.54±0.00	5.13±0.01	6.54±0.01	5.40±0.01	10.29±0.00	8.09±0.02	5.61±0.01	15.78±0.01	10.08±0.00
	B	MP on $\mathcal{G}_p$	24.22±0.10	33.25±0.13	40.05±0.13	17.61±0.08	42.19±0.23	41.60±0.16	14.64±0.15	49.36±0.17	43.83±0.12
	C	MP on $\mathcal{G}_p$ & $\mathcal{G}_n$	24.70±0.14	32.90±0.09	40.52±0.20	17.70±0.22	42.67±0.27	41.59±0.09	14.92±0.14	50.12±0.18	44.03±0.13
	D	Variant-C + GCL	24.94±0.18	34.04±0.22	41.21±0.06	19.37±0.14	45.81±0.04	44.10±0.28	15.88±0.19	53.36±0.06	46.28±0.18
	PANE-GNN	Variant-D + Filter	25.85±0.10	34.61±0.11	41.91±0.20	19.39±0.17	46.03±0.07	44.13±0.05	16.16±0.16	53.47±0.15	46.55±0.10

Table 3: Results (%) of baselines and our method on ML-1M, Amazon-Book, Yelp, and KuaiRec. In each column, the best result is bolded. The results of the methods marked with "†" are from [30]. For the methods marked with "⋆", we run each of them five times with default hyperparameter settings.

Dataset	Method	$K=5$			$K=10$			$K=15$		
		Precision@K	*Recall@K*	*nDCG@K*	*Precision@K*	*Recall@K*	*nDCG@K*	*Precision@K*	*Recall@K*	*nDCG@K*
ML-1M	NGCF†	29.73±0.43	10.99±0.26	32.38±0.45	24.77±0.23	17.48±0.25	30.31±0.33	21.74±0.22	22.29±0.27	29.85±0.29
	LR-GCCF†	30.52±0.33	11.40±0.23	33.30±0.44	25.39±0.27	18.02±0.31	31.17±0.39	22.20±0.25	22.92±0.46	30.66±0.42
	LightGCN†	32.18±0.22	12.06±0.11	35.19±0.23	26.79±0.13	19.09±0.16	32.97±0.18	23.49±0.16	24.32±0.29	32.49±0.22
	SGCN†	24.84±033	9.10±0.17	26.83±0.35	18.73±0.20	14.92±0.26	25.47±0.24	18.73±0.20	19.32±0.37	25.30±0.26
	SiReN†	33.28±0.54	12.79±0.27	36.37±0.55	27.74±0.37	20.16±0.33	34.23±0.47	24.44±0.25	25.69±0.29	33.88±0.40
	PANE-GNN (Ours)⋆	**33.66±0.14**	**13.26±0.17**	**36.90±0.25**	**27.97±0.14**	**20.50±0.18**	**34.70±0.13**	**24.66±0.22**	**25.95±0.09**	**34.37±0.14**
Amazon-Book	NGCF†	4.63±0.14	3.20±0.07	5.18±0.11	3.91±0.14	5.32±0.08	5.62±0.10	4.03±1.27	7.06±0.07	6.18±0.09
	LR-GCCF†	4.69±0.16	3.24±0.02	5.27±0.12	3.99±0.14	5.44±0.05	5.74±0.09	3.57±0.13	7.21±0.04	6.31±0.08
	LightGCN†	5.29±0.15	3.62±0.07	5.96±0.11	4.43±0.13	5.95±0.08	6.38±0.07	3.93±0.11	7.81±0.08	6.98±0.07
	SGCN†	3.90±0.23	2.67±0.12	4.33±0.24	3.36±0.18	4.54±0.18	4.75±0.23	3.04±0.16	6.09±0.21	5.27±0.24
	SiReN†	6.78±0.25	4.74±0.05	7.66±0.02	5.65±0.25	7.75±0.08	8.23±0.13	4.97±0.18	10.09±0.11	8.97±0.11
	PANE-GNN (Ours)⋆	**7.31±0.05**	**4.95±0.19**	**8.14±0.08**	**6.07±0.22**	**8.05±0.18**	**8.69±0.26**	**5.32±0.20**	**10.41±0.05**	**9.45±0.11**
Yelp	NGCF†	2.85±0.12	2.26±0.07	3.29±0.10	2.43±0.09	3.83±0.10	3.68±0.10	2.19±0.07	5.15±0.08	4.13±0.09
	LR-GCCF†	3.03±0.14	2.40±0.06	3.51±0.13	2.58±0.10	4.05±0.09	3.92±0.11	2.32±0.08	5.43±0.10	4.39±0.12
	LightGCN†	3.33±0.11	2.59±0.04	3.86±0.09	2.81±0.09	4.35±0.07	4.27±0.08	2.51±0.08	5.82±0.10	4.76±0.08
	SGCN†	2.93±0.10	2.26±0.06	3.32±0.10	2.56±0.08	3.95±0.11	3.77±0.10	2.32±0.07	5.38±0.19	4.26±0.12
	SiReN†	4.20±0.09	3.32±0.05	4.88±0.07	3.52±0.07	5.54±0.11	5.39±0.07	3.14±0.06	7.37±0.12	6.00±0.06
	PANE-GNN (Ours)⋆	**4.75±0.11**	**3.56±0.06**	**5.49±0.11**	**3.93±0.13**	**5.89±0.07**	**5.94±0.14**	**3.51±0.09**	**7.83±0.05**	**6.59±0.10**
KuaiRec	NGCF⋆	2.05±0.18	5.05±0.20	3.85±0.09	1.88±0.04	8.39±0.17	5.10±0.18	1.69±0.21	10.47±0.14	5.80±0.14
	LR-GCCF⋆	4.84±0.21	11.39±0.25	9.37±0.17	3.60±0.09	15.43±0.11	10.79±0.14	2.96±0.09	17.80±0.18	11.60±0.22
	LightGCN⋆	23.83±0.19	33.76±0.22	39.11±0.07	17.58±0.05	43.16±0.14	41.04±0.20	14.84±0.21	50.95±0.12	43.65±0.15
	SGCN⋆	0.16±0.01	0.13±0.00	0.17±0.00	0.14±0.01	0.20±0.00	0.18±0.01	0.13±0.02	0.27±0.00	0.20±0.01
	SiReN⋆	24.50±0.08	33.33±0.10	40.07±0.08	17.73±0.22	42.75±0.17	41.44±0.13	14.81±0.16	49.90±0.05	43.72±0.09
	PANE-GNN (Ours)⋆	**25.85±0.10**	**34.61±0.11**	**41.91±0.20**	**19.39±0.17**	**46.03±0.07**	**44.13±0.05**	**16.16±0.16**	**53.47±0.15**	**46.55±0.10**

PANE-GNN: Unifying Positive and Negative Edges in Graph Neural Networks for Recommendation - 2023: Limitations & Ideas to improve

Limitations:

- **Handling of Negative Feedback and Graph Structure:** The model struggles with effectively incorporating negative feedback due to the assumptions of homophily and balance theory, which do not apply well in signed bipartite graphs. These graphs, often used in recommender systems, complicate direct integration of negative interactions.
- **Risk of Oversmoothing:** The PANE-GNN model faces a risk of oversmoothing, where, with increasing message-passing layers, node representations become overly similar. This can lead to a loss of distinguishing information between nodes, limiting the model's expressiveness and accuracy in making personalized recommendations.

Ideas to improve:

- **Implement a more targeted negative sampling strategy,** which would focus on meaningful disinterest connections. By identifying relevant negative connections, the model can better learn user disinterest patterns, reducing reliance on broad assumptions like homophily and balance theory.
- **Techniques to Mitigate Oversmoothing:** Use methods like DropEdge, which randomly drops edges during training, or apply node-level regularization to preserve diversity in node embeddings.

Learning and Optimization of Implicit Negative Feedback for Industrial Short-video Recommender System - 2023: Architecture

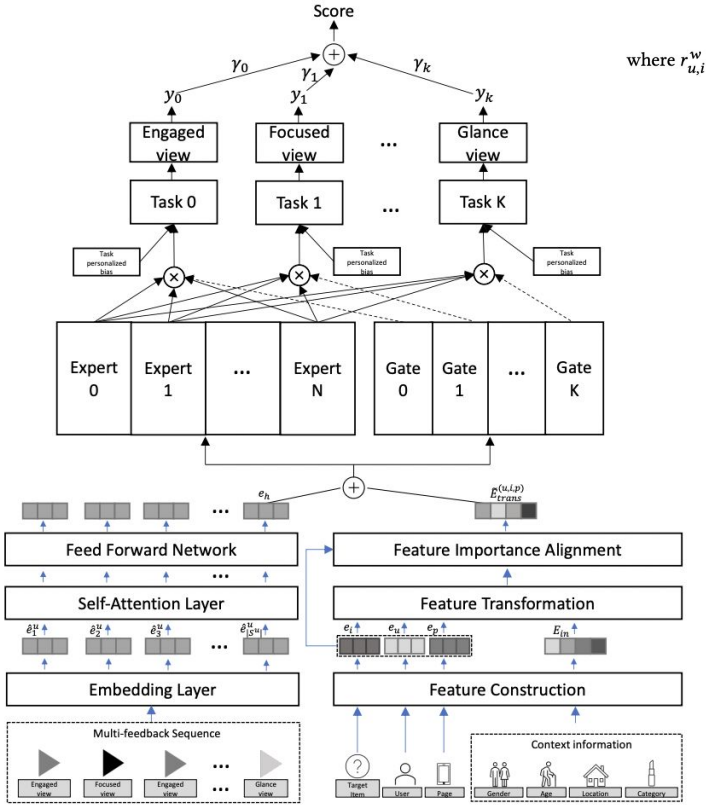


Figure 1: Illustration of our proposed system.

$$R_u^w = \{r_{u,1}^w, r_{u,2}^w, \cdots, r_{u,|S^u|}^w\}, w \in \{e, f, g\},$$

where $r_{u,i}^w$ is either 1 or 0, corresponding to True and False.

- R represent each feedback type, e - positive feedback, f - stronger positive feedback, g - negative feedback

$$E^u = (e_1^u, e_2^u, \cdots, e_{|S^u|}^u).$$

- item embs

$$\hat{E}^u = [E^u; R_u^e; R_u^f; R_u^g] + \mathbf{p}.$$

- $\hat{E}^u$ - self-attention encoder input, p - a learnable position embedding for position information.

$$\mathbb{L}_k = -\mathbb{Y}_k^+ \log \mathbf{y}_k - (1 - \mathbb{Y}_k^-) \log(1 - \mathbf{y}_k), ; \mathbb{L}^{'} = \sum_{k=1}^{K} \lambda_k \mathbb{L}_k, \sum_{k=1}^{K} \lambda_k = 1.$$

2.4.2 Fusion strategy for online serving. Our proposed method serves in the ranking phase of the real-world recommender system, of which the final output is the ranking list exposed to the users. For the online serving in the real system, we adopt a simple yet effective fusion strategy as **score** $= \sum_{k=1}^{K} \gamma_k \mathbf{y}_k$, where the selection γ_k depends on the real-world requirements; for example, the γ_k value for negative feedback should be a negative number. With the fused score, we generate the L-length ranking list from the candidate pool (generated by the previous Recall Phase in the recommendation engine) with the L highest scores.

Learning and Optimization of Implicit Negative Feedback for Industrial Short-video Recommender System - 2023: Dataset & Results

A/B testing was provided on Kuaishou platform on Kuaishou's Discover and Featured pages. Baseline. To evaluate the effect of negative feedback, we remove the mixed feedback encoder from the proposed model to serve as the experimental baseline model.

3.1.2 Main results. Our results, detailed in Table 2, reveal two key insights:

- **By incorporating user negative feedback, our model outperforms the baseline in all measured metrics.** Our model successfully identifies user intent and context-aware preferences, leading to more active users, increasing video viewing time, and enhancing user engagement (evident from more likes and comments), with relative improvements of 0.055% and 0.160%, respectively.
- **Integrating implicit user negative feedback, such as Glance Video Viewing, helps reduce explicit negative behaviors.** A notable decrease of 0.049% users and 0.251% total reducing times is observed in the use of Kuaishou's "reduce similar videos" function, which is intended to collect explicit negative feedback. Thus, our method effectively handles user feedback, enhancing the user experience.

Table 2: Main results of A/B performance on user multi-feedback modeling. Reduction refers to user reports reducing similar recommendations. ↓ denotes that the lower the value, the more satisfied the user is with the recommended video.

Method	Active Users	Play Duration	Players Number	Like Users	Like Times	Comment Times	Reduction Users ↓	Reduction Times ↓
Improvements rate	+0.055%	+0.160%	+0.070%	+0.046%	+0.042%	+0.034%	-0.049%	-0.251%

Table 3: The result of the user's response to the user satisfaction questionnaire in the system popup. ↓ represents negative metrics, in other words, the performance is improved with lower values.

Method	Questionnaire Pos / Neg feedback	Video Pos / Neg feedback	Video Likes	Questionnaire Dislikes ↓	Video Dislikes ↓
Improvements rate	+1.060%	+0.089%	+0.048%	-1.552%	-0.033%

Table 4: Performance of our model under single-column page and double-column page scenarios. ↓ denotes that the lower the value, the better performance our model achieved.

Page	Forward	Comment	Negative	Visitors	Players	Reduction ↓
Single-Column	+0.194%	+0.102%	-0.867%	+0.060%	+0.079%	-0.093%
Double-Column	+0.776%	+0.042%	-2.045%	+0.041%	+0.057%	-2.182%

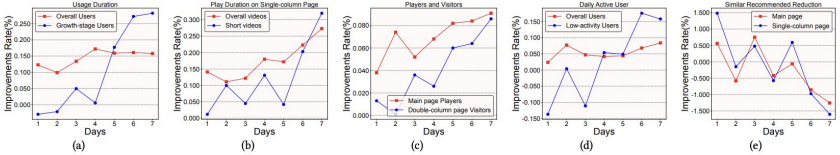


Figure 2: The performance improvement trend of our model in a one-week window. (a) The improvement trend of user's usage duration. (b) The improvement trend of video's playing duration on single-column pages. (c) The improvement trend of the main page's players and double-column page's visitors. (d) The improvement trend of the number of daily active users. (e) The improvement trend of the reduction number of similar recommendations, of which the lower number the better performance.

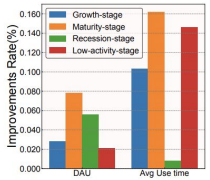
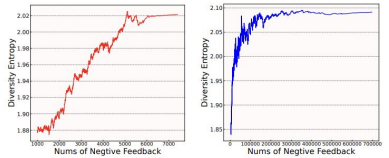


Figure 3: Daily active users (DAU) and user's average using time (Avg Use time) on different user engagement levels.



(a) Result of low exposure users. (b) Result of high exposure users.

Figure 4: Relationship between negative feedback and diversity entropy in the long-term aspect.

Learning and Optimization of Implicit Negative Feedback for Industrial Short-video Recommender System - 2023: Limitations & Ideas to improve

Limitations:

- **Simplified Feedback Categories Limit Granularity in User Behavior Modeling:** The model categorizes feedback into only three types (EVV, FVV, and GVV), based solely on viewing duration thresholds.

Ideas to improve:

- **Incorporating content-based recommendations**—such as analyzing genre, visual style, or creator characteristics—would enable the system to recommend new or niche content more effectively, regardless of historical engagement data.
- **To improve behavioral insight,** the model could incorporate additional interactions, like rewinding, pausing, and duration-based trends (e.g., skipping early vs. late in the video).

Multi-Behavior Sequential Recommendation with Temporal Graph Transformer - 2022:

Architecture

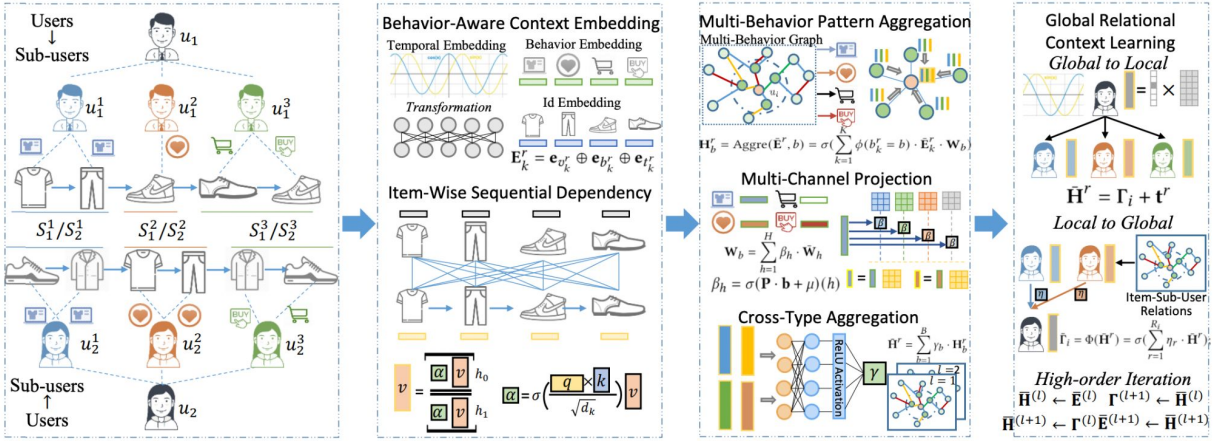


Figure 2: Model flow of the proposed TGT. (a) Hierarchical graph structure between user and item with respect to both short-term and long-term multi-behavior relations. (b) Dynamic individual interest modeling with behavior-aware sequential context. (c) Multi-behavior pattern aggregation which jointly preserves behavior semantics and cross-type behavior dependency. (d) Global relational context injection with the high-order embedding propagation paradigm.

Algorithm 1: Model Inference of TGT Framework

Input: user-item interaction sequences S_i, sub-user size $|S_i^r|$, graph iterations L, target behavior B, sample number C, maximum epoch number E, regularization weight λ, learning rate ρ

Output: trained parameters in Θ

Initialize all parameters in Θ
Divide S_i into sub-sequences S_i^r of size $|S_i^r|$
for $e = 1$ *to* E **do**
  Generate behavior-aware context embeddings $\mathbf{E}_k^r$
  Capture item-wise sequential dependency to yield $\bar{\mathbf{E}}_k^r$
  for $l = 1$ *to* L **do**
    Compute sub-users' behavior embeddings $\mathbf{H}_b^r$
    Conduct cross-type aggregation to get $\bar{\mathbf{H}}^r$
    Aggregate for the global user embedding $\bar{\mathbf{\Gamma}}_i$
    Refine the sub-user embeddings $\bar{\mathbf{H}}^r$ using $\bar{\mathbf{\Gamma}}_i$
    Update the item embeddings $\bar{\mathbf{E}}_j$ using $\bar{\mathbf{H}}^r$
  end
  Integrate the cross-order graph embeddings $\tilde{\mathbf{\Gamma}}_i, \tilde{\mathbf{H}}^r, \tilde{\mathbf{E}}_j$
  Draw a mini-batch $\mathbf{U}$ from all users, each with C positive-negative samples
  $\mathcal{L} = \lambda \cdot \|\mathbf{\Theta}\|_F^2$
  for *each* $u_i \in \boldsymbol{U}$ **do**
    Compute predictions $\Pr_{i,p_c}, \Pr_{i,n_c}$
    $\mathcal{L} += \sum_{c=1}^{C} \max(0, 1 - \Pr_{i,p_c} + \Pr_{i,n_c})$
  end
  for *each parameter* $\theta \in \mathbf{\Theta}$ **do**
    $\theta = \theta - \rho \cdot \partial\mathcal{L}/\partial\theta$
  end
end
return all parameters Θ

Multi-Behavior Sequential Recommendation with Temporal Graph Transformer - 2022:

Dataset & Results

Datasets: Taobao-Data, IJCAI-Contest Data; Eval metrics: NDCG@N, Recall@N.

Table 2: Statistics of experimented datasets

Dataset	# of Users	# of Items	# of Interactions
Taobao	147894	99037	7658926
IJCAI	423423	874328	36203512

Methods	Taobao Data										IJCAI Contest									
	Top @ 1		Top @ 5				Top @ 10				Top @ 1		Top @ 5				Top @ 10			
	HR	Imp	HR	Imp	NDCG	Imp	HR	Imp	NDCG	Imp	HR	Imp	HR	Imp	NDCG	Imp	HR	Imp	NDCG	Imp
BPR	0.048	139%	0.209	39%	0.143	26%	0.295	53%	0.179	47%	0.073	102%	0.184	117%	0.130	125%	0.257	102%	0.153	116%
NCF	0.061	89%	0.231	26%	0.157	15%	0.325	39%	0.201	31%	0.139	6%	0.351	14%	0.255	15%	0.459	13%	0.294	12%
DeepFM	0.059	95%	0.223	30%	0.142	27%	0.328	38%	0.175	50%	0.138	7%	0.332	20%	0.244	20%	0.469	11%	0.290	14%
GRURec	0.081	42%	0.237	22%	0.161	12%	0.328	38%	0.187	41%	0.107	38%	0.284	41%	0.194	51%	0.418	24%	0.247	34%
Caser	0.094	22%	0.239	21%	0.168	7%	0.364	24%	0.197	34%	0.129	14%	0.295	35%	0.208	41%	0.422	23%	0.252	31%
NARM	0.085	35%	0.220	32%	0.151	19%	0.320	41%	0.181	45%	0.107	38%	0.289	38%	0.200	47%	0.396	31%	0.229	44%
SASRec	0.089	29%	0.250	16%	0.169	7%	0.353	28%	0.203	30%	0.114	29%	0.306	30%	0.216	36%	0.420	24%	0.247	34%
Bert4Rec	0.113	2%	0.265	9%	0.174	3%	0.369	22.5%	0.229	15%	0.141	4%	0.356	12%	0.261	12%	0.467	11%	0.297	11%
Chorus	0.100	15%	0.235	23%	0.171	5%	0.388	17%	0.232	13%	0.140	5%	0.345	15.7%	0.247	19%	0.457	14%	0.283	17%
HyRec	0.088	31%	0.228	27%	0.161	12%	0.324	40%	0.191	38%	0.137	8%	0.323	24%	0.229	28%	0.442	17%	0.266	24%
MAGNN	0.108	6%	0.235	23%	0.174	3%	0.320	41%	0.199	32%	0.127	16%	0.291	37%	0.212	38%	0.392	32%	0.245	35%
SR-GNN	0.086	34%	0.240	21%	0.162	11%	0.332	36%	0.191	38%	0.133	11%	0.316	26%	0.214	37%	0.431	20%	0.263	26%
GCGNN	0.087	32%	0.250	16%	0.168	7%	0.349	30%	0.201	31%	0.136	8%	0.289	38%	0.210	40%	0.428	21%	0.257	28%
MGNN	0.096	20%	0.259	12%	0.172	5%	0.374	21%	0.219	20%	0.138	7%	0.244	64%	0.248	18%	0.457	14%	0.286	15%
HGT	0.101	14%	0.261	11%	0.174	3%	0.364	24%	0.216	22%	0.128	15%	0.309	29%	0.207	41.5%	0.450	15%	0.265	25%
NMTR	0.079	46%	0.218	33%	0.147	22%	0.332	36%	0.179	47%	0.141	4%	0.360	10.8%	0.254	15%	0.481	8%	0.304	9%
MATN	0.081	42%	0.226	29%	0.153	17%	0.354	28%	0.209	26%	0.142	4%	0.375	6%	0.273	7%	0.489	6%	0.309	7%
MBGCN	0.110	5%	0.259	12%	0.172	5%	0.369	23%	0.222	19%	0.137	8%	0.332	20%	0.228	31%	0.463	12%	0.277	19%
TGT	**0.115**	–	**0.290**	–	**0.180**	–	**0.452**	–	**0.263**	–	**0.148**		**0.399**	–	**0.293**	–	**0.519**	–	**0.330**	–

Learn more about the comparison methods on page 8 (didn't fit, a lot of data)

Multi-Behavior Sequential Recommendation with Temporal Graph Transformer - 2022:

Limitations & Ideas to improve

Limitations:

- **Overfitting in Multi-Channel Projection:** The model includes a multi-channel projection layer to handle multi-behavior pattern aggregation, which may lead to overfitting when using a high number of projection channels. The model performance reportedly declines when the number of channels is increased beyond a certain point.
- **Static Item and Positional Embeddings:** The authors mention that item embeddings are initially static (based on unique item IDs) and describe an approach using positional encoding inspired by sinusoidal functions. This static embedding approach may limit the model's ability to adapt to evolving item relevance or new item popularity trends over time.

Ideas to improve:

- **Introducing regularization techniques** such as dropout within the multi-channel projection layer could reduce overfitting.
- **Using dynamically updated embeddings** for items could help the model remain responsive to recent changes in user interactions. Techniques like online learning or adaptive embedding updates based on recent interactions could help the model capture evolving preferences more accurately.

KGUF: Simple Knowledge-aware Graph-based Recommender with User-based Semantic Features Filtering - 2024: Architecture

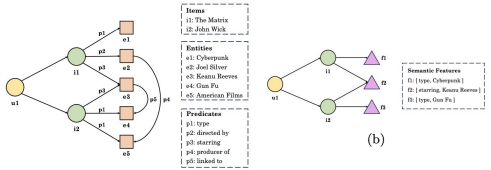


Fig. 1: An example of how KGUF models the item representation. Figure 1a shows the entities in the knowledge graph linked to the items. In figure 1b, the most relevant semantic features selected by KGUF are reported.

A knowledge graph stores structured information of real-world facts in the form of a heterogeneous graph which can be used to retrieve item attributes [6, 13]. Formally, it is presented as a set of triples in the form $\sigma \xrightarrow{\rho} \omega$. Each triple indicates that the head entity σ is connected to the tail entity ω through the relation ρ. Given a set of entities $\mathcal{V}$ and a set of relations $\mathcal{R}$, a knowledge graph $\mathcal{KG}$ can be represented as follows:

$$\mathcal{KG} = \{\sigma \xrightarrow{\rho} \omega \mid \sigma, \omega \in \mathcal{V}, \rho \in \mathcal{R}\}. \quad (1)$$

Mapping items in the catalog using knowledge graphs, i.e., $\mathcal{I} \subseteq \mathcal{V}$, enables the descriptions of items based on semantic features.

Formally, let $\mathcal{I}_u^+ \subseteq \mathcal{I}$ denote the set of items positively rated by each user u in the set $\mathcal{U}$ of our recommendation problem, and let $\mathcal{I}_u^- = \mathcal{I} \setminus \mathcal{I}_u^+$ be the set of items not enjoyed by user u. Let $\mathcal{J}_u^- \subseteq \mathcal{I}_u^-$ be a selection of items randomly sampled from $\mathcal{I}_u^-$, such that $|\mathcal{J}_u^-| = |\mathcal{I}_u^+|$. Finally, $\mathcal{J}_u = \mathcal{I}_u^+ \cup \mathcal{J}_u^-$ is the set of items selected to depict the past behavior of the user.

For each user u, the set of semantic features $\mathcal{F}_u = \bigcup_{i \in \mathcal{J}_u} \mathcal{F}_i$ is used to compute the user decision tree T_u that, according to the definition of Information Gain [3], is able to find the minimum subset of semantic features $\mathcal{F}_u^{T_u} \subseteq \mathcal{F}_u$ that can distinguish whether an item belongs to $\mathcal{I}_u^+$ or $\mathcal{I}_u^-$. Finally, each item i is formally described by the subset of its semantic features that appear in at least one user decision tree:

$$\mathcal{F}_i = \{\langle \rho, \omega \rangle | i \xrightarrow{\rho} \omega \in \mathcal{KG}\}.$$
$$\mathcal{F}_i^* = \mathcal{F}_i \cap \bigcup_{u \in \mathcal{U}} \mathcal{F}_u^{T_u}$$

2

$$\mathbf{e}_u^{(l)} = \sum_{i \in \mathcal{N}(u)} \frac{1}{\sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(i)|}} \mathbf{e}_i^{(l-1)},$$
$$\mathbf{e}_i^{(l)} = \frac{\alpha}{|\mathcal{F}_i^*|} \sum_{f \in \mathcal{F}_i^*} \mathbf{e}_f + (1-\alpha) \sum_{u \in \mathcal{N}(i)} \frac{1}{\sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(i)|}} \mathbf{e}_u^{(l-1)},$$

where α serves as a weight factor for balancing content and collaborative contributions in item representations.

After completing L propagation hops, the user and item embeddings are derived as follows:

$$\mathbf{e}_u = \sum_{l=0}^{L} \frac{1}{1+l} \mathbf{e}_u^{(l)}, \quad \mathbf{e}_i = \sum_{l=0}^{L} \frac{1}{1+l} \mathbf{e}_i^{(l)}. \quad (7)$$

The introduction of the scaling factor $1/(1+l)$ serves to mitigate the oversmoothing problem that may arise with an increased number of explored hops [18].

$$\hat{r}_{ui} = \mathbf{e}_u^\top \mathbf{e}_i.$$

To optimize the model parameters, we utilize the pairwise optimization algorithm Bayesian Personalized Ranking (BPR). This algorithm operates on the assumption that a user u favors a consumed item i^+ over a non-consumed item i^-. It optimizes the model by maximizing, for each pair of i^+ and i^-, a function based on the difference $r_{ui^+} - r_{ui^-}$. Specifically, by constructing a training set $\mathcal{T} = \{(u, i^+, i^-) | (u, i^+) \in \mathcal{R}, (u, i^-) \notin \mathcal{R}, i^- \in \mathcal{I}\}$, BPR aims to optimize the following loss, where σ represents the sigmoid function:

$$\mathcal{L}_{\text{BPR}} = \sum_{(u,i^+,i^-) \in \mathcal{T}} -\ln \sigma(\hat{r}_{ui^+} - \hat{r}_{ui^-}), \quad (9)$$

Finally, we use a joint learning approach to include the L2-regularization loss as follows:

$$\mathcal{L} = \mathcal{L}_{\text{BPR}} + \lambda \|\Theta\|_2^2, \quad (10)$$

here Θ encompass all the learnable parameter, specifically $\Theta = \{\mathbf{e}_u, \mathbf{e}_i, \mathbf{e}_f | u \in \mathcal{U}, i \in \mathcal{I}, f \in \bigcup_{i \in \mathcal{I}} \mathcal{F}_i^*\}$, while λ controls the regularization term.

3

OpenSource: https://github.com/sisinflab/KGUF

1

KGUF: Simple Knowledge-aware Graph-based Recommender with User-based Semantic Features Filtering - 2024: Dataset & Results

Datasets: *MovieLens*, *Yahoo! Movies*, *Facebook Books*;

Table 1: Accuracy performance comparison of KGUF with the selected baselines. All the models are grouped into their respective categories.

	Facebook Books			Yahoo! Movies			MovieLens 1M		
	nDCG	HR	Recall	nDCG	HR	Recall	nDCG	HR	Recall
Random	0.0074	0.0256	0.0110	0.0065	0.0342	0.0090	0.0096	0.0836	0.0036
MostPop	0.0787	0.2535	0.1117	0.1185	0.3238	0.1501	0.1730	0.6537	0.0773
BPR-MF	0.0945	0.2973	0.1358	0.2303	0.5228	0.2843	0.3100	0.8811	0.1627
Item-kNN	0.1039	0.3353	0.1556	0.2258	0.5033	0.2658	0.2932	0.8504	0.1490
KGFlex	0.0712	0.2235	0.1001	0.1758	0.4360	0.2022	0.0111	0.1058	0.0053
kaHFM	0.0843	0.2776	0.1221	0.2085	0.5064	0.2621	0.2992	0.8654	0.1526
LightGCN	0.0969	0.3170	0.1435	0.2230	0.5085	0.2712	0.3081	0.8813	0.1614
DGCF	0.1001	0.3192	0.1444	0.2356	0.5308	0.2848	0.3124	0.8819	0.1637
KGIN	0.1023	0.3112	0.1448	0.2368	0.5500	0.2971	0.3082	0.8854	0.1638
KGTORe	**0.1188**	**0.3601**	**0.1698**	0.2471	0.5566	0.2944	0.3212	0.8957	0.1710
KGUF	0.1156	0.3594	0.1697	**0.2561**	**0.5685**	**0.3058**	**0.3277**	**0.8998**	**0.1755**

5.2 Baselines

We empirically validate the effectiveness of KGUF by comparing its performance in terms of recommendation accuracy with different baselines from five related categories described in the following.

Unpersonalized We test Random and Most Popular recommendations as a basis for discerning significance and behaviors influenced by popularity.

Collaborative Filtering BPR-MF [28] and Item-kNN [22] are two widely recognized collaborative approaches. The former follows a pair-wise learning approach as KGUF, and the latter exploits similarities between items by computing the nearest neighbors.

Knowledge-Aware Among these models, we selected KaHFM [4] and KGFlex [3, 12], since they share with KGUF the feature representation but differ in the way they exploit it. KaHFM uses knowledge for initializing latent factors in factorization machines. KGFlex, instead, models the user-item interaction by means of a sparse collection of the most relevant semantic features.

Graph-Based LightGCN [18] rethinks the classic GCN aggregation schema by discarding feature transformation and non-linear activation functions. It results in linear aggregations with improved performance. DGFC [41] introduces a graph disentangling strategy for intercepting hidden user intents to model intent-dependent user-item interactions.

Graph-based with Knowledge KGIN [40] exploits the KG for computing users' intents as a weighted combination of semantic relations. User-item connections are interpreted by means of a limited set of relevant intents. KGTORe [25] mines the user motivations by looking at the discriminant semantic features in her history. These explicit features are then used to enrich the user and the item node representation.

(a)

Configuration	nDCG@10
$KGUF_{\alpha=.2}$	0.25255
$KGUF_{\alpha=.4}$	**0.25640**
$KGUF_{\alpha=.6}$	0.25233
$KGUF_{\alpha=.8}$	0.25322

(b)

Setting	nDCG@10
$KGUF_{\text{w / o\_knowledge}}$	0.23031
$KGUF_{\text{w / o\_collaborative}}$	0.25074
$KGUF_{\alpha=.4}$	**0.25640**

KGUF: Simple Knowledge-aware Graph-based Recommender with User-based Semantic Features Filtering - 2024: Limitations & Ideas to improve

Limitations:

- **Over-reliance on Decision Trees for Feature Selection Evidence:** This approach can lead to computational inefficiency and may overfit specific features. Furthermore, this feature selection may not adapt well to new data due to the static nature of decision trees.
- **Scalability Concerns with Increased Knowledge Graph Complexity:** KGUF uses recursive propagation across multi-hop neighbors, integrating various semantic features in item embeddings. This increases computational costs and may hinder scalability with larger KGs or datasets. Lead to excessive computation, especially in real-time recommendation scenarios.
- **Potential for Oversmoothing in Multi-hop Propagation:** KGUF mitigates oversmoothing in multi-hop aggregations with a layer-wise scaling factor, but this solution may not fully prevent oversmoothing, especially in high-depth graphs, where deeper propagation can blur distinctions between node embeddings.

Ideas to improve:

- **Consider replacing decision trees with a more adaptive filtering mechanism**, such as reinforcement learning-based feature selection, allowing the model to dynamically adjust its feature filtering as it learns.
- **Explore the use of graph sampling techniques:** use GraphSAGE or mini-batching, which can limit propagation to high-impact nodes, reducing computational demands without sacrificing embedding quality.
- **Apply gating mechanisms on deeper layers** to selectively retain meaningful distinctions in representations across propagation layers.

Explicit Behavior Interaction with Heterogeneous Graph for Multi-behavior Recommendation - 2024: Architecture

Fig. 2 Overview of MB-EBIH model. *(user, behavior) and (item, behavior)* denote the explicit behavior interactions. $\|$ is the concatenate operation. $\otimes$ is the inner product operation

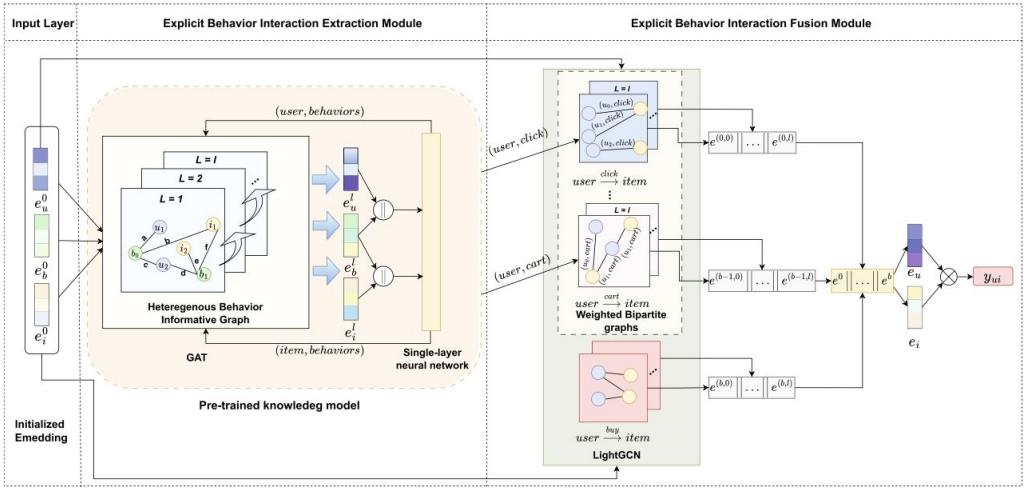
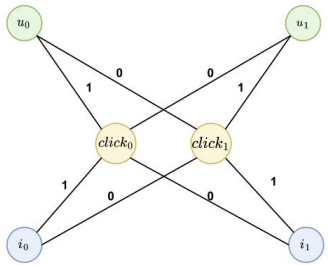


Fig. 3 Process of constructing heterogeneous graph, $click_0$ and $click_1$ denotes the negative and the positive feedback signals of *click*

User & Item ID		Behavior	Label
user_id	item_id	click	buy
u_0	i_0	1	0
u_0	i_0	0	1
u_1	i_1	1	1
u_1	i_1	0	0

Multi-Behavior history data



Constructed Heterogeneous Graph

Explicit Behavior Interaction with Heterogeneous Graph for Multi-behavior Recommendation - 2024: Dataset & Results

Datasets: Beibei, Tmall, IJCAI15, QK-article(github);

Table 1 Statistics of the datasets used in our experiments

Dataset	User#	Item#	Interaction#	Behavior type#
Beibei	21,716	7,977	3.3×10^6	{Click,Cart,Buy}
Tmall	41,738	11,953	2.3×10^6	{Click,Cart,Collect,Buy}
IJCAI15	55,038	28,728	7.5×10^6	{Click,Cart,Collect,Buy}
QK-article	40,343	19,218	2.4×10^6	{Click,Share,Follow,Like}

Table 2 Overall performance comparison between MB-EBIH and baseline models on four datasets

Dataset	Metric	Single-behavior methods			Multi-behavior methods							MB-EBIH
		MF-BPR	NGCF	LightGCN	NMTR	MBGCN	MATN	MB-GMN	GNMR	CRGCN	MB-CGCN	
Beibei	Recall@10	0.0868	0.0901	0.0925	0.0937	0.1249	0.0942	0.1040	0.1105	0.1215	0.1539	**0.1911**
	NDCG@10	0.0414	0.0521	0.0538	0.0571	0.0745	0.0579	0.0586	0.0635	0.0986	0.1095	**0.1235**
	Recall@20	0.1244	0.1463	0.1517	0.1559	0.1925	0.1604	0.1608	0.1635	0.2173	0.2362	**0.2807**
	NDCG@20	0.0641	0.0652	0.0698	0.0796	0.0937	0.0790	0.0740	0.0794	0.1330	0.1547	**0.1674**
	Recall@40	0.1965	0.2311	0.2328	0.2644	0.2938	0.2610	0.2863	0.2872	0.2778	0.3388	**0.3694**
	NDCG@40	0.0817	0.0853	0.0887	0.0995	0.1157	0.1071	0.1078	0.1096	0.1585	0.1910	**0.1991**
Tmall	Recall@10	0.0263	0.0391	0.0417	0.0470	0.0692	0.0562	0.0893	0.0601	0.0901	0.0925	**0.0985**
	NDCG@10	0.0151	0.0225	0.0291	0.0293	0.0424	0.0367	0.0462	0.0388	0.0568	0.0594	**0.0644**
	Recall@20	0.0355	0.0491	0.0553	0.0672	0.0947	0.0790	0.1072	0.0838	0.1136	0.1231	**0.1280**
	NDCG@20	0.0176	0.0356	0.0333	0.0360	0.0493	0.0408	0.0592	0.0452	0.0631	0.0696	**0.0732**
	Recall@40	0.0481	0.0716	0.0719	0.0982	0.1275	0.1042	0.1345	0.1190	0.1543	0.1594	**0.1664**
	NDCG@40	0.0200	0.0350	0.0361	0.0412	0.0558	0.0517	0.0674	0.0529	0.0749	0.0778	**0.0803**
IJCAI15	Recall@10	0.0217	0.0281	0.0286	0.0314	0.0468	0.0350	0.0558	0.0417	0.0562	0.0701	**0.0718**
	NDCG@10	0.0116	0.0153	0.0180	0.0211	0.0304	0.0217	0.0376	0.0272	0.0351	0.0428	**0.0500**
	Recall@20	0.0311	0.0351	0.0387	0.0461	0.0661	0.0412	0.0750	0.0600	0.0884	0.0891	**0.0939**
	NDCG@20	0.0191	0.0198	0.0211	0.0263	0.0358	0.0242	0.0432	0.0393	0.0412	0.0504	**0.0563**
	Recall@40	0.0489	0.0501	0.0519	0.0707	0.0906	0.0714	0.0960	0.0779	0.0924	0.1135	**0.1221**
	NDCG@40	0.0225	0.0231	0.0240	0.0309	0.0415	0.0316	0.0518	0.0327	0.0510	0.0598	**0.0629**
QK-article	Recall@10	0.0615	0.0641	0.0692	0.0704	0.0711	0.0708	0.0946	0.0860	0.1173	0.1359	**0.1509**
	NDCG@10	0.0333	0.0349	0.0357	0.0366	0.0384	0.0380	0.0495	0.0488	0.0691	0.0796	**0.0892**
	Recall@20	0.1021	0.1063	0.1064	0.1107	0.1178	0.1122	0.1402	0.1366	0.1767	0.2005	**0.2205**
	NDCG@20	0.0445	0.0465	0.0493	0.0502	0.0510	0.0506	0.0622	0.0581	0.0748	0.0934	**0.1084**
	Recall@40	0.1673	0.1745	0.1780	0.1821	0.1844	0.1838	0.2202	0.2084	0.2534	0.2846	**0.3096**
	NDCG@40	0.0589	0.0613	0.0625	0.0650	0.0662	0.0658	0.0834	0.0790	0.0907	0.1121	**0.1282**

Table 3 Effect of GAT in MB-EBIH (results based on $K = 10$)

Dataset	Beibei		Tmall		IJCAI15		QK-article	
Method	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
GraphSage	0.1458	0.0867	0.0722	0.0419	0.0619	0.0415	0.1329	0.0765
k-GNNs	0.1838	0.1218	0.0782	0.0469	0.0662	0.0440	0.1292	0.0743
LEConv	0.1866	0.1221	0.0796	0.0482	0.0643	0.0419	0.1387	0.0805
GAT	**0.1911**	**0.1235**	**0.0985**	**0.0644**	**0.0718**	**0.0500**	**0.1509**	**0.0892**

Best values are highlighted in bold

Table 4 Effects of different explicit behavior interactions (results based on $K = 10$)

Dataset	Beibei		Tmall		IJCAI15		QK-article	
Variants	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
Base model	**0.1911**	**0.1235**	**0.0985**	**0.0644**	**0.0718**	**0.0500**	**0.1509**	**0.0892**
w/o. click	0.1899	0.1218	0.0703	0.0432	0.0479	0.0310	0.0757	0.0412
w/o. cart(follow)	0.1229	0.0706	0.0951	0.0593	0.0706	0.0485	0.1447	0.0843
w/o. collect(share)	/	/	0.0899	0.0550	0.0712	0.0492	0.1488	0.0862
w/o. cart,click	0.1170	0.0672	0.0674	0.0419	0.0503	0.0329	/	
w/o. cart,collect	/		0.0880	0.0537	0.0596	0.0386		
w/o. collect,click			0.0677	0.0411	0.0467	0.0295		
w/o. follow,click			/				0.0747	0.0406
w/o. follow,share							0.1436	0.0826
w/o. share,click							0.0707	0.0378
w/o. collect,click,cart	/		0.0631	0.0396	0.0413	0.0257	/	
w/o. share,click,follow			/				0.0699	0.0376

Explicit Behavior Interaction with Heterogeneous Graph for Multi-behavior Recommendation - 2024: Limitations & Ideas to improve

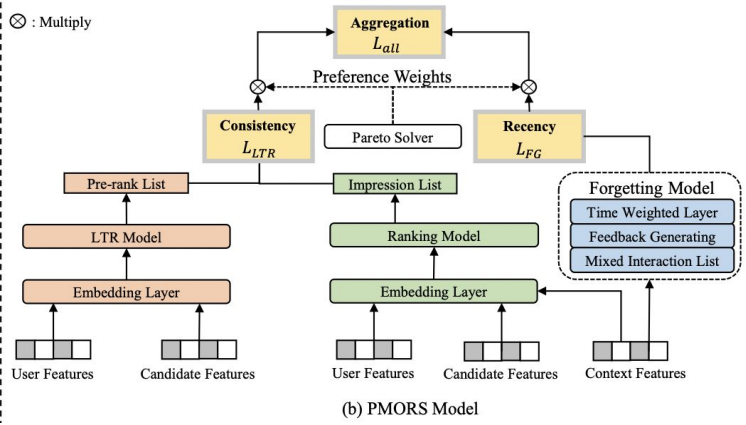
Limitations:

- **Sensitivity to Hyperparameters:** The authors conducted sensitivity analysis on parameters, such as node dimensions and GAT layers, indicating that model performance is sensitive to these settings. For example, changes in GAT layers affected accuracy due to over-smoothing with more layers, which could lead to overfitting or underfitting in varied datasets.
- **Temporal Dynamics Exclusion:** The model doesn't address the evolving nature of user behaviors over time, which limits its applicability in dynamic environments where user preferences frequently change.

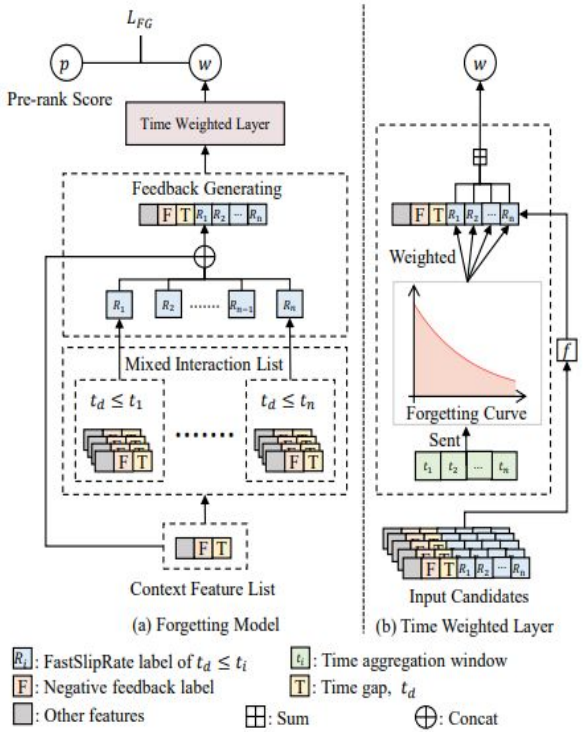
Ideas to improve:

- **Automated Hyperparameter Tuning:** To address parameter sensitivity, employing automated tuning (e.g., AutoML) could help optimize dimensions and layer settings for diverse datasets.
- **Integrating Temporal Components:** Adding a time-aware module or recurrent layers could better capture evolving user behavior, enhancing recommendation relevance for applications sensitive to recent interactions

Pareto-based Multi-Objective Recommender System with Forgetting Curve: Limitations & Ideas to improve - 2023: Architecture



(b) PMORS Model



1) Pareto Dominance & Pareto Optimality:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{t} \alpha_i \mathcal{L}_i(\boldsymbol{\theta})$$

2) Karush-Kuhn-Tucker (KKT) conditions

$$(1)\ \sum_{i=1}^{t} \alpha_i \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\boldsymbol{\theta}) = 0$$

$$(2)\ \sum_{i=1}^{t} \alpha_i = 1,\ \alpha_i \ge 0, \text{for } i = 1, \ldots, t.$$

3) Ebbinghaus Forgetting Curve

$$R(t) = e^{-\frac{t}{S}},$$

Loss Pareto:

$$\min_{\alpha_1,\ldots,\alpha_t} \left\| \sum_{i=1}^{t} \alpha_i \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \right\|_2^2,$$

$$s.t., \sum_{i=1}^{t} \alpha_i = 1, \alpha_i \ge 0, for\ i = 1, \ldots, t.$$

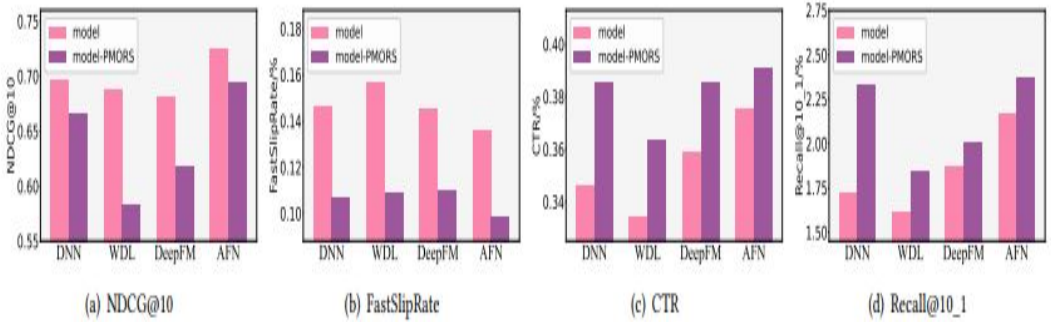
Pareto-based Multi-Objective Recommender System with Forgetting Curve: Limitations & Ideas to improve - 2023: Dataset & Results

Datasets: KuaiRec dataset; Eval metrics: NDCG@10,Recall@10_1, CTR and FastSlipRate as metrics for PMORS

Offline results

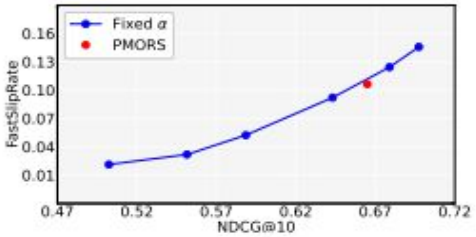
Table 1: Overall Performance Results on KuaiRec Dataset. Impr. indicates the relative improvement compared to RankNet, results are statistically significant with $p < 0.05$.

Model	NDCG@10	Impr.	FastSlipRate	Impr.	CTR	Impr.	Recall@10_1	Impr.
RankNet	0.6974	-	0.1458%	-	0.3458%	-	1.7293%	-
LambdaMART	**0.7067**	**1.32%**	0.1662%	14.01%	0.3686%	6.59%	1.9440%	12.42%
ESMM	0.5107	-26.77%	**0.0218%**	**-85.01%**	0.2932%	-15.22%	1.2962%	-25.05%
MMOE	0.5515	-20.92%	0.0710%	-51.33%	0.2987%	-13.63%	1.4986%	-13.34%
PO-EA	0.6131	-12.09%	0.1024%	-29.78%	0.3804%	9.99%	2.1470%	24.16%
PE-LTR	0.6507	-6.70%	0.1021%	-29.95%	0.3831%	10.77%	2.2530%	30.28%
CL	0.6907	-0.96%	0.1554%	6.61%	0.3582%	3.57%	2.0632%	19.31%
PMORS	0.6650	-4.65%	0.1064%	-26.99%	**0.3851%**	**11.35%**	**2.3386%**	**35.24%**



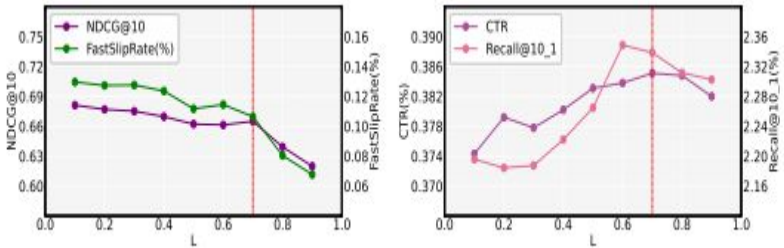
(a) NDCG@10 (b) FastSlipRate (c) CTR (d) Recall@10_1

Online results



Consistency and Recency in Ablation Study

Evaluation Metrics with Different L



(a) NDCG,FastSlipRate (b) CTR,Recall@10_1

Table 3: Online Deployment Results.

GMV	FastSlipRate	FastSlipAction	CTR	Recall@10_1
+1.45%	-0.63%	-0.50%	+0.89%	+3.0%

Pareto-based Multi-Objective Recommender System with Forgetting Curve: Limitations & Ideas to improve - 2023

Limitations:

Static Forgetting Curve Leading to Suboptimal Temporal Adaptation: The use of a static Ebbinghaus forgetting curve function, constrains the model's ability to dynamically adapt to rapid changes in user preferences over time. This can result in temporal lag in recommendation relevance, as the forgetting rate S remains constant, failing to account for variations in user behavior that could require faster or slower decay.

High Computational Complexity in Multi-Objective Pareto Optimization:The Pareto-based multi-objective optimization framework with constraints introduces computational burdens, especially for large-scale datasets. The necessity to satisfy Karush-Kuhn-Tucker (KKT) conditions in each iteration increases computational costs, potentially limiting the model's scalability in high-dimensional recommendation environments.

Vulnerability to Noise in Feedback Metrics: The model's reliance on feedback-based metrics can amplify the impact of noisy or erroneous interactions. High sensitivity to such metrics may lead to instability in the model's outputs, especially when feedback data contains outliers or mislabels, which can distort recommendations.

Ideas to improve:

Introduce an adaptive mechanism for the forgetting rate: allowing S to vary based on recent user interactions. One approach is to use a reinforcement learning framework where S is adjusted dynamically to minimize a reward function that tracks recommendation relevance

Consider incorporating stochastic gradient descent (SGD) or mini-batch gradient descent in the Pareto optimization process. Instead of computing full gradients across all objectives at each step, approximate gradients could be computed over sampled subsets, preserving optimization accuracy while reducing computation time

Apply a regularization term to the FastSlipRate to penalize extreme feedback fluctuations, improving stability in the presence of noisy data. Additionally, integrate noise filtering methods, such as anomaly detection algorithms, to pre-process feedback data and reduce the influence of outliers or mislabels on feedback metrics

Multi-behavior Session-based Recommendation via Graph Reinforcement Learning - 2023:

Architecture (combining GNN and RL)

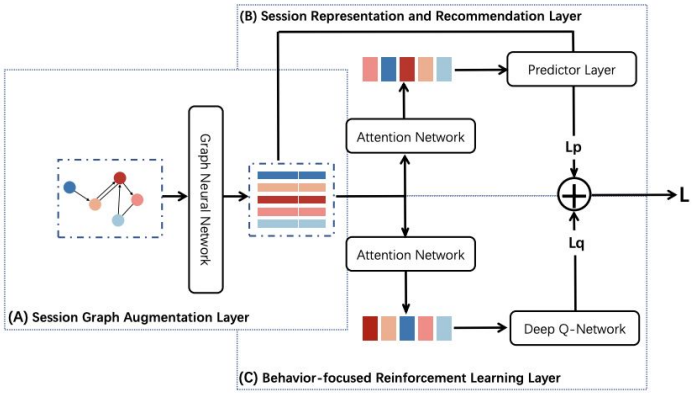


Figure 1: The model architecture of MB-GRl. (A) integrates the transition patterns of all items within a session along with the activated contextual information into the presentation of each item. (B) extract session representation that includes both the long-term preferences and current interests and make recommendation. (C) acts as a regularizer to fine-tune the recommendation performance.

Algorithm 1 Training procedure of MB-GRL

Input: : Replay buffer B; Target network update rate ϵ
Output: All parameters in learning space Θ
Initialize all trainable parameters
target network parameters $\theta' \leftarrow$ policy network parameters θ
repeat
Draw a mini-batch of $(s_t, a_t, r_t, d, s_{t+1}) \in B$
Get item embedding according to Eq.[2-3]
Get session representation $\mathbf{h}_s$ according to Eq.[4, 5]
Get state representation $\mathbf{st}_t, \mathbf{st}_{t+1}$ according to Eq.[9, 10]
$\mathbf{a}_{t+1} = \underset{\mathbf{a}_{t+1}}{agrmax}\, Q_\theta(\mathbf{s}_{t+1}, \mathbf{a}_t)$
$l_{RL} = (r_t + \gamma Q_{\theta'}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t))^2$
Calculate loss l according to [8,16]
Perform updates by $\nabla_\Theta L$
Update the target network parameters: $\theta'_q = \epsilon\theta_q + (1-\epsilon)\theta'_q$
until converge

- GNN encodes information about the transitions of elements from the session.
- Attention network gets a view of the session and generates recommendations.
- Deep Q-Network is used as a regularizer to improve recommendations for certain types of behavior.

Multi-behavior Session-based Recommendation via Graph Reinforcement Learning - 2023:

Dataset & Results

Datasets: Yoochoose, RetailRocket;

Table 2: Top-k recommendation performance comparison of different models (k=5, 10, 20) on Yoochoose and RetailRocket. NG is short for NDCG. Boldface denotes the highest score.

Yoochoose	purchase						click					
	HR@5	HR@10	HR@20	NG@5	NG@10	NG@20	HR@5	HR@10	HR@20	NG@5	NG@10	NG@20
GRU4Rec	0.371	0.479	0.560	0.260	0.295	0.316	0.310	0.410	0.495	0.213	0.245	0.267
SR-GNN	**0.534**	**0.641**	**0.729**	**0.384**	**0.419**	**0.441**	**0.432**	**0.555**	**0.653**	**0.300**	**0.340**	**0.365**
SR-GNN	0.534	0.641	0.729	0.384	0.419	0.441	0.432	0.555	0.653	0.300	0.340	0.365
GC-SAN	0.486	0.598	0.689	0.348	0.384	0.407	0.394	0.511	0.608	0.276	0.314	0.339
MBGCN	0.318	0.421	0.496	0.221	0.255	0.274	0.320	0.426	0.516	0.218	0.253	0.275
MGNN-Spred	0.447	0.562	0.656	0.319	0.356	0.380	0.307	0.419	0.519	0.208	0.245	0.270
SASRec-SQN	0.434	0.551	0.644	0.307	0.344	0.367	0.327	0.426	0.507	0.226	0.258	0.279
SASRec-SA2C	0.471	0.576	0.665	0.339	0.373	0.396	0.344	0.440	0.519	0.241	0.272	0.292
CP4Rec	0.424	0.546	0.642	0.294	0.334	0.358	0.315	0.413	0.495	0.217	0.249	0.269
CP4Rec-CDARL	0.445	0.564	0.652	0.311	0.350	0.373	0.328	0.428	0.509	0.226	0.259	0.280
MB-GRL (ours)	0.598	0.685	0.754	0.433	0.461	0.477	0.435	0.558	0.655	0.303	0.343	0.367
95% CI	0.012	0.008	0.003	0.013	0.010	0.007	0.002	0.002	0.001	0.002	0.002	0.002
imporve	11.906%	6.871%	3.454%	12.701%	10.016%	8.176%	0.733%	0.571%	0.325%	0.875%	0.757%	0.607%
RetailRoctet	purchase						click					
	HR@5	HR@10	HR@20	NG@5	NG@10	NG@20	HR@5	HR@10	HR@20	NG@5	NG@10	NG@20
GRU4Rec	0.281	0.312	0.346	0.241	0.251	0.260	0.185	0.218	0.253	0.150	0.161	0.170
SASRec	0.527	0.592	0.634	0.430	0.451	0.462	0.254	0.309	0.357	0.193	0.211	0.223
SR-GNN	0.421	0.454	0.488	0.366	0.376	0.385	0.295	0.319	0.343	0.260	0.268	0.274
GC-SAN	0.472	0.533	0.590	0.391	0.410	0.425	**0.322**	**0.378**	**0.431**	**0.259**	**0.277**	**0.291**
MBGCN	0.093	0.127	0.176	0.065	0.076	0.088	0.058	0.085	0.124	0.039	0.048	0.057
MGNN-Spred	0.240	0.300	0.360	0.176	0.195	0.210	0.185	0.244	0.314	0.133	0.152	0.170
SASRec-SQN	0.568	0.620	0.662	0.462	0.481	0.491	0.276	0.330	0.380	0.210	0.228	0.241
SASRec-SA2C	**0.593**	**0.644**	**0.680**	**0.508**	**0.525**	**0.534**	0.287	0.341	0.389	0.224	0.242	0.254
CP4Rec	0.536	0.601	0.649	0.436	0.456	0.467	0.261	0.316	0.368	0.199	0.217	0.230
CP4Rec-CDARL	0.582	0.639	0.685	0.479	0.498	0.509	0.280	0.338	0.391	0.213	0.231	0.245
MB-GRL (ours)	0.687	0.733	0.767	0.587	0.602	0.611	0.435	0.495	0.550	0.350	0.370	0.384
95% CI	0.016	0.016	0.016	0.019	0.017	0.016	0.012	0.016	0.019	0.008	0.010	0.011
improve	15.87%	13.76%	12.82%	15.64%	14.72%	14.44%	35.01%	30.90%	27.70%	35.28%	33.52%	31.94%

Table 1: Statistics of datasets used in experiments

Statistic	Yoochoose	RetailRocket
No. of items	200,000	195,523
No. of sessions	26,702	70,852
Avg. of session length	5.75	6.88
No. of click	1,110,965	1,176,680
No. of purchase	43,946	57,269

Table 3: Top-k recommendation performance comparison of different models (k=5, 10, 20) on Yoochoose. Boldface denotes the highest score.

models	purchase						click					
	hr@5	hr@10	hr@20	ndcg@5	ndcg@10	ndcg@20	hr@5	hr@10	hr@20	ndcg@5	ndcg@10	ndcg@20
TD	0.562	0.666	0.745	0.404	0.438	0.458	**0.438**	**0.560**	**0.657**	**0.305**	**0.345**	**0.369**
OA	**0.570**	**0.668**	**0.746**	**0.410**	**0.442**	**0.462**	0.437	0.559	0.657	0.305	0.344	0.368
SP	0.388	0.517	0.612	0.255	0.296	0.321	0.266	0.385	0.499	0.170	0.208	0.237
MB-GRL	0.606	0.691	0.755	0.433	0.470	0.484	0.435	0.558	0.655	0.302	0.342	0.367

Multi-behavior Session-based Recommendation via Graph Reinforcement Learning - 2023:

Limitations & Ideas to improve

Limitations:

- **Limitation: Absence of Negative Reward in Behavior Regularization:** “The absence of negative rewards does not affect the model because the behavior-focused reinforcement learning layer is trained on positive actions”. Here, the authors describe how their model only incorporates positive rewards, omitting penalties for non-ideal outcomes.
- **A session-based approach without taking into account inter-session interactions:** "Session-based recommendation (SBR) is an approach to recommendation systems that considers a user's previous interactions within a session to generate personalized recommendations." The article describes that the model is based on session data, focusing on user actions within a single session. This approach limits the model's ability to account for interactions that occurred outside of a single session. Therefore, she does not see the full picture of the user's preferences, which are formed on the basis of all his sessions. When modeling recommendations for individual sessions, the model may ignore long-term preferences or repetitive actions specific to the user in different sessions.

Ideas to improve:

- **Introducing a penalty system** could help the model avoid non-beneficial recommendations by incorporating both positive and negative feedback signals.
- **To improve the model, a mechanism could be implemented that takes into account inter-sessional communications**. For example, using long-term memory (LSTM) or recurrent neural networks (RNN) to save the history of user interactions between sessions can help to take into account long-term preferences. An alternative would be to use User-based Graph Neural Networks (GNN), where graphs are built not only based on a single session, but also span multiple user sessions to reflect a broader picture of preferences.

A Metric Learning Perspective on the Implicit Feedback-Based Recommendation Data Imbalance Problem - 2024: Architecture

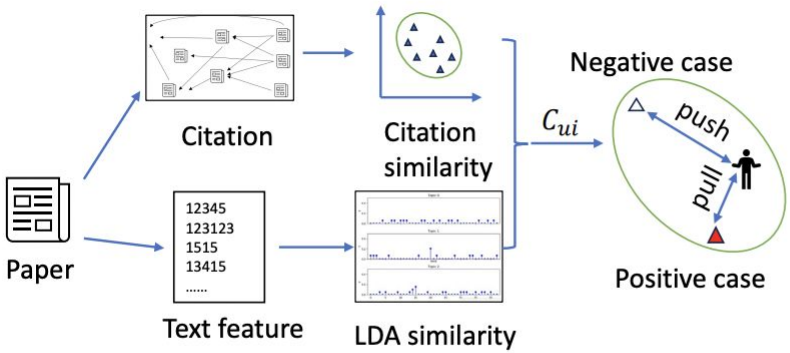


Figure 1. Context-aware paper recommendation by metric learning based on push and pull operations. Context factors were used to speed up the pull operation for positive samples to accelerate convergence. For negative samples, unbiased global sampling was utilized.

$$min_{u^*,v^*}\ L_{push}\ (s) + \omega\ L_{pull}\ (s)$$

$$s.t.\ \ \|p_u^*\| \leq 1\ and\ \|q_i^*\| \leq 1$$

$$L_{pull}\ (s) = \sum_{(u,i\in R)} \|1 - C_{ui} {p_u}^T q_i\|_2^2$$

where $\|C_{ui}\|_2^2 \leq 1$; the more relevant the content is, the closer $\|C_{ui}\|_2^2$ is to 1.

$$L_{push}\ (s) = \sum_{(u,i\notin R)} \|0 - {p_u}^T q_i\|_2^2.$$

Algorithm 1 Context-aware element-wise alternating least squares algorithm

Input: $\{r_{ui}\}$: interaction matrix; $\{\omega_{ij}\}$: weight matrix; $D(u_i, v_j)$: user's text preference matrix for papers; $N(u_i, v_j)$: user citation preference matrix for papers; K_d: latent vector dimension

Output: optimal $\{p\}$: latent vector matrix for users; $\{q\}$: latent vector matrix for papers

Randomly initialize $\{p\},\{q\}$;
repeat
  compute cache matrix for $\{q\} X^q = \sum_{i=1}^{|I|} q_{if} q_{ik}$;
  For u from 1:|U| Do
    For f from 1:K Do
      $p_{uf} \leftarrow (10)$
    END
  END
  compute cache matrix for $\{p\} X^p = \sum_{u=1}^{|U|} p_{uf} p_{uk}$;
  For u from 1:|I| Do
    For f from 1:K Do
      $q_{if} \leftarrow (13)$
    END
  END
until CONVERGE

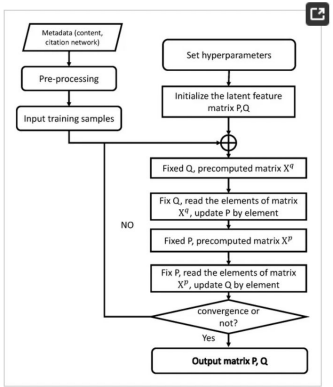


Figure 2. The image shows the optimization of context-aware element-wise alternating least squares, where we use the cache matrix to improve computational efficiency.

A Metric Learning Perspective on the Implicit Feedback-Based Recommendation Data Imbalance Problem - 2024: Dataset & Results

Datasets: citeUlike;

Table 4. Performance comparison on citeUlike-a.

	ND@5	ND@10	re@5	re@10
userKNN	0.0235	0.0326	0.0191	0.0429
itemKNN	0.0106	0.0138	0.0067	0.0151
PMF	0.0168	0.0200	0.0151	0.0228
BPR	0.0196	0.0252	0.0199	0.0333
ALS	0.0593	0.0714	0.0408	0.0824
eALS	0.0583	0.0702	**0.0451**	0.0765
CDR	0.0364	0.0442	0.0262	0.0497
CTR	0.0545	0.0637	0.0377	0.0710
ours	**0.0612**	**0.0744**	0.0439	**0.0873**

Superior results are bolded.

Table 5. Performance comparison on citeUlike-t.

	ND@5	ND@10	re@5	re@10
userKNN	0.0226	0.0334	0.0287	0.0585
itemKNN	0.0089	0.0157	0.0130	0.0317
PMF	0.0483	0.0509	0.0542	0.0613
BPR	0.0178	0.0232	0.0167	0.0332
ALS	**0.0510**	0.0598	0.0334	0.0696
eALS	0.0507	0.0614	0.0367	0.0728
CDR	0.0324	0.0411	0.0245	0.0476
CTR	0.0497	0.0547	0.0345	0.0695
ours	0.0507	**0.0634**	**0.0388**	**0.0773**

Superior results are bolded.

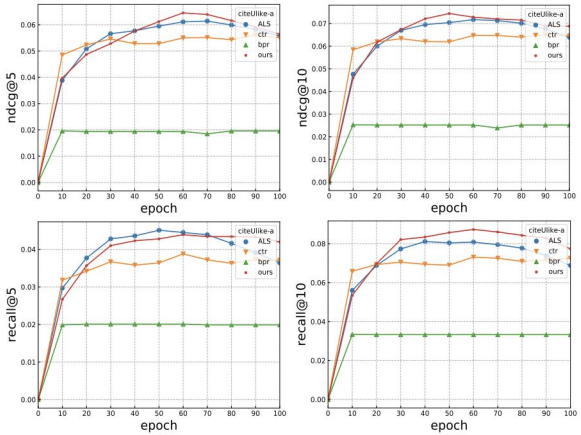


Figure 5. Prediction accuracy of four implicit feedback methods ($K_d = 50$).

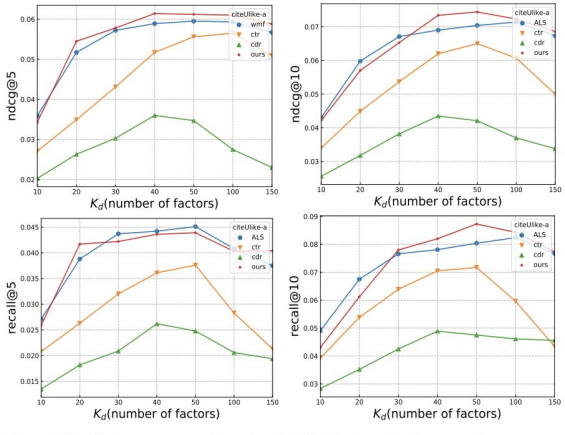


Figure 6. Prediction accuracy of four implicit feedback methods across K_d.

Table 6. Ablation experiment (citeUlike-a).

Metric	eALS	+Topic	+Citation	Our Method
NDCG@5	0.0593	0.0595	0.0604	0.0612
NDCG@10	0.0714	0.0724	0.0738	0.0744
recall@5	0.0408	0.0425	0.0426	0.0439
recall@10	0.0824	0.0824	0.0858	0.0873

A Metric Learning Perspective on the Implicit Feedback-Based Recommendation Data Imbalance Problem - 2024: Limitations & Ideas to improve

Limitations:

- **Random Negative Sampling in Metric Learning:** The approach for negative sampling relies on random sampling and ternary loss, which might select triples with minimal informative content. This random approach can lead to suboptimal results, especially with severely imbalanced datasets (e.g., academic recommendations) where undertrained and sparse data dominate.
- **Loss Function Design:** While the ternary loss function facilitates push-pull dynamics in training, its effectiveness is constrained by unstructured negative sample mining. This limitation may lead to suboptimal differentiation between positive and negative samples, potentially degrading the system's recommendation accuracy.

Ideas to improve:

- Implement a mechanism to prioritize "hard negatives" (samples close to the decision boundary) that provide the most learning signal.
- Use an adaptive sampling strategy that dynamically selects negatives based on model predictions during training, ensuring that the selected negatives contribute meaningfully to learning.
- Group items into clusters using similarity metrics and focus on sampling negatives from clusters near positive samples. This ensures more relevant and challenging negatives are included.

[Negative Can Be Positive: Signed Graph Neural Networks for Recommendation - 2023]: Architecture

Bayesian personalized ranking loss function:

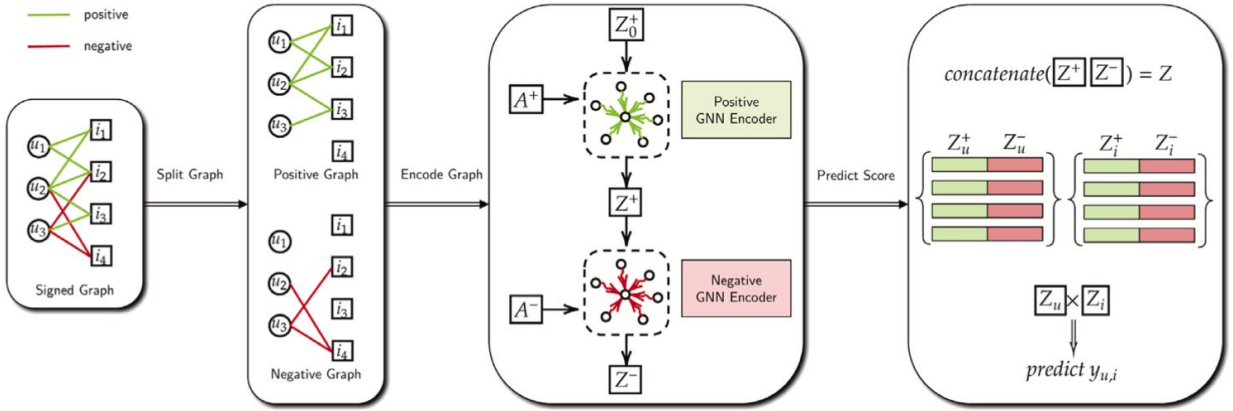
$$\hat{y}_{u,i} = z_u^\top z_i,$$

$$\mathcal{L}_{bpr} = \sum_{(u,i_1,i_2)\in O} -\log\sigma\left(\hat{y}_{u,i_1} - \hat{y}_{u,i_2}\right),$$

$$\mathcal{L}_{sbpr} = \sum_{(u,i_1,i_2)\in O} -\log\sigma\left(s\cdot\hat{y}_{u,i_1} - \hat{y}_{u,i_2}\right), s = \begin{cases} 1, & \text{if } y_{u,i_1} = 1 \\ c, & \text{if } y_{u,i_1} = -1, \end{cases}$$

$$O = \left\{(u,i_1,i_2) \mid \mathcal{A}_{u,i_1} \neq 0, \mathcal{A}_{u,i_2} = 0\right\}$$

the interacted pair (u, i_1) and non-interacted pair (u, i_2).



Sign Cosine loss function:

$$\mathcal{L}_{sign}(z_u, z_i, y_{u,i}) = \begin{cases} 1 - \cos\left(z_u, z_i\right), & \text{if } y_{u,i} = 1 \\ \omega \cdot \max\left(0, \cos\left(z_u, z_i\right) - \mu\right), & \text{if } y_{u,i} = -1, \end{cases}$$

Fig. 4. The demonstration of our SiGRec. SiGRec includes a positive GNN encoder f^+ and a negative GNN encoder f^-. The final embeddings are the concatenation of positive embeddings and negative embeddings.

where margin μ is the number from −1 to +1, ω is the weight of negative samples. μ is a flexible hyperparameter that controls the consistency of the directions of the two representations zu and zi.

Overall objective function:

$$\mathcal{L} = \mathcal{L}_{bpr} + \lambda_{sign}\mathcal{L}_{sign} + \lambda_{reg}\mathcal{L}_{reg},\ \mathcal{L}_{reg} \text{ - L2 reg}, \lambda_{reg} = 1e^{-4}$$

Negative Can Be Positive: Signed Graph Neural Networks for Recommendation - 2023:

Dataset & Results

Datasets: Amazon-Book, Yelp, Zhihu, WeChat;

Table 3
Performance comparison of different models on four datasets. The performances are the average of five experiments from different seed sets. The best method is **bold**, and the second best is underlined.

Dataset	Metric	BPRMF	NeuMF	NGCF	DGCF	LightGCN	SiReN	SiGRec	Improv (%)	Power (%)
Amazon-Book	P@10	0.0427	0.0314	0.0415	0.0496	0.0549	0.0536	**0.0565***	2.77	99.9
	R@10	0.0586	0.0427	0.0560	0.0672	0.0744	0.0741	**0.0764***	2.66	95.6
	nDCG@10	0.0614	0.0423	0.0588	0.0715	0.0797	0.0773	**0.0815***	2.16	86.2
	P@15	0.0380	0.0289	0.0372	0.0437	0.0484	0.0476	**0.0499***	3.27	100
	R@15	0.0773	0.0580	0.0746	0.0876	0.0968	0.0972	**0.0999***	2.84	84.9
	nDCG@15	0.0672	0.0476	0.0647	0.0777	0.0864	0.0846	**0.0885***	2.50	95.9
	P@20	0.0347	0.0269	0.0342	0.0398	0.0438	0.0434	**0.0454***	3.70	100
	R@20	0.0930	0.0712	0.0906	0.1051	0.1158	0.1169	**0.1199***	2.56	89.3
	nDCG@20	0.0726	0.0523	0.0702	0.0836	0.0926	0.0913	**0.0952***	2.76	99.1
Yelp	P@10	0.0287	0.0210	0.0298	0.0335	0.0358	0.0367	**0.0402***	9.57	100
	R@10	0.0449	0.0331	0.0462	0.0524	0.0557	0.0579	**0.0626***	8.03	100
	nDCG@10	0.0427	0.0305	0.0441	0.0502	0.0535	0.0555	**0.0602***	8.49	100
	P@15	0.0261	0.0196	0.0269	0.0302	0.0321	0.0328	**0.0360***	9.75	100
	R@15	0.0610	0.0458	0.0624	0.0703	0.0745	0.0772	**0.0836***	8.39	100
	nDCG@15	0.0482	0.0351	0.0496	0.0562	0.0597	0.0618	**0.0671***	8.50	100
	P@20	0.0242	0.0185	0.0251	0.0278	0.0296	0.0303	**0.0331***	9.36	100
	R@20	0.0749	0.0575	0.0769	0.0859	0.0913	0.0945	**0.1020***	7.88	100
	nDCG@20	0.0531	0.0392	0.0547	0.0617	0.0656	0.0679	**0.0735***	8.23	100
Zhihu	P@10	0.0237	0.0213	0.0276	0.0279	0.0301	0.0368	**0.0392***	6.40	99.8
	R@10	0.0394	0.0356	0.0464	0.0467	0.0506	0.0637	**0.0681***	7.00	100
	nDCG@10	0.0348	0.0315	0.0413	0.0417	0.0451	0.0556	**0.0603***	8.46	100
	P@15	0.0216	0.0196	0.0257	0.0254	0.0276	0.0336	**0.0355***	5.69	99.9
	R@15	0.0536	0.0488	0.0641	0.0633	0.0694	0.0862	**0.0913***	5.95	100
	nDCG@15	0.0402	0.0365	0.0481	0.0479	0.0523	0.0642	**0.0691***	7.60	100
	P@20	0.0203	0.0184	0.0240	0.0236	0.0257	0.0313	**0.0328***	5.09	99.8
	R@20	0.0668	0.0607	0.0798	0.0778	0.0859	0.1067	**0.1115***	4.50	99.8
	nDCG@20	0.0454	0.0412	0.0542	0.0537	0.0588	0.0722	**0.0771***	6.78	99.9
WeChat	P@10	0.0516	0.0469	0.0530	0.0549	0.0562	0.0474	**0.0582***	3.41	79.6
	R@10	0.0882	0.0804	0.0912	0.0943	0.0970	0.0821	**0.1001***	3.20	63.6
	nDCG@10	0.0786	0.0708	0.0808	0.0843	0.0871	0.0716	**0.0890**	2.13	28.9
	P@15	0.0463	0.0429	0.0477	0.0494	0.0507	0.0430	**0.0518**	2.13	51.7
	R@15	0.1184	0.1096	0.1221	0.1271	0.1306	0.1109	**0.1353**	1.85	25.2
	nDCG@15	0.0910	0.0829	0.0936	0.0978	0.1011	0.0836	**0.1044**	1.52	19.3
	P@20	0.0425	0.0397	0.0440	0.0452	0.0467	0.0401	**0.0479**	2.14	72.9
	R@20	0.1440	0.1346	0.1494	0.1536	0.1591	0.1374	**0.1640**	2.17	50.1
	nDCG@20	0.1014	0.0930	0.1046	0.1086	0.1126	0.0943	**0.1159**	1.67	31.6

*Means that p-value ≤ 0.05 for two-tailed paired t-tests.

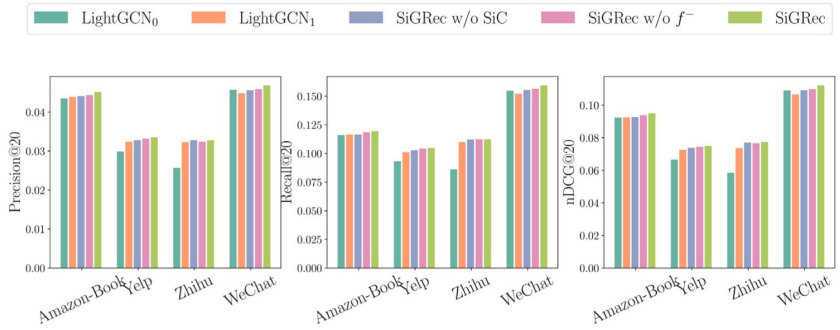


Fig. 5. Performance of SiGRec on four datasets without negative GNN encoder f^- and Sign Cosine (SiC) loss function.

Table 4
The performance of different λ_{sign} and negative weight ω (P@20, R@20, nDCG@20).

Dataset	$\lambda_{sign} = 0.1$	$\lambda_{sign} = 0.1$	$\lambda_{sign} = 1$	$\lambda_{sign} = 10$
Amazon-Book	$\omega = 1$	(0.0441,0.1164,0.0927)	(0.0387,0.1014,0.0782)	(0.0185,0.0469,0.0344)
	$\omega = 300$	(0.0451,0.1196,0.0951)	(0.0372,0.0979,0.0761)	(0.0096,0.0245,0.0187)
	$\omega = 1000$	**(0.0453,0.1207,0.0954)**	(0.0361,0.0949,0.0739)	(0.0094,0.0240,0.0179)
Yelp	$\omega = 1$	**(0.0335,0.1050,0.0751)**	(0.0294,0.0902,0.0633)	(0.0166,0.0500,0.0355)
	$\omega = 300$	(0.0331,0.1037,0.0743)	(0.0272,0.0850,0.0600)	(0.0162,0.0493,0.0354)
	$\omega = 1000$	(0.0332,0.1037,0.0745)	(0.0243,0.0764,0.0534)	(0.0041,0.0111,0.0082)
Zhihu	$\omega = 1$	**(0.0329,0.1132,0.0776)**	(0.0313,0.1066,0.0728)	(0.0246,0.0834,0.0560)
	$\omega = 300$	(0.0326,0.1112,0.0769)	(0.0314,0.1077,0.0732)	(0.0081,0.0250,0.0170)
	$\omega = 1000$	(0.0326,0.1111,0.0765)	(0.0306,0.1050,0.0711)	(0.0088,0.0274,0.0184)
WeChat	$\omega = 1$	(0.0457,0.1558,0.1097)	**(0.0468,0.1598,0.1115)**	(0.0312,0.1064,0.0725)
	$\omega = 300$	(0.0456,0.1555,0.1098)	(0.0438,0.1507,0.1058)	(0.0326,0.1110,0.0741)
	$\omega = 1000$	(0.0460,0.1559,0.1094)	(0.0436,0.1491,0.1045)	(0.0305,0.1031,0.0691)

Negative Can Be Positive: Signed Graph Neural Networks for Recommendation - 2023:

Limitations & Ideas to improve

Limitations:

- **Insufficient Handling of Diverse Negative Feedback Types:** Model differentiates three specific types of negative feedback (low rating, skipping content, clicking dislike), but these are treated within the same general framework without unique adjustments or mechanisms tailored for each type. This design may limit the model's effectiveness in capturing the nuances and contextual significance of each feedback type.
- **Static Graph Assumption Limits Temporal Adaptability:** The model represents feedback through a static signed bipartite graph, limiting its applicability in real-world recommendation systems where user preferences evolve over time. This static approach prevents the model from adapting to changes in user-item interactions dynamically.

Ideas to improve:

- **Expanding Types of Negative Feedback:** Develop a hierarchical feedback classification that assigns various levels of intensity to different types of negative feedback. This would allow the model to weigh negative feedback according to its severity.
- **Modeling Dynamic Feedback in Real-Time:** Incorporate sequential or temporal graph neural network models to account for the evolution of user preferences over time. This approach could capture dynamic changes in feedback, aligning the model more closely with real-world recommendation systems or explore reinforcement learning techniques.

Reinforced Negative Sampling over Knowledge Graph for Recommendation - 2020:

Architecture

github: https://github.com/xiangwang1223/kgpolicy

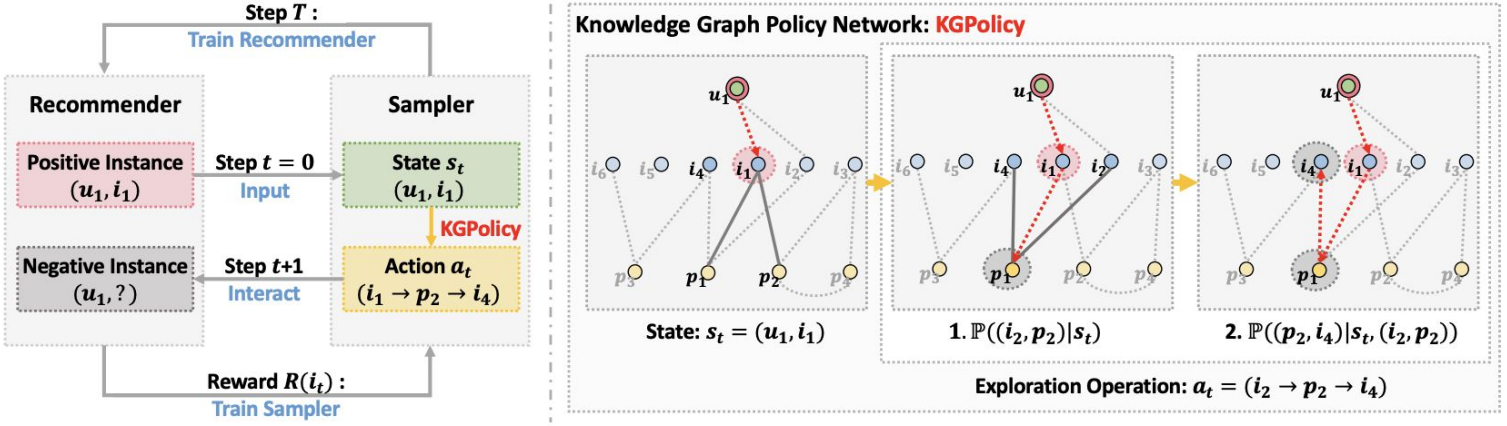


Figure 2: Illustration of the proposed knowledge-aware negative sampling framework. The left subfigure shows the model framework, and the right subfigure presents the proposed KGPolicy network. Best view in color.

$\hat{y}_{ui} = f_R(u, i) = \mathbf{r}_u^\top \mathbf{r}_i$, $\hat{y}_{ui}$ · prediction score for an (u,i) interaction; $f_R(\cdot)$ - abstracted as the interaction function with recommender parameters Θ_R; $\mathbf{r}_u \in \mathbb{R}^d$ and $\mathbf{r}_i \in \mathbb{R}^d$ ID embeddings of user u and item i, respectively; d is the embedding size

$\min_{\Theta_R} \sum_{(u,i) \in O^+} \mathbb{E}_{j \sim f_S(u,i,\mathcal{G})} - \ln \sigma\left(f_R(u,i) - f_R(u,j)\right)$, $\sigma(\cdot)$ is the sigmoid function, $O^+ = \{(u,i)|u \in \mathcal{U}, i \in \mathcal{I}\}$ - where each (u,i) pair indicates a historical interaction between user u and positive item i

$\mathcal{G} = \{(e, e')|e, e' \in \mathcal{E}\}$, where $\mathcal{E}$ is the entity set

Markov Decision Process (MDP) $M = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\}$

$\pi = (e_1, e_2, \cdots, e_T)$, where the positive item i is the first node in a_0; et is the last item node of the t-th step. At the terminal step T , e_T is used as the final negative item to optimize the recommender.

$\mathbb{P}\left(s_{t+1} = (u, e_{t+1})|s_t = (u, e_t), a_t = (e_t \rightarrow e'_t \rightarrow e_{t+1})\right) = 1$ $\mathcal{R}(e_t) = f_R(u, e_t) + g_R(i, e_t)$, $\max_{\Theta_S} \sum_{(u,i) \in O^+} \mathbb{E}_\pi[\sum_{t=1}^{T} \lambda^{t-1} \mathcal{R}(e_t)]$,

Recommender

$\mathbf{h}_e^{(l)} = \rho\left(\mathbf{W}^{(l)}(\mathbf{h}_e^{(l-1)} || \mathbf{h}_{\mathcal{N}_e}^{(l-1)})\right)$, where $\mathbf{h}_e^{(l)} \in \mathbb{R}^{d_l}$ is the representation after l steps of embedding propagation, d_l denotes the embedding size, $\mathbf{W}^{(l)} \in \mathbb{R}^{d_l \times 2d_{l-1}}$ - is the weight matrix to distill useful information; || is the concatenation operation and ρ is a nonlinear activation function set as LeakyReLU here; $\mathbf{h}_e^{(0)}$ epresents ID embedding, while $\mathbf{h}_{\mathcal{N}_e}^{(l-1)}$ is the information being propagated from e's neighbors as: $\mathbf{h}_{\mathcal{N}_e}^{(l-1)} = \sum_{\frac{1}{\cdots}} \mathbf{h}_{e'}^{(l-1)}, \mathcal{N}_e = \{e'|(e, e') \in \mathcal{G} \text{ or } (e, e') \in O^+\}$

$\mathbb{P}(a_t|s_t) = \mathbb{P}((e_t, e'_t)|s_t) \cdot \mathbb{P}((e'_t, e_{t+1})|s_t, (e_t, e'_t)), s_t = (u, e_t)$, established representations for node e_t and its neighbors $\mathcal{N}_{e_t}$ where $\mathbb{P}(a_t|s_t)$ represents the confidence or probability of e_{t+1} being negative; $\mathbb{P}(e_t, e'_t)$ and $\mathbb{P}(e'_t, e_{t+1})$ separately model the confidence of each exploration step. → $p(e_t, e'_t) = \mathbf{h}_u^\top \rho(\mathbf{h}_{e_t} \odot \mathbf{h}_{e'_t})$, where ⊙ is the element-wise product. → $\mathbb{P}((e_t, e'_t)|s_t) = \frac{\exp(p(e_t, e'_t))}{\sum_{e''_t \in \mathcal{N}_{e_t}} \exp(p(e_t, e''_t))}$ → $p(e'_t, e_{t+1}) = \mathbf{h}_u^\top \rho(\mathbf{h}_{e'_t} \odot \mathbf{h}_{e_{t+1}})$. → $\mathbb{P}((e'_t, e_{t+1})|s_t, (e_t, e'_t)) = \frac{\exp(p(e'_t, e_{t+1}))}{\sum_{e''_{t+1} \in \mathcal{N}_{e'_t}} \exp(p(e'_t, e''_{t+1}))}$.

Reinforced Negative Sampling over Knowledge Graph for Recommendation - 2020: Dataset & Results

Datasets: Amazon-Book, Yelp, Last-FM;

Table 2: Comparison with Negative Samplers.

	Yelp2018		Last-FM		Amazon-Book	
	recall	ndcg	recall	ndcg	recall	ndcg
RNS	0.0465	0.0575	0.0661	0.1063	0.1153	0.0754
PNS	0.0166	0.0220	0.0668	0.0984	0.1127	0.0730
DNS	**0.0687**	**0.0839**	**0.0877**	**0.1381**	**0.1518**	**0.1033**
IRGAN	0.0628	0.0767	0.0642	0.1070	0.1253	0.0833
AdvIR	0.0590	0.0744	0.0810	0.1304	0.1427	0.0967
NMRN	0.0565	0.0691	0.0719	0.1151	0.1305	0.0875
RWS	0.0488	0.0611	0.0667	0.1076	0.1185	0.0760
KGPolicy	**0.0747***	**0.0921***	**0.0932***	**0.1472***	**0.1572***	**0.1089***
%Improv.	8.73%	9.77%	6.27%	6.59%	3.56%	5.42%

Table 3: Comparison with KG-based Recommenders.

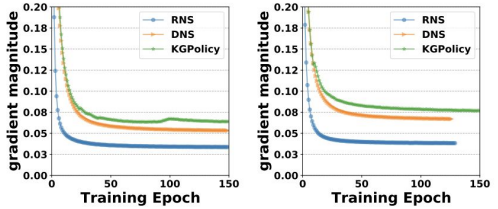
	Yelp2018		Last-FM		Amazon-Book	
	recall	ndcg	recall	ndcg	recall	ndcg
NFM	0.0660	0.0810	0.0829	0.1213	0.1366	0.0914
CKE	0.0657	0.0805	0.0736	0.1184	0.1344	0.0885
RippleNet	0.0664	0.0822	0.0791	0.1238	0.1336	0.0910
KGAT	**0.0712**	**0.0867**	**0.0870**	**0.1325**	**0.1489**	**0.1006**
KGPolicy	**0.0747***	**0.0921***	**0.0932***	**0.1472***	**0.1572***	**0.1089***
%Improv.	4.92%	6.22%	7.12%	11.09%	5.60%	8.25%

Table 4: Impact of exploration operation numbers.

	Yelp2018		Last-FM		Amazon-Book	
	recall	ndcg	recall	ndcg	recall	ndcg
KGPolicy-1	0.0738	0.0878	0.0891	0.1409	0.1520	0.1053
KGPolicy-2	**0.0747***	**0.0921***	**0.0932***	**0.1472***	**0.1572***	**0.1089***
KGPolicy-3	0.0730	0.0879	0.0928	0.1450	0.1551	0.1076
KGPolicy-4	0.0729	0.0878	0.0919	0.1437	0.1546	0.1059

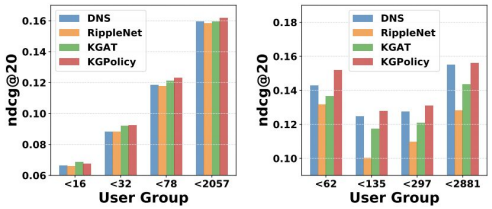
Table 5: Impacts of reward functions.

	Yelp2018		Last-FM		Amazon-Book	
	recall	ndcg	recall	ndcg	recall	ndcg
S-Reward	0.0731	0.0865	0.0899	0.1411	0.1559	0.1071
P-Reward	0.0721	0.0859	0.0918	0.1443	0.1558	0.1068



(a) gradient magnitude in Yelp2018 (b) gradient magnitude in Last-FM

Figure 3: Training process *w.r.t.* average gradient magnitude.



(a) ndcg on Yelp2018 (b) ndcg on Last-FM

Figure 4: Performance comparison *w.r.t.* sparsity levels.

Reinforced Negative Sampling over Knowledge Graph for Recommendation - 2020:

Limitations & Ideas to improve

Limitations:

- **Subsampling-Induced Errors:** Subsampling used to reduce computational costs can introduce information loss and errors, reducing the method's competitiveness.
- **Dependence on Predefined Reward Functions:** Static reward functions may not adapt effectively to varying user preferences or new interaction data.
- **Exploration Space Complexity:** The growth in the exploration space leads to computational overhead, particularly for large-scale KGs with a high number of entities and relations.
- **Scalability for Large KGs:** The longer training time for KGPolicy compared to other methods indicates scalability challenges. ("Empirically, RNS, DNS, IRGAN, AdvIR, NMRN, and KGPolicy cost around 22s, 116s, 155s, 175s, 172s, and 232s per training epoch on the largest Yelp2018 dataset")

Ideas to improve:

- (?) Incorporating a hierarchical sampling method could reduce exploration complexity by clustering nodes and focusing on the most relevant subsets of the graph.
- **Scalability for Large KGs:** Parallelize the exploration operations using distributed graph processing frameworks like DGL (Deep Graph Library) to handle large-scale KGs efficiently.
- (?) Integrate auxiliary user interaction data (e.g., viewing without clicking or partial engagement metrics) to better distinguish true negatives from potential positives.
- Design a reward function that combines user-based features (e.g., interest categories) and item popularity trends to refine the negative sampling quality.

RevGNN: Negative Sampling Enhanced Contrastive Graph Learning for Academic Reviewer Recommendation - 2024: Architecture

Stage 1

GCN(Submission-scholar bipartite graph)

$$\boldsymbol{h}_u^{b,(l)} = \hat{\boldsymbol{A}}\boldsymbol{h}^{b,(l-1)}\boldsymbol{W}_b^{(l)} \text{ where } H_b^{(l)} = [\boldsymbol{h}_1^{b,(l)}, ..., \boldsymbol{h}_n^{b,(l)}]^T \in \mathbb{R}^{n\times d}$$

is the representation of the nodes derived at l-th layer,

$$\hat{y}_{u,v}^b = ReLU(\boldsymbol{h}_u^b \odot \boldsymbol{h}_v^b),\ \boldsymbol{h}_u^b = \sum_{i=1}^{l} \boldsymbol{h}_u^{b,(i)},\quad \boldsymbol{h}_v^b = \sum_{i=1}^{l} \boldsymbol{h}_v^{b,(i)}.$$

Loss: $\mathcal{L}_{beh} = -\sum_{(u,v)\in E_{review}} y_{u,v}\log\hat{y}_{u,v}^b + (1-y_{u,v})\log(1-\hat{y}_{u,v}^b),$

Knowledge Encoder: The text title and abstract of a submission provide a refined theme description with scientific key points. Utilize OAG-LM (also known as OAG-BERT).

github: https://github.com/THUDM/Reviewer-Rec

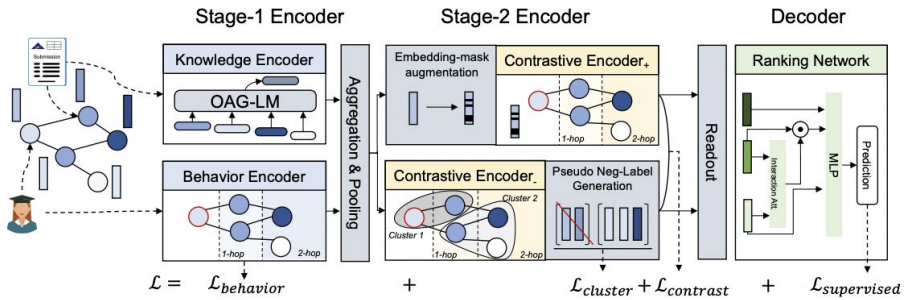


Fig. 2. The framework of RevGNN.

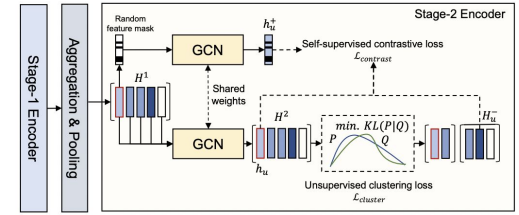


Fig. 3. The stage-2 encoding process. In this stage, the embedding representation of nodes is adjusted by graph contrastive learning. Particularly, we design a Pseudo Neg-Label Sampling strategy to avoid false negative edges for the observation-insufficient scholar-submission interaction graph.

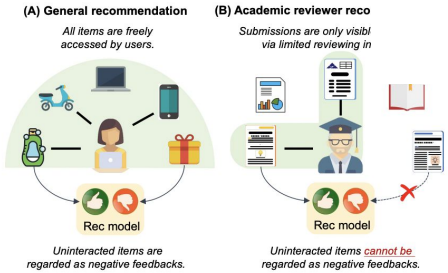


Fig. 1. Different visibility of items between (A) general and (B) academic reviewer recommendation.

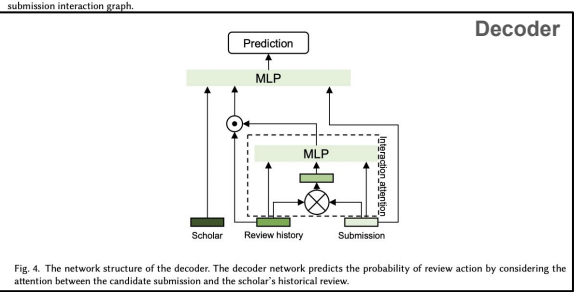


Fig. 4. The network structure of the decoder. The decoder network predicts the probability of review action by considering the attention between the candidate submission and the scholar's historical review.

Algorithm 1: RevGNN training algorithm

Input : Heterogeneous academic graph $\mathcal{G}$, Node feature X
Output: Optimized model parameters Θ

Initialize Θ ;
while Θ *does not converge* **do**
  // Stage-1 Preference Encoder
  Calculate the behaviour loss: $\mathcal{L}_{beh} \leftarrow$ Eq. (9) ;
  Derive the preference embedding: $\boldsymbol{h}_u^b \leftarrow$ Eq. (6) ;
  // Stage-1 Knowledge Encoder
  Derive the knowledge embedding: $\boldsymbol{h}_u^k \leftarrow$ OAG-LM(u, v) ;
  // Aggregation and Pooling
  $\boldsymbol{h}_u^1 = AGG(\boldsymbol{h}_u^b, \boldsymbol{h}_u^k)$;
  // Stage-2 Encoder
  for $i : 1 \rightarrow m$ **do**
    // Positive Sample
    $\boldsymbol{h}_u^+ \leftarrow$ Eq. (11) ;
    cluster ;
    for $i : 1 \rightarrow C$ **do**
      // Calculate the cluster loss
      Calculate the cluster loss: $\mathcal{L}_{clus} \leftarrow$ Eq. (13) ;
      // Pseudo Neg-Label Sample
      $\boldsymbol{h}_u^- \leftarrow$ Eq. (16) ;
      // Calculate the contrastive loss
      Calculate the cluster loss: $\mathcal{L}_{cl} \leftarrow$ Eq. (17) ;
  // Interaction-Oriented Recommendation Decoder
  Calculate the recommendation loss: $\mathcal{L}_{sup} \leftarrow$ Eq. (20) ;
  Update RevGNN: $\Theta = \mathcal{L} + \nabla\mathcal{L} \leftarrow$ Eq. (21) ;
Return Θ.

RevGNN: Negative Sampling Enhanced Contrastive Graph Learning for Academic Reviewer Recommendation - 2024: Dataset & Results

Datasets: Frontiers-4k, Frontiers-8k;

Dataset	Frontiers-4k				Frontiers-8k			
Metrics	R@20	N@20	HR@20	P@20	R@20	N@20	HR@20	P@20
TF-IDF	0.0122	0.0084	N/A	0.0022	N/A	N/A	N/A	N/A
TPMS	0.0802	0.0264	0.0832	0.0043	0.0546	0.0187	0.0573	0.0029
Wide&Deep	0.0132	0.0054	0.0267	0.0014	0.0070	0.0036	0.0140	0.0007
DeepFM	0.0118	0.0045	0.0238	0.0013	0.0070	0.0034	0.0140	0.0007
ENMF	0.0123	0.0074	0.0253	0.0013	0.0465	0.0224	0.0963	0.0049
RecVAE	0.0814	0.0384	0.1664	0.0086	0.0789	0.0418	0.1584	0.0083
LightGCN	0.0457	0.0219	0.0936	0.0048	0.0422	0.0213	0.0841	0.0044
SGL	0.0524	0.0257	0.1070	0.0053	0.0409	0.0202	0.0835	0.0043
SimGCL	0.0470	0.0212	0.0949	0.0049	0.0383	0.0184	0.0772	0.0039
SHT	0.0623	0.0364	0.1262	0.0053	0.0557	0.0337	0.1123	0.0030
ApeGNN_HK	0.0676	0.0420	0.0684	0.0035	0.0572	0.0322	0.0603	0.0030
ApeGNN_PPR	0.0705	0.0416	0.0713	0.0036	0.0477	0.0287	0.0512	0.0026
RevGNN	**0.1076**†	**0.0522**†	**0.2140**†	**0.0116**†	**0.0967**†	**0.0470**†	**0.1895**†	**0.0100**†
%Improv.	32.19%	24.29%	28.61%	34.88%	22.56%	12.44%	19.63%	20.48%

Table 4. Overall performance comparison between RevGNN and baselines. Marker † indicates the statistical significance between RevGNN and the most competitive baseline (p-value < 0.001).

Dataset	NIPS			
Metrics	R@20	N@20	HR@20	P@20
TF-IDF	N/A	N/A	N/A	N/A
TPMS	0.0912	0.0583	0.2104	0.0169
Wide&Deep	0.1219	0.0610	0.2541	0.0203
DeepFM	0.1219	0.0610	0.2541	0.0203
ENMF	0.0070	0.0029	0.0247	0.0012
RecVAE	0.1045	0.0805	0.4890	0.0315
LightGCN	0.0945	0.0543	0.2226	0.0182
SGL	0.0995	0.0666	0.2253	0.0125
SimGCL	0.1408	0.0880	0.4411	0.0242
SHT	0.1142	0.0564	0.2396	0.0110
ApeGNN_HK	0.1267	0.0721	0.2962	0.0179
ApeGNN_PPR	0.1024	0.0754	0.2469	0.0154
RevGNN	**0.1526**†	**0.1246**†	**0.6099**†	**0.0420**†
%Improv.	8.38%	41.59%	24.72%	33.33%

Table 5. Ablation Result of RevGNN.

Dataset	Frontiers-4k				Frontiers-8k			
Metrics	R@20	N@20	HR@20	P@20	R@20	N@20	HR@20	P@20
-Knowl.	0.0868	0.0431	0.1798	0.0094	0.0813	0.0401	0.1627	0.0084
-Behav.	0.0887	0.0428	0.1798	0.0094	0.0856	0.0439	0.1718	0.0089
-Stage-2	0.0992	0.0505	0.2021	0.0103	0.0944	0.0451	0.1883	0.0099
RevGNN	**0.1076**	**0.0522**	**0.2140**	**0.0116**	**0.0967**	**0.0470**	**0.1895**	**0.0100**

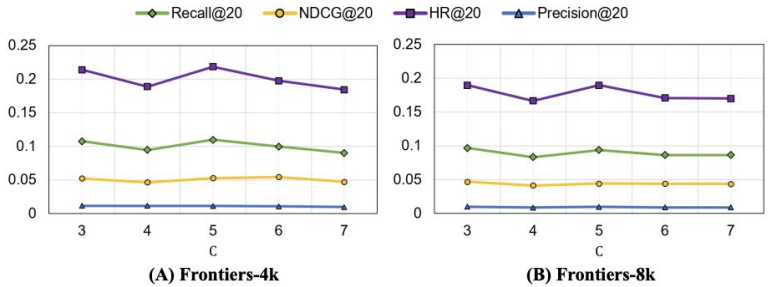


Fig. 5. The variation of performance with different numbers of cluster C.

RevGNN: Negative Sampling Enhanced Contrastive Graph Learning for Academic Reviewer Recommendation - 2024: Limitations & Ideas to improve

Limitations:

- **False Assumption of Negative Interactions:** The assumption that unobserved reviewer-submission interactions are "negative samples" is overly simplistic and inaccurate. This results in false negatives that mislead the model during training.
- **Lack of Fine-Tuning for Pre-trained Models:** The OAG-LM (Open Academic Graph Language Model) is used to encode prior knowledge but is not fine-tuned for the specific dataset or task.
- **Absence of Reviewer Availability Modeling:** The method does not account for reviewer workload or availability, which are critical factors in practical reviewer assignment.

Ideas to improve:

- (?) Use a probabilistic labeling system to differentiate between unobserved interactions and actual negatives based on auxiliary metadata such as submission keywords or reviewer expertise overlap.
- Perform lightweight fine-tuning of the OAG-LM using a validation subset to align it better with the task without risking data leakage.
- Introduce a reviewer workload prediction module using historical review acceptance/rejection rates to filter out unavailable candidates.

Positive, Negative and Neutral: Modeling Implicit Feedback in Session-based News Recommendation - 2022: Architecture

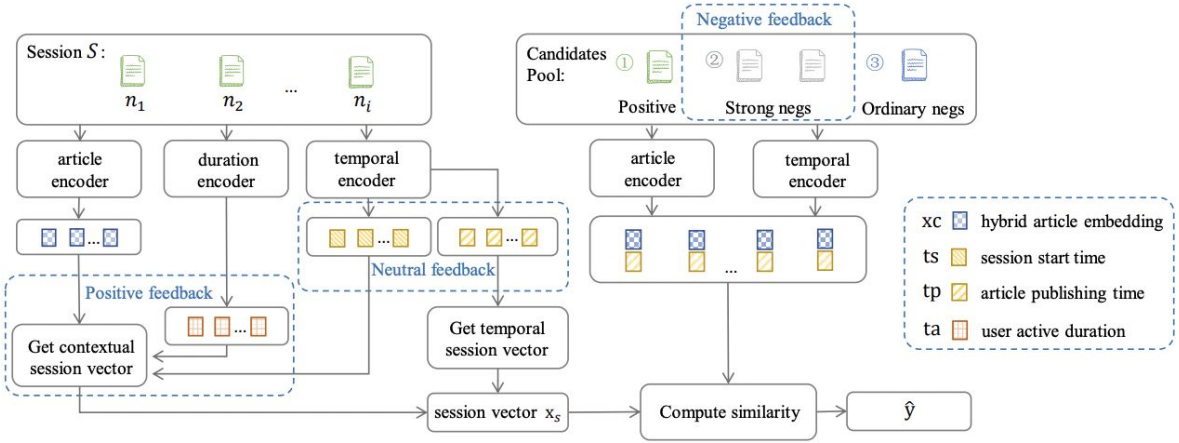


Figure 2: The architecture of our model. Squares in the figure represent vectors, and their colors refer to the different encoders that produce them.

Positive feedback: Derived from the time users spend on articles.
Negative feedback: Estimated from articles skipped in the impression list.
Neutral feedback: Modeled using session start time and article publishing time.

Temporal Encoder:

- Captures temporal patterns (e.g., weekday, hour) in session start time and publishing time.
- Helps model user routines and the timeliness of articles.

Duration Encoder:

- Bucketizes reading time into discrete categories for better representation of user engagement.
- Converts reading durations into embeddings to quantify positive feedback.

Positive, Negative and Neutral: Modeling Implicit Feedback in Session-based News Recommendation - 2022: Dataset & Results

Datasets: Adressa, Globo, MIND;

Table 3: Main and ablation results ($k = 20$ by default in our all tables). All results are averaged over all folds. The best baseline result on each metric is marked with $*$ and overall best results are bolded. The "Ours" is our whole model and (-) means to ablate the corresponding module, where "neut", "pos" and "neg" respectively refer to our neutral, positive and negative feedback modules. The last column is to replace our negative sampling strategy with random sampling. ↓ indicates performance drop over the whole model.

Datasets	Metrics	CBCF	STAN	GRU4Rec	SASRec	SRGNN	SGNNHN	STAMP	Ours	(-)neut	(-)pos	(-)neg	random
Adressa	HR	0.0957	0.1130	0.1120	0.1205	0.1152	0.1285	0.1287*	**0.1658**	0.1344↓	0.1619↓	0.1658	0.1646↓
	NDCG	0.0341	0.0500	0.0511	0.0509	0.0536	0.0562	0.0575*	**0.0730**	0.0613↓	0.0690↓	0.0720↓	0.0693↓
	ILD	0.2337	0.2409	0.8170	0.7856	**0.8611***	0.8403	0.8445	0.8085	0.8204	0.8249	0.8237	0.8234
	unEXP	0.2509	0.2407	0.6949	0.8010	0.4754	0.8059*	0.5728	**0.8279**	0.8243↓	0.8333	0.8267↓	0.8346
Globo	HR	0.1185	0.1273	0.1280	0.1409	0.1280	0.1414	0.1435*	**0.1852**	0.1460↓	0.1817↓	0.1821↓	0.1847↓
	NDCG	0.0474	0.0647	0.0599	0.0620	0.0627	0.0611	0.0698*	**0.0936**	0.0727↓	0.0907↓	0.0940	0.0933↓
	ILD	0.3874	0.3087	0.9377	**0.9864***	0.9248	0.9415	0.7980	0.8702	0.8362↓	0.8685↓	0.8927	0.8739
	unEXP	0.3730	0.2921	**0.9771***	0.9690	0.6383	0.9467	0.8437	0.8358	0.8142↓	0.8252↓	0.8489	0.8317↓
MIND	HR	0.0315	0.0312	0.0338	0.0355	0.0334	0.0366	0.0371*	**0.0495**	0.0445↓	-	0.0471↓	0.0457↓
	NDCG	0.0110	0.0142	0.0132	0.0139	0.0144	0.0122	0.0151*	**0.0211**	0.0198↓	-	0.0180↓	0.0204↓
	ILD	0.7166	0.3193	0.8662	0.8562	0.8706	0.8775*	0.8452	**0.8813**	0.8779↓	-	0.8808↓	0.8858
	unEXP	0.6039	0.1064	0.8578	**0.8654***	0.4508	0.4514	0.7544	0.8617	0.8415↓	-	0.8623	0.8680

[1] We conduct t significance test between the best score (if ours) and the second-best score for the main results, and the improvement is strongly significant as $p < 0.001$. Between the results of the whole model and the ablation model, the decline is significant as $p < 0.01$.

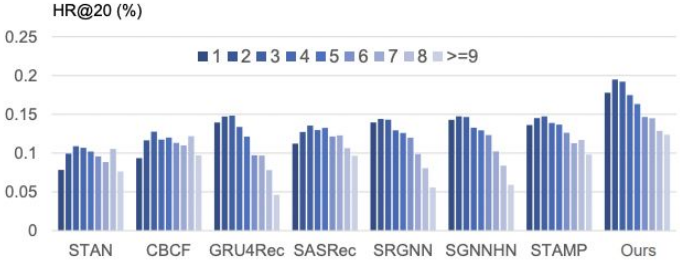
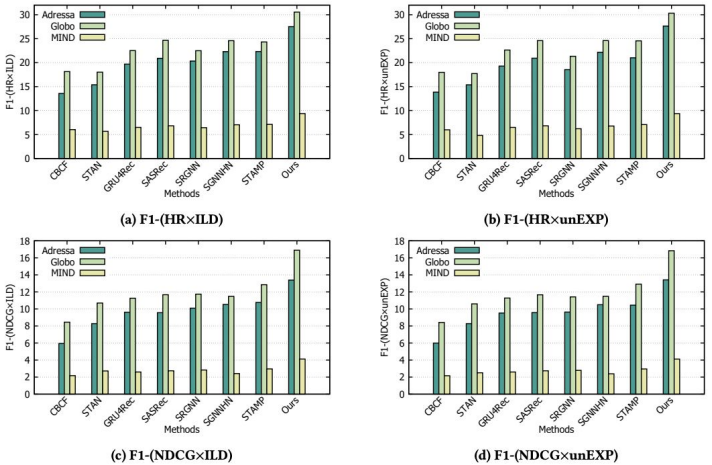


Figure 6: Accuracy with different session lengths.



(a) F1-(HR×ILD) (b) F1-(HR×unEXP)

(c) F1-(NDCG×ILD) (d) F1-(NDCG×unEXP)

Figure 3: The graphical comparison of all methods on 4 different metrics and 3 different datasets."Ours" is our approach.

Positive, Negative and Neutral: Modeling Implicit Feedback in Session-based News Recommendation - 2022: Limitations & Ideas to improve

Limitations:

- **Assumption of Uniform Reading Speeds:** The approach estimates user preference using active reading time but does not account for variations in individual reading speeds, leading to potential bias in interpreting positive feedback.
- **Potential Noise in Long Sessions:** For longer sessions, the model performance drops due to noise introduced by diverse user interests, leading to less accurate session representations.

Ideas to improve:

- Incorporate personalized reading speed profiles using historical data or demographic information where available.
- Implement a hierarchical attention mechanism that prioritizes recent clicks while retaining broader session context.
- Segment sessions dynamically into sub-sessions based on user activity patterns to reduce noise from unrelated interactions.

SIGformer: Sign-aware Graph Transformer for Recommendation - 2024: Architecture

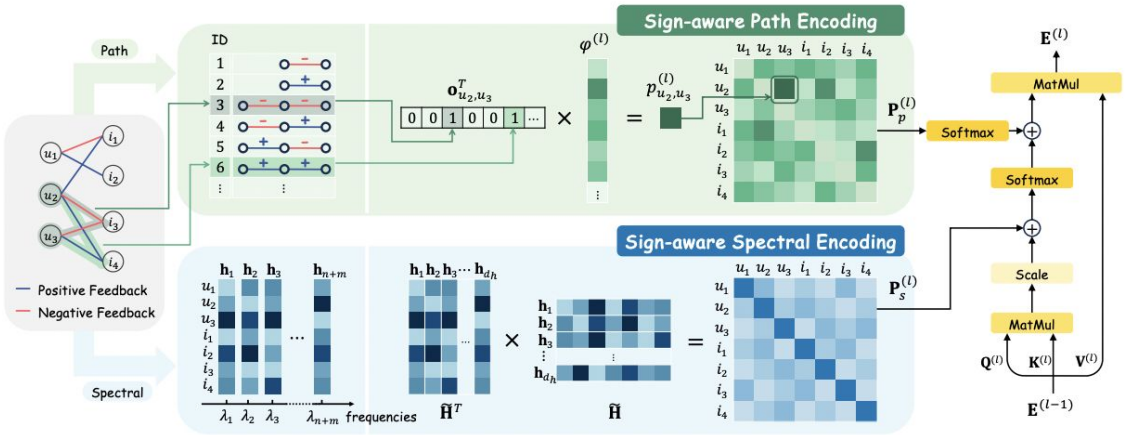


Figure 2: The illustration of our proposed sign-aware path encoding and sign-aware spectral encoding in SIGformer.

Overview of SIGformer

$$\mathbf{E}^{(0)} = [\underbrace{\mathbf{e}_{u_1}^{(0)}, \cdots, \mathbf{e}_{u_n}^{(0)}}_{\text{user embeddings}}, \underbrace{\mathbf{e}_{i_1}^{(0)}, \cdots, \mathbf{e}_{i_m}^{(0)}}_{\text{item embeddings}}]^T.$$

$$\mathbf{Q}^{(l)} = \mathbf{K}^{(l)} = \mathbf{V}^{(l)} = \mathbf{E}^{(l-1)}$$

$$\mathbf{E}^{(l)} = \frac{1}{2}(\text{softmax}(\frac{\mathbf{Q}^{(l)}(\mathbf{K}^{(l)})^T}{\sqrt{d}} + \mathbf{P}_s^{(l)}) + \text{softmax}(\mathbf{P}_p^{(l)}))\mathbf{V}^{(l)}$$

$$\mathbf{E} = \frac{1}{L+1}\sum_{0\le l\le L}\mathbf{E}^{(l)} \qquad \hat{y}_{ui} = \mathbf{e}_u^T\mathbf{e}_i$$

Sign-aware Spectral Encoding (SSE)

$\mathbf{L} = \frac{1}{1-\alpha}(\mathbf{L}^+ - \alpha\mathbf{L}^-)$ $\mathbf{L}^+$ denote the Laplacian matrix of $\mathcal{G}^+$, defined as $\mathbf{L}^+ = \mathbf{I} - (\mathbf{D}^+)^{-\frac{1}{2}}\mathbf{A}^+(\mathbf{D}^+)^{-\frac{1}{2}}$

$\mathbf{L} = \mathbf{H}^T\mathbf{\Lambda}\mathbf{H}$, $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{n+m}]^T$ $\mathbf{P}_s^{(l)} = \theta^{(l)}\tilde{\mathbf{H}}^T\tilde{\mathbf{H}}$, $\tilde{\mathbf{H}} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{d_h}]^T$

$\mathbf{v} = \sum_{1\le k\le n+m}\varepsilon_k\mathbf{h}_k$ where ε_k represents the strength of the signal on the component $\mathbf{h}_k$. The effect of introducing $\bar{\mathbf{P}}_s^{(l)}$ can be described as:

$$\bar{\mathbf{P}}_s^{(l)}\mathbf{v} = (\sum_{1\le k\le d_h}\theta^{(l)}\mathbf{h}_k\mathbf{h}_k^T)(\sum_{1\le k\le n+m}\varepsilon_k\mathbf{h}_k) = \theta^{(l)}\sum_{1\le k\le d_h}\varepsilon_k\mathbf{h}_k$$

where only the low-frequency components $(h_1, h_2, \cdots, h_{d_k})$

Sampling for Acceleration:
Authors utilize a random walk strategy on the signed graph to pick up nodes for aggregation and concurrently record the walked path for computing $\bar{\mathbf{P}}_p^{(l)}$. The rapid updating of user/item embeddings as follows:

$$\mathbf{e}_v^{(l)} = \frac{1}{2}\sum_{w\in\mathcal{S}_v}\left(\text{softmax}\left(\frac{(\mathbf{e}_v^{(l-1)})^T\mathbf{e}_w^{(l-1)}}{\sqrt{d}} + \theta^{(l)}m_{vw}\right) + \text{softmax}(\varphi_{t_{vw}})\right)\mathbf{e}_w^{(l-1)}$$

$$\mathcal{L} = -\sum_{(u,i)\in\mathcal{E}^+}\ln\sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \sum_{(u,i)\in\mathcal{E}^-}\ln\sigma(\beta(\hat{y}_{ui} - \hat{y}_{uj}))$$

SIGformer: Sign-aware Graph Transformer for Recommendation - 2024: Dataset & Results

Datasets: : Amazon-CDs, Amazon-Music, Epinions, KuaiRec, KuaiRand;

Table 2: Performance comparison between SIGformer and baselines. The best result is bolded and the runner-up is underlined. The mark '*' suggests the improvement is statistically significant with $p < 0.05$.

		Amazon-CDs		Amazon-Music		Epinions		KuaiRec		KuaiRand	
		Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
Unsigned Graph-based RS	LightGCN	0.1325	0.0781	0.2725	0.1601	0.0854	0.0510	0.0826	0.0499	0.1197	0.0588
	LightGCL	0.1040	0.0591	0.2921	0.1648	0.0864	0.0516	0.0848	0.0520	0.1291	0.0628
	XSimGCL	0.1346	0.0796	0.2848	0.1683	0.0887	0.0558	0.0863	0.0522	0.1293	0.0641
	GFormer	0.1366	0.0812	0.2807	0.1648	**0.0978**	**0.0602**	0.0864	0.0520	0.1083	0.0532
Sign-aware Graph-based RS	SiReN	0.1369	0.0801	0.2880	0.1725	0.0804	0.0492	0.0826	0.0473	0.1167	0.0571
	SiGRec	0.1092	0.0648	0.1591	0.0896	0.0738	0.0475	0.0497	0.0314	0.1266	0.0699
	PANE-GNN	0.1361	0.0810	0.2691	0.1605	0.0532	0.0301	0.0806	0.0514	0.1066	0.0522
Signed Graph Embedding Methods	SBGNN	0.0183	0.0100	0.0641	0.0325	0.0249	0.0143	0.0797	0.0469	0.0750	0.0361
	SLGNN	0.0283	0.0148	0.1498	0.0788	0.0585	0.0336	0.0865	0.0508	0.1082	0.0520
Graph Transformer	SGFormer	0.0492	0.0275	0.2402	0.1373	0.0588	0.0343	0.0840	0.0504	0.0883	0.0423
	SignGT	0.0231	0.0121	0.1283	0.0666	0.0521	0.0300	0.0861	0.0515	0.0927	0.0439
Our Method	SIGformer	**0.1412***	**0.0828***	**0.3091***	**0.1827***	0.0974	0.0585	**0.0908***	**0.0539***	**0.1494***	**0.0722***
		+3.09%	+1.96%	+5.81%	+5.87%	-0.41%	-2.77%	+5.05%	+3.32%	+15.61%	+3.33%

Table 3: The results of the ablation study, where positional encodings or negative interactions are removed respectively.

	Negative Interactions?	Spectral Encoding?	Path Encoding?	Amazon-CDs		Amazon-Music		Epinions		KuaiRec		KuaiRand	
				Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
SIGformer-w/o-Neg		✓	✓	0.1349	0.0775	0.2937	0.1738	0.0824	0.0477	0.0708	0.0433	0.1173	0.0545
SIGformer-w/o-En	✓			0.1355	0.0779	0.2932	0.1698	0.0894	0.0526	0.0728	0.0448	0.1413	0.0661
SIGformer-w/o-SPE	✓	✓		0.1380	0.0798	0.2988	0.1744	0.0959	0.0574	0.0862	0.0520	0.1471	0.0697
SIGformer-w/o-SSE	✓		✓	0.1381	0.0812	0.2947	0.1758	0.0945	0.0566	0.0866	0.0515	0.1457	0.0703
SIGformer	✓	✓	✓	**0.1412**	**0.0828**	**0.3091**	**0.1827**	**0.0974**	**0.0585**	**0.0908**	**0.0539**	**0.1494**	**0.0722**

SIGformer: Sign-aware Graph Transformer for Recommendation - 2024: Limitations & Ideas to improve

Limitations:

- **Computational Complexity of Sampling Strategy:** The random-walk-based sampling strategy, while efficient, introduces variance and can potentially exclude valuable interactions. This limits the reproducibility and robustness of the embeddings generated for certain user-item interactions.
- **Overemphasis on Path Length:** The model's consideration of path lengths up to L_p = 6 may dilute the importance of direct, high-signal interactions in the graph. Excessively long paths introduce noise rather than meaningful collaborative signals.
- **Over-Simplified Attention Mechanism:** By omitting the query, key, and value projection matrices in the transformer, SIGformer reduces computational complexity but potentially sacrifices the expressiveness of the attention mechanism.
- **Lack of Content-Aware Positional Encoding:** While the positional encodings effectively capture structural properties, they do not incorporate contextual or content information about users or items, which could be crucial for certain recommendation tasks.

Ideas to improve:

- Use importance sampling techniques to prioritize nodes and paths that contribute higher collaborative signals.
- Incorporate adaptive sampling based on node centrality or interaction frequency to reduce variance.
- Introduce a decay factor for longer paths to weigh them less heavily, reducing noise from distant connections.
- Reintroduce projection matrices \(W_Q, W_K, W_V \) with lightweight approximations (e.g., low-rank factorization) to maintain computational efficiency.
- Experiment with multi-head attention variants to improve expressiveness without significant overhead.
- Incorporate item/user features (e.g., category, metadata) into the positional encodings to enrich node representations.
- Explore hybrid encodings combining structural and content-based information for better generalization.

Amplify Graph Learning for Recommendation via Sparsity Completion - 2024: Architecture

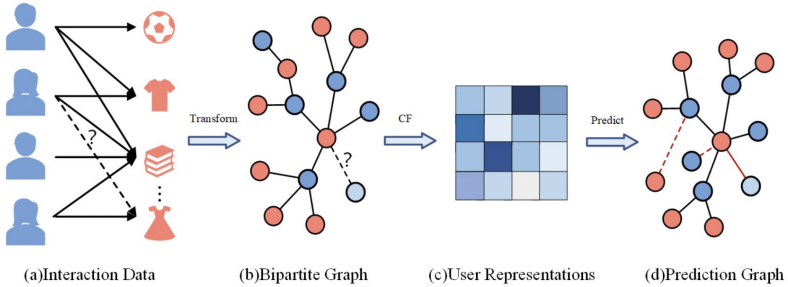


Figure 1: An illustration of the matrix completion algorithm in recommendation systems, demonstrating the transformation from interactions to bipartite graphs for prediction purposes.

$$\mathcal{L}_{REC} = \sum_{u \in U, i, j \in V} - \ln sigmoid(\hat{P}_{ui} - \hat{P}_{uj}).$$

$$\mathcal{L}_{MC} = D_{KL}(W_U \mid\mid \hat{W}_U) + D_{KL}(W_V \mid\mid \hat{W}_V).$$

$$\mathcal{L}_{MC} = \sum_{i,j=1}^{M} W_{Uij}\left(\log \frac{W_{Uij}}{\hat{W}_{Uij}}\right) + \sum_{i,j \in V}^{N} W_{Vij}\left(\log \frac{W_{Vij}}{\hat{W}_{Vij}}\right).$$

$$\mathcal{L}_{VAE} = \mathbb{E}_{q_\phi(\mathbf{z}_u \mid \hat{P}_u)}\left[\log p_\theta(\mathbf{z}_u \mid \hat{P}_u)\right] - D_{KL}(q_\phi(\mathbf{z}_u \mid \hat{P}_u) \mid\mid p_\theta(\mathbf{z}_u)).$$

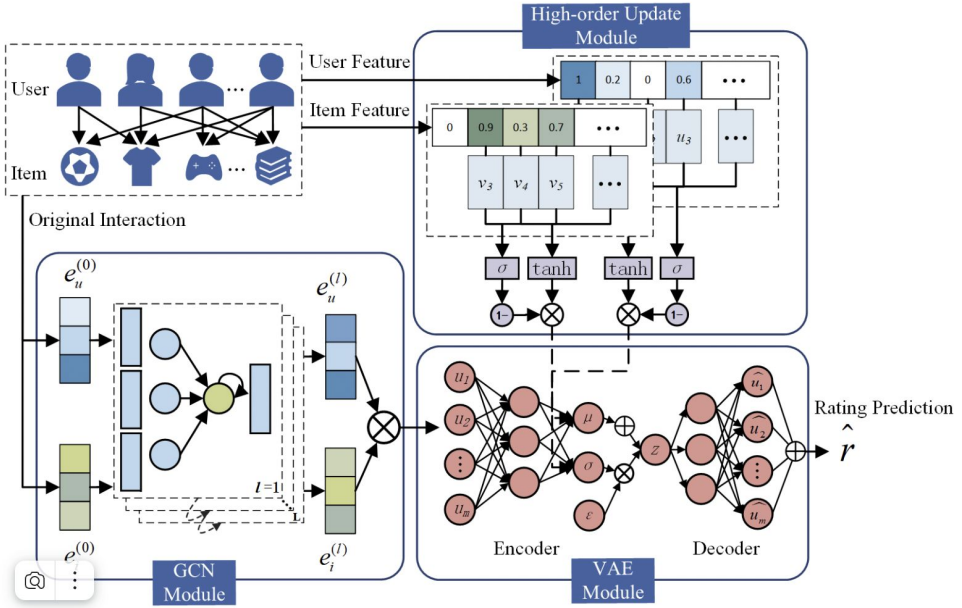


Figure 2: The illustration of our proposed AGL-SC framework, which consists of a GCN module for initial feature extraction from original user-item interactions, high-order update module for incorporating high-order constraints, and a VAE module for generating predictions.

$$\mathcal{L}_{AGL-SC} = \mathcal{L}_{REC} + \lambda \mathcal{L}_{MC} + \beta \mathcal{L}_{VAE}.$$

github: https://github.com/yp8976/AGL_SC

Amplify Graph Learning for Recommendation via Sparsity Completion - 2024: Dataset & Results

Datasets: MovieLens100K, MovieLens1M, Amazon Electronics, Yelp;

TABLE I: Overall performance comparison. Improv. denotes the relative improvements over the best baselines.

Model	Amazon-Electronics			ML-100K			ML-1M			Ye	
	R@20	N@20	N@40	R@20	N@20	N@40	R@20	N@20	N@40	R@20	N@
DeepWalk	0.0725	0.0579	0.0654	0.2832	0.2371	0.2758	0.1348	0.1057	0.1626	0.0476	0.0
Node2Vec	0.0799	0.0534	0.0724	0.3005	0.2568	0.2974	0.1475	0.1186	0.1677	0.0452	0.0
NGCF	0.1286	0.0710	0.0830	0.3413	0.3034	0.3341	0.2475	0.2457	0.3028	0.0579	0.0
LightGCN	0.1259	0.0715	0.0855	0.3422	0.3097	0.3348	0.2511	0.2507	0.3072	0.0617	0.0
DGCF	0.1298	0.0714	0.0853	0.3433	0.3093	0.3386	0.2576	0.2504	0.3069	0.0649	0.0
UltraGCN	0.1291	0.0791	0.0894	0.3445	0.3124	0.3574	0.2640	0.2513	0.3092	0.0654	0.0
SVD-GCN	0.1318	0.0809	0.0913	0.3505	0.3145	0.3601	0.2737	0.2552	0.3112	0.0683	0.0
VGCL	0.1362	0.0824	0.0935	0.3524	0.3152	0.3606	0.2756	0.2598	0.3120	0.0687	0.0
AGL-SC(Pytorch)	**0.1413**	**0.0850**	**0.0967**	**0.3598**	**0.3226**	**0.3683**	**0.2834**	**0.2653**	**0.3164**	**0.0703**	**0.0**
AGL-SC(MindSpore)	**0.1139**	**0.0459**	**0.0671**	**0.3506**	**0.3069**	**0.3233**	**0.2551**	**0.2671**	**0.2787**	**0.0701**	**0.0**
%Improv.	3.74%	3.16%	3.42%	2.10%	2.35%	2.14%	2.83%	2.12%	1.4%	2.34%	3.8
p-**value**	2.81e-12	4.59e-9	3.84e-9	1.71e-9	3.72e-8	1.24e-9	1.02e-6	6.84e-6	9.02e-8	9.64e-9	1.2 1

TABLE III: Performance comparison among DGCF and different AGL-SC variants.

Model	MovieLens1M		Yelp	
	R@20	N@20	R@20	N@20
DGCF	0.2576	0.2504	0.0649	0.0530
w / o **both**	0.2471	0.2390	0.0542	0.0454
w / o **VAE**	0.2517	0.2449	0.0544	0.0514
w / o **FM**	0.2593	0.2599	0.0674	0.0563
AGL-SC(Pytorch)	0.2834	0.2653	0.0703	0.0587
AGL-SC(MindSpore)	0.2551	0.2671	0.0655	0.0479

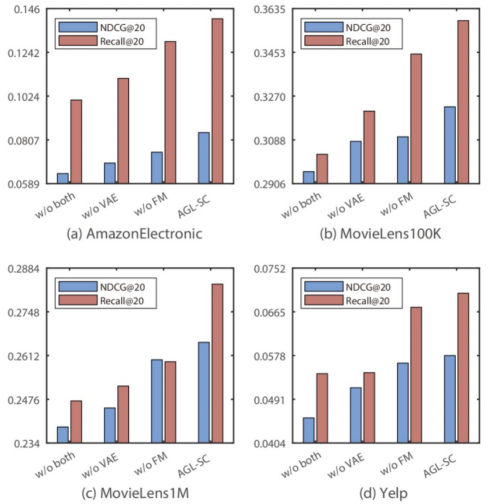


Figure 3: Performance of ablated models on datasets in terms of Recall@20 and NDCG@20.

Amplify Graph Learning for Recommendation via Sparsity Completion - 2024: Limitations & Ideas to improve

Limitations:

- **Suboptimal High-Order Interaction Mining Limitation:** High-order interactions are derived through a factorization-based method, which may oversimplify complex relationships between nodes. This approach assumes linear interactions between high-order features, which might not be sufficient for capturing non-linear dependencies.
- **Potential Overfitting in Sparsity Completion Limitation:** The sparsity completion relies heavily on multi-order feature integration through the VAE module. However, the lack of explicit regularization mechanisms in the high-order constraint module increases the risk of overfitting, particularly in dense regions of the graph.

Ideas to improve:

- Replace the factorization-based method with a graph attention network (GAT) to dynamically model high-order interactions. Attention mechanisms can weight relationships more effectively, improving the representation of non-linear dependencies.
- Incorporate explicit regularization terms or dropout layers in the high-order constraint module to mitigate overfitting. Additionally, using a sparsity-aware loss function could prevent overemphasis on dense regions of the graph.

Graph Contrastive Learning with Multi-Objective for Personalized Product Retrieval in Taobao Search - 2017: Architecture

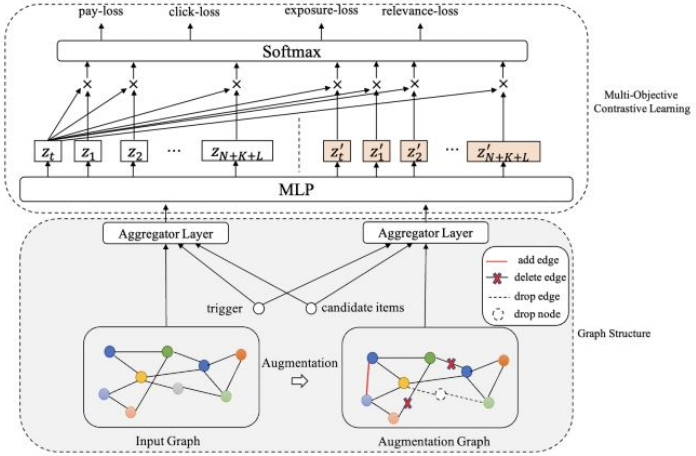


Figure 1: GCL-MO:Graph Contrastive Learning with Multi-Objective model architecture. Triggers and candidate items are aggregated from original graph and augmented graph obtaining two vectors. In addition to the two vectors z_t and z'_t of trigger, there are $2 \times (N + K + L)$ vectors for candidate items, where z_i represents the vector formed from the aggregation of original graph, z'_i denotes a vector aggregated from augmented graph. z_t computes the inner-product with all other vectors and constructs four contrastive loss functions.

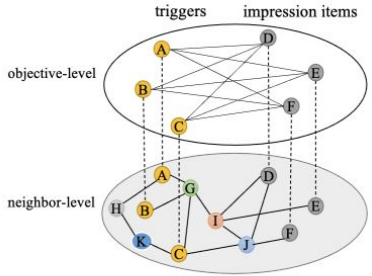


Figure 2: The two-level structure of items relationship.

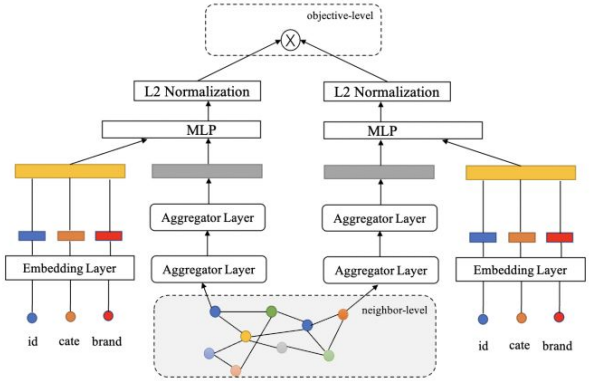


Figure 3: An illustration the generation process of the item vector in GCL-MO model. The source of item's embedding has two parts, one part is the embedding of its ID and other features, and the other is aggregated from the neighbor nodes.

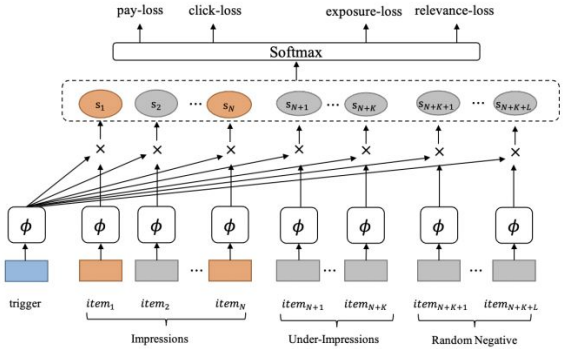


Figure 4: Multi-objective retrieval model GCL-MO with multi-positive training samples.

Graph Contrastive Learning with Multi-Objective for Personalized Product Retrieval in Taobao Search - 2017: Dataset & Results

Datasets: Taobao;

Table 1: Offline performance of GCL-MO and baseline models,where K is 6,000.

Model	$Recall@K$	$Recall_p@K$	P_{good}
GCL-MO	0.5732	0.5957	0.2845
LINE	0.5472(-2.60%)	0.5596(-3.61%)	0.2446(-3.99%)
EGES	0.5537(-1.95%)	0.5703(-2.54%)	0.2587(-2.58%)
GraphSAGE	0.5618(-1.14%)	0.5845(-1.12%)	0.2687(-1.58%)
NGCF	0.5634(-0.98%)	0.5854(-1.03%)	0.2693(-1.52%)
NIA-GCN	0.5647(-0.85%)	0.5859(-0.98%)	0.2702(-1.43%)
LightGCN	0.5651(-0.81%)	0.5864(-0.93%)	0.2706(-1.39%)
DGCF	0.5655(-0.77%)	0.5868(-0.89%)	0.2715(-1.30%)
UltraGCN	0.5661(-0.71%)	0.5871(-0.86%)	0.2723(-1.22%)
MO-UltraGCN	0.5695(-0.37%)	0.5917(-0.40%)	0.2769(-0.76%)
OL GCL-MO	0.5703(-0.29%)	0.5923(-0.34%)	0.2816(-0.29%)

Table 2: Offline performance of GCL-MO and its ablation models,where K is 6,000.

Model	$Recall@K$	$Recall_p@K$	P_{good}
-relevance loss	0.5755(+0.23%)	0.5968(+0.11%)	0.2413(-4.32%)
-exposure loss	0.5616(-1.16%)	0.5813(-1.44%)	0.2834(-0.11%)
-click loss	0.5587(-1.45%)	0.5848(-1.09%)	0.2796(-0.49%)
-purchase loss	0.5607(-1.25%)	0.5841(-1.16%)	0.2831(-0.14%)

Table 3: Offline performance of GCL-MO model and other models(without contrastive loss) about P_l.

Model	$Recall@K$	$Recall_p@K$	P_l
GCL-MO	0.5732	0.5957	0.1445
GL-MO	0.5681(-0.51%)	0.5914(-0.43%)	0.1364(-0.81%)
AP GCL-MO	0.5703(-0.29%)	0.5937(-0.20%)	0.1411(-0.34%)
LightGCN	0.5651(-0.81%)	0.5864(-0.93%)	0.1327(-1.18%)
DGCF	0.5655(-0.77%)	0.5868(-0.89%)	0.1339(-1.06%)
UltraGCN	0.5661(-0.71%)	0.5871(-0.86%)	0.1354(-0.91%)

Table 4: Online performance of GCL-MO and its ablation models.

Model	GMV	P_{good}	P_{click}	$P_{purchase}$
GCL-MO	+0.36%	+0.78%	+1.53%	+0.98%
GL-MO	+0.14%	+0.37%	+0.65%	+0.37%
OL GCL-MO	+0.16%	+0.39%	+0.71%	+0.43%

Table 5: Online performance of GCL-MO in long-tail items.

Model	$P_{l-impression}$	$P_{l-click}$	$P_{l-purchase}$
GCL-MO	+3.38%	+2.76%	+1.37%
GL-MO	+0.65%	+0.39%	+0.17%

Graph Contrastive Learning with Multi-Objective for Personalized Product Retrieval in Taobao Search - 2017: Limitations & Ideas to improve

Limitations:

- **Imbalanced Objective Optimization Limitation:** The multi-objective learning framework optimizes for relevance, exposure, click, and purchase, but assigning fixed weights

Ideas to improve:

- Use adaptive weighting mechanisms to dynamically adjust w, based on real-time performance metrics.

Challenging the Myth of Graph Collaborative Filtering: a Reasoned and Reproducibility-driven Analysis - 2023: Critical Review of the Article: Positive Points

github: https://github.com/sisinflab/Graph-RSs-Reproducibility

1. Focus on Reproducibility

The paper emphasizes the issue of reproducibility in graph-based recommender systems. It rigorously evaluates six models (e.g., NGCF, LightGCN, and SGL) across three datasets (Gowalla, Yelp 2018, Amazon Book) to provide objective benchmarks. For example, on the Yelp 2018 dataset, RP3β achieved a Recall@20 of 0.0647, outperforming LightGCN's 0.0639, challenging assumptions about graph-based models' superiority.

2. Comprehensive Experimental Setup

The authors propose a well-designed experimental framework, incorporating multiple metrics (e.g., Recall@20) and deep comparisons. This highlights the strengths and limitations of both graph-based and traditional collaborative filtering methods.

Graph-based CF solutions tested against unpersonalized (i.e., reference) and classical CF approaches on Gowalla, Yelp 2018, and Amazon Book. While results for the graph-based approaches have been directly reported from our reproducibility study (see above), classical CF recommender systems have been fine-tuned on the two datasets to find their best configurations. Boldface and underline refer to best and second-to-best values, respectively.

Families	Models	Gowalla		Yelp 2018		Amazon Book	
		Recall	nDCG	Recall	nDCG	Recall	nDCG
Reference	MostPop	0.0416	0.0316	0.0125	0.0101	0.0051	0.0044
	Random	0.0005	0.0003	0.0005	0.0004	0.0002	0.0002
Classic CF	UserkNN	0.1685	0.1370	0.0630	0.0528	0.0582	0.0477
	ItemkNN	0.1409	0.1165	0.0610	0.0507	0.0634	0.0524
	$RP^3\beta$	0.1829	0.1520	0.0671	0.0559	0.0683	0.0565
	$EASE^R$ *	0.1661	0.1384	0.0655	0.0552	**0.0710**	0.0567
Graph CF	NGCF	0.1556	0.1320	0.0556	0.0452	0.0319	0.0246
	DGCF	0.1736	0.1477	0.0621	0.0505	0.0384	0.0295
	LightGCN	0.1826	0.1545	0.0629	0.0516	0.0419	0.0323
	SGL	—	—	0.0669	0.0552	0.0474	0.0372
	UltraGCN	**0.1863**	**0.1580**	0.0672	0.0553	0.0688	0.0561
	GFCF	0.1849	0.1518	**0.0697**	**0.0571**	**0.0710**	**0.0584**

**Results for $EASE^R$ on Amazon Book are taken from BARS Benchmark [96].*

Challenging the Myth of Graph Collaborative Filtering: a Reasoned and Reproducibility-driven Analysis - 2023: **Critical Review of the Article: Negative Points**

1. Overfitting Risk

Models like NGCF show overfitting tendencies, especially on smaller datasets or when the negative feedback ratio is high. This diminishes their generalizability to unseen data.

Tab. Results of replicability study on Gowalla, Yelp 2018, and Amazon Book for the selected state-of-the-art graph-based recommender systems. Calculate the performance shift between conducted experiments and the original ones (as reported in their papers). Note that models have been sorted out according to the chronological order.

Datasets	Models	Ours		Original		Performance Shift	
		Recall	nDCG	Recall	nDCG	Recall	nDCG
Gowalla	NGCF	0.1556	0.1320	0.1569	0.1327	$-1.3 \cdot 10^{-03}$	$-7 \cdot 10^{-04}$
	DGCF	0.1736	0.1477	0.1794	0.1521	$-5.8 \cdot 10^{-03}$	$-4.4 \cdot 10^{-03}$
	LightGCN	0.1826	0.1545	0.1830	0.1554	$-4 \cdot 10^{-04}$	$-9 \cdot 10^{-04}$
	SGL*	—	—	—	—	—	—
	UltraGCN	0.1863	0.1580	0.1862	0.1580	$+1 \cdot 10^{-04}$	0
	GFCF	0.1849	0.1518	0.1849	0.1518	0	0
Yelp 2018	NGCF	0.0556	0.0452	0.0579	0.0477	$-2.3 \cdot 10^{-03}$	$-2.5 \cdot 10^{-03}$
	DGCF	0.0621	0.0505	0.0640	0.0522	$-1.9 \cdot 10^{-03}$	$-1.7 \cdot 10^{-03}$
	LightGCN	0.0629	0.0516	0.0649	0.0530	$-2 \cdot 10^{-03}$	$-1.4 \cdot 10^{-03}$
	SGL	0.0669	0.0552	0.0675	0.0555	$-6 \cdot 10^{-04}$	$-3 \cdot 10^{-04}$
	UltraGCN	0.0672	0.0553	0.0683	0.0561	$-1.1 \cdot 10^{-03}$	$-8 \cdot 10^{-04}$
	GFCF	0.0697	0.0571	0.0697	0.0571	0	0
Amazon Book	NGCF	0.0319	0.0246	0.0337	0.0261	$-1.8 \cdot 10^{-03}$	$-1.5 \cdot 10^{-03}$
	DGCF	0.0384	0.0295	0.0399	0.0308	$-1.5 \cdot 10^{-03}$	$-1.3 \cdot 10^{-03}$
	LightGCN	0.0419	0.0323	0.0411	0.0315	$+8 \cdot 10^{-04}$	$+8 \cdot 10^{-04}$
	SGL	0.0474	0.0372	0.0478	0.0379	$-4 \cdot 10^{-04}$	$-7 \cdot 10^{-04}$
	UltraGCN	0.0688	0.0561	0.0681	0.0556	$+7 \cdot 10^{-04}$	$+5 \cdot 10^{-04}$
	GFCF	0.0710	0.0584	0.0710	0.0584	0	0

*Results are not provided since SGL was not originally trained and tested on Gowalla [78].

2. Scalability Challenges

Graph-based models, such as NGCF and LightGCN, have high computational costs, especially on large datasets like Amazon Book. The increased complexity in encoding negative feedback exacerbates this issue, making real-world deployment challenging.

3. Contextual Data Dependency

The study demonstrates that results vary significantly across datasets. For instance, while LightGCN outperforms RP3β on Gowalla, it underperforms on Yelp 2018 (Recall@20 of 0.0639 vs. 0.0647), highlighting domain-specific limitations.

Challenging the Myth of Graph Collaborative Filtering: a Reasoned and Reproducibility-driven Analysis - 2023: **Specific Takeaways for Implementation in the Current Study**

Limitations:

- **Scalability of Graph Models**
  Graph-based models, such as NGCF and LightGCN, demonstrate high computational costs, particularly on large-scale datasets like Amazon Book. This limits their deployment in real-world systems with massive user-item interactions.
- **Reproducibility Issues**
  As shown in the replicability study, there are significant discrepancies between reported and reproduced results, with performance shifts of up to -2.5% in Recall. This highlights challenges in ensuring consistent benchmarking.
- **Contextual Dependency of Results**
  The effectiveness of models is highly dependent on dataset characteristics. For instance, UltraGCN performs well on Gowalla but falls behind RP3β on Yelp 2018. This suggests a lack of generalizability across different domains or regions.
-

Ideas to improve:

- **Optimization for Scalability**
  Employ graph sampling techniques, approximate nearest neighbor search, or distributed frameworks to handle computational complexity and enable scalability to larger datasets.
- **Standardized Benchmarks for Reproducibility**
  Introduce a unified experimental pipeline with open-source implementations and clearly defined evaluation protocols to minimize discrepancies in reported results.
- **Hybrid Approaches for Generalizability**
  Combine graph-based models with classical methods (e.g., RP3β + UltraGCN) to leverage the strengths of both. This could improve performance across datasets with varying characteristics.
- **Realistic Negative Feedback Modeling**
  Move beyond synthetic negative feedback by incorporating real-world signals (e.g., dwell time, skipped recommendations) to better capture user preferences and interactions.

Multi-behavior Self-supervised Learning for Recommendation - 2023: **Architecture**

Loss functions:

Main loss: Loss for recommendation is based on the non-sampling recommendation approach.

$$\mathcal{L}_{rec}^{k} = \sum_{u\in\mathcal{B}}\sum_{i\in\mathcal{I}_u^{k+}}\left((c_i^{k+}-c_i^{k-})(\hat{x}_{u,i}^{k})^2 - 2c_i^{k+}\hat{x}_{u,i}^{k}\right)$$

$$+\sum_{m=1}^{d}\sum_{n=1}^{d}\left((\mathbf{e}_k^{(m)}\mathbf{e}_k^{(n)})(\sum_{u\in\mathcal{B}}\mathbf{e}_{u,k}^{(m)}\mathbf{e}_{u,k}^{(n)})(\sum_{i\in\mathcal{I}}\mathbf{e}_{i,k}^{(m)}\mathbf{e}_{i,k}^{(n)})\right),$$

It takes into account: Interaction of users and elements. Weight coefficients depending on the types of behavior.

Auxiliary tasks (SSL): Inter-behavior SSL:

$$\mathcal{L}_{ssl_1,user}^{K,k} = \sum_{u\in\mathcal{U}} -\log\frac{\exp(\phi(\mathbf{e}_{u,K},\mathbf{e}_{u,k})/\tau)}{\sum_{v\in\mathcal{U}\setminus FN(u)}\exp(\phi(\mathbf{e}_{u,K},\mathbf{e}_{v,k})/\tau)},$$

Contrastive loss that minimizes the distance between representations related to similar nodes in the graph (different types of behavior).
Intra-behavior SSL :

$$\mathcal{L}_{ssl_2,user} = \sum_{u\in\mathcal{U}} -\log\frac{\exp(\phi(\mathbf{e}_{u,K}^{1},\mathbf{e}_{u,K}^{2})/\tau)}{\sum_{v\in\mathcal{U}}\exp(\phi(\mathbf{e}_{u,K}^{1},\mathbf{e}_{v,K}^{2})/\tau)}.$$

And finally:

$$L = L_{rec} + \sum_{k=1}^{K-1}\mu_k L_{ssl}^{k} + L_{ssl2}$$

Optimization - Hybrid gradient manipulation:

Correction of gradient direction

$$\mathbf{G}_{aux,i}^{t} = \mathbf{G}_{aux,i}^{t} - \frac{\mathbf{G}_{aux,i}^{t}\cdot\mathbf{G}_{tar}^{t}}{||\mathbf{G}_{tar}^{t}||^2}\mathbf{G}_{tar}^{t}.$$

Balancing of gradients

$$\mathbf{G}_{aux,i}^{t} = r*\frac{||\mathbf{G}_{tar}^{t}||}{||\mathbf{G}_{aux,i}^{t}||}\mathbf{G}_{aux,i}^{t} + (1-r)*\mathbf{G}_{aux,i}^{t}.$$

Graph operations:

Propagation of embeddings in a graph

$$\mathbf{e}_{u,k}^{(l+1)} = \text{LeakyRelu}(\mathbf{W}^{(l)}\cdot\text{mean}(\{\mathbf{e}_{i,k}^{(l)}\odot\mathbf{e}_{k}^{(l)} : i\in\mathcal{N}_{u,k}\})),$$

Modeling dependencies between behavior

$$\mathbf{a}_{u,k} = \text{softmax}((\mathbf{W}_2^k)^T\tanh((\boldsymbol{e}_u\mathbf{W}_1^k)^T))$$

Algorithm 1: The Hybrid Manipulation on Gradients.

Input: Initial shared parameters θ_0; Relax factor r; Learning rate η; Auxiliary task loss $\mathcal{L}_{aux,i} \in \{\mathcal{L}_{ssl1}^{K,1}, ..., \mathcal{L}_{ssl1}^{K,K-1}, \mathcal{L}_{ssl2}\}$; Target task loss $\mathcal{L}_{tar} = \mathcal{L}_{rec}$;
Output: Updated shared parameters θ_T

for $t = 1$ *to* T **do**
  $\mathbf{G}_{tar}^{t} = \nabla_\theta\mathcal{L}_{tar}^{t}$
  for $i = 1$ *to* K **do**
    $\mathbf{G}_{aux,i}^{t} = \nabla_\theta\mathcal{L}_{aux,i}^{t}$
    if $||\mathbf{G}_{aux,i}^{t}|| > ||\mathbf{G}_{tar}^{t}||$ **then**
      $\mathbf{G}_{aux,i}^{t} \leftarrow \mathbf{G}_{aux,i}^{t} - \frac{\mathbf{G}_{aux,i}^{t}\cdot\mathbf{G}_{tar}^{t}}{||\mathbf{G}_{tar}^{t}||^2}\mathbf{G}_{tar}^{t}$
      $\mathbf{G}_{aux,i}^{t} \leftarrow r*\frac{||\mathbf{G}_{tar}^{t}||}{||\mathbf{G}_{aux,i}^{t}||}\mathbf{G}_{aux,i}^{t} + (1-r)*\mathbf{G}_{aux,i}^{t}$
    end
  end
  $\mathbf{G}^{t} \leftarrow \mathbf{G}_{tar}^{t} + \mathbf{G}_{aux,1}^{t} + ... + \mathbf{G}_{aux,K}^{t}$
  $\theta_t \leftarrow \theta_{t-1} - \eta*\mathbf{G}^{t}$
end
return θ^T

Multi-behavior Self-supervised Learning for Recommendation - 2023: **Dataset & Results**

Datasets: Beibei, Taobao, Tmall, IJCAI-Contest, Videos; Eval metrics: NDCG@K,Recall@K

Table 1: The statistics of datasets.

Dataset	#User	#Item	#Interactions	Interaction Behavior Type
Beibei	21,716	7,977	3,338,068	{Page View, Cart, Purchase}
Taobao	48,749	39,493	1,952,931	{Page View, Cart, Purchase}
Tmall	31,882	31,232	1,451,219	{Page View, Favorite, Cart, Purchase}
IJCAI-Contest	17,435	35,920	799,368	{Page View, Favorite, Cart, Purchase}
Videos	29,197	23,251	2,002,201	{Click, Like, Comment, Download }

1) Performance Comparison

Our model MBSSL generally outperforms all the baselines on all datasets, with the improvements over the best baseline ranging from 18.64% to 19.71% in terms of Recall@10 and NDCG@10. The significant improvements can be attributed to two main reasons: i) Through inter-behavior and intra-behavior SSL, our model effectively addresses the data sparsity under the target behavior and noisy interactions from auxiliary behaviors, which are two key problems of multi-behavior recommendation. ii) The proposed hybrid manipulation method on gradients empowers the capability of balancing the optimization between the SSL task and the main supervised recommendation task, which further preserves the recommendation performance.

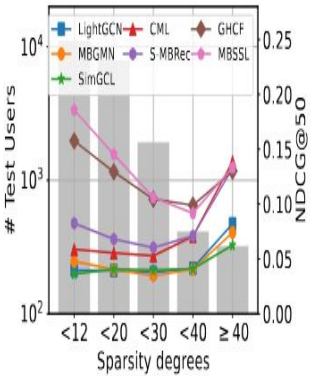
Interestingly, we find that some multi-behavior models (e.g., MATN, MBGMN) may perform worse than some single-behavior models which treat all the behaviors as the same type

2) Ablation Study

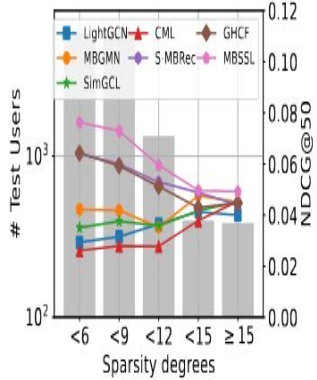
Table 3: The experimental results of ablation study.

Data	Beibei		Taobao	
Metric	Recall@10	NDCG@10	Recall@10	NDCG@10
w/o CDM	0.1923	0.1059	0.0936	0.0520
w/o SSL_{inter}	0.2025	0.1108	0.0902	0.0515
w/o SSL_{intra}	0.1994	0.1027	0.0943	0.0526
w/o HMG	0.2086	0.1131	0.0936	0.0530
MBSSL	**0.2229**	**0.1277**	**0.1027**	**0.0576**

3) Robustness Analysis



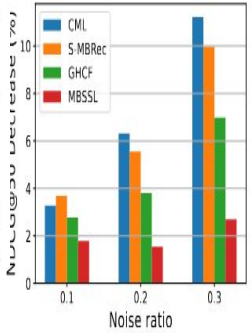
(a) Beibei NDCG@50



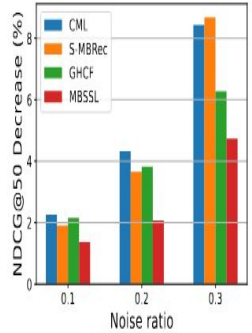
(b) Taobao NDCG@50

4) Impact of Hybrid Strategies

Data	Beibei		Taobao	
Metric	Recall@10	NDCG@10	Recall@10	NDCG@10
Strategy A	0.2081	0.1155	0.0950	0.0543
Strategy B	0.2196	0.1247	0.0994	0.0567
Strategy C	0.2114	0.1217	0.0942	0.0543
MBSSL	**0.2229**	**0.1277**	**0.1027**	**0.0576**



(a) Beibei



(b) Taobao

Multi-behavior Self-supervised Learning for Recommendation - 2023: Limitations & Ideas to improve

Limitations:

- Gradients may conflict with the primary task, which can degrade model performance. This is especially noticeable when the gradient scale and direction are very different. Auxiliary

- Approach uses fixed weights for auxiliary tasks. These weights do not automatically adapt to changes in the data or during training.

- Interactions between users and items are modeled as simple pairs, which may not be sufficient for complex behavior scenarios, such as those where temporal dependencies or group actions are important.

- The model does not account for more complex graph structures that could better describe the relationships between different behaviors.

- Data Quality Limitation

Ideas to improve:

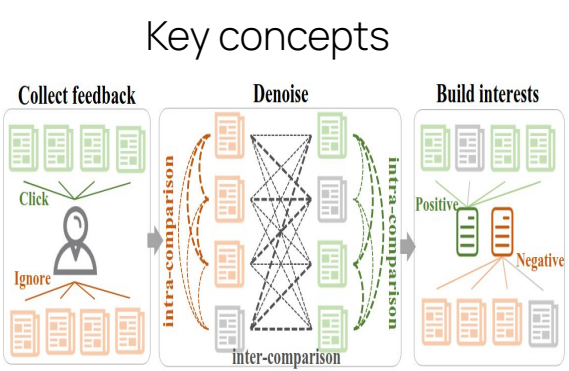
- Develop a more intelligent gradient management mechanism to avoid conflicts between auxiliary tasks and the main task, for example using GradNorm or Meta-Learning methods for adaptive task weighting.

- Enable dynamic optimization of the weights of auxiliary tasks depending on the learning progress or the structure of the data. This will help to balance the importance of tasks at different stages of learning.

- Extend the model to account for temporal patterns in the data, for example using Temporal Graph Networks or Sequential Attention mechanisms.

- Integration with Heterogeneous Graphs

Denoising Neural Network for News Recommendation with Positive and Negative Implicit Feedback - 2023: **Architecture**



Collect feedback:
- Positive examples (clicks).
- Negative examples (ignoring).

Denoising process:
- Intra-comparison and inter-comparison are used.

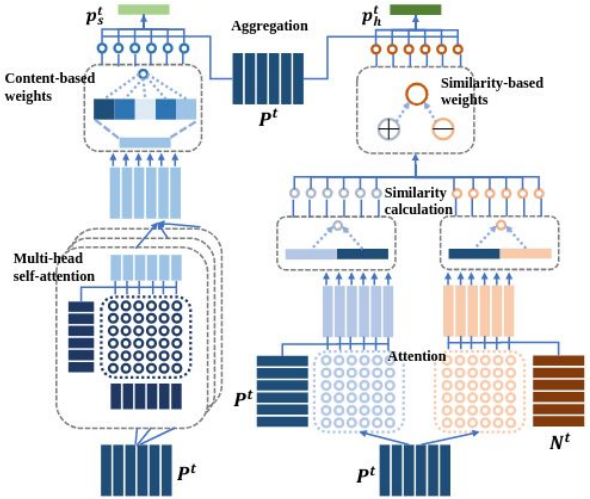
Build interests:
- Classification of feedback into positive and negative.

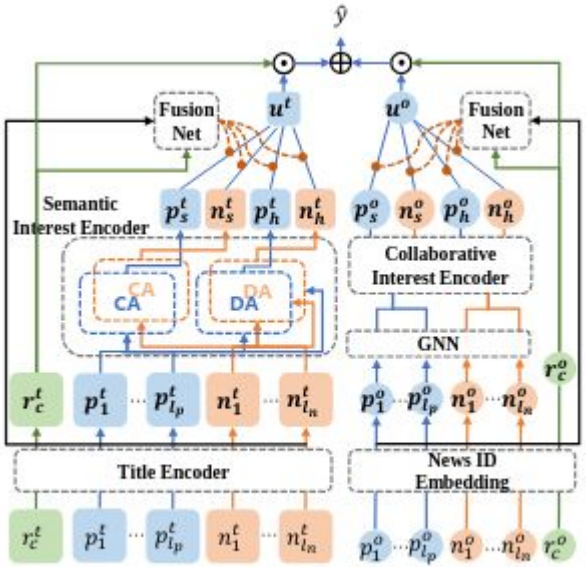
(This is a data preprocessing step to identify noise and improve the quality of the model)

Semantic Interest Encoder - contains two modules:
- CA (Content Aggregator)
- DA (Denoising Aggregator)

Collaborative Interest Encoder - uses graph neural networks to discover hidden dependencies between news items.

Fusion Net - combines the outputs from Semantic Interest Encoder and Collaborative Interest Encoder to form the final recommendations.





Aggregators in Semantic Interest Encoder
Content-based Aggregator (CA):
Uses content-based weights to analyze news. Includes multi-head attention to select the most relevant news. Denoising Aggregator (DA): Considers similarity-based weights to remove noise in preferences. CA and DA together form user representations, but with different approaches: one analyzes content, the other analyzes noise.

Denoising Neural Network for News Recommendation with Positive and Negative Implicit Feedback - 2023: Dataset & Results

Datasets: MIND; Eval metrics: AUC, MRR, nDCG@5, nDCG@10

1. Performance Comparison

Models	AUC	MRR	nDCG@5	nDCG@10
LibFM	60.48	26.38	27.75	34.63
DeepFM	62.18	27.26	29.08	35.68
DKN	64.00	28.98	31.49	38.22
LSTUR	65.31	30.31	33.34	39.86
NPA	64.35	29.61	32.88	39.23
DEERS	65.29	30.78	33.78	40.09
DFN	63.11	29.14	31.88	38.33
GERL	64.08	29.34	32.50	38.96
NAML	65.84	30.60	33.89	40.23
NRMS	65.46	30.73	33.78	40.13
NAML + TCE	65.95	30.66	33.93	40.52
NRMS + TCE	65.84	31.58	34.93	41.26
DRPN	**67.30**	**32.68**	**36.27**	**42.33**

Table 2: Performance comparison of all methods. Best results are highlighted in bold.

"First, our proposed model, DRPN, outperforms all baselines on the news recommendation datasets. Second, among all baselines, the methods which use the deep neural networks to model the news (i.e., DKN, NPA, LSTUR, DFN, DEERS, NAML, GERL, NRMS) perform better than the feature-based methods (e.g., LibFM and DeepFM)."

2. Ablation Study

Models	AUC	MRR	nDCG@5	nDCG@10
DRPN	**67.30**	**32.68**	**36.27**	**42.33**
DRPN-D	66.51	31.09	35.40	41.63
DRPN-G	66.82	31.90	35.34	41.62
DRPN-DG	66.14	31.17	34.57	40.97
DRPN-N	66.38	31.39	34.79	41.13
DRPN-P	65.89	31.11	34.38	40.59

Table 3: Performance comparison of all variants of DRPN. Best results are highlighted in bold.

- DRPN (full model) gives maximum metric values.
- Removing any modules (for example, denoising aggregator) leads to a decrease in performance.

Denoising Neural Network for News Recommendation with Positive and Negative Implicit Feedback - 2023: Limitations & Ideas to improve

Limitations:

- The use of complex components such as attention mechanisms and denoising makes it difficult to explain recommendations to the user.

- DRPN is computationally intensive due to its complex structure including GNNs, denoising mechanisms and multi-task learning.

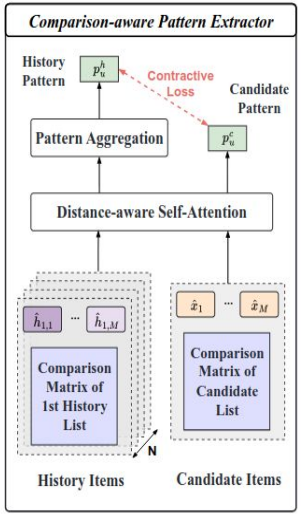
Ideas to improve:

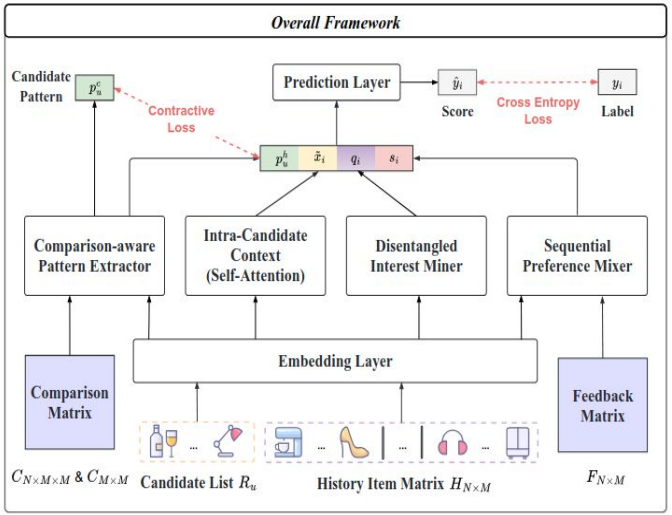
- Add temporal dependencies to the data processing (e.g. using Temporal Graph Networks) to better account for changes in user preferences.

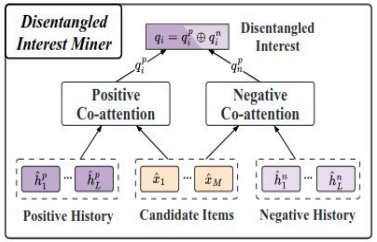
- Include additional data such as text descriptions of news items, images or metadata to improve understanding of user interests.

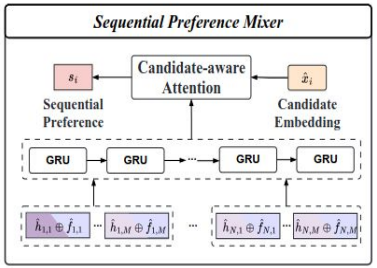
- Simplify the model architecture to improve its scalability, for example by using lightweight GNN (LightGCN) or optimizing graph computing.

Beyond Positive History: Re-ranking with List-level Hybrid Feedback - 2024: **Architecture**









1. Disentangled Interest Miner
an attention mechanism to extract significant features from the interaction history.

2. Sequential Preference Mixer (SPM):
recurrent neural networks (RNN) for modeling temporal dependencies.

$$s_i = \sum_{j=1}^{NM} w_j \tilde{h}_j^s = \sum_{j=1}^{NM} a(\hat{x}_i, \tilde{h}_j^s)\tilde{h}_j^s,$$

3. Comparison-aware Pattern Extractor (CPE):
An attention mechanism for identifying the importance of items in a list.

$$\mathcal{L}_{info} = -\frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}} \log \frac{\exp(p_u^c \cdot p_u^h/\tau)}{\exp(\sum_{u'\in\mathcal{U}} p_{u'}^c \cdot p_u^h/\tau)}.$$

4. Prediction and Optimization

The optimization function for model can be divided into two parts. One part is the utility loss, which enhances the accuracy of the re-ranking, and the other is the pattern contractive loss concerning user behavior patterns. We adopt info NCE loss for pattern contractive loss and aim to reduce the distance between the user's historical pattern and the candidate pattern, thereby making the user's behavior on the candidate set more similar to their historical behavior.

$$\mathcal{L}_{util} = -\frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\sum_{i=1}^{M} y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i), \qquad \mathcal{L} = \mathcal{L}_{util} + \beta\mathcal{L}_{info},$$

Beyond Positive History: Re-ranking with List-level Hybrid Feedback - 2024: Dataset & Results

Datasets: PRM public, MovieLens-20M; Eval metrics: MAP@K, NDCG@K, Click@K

Table 1: Overall performance on two benchmark datasets. The best result is given in bold, while the second-best value i underlined. The symbol * indicates statistically significant improvement over the best baselines(t-test with $p < 0.05$).

Model	PRM Public						MovieLens-20M					
	@5			@10			@5			@10		
	MAP	NDCG	Click	MAP	NDCG	Click	MAP	NDCG	Click	MAP	NDCG	Click
DLCM	0.3074	0.2930	0.6180	0.3236	0.3456	0.8853	0.7327	0.6836	2.7447	0.6953	0.7956	3.9795
SAR	0.3092	0.2965	0.6414	0.3245	0.3546	0.9362	0.7276	0.6751	2.7176	0.6879	0.7911	3.9562
PRM	0.3173	0.2958	0.6174	0.3290	0.3465	0.8794	0.7373	0.6896	2.7621	0.7004	0.7989	3.9752
SetRank	0.3246	0.3055	0.6442	0.3360	0.3636	0.9431	0.7324	0.6834	2.7425	0.6957	0.7955	3.9705
SRGA	0.3196	0.3026	0.6356	0.3366	0.3567	0.9069	0.7378	0.6893	2.7675	0.7001	0.7989	3.9884
DFN	0.3243	0.3083	0.6531	0.3386	0.3636	0.9352	0.7488	0.7015	2.7968	0.7112	0.8065	4.0022
RACP	0.3169	0.3019	0.6411	0.3344	0.3579	0.9250	0.7502	0.7016	2.7990	0.7115	0.8070	4.0017
PEAR	0.3133	0.2987	0.6388	0.3281	0.3571	0.9369	0.7446	0.6965	2.7809	0.7070	0.8036	3.9892
MIR	0.3284	0.3113	0.6546	0.3431	0.3663	0.9353	0.7473	0.6987	2.7872	0.7088	0.8050	3.9999
PIER	0.3235	0.3096	0.6586	0.3414	0.3682	0.9554	0.7492	0.7019	2.7996	0.7113	0.8067	3.9994
RELIFE	**0.3393***	**0.3225***	**0.6820***	**0.3551***	**0.3805***	**0.9792***	**0.7585***	**0.7117***	**2.8292***	**0.7202***	**0.8130***	**4.0122**

Table 2: Comparison of RELIFE and its variants. The best result is given in bold, while the second-best value is underlined.

Model	PRM Public						MovieLens-20M					
	@5			@10			@5			@10		
	MAP	NDCG	Click	MAP	NDCG	Click	MAP	NDCG	Click	MAP	NDCG	Click
RELIFE-DIM	0.3358	0.3195	0.6756	0.3513	0.3769	0.9659	0.7513	0.7039	2.8064	0.7134	0.8080	4.0071
RELIFE-CPE	0.3332	0.3184	0.6733	0.3494	0.3756	0.9627	0.7507	0.7035	2.8053	0.7124	0.8075	4.0072
RELIFE-SPM	0.3388	0.3219	0.6810	0.3538	0.3760	0.9551	0.7563	0.7083	2.8138	0.7175	0.8112	4.0096
RELIFE-ICC	0.3353	0.3162	0.6611	0.3495	0.3656	0.9132	0.7568	0.7102	2.8265	0.7182	0.8116	4.0040
RELIFE-CL	0.3288	0.3154	0.6735	0.3453	0.3744	0.9759	0.7490	0.7022	2.8041	0.7117	0.8067	4.0055
RELIFE-PAT	0.3388	**0.3236**	**0.6846**	0.3545	**0.3817**	0.9785	0.7578	0.7115	2.8276	0.7194	0.8125	**4.0158**
RELIFE	**0.3393**	0.3225	0.6820	**0.3551**	0.3805	**0.9792**	**0.7585**	**0.7117**	**2.8292**	**0.7202**	**0.8130**	4.0122

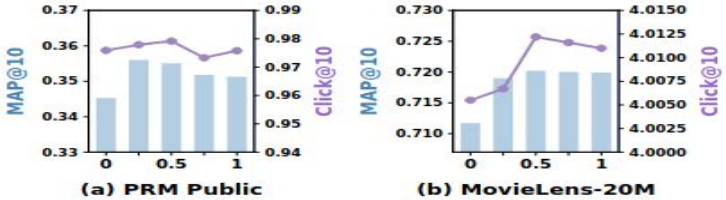
Table 1 - The RELIFE model shows the best results on two datasets: PRM Public and MovieLens-20M. RELIFE statistically significantly (p<0.05) outperforms most existing methods (e.g. DFN, RACF, PEAR).

Table 2 - An analysis of RELIFE components through its variants (e.g. RELIFE-DIM, RELIFE-CPE) shows that each module contributes to performance.

Figure 3 - Number of historical lists: Increasing the number of historical lists from improves MAP and Click@10 metrics, but further increases (to 5 and 6 lists) lead decreased performance.

Figure 4 - Weight β of InfoNCE loss: The optimal value β=0.5 leads to the highes results for both metrics (MAP@10 and Click@10). Too small (β=0.1) or too large values reduce efficiency.

Figure 3: The impact of the number of history lists.



Beyond Positive History: Re-ranking with List-level Hybrid Feedback - 2024: Limitations & Ideas to improve

Limitations:

- RELIFE outperforms on two datasets (PRM Public and MovieLens-20M), but the model may show less consistent results on other datasets that have different data structures or user behavior
- RELIFE requires more inference time (61.10 ms on PRM Public) and uses more GPU memory (10.04 G) compared to its competitors. This may be a problem for resource-constrained systems or when processing in real-time
- Scalability

Ideas to improve:

- Automate parameter selection, such as the β weight for InfoNCE Loss, using Bayesian optimization or gradient search methods.
- Extend the model to account for temporal aspects of user preferences, such as Temporal Graph Networks or Transformer architectures for temporal sequences.
- Implement optimized architectures, such as LightGCN or sparse computing, to reduce inference time and memory consumption.
- Apply distillation knowledge to simplify the model without significant performance penalties.

TempGNN: Temporal Graph Neural Networks for Dynamic Session-Based Recommendations - 2023: Architecture *there is no negative feedback

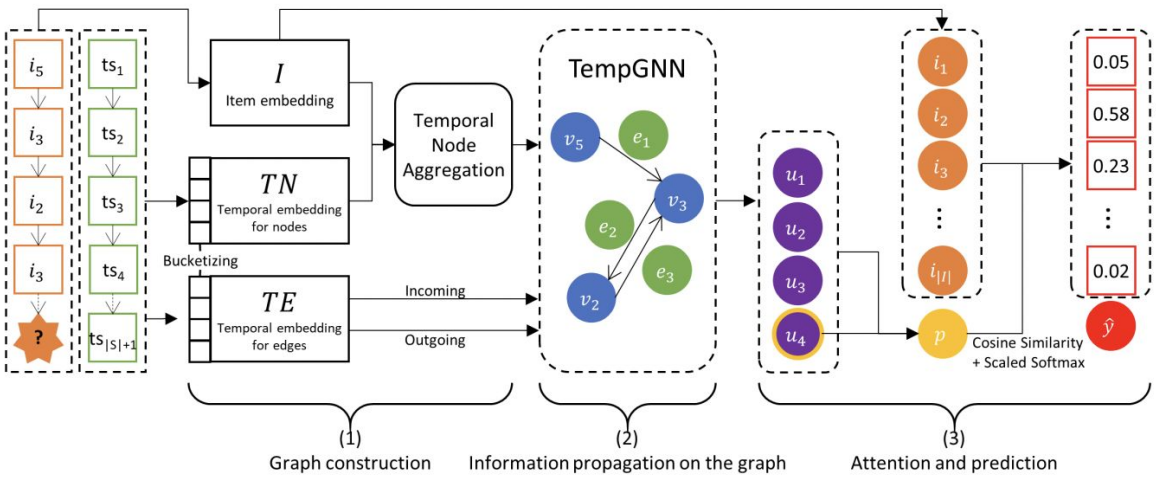


Figure 3: The overall workflow of our model, TempGNN. It consists of three main processes. It takes a prediction timing as input and outputs recommended probabilities for candidate items.

$$\mathcal{L} = -\sum_{j=1}^{|I|} y[j] log\,(\hat{y}[j]),$$

where $y[j] \in \{0,1\}$ is a target that indicates whether the next click is the j-th item or not.

The information exchange between nodes is guided by:

- Temporal Embedding for Edges (TE): Time intervals influence the weight of information shared between nodes.
- Temporal Node Aggregation (TN): Encodes how distant a click is from the prediction point, using gated mechanisms to combine item and time features.

TempGNN: Temporal Graph Neural Networks for Dynamic Session-Based Recommendations - 2023: Dataset & Results

Datasets: Yoochoose, Diginetica;

Table 3: Performance of temporal embedding methods. Q means quantile bucketizing for time, A is an activation function, and G is a gate network when applying temporal embeddings.

	Yoochoose 1/64		Yoochoose 1/4		Diginetica	
	R@20	M@20	R@20	M@20	R@20	M@20
Base	71.90	32.58	72.99	33.31	55.85	19.67
Position	71.86	31.84	72.88	32.24	55.68	19.43
Constant	72.27	32.57	73.38	33.75	55.85	19.76
Bucket	72.49	32.93	73.42	33.46	55.94	19.83
Q	72.57	33.44	**73.53**	34.08	55.90	19.83
Q+A	72.56	33.50	73.47	34.07	56.00	19.82
Q+G	72.43	33.43	73.52	34.08	56.07	19.91
Q+A+G	**72.60**	**33.58**	73.52	**34.19**	**56.08**	**19.96**

Table 2: Overall performance for three datasets. A bold-faced number indicates the best score and the second performer is underlined in each column.

		Yoochoose 1/64				Yoochoose 1/4				Diginetica			
		R@20	M@20	R@5	M@5	R@20	M@20	R@5	M@5	R@20	M@20	R@5	M@5
RNN-based	GRU4Rec	62.03	23.34	37.04	20.74	67.63	27.32	42.69	24.71	34.25	9.45	14.71	7.58
	CSRM	70.20	29.77	46.05	27.20	70.50	29.23	45.37	26.58	51.51	17.20	26.55	14.76
Attention-based	STAMP	68.64	29.89	45.65	27.47	70.62	30.36	46.53	27.83	47.66	15.54	24.16	13.25
	SR-IEM	70.86	31.59	47.95	29.16	71.02	30.49	46.69	27.92	51.70	17.14	26.46	14.66
GNN-based	SR-GNN	70.38	30.71	47.08	28.26	71.39	30.96	47.07	28.40	51.46	17.54	26.94	15.11
	NISER+	71.36	31.91	48.21	29.46	72.74	32.09	48.82	29.55	54.39	19.20	29.15	16.70
	SGNN-HN	71.88	31.94	48.40	29.46	72.92	32.69	48.78	30.13	55.56	19.44	29.72	16.88
	TempGNN	**72.60**	**33.58**	**48.88**	**31.09**	**73.52**	**34.19**	**49.62**	**31.67**	**56.08**	**19.96**	**30.25**	**17.39**

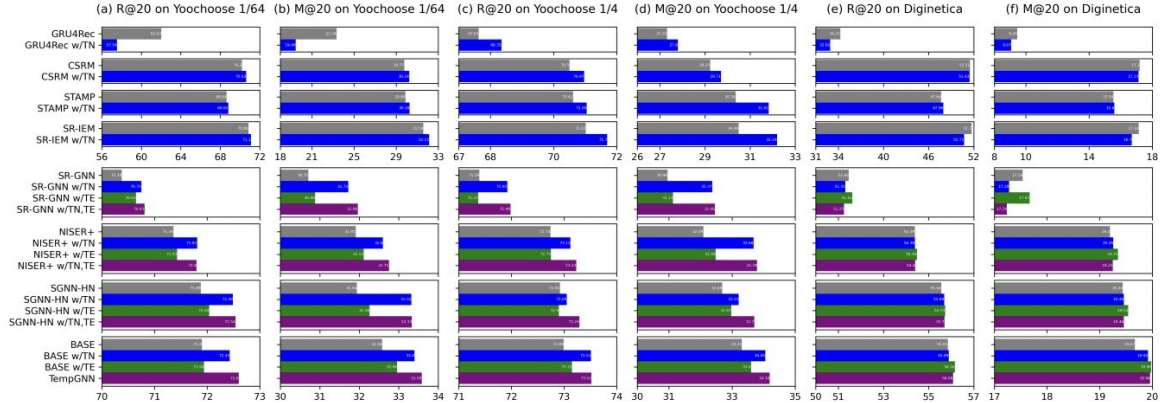


Figure 4: Performance of baseline models with temporal embedding using three datasets. Gray bars indicate the results of basic models. The results of models with TN are shown as blue bars, the ones of models with TE are shown as green bars, and purple bars indicate performance when both are used together.

TempGNN: Temporal Graph Neural Networks for Dynamic Session-Based Recommendations - 2023: Limitations & Ideas to improve

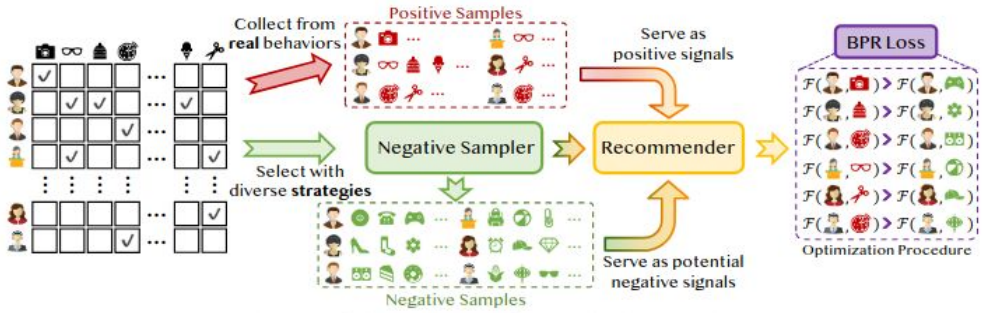
Limitations:

- **Discrete Temporal Bucketization:** TempGNN employs discrete bucketization to represent temporal differences, which loses granularity and may fail to capture subtle variations in time-sensitive interactions.
- **Dataset-Specific Sensitivity:** TempGNN's performance varies significantly between datasets, particularly in its reliance on temporal embeddings.
- O**ver-Simplified Temporal-Item Aggregation:** The gating mechanism (Equation 3) aggregates item and temporal embeddings linearly, assuming a fixed relationship, which may fail to capture complex temporal dependencies or nonlinear interactions.
- **Limited Long-Term Dependency Modeling:** TempGNN focuses on short-term session-based recommendations but does not address long-term user behavior or preferences.
- **Temporal Bias in Popular Items:** Temporal embedding strategies may amplify biases toward frequently interacted items due to fixed bucket allocation and uneven item distributions.

Ideas to improve:

- Replace discrete bucketization with continuous temporal embeddings (e.g., Time2Vec, sinusoidal functions).
- Dynamically allocate temporal buckets based on data distribution (e.g., clustering timestamps or frequency-aware binning).
- Add a memory network or dual-network framework to capture both short-term session dynamics and long-term user preferences.

Negative Sampling in Recommendation: A Survey and Future Directions - 2024: Description of the methodology in the context of negative feedback



Negative Sampling and Negative Feedback:

1. The article distinguishes between implicit negative feedback (e.g. "not viewed") and explicit negative feedback (e.g. "dislike").
2. Explicit Negative Feedback: is considered a more reliable source of information, as it clearly indicates a lack of interest in the item.
3. To deal with explicit feedback, the article emphasizes the need for an adaptive approach to account for different types of negative interactions.
4. Negative sampling is discussed as a method that selects items for training a model from implicit negative signals, balancing model complexity and recommendation accuracy.

Metrics for assessing negative feedback:

1.The impact of negative examples on ranking metrics such as nDCG and MAP is considered.

2.The need to adapt loss functions to account for explicit negative signals is noted (e.g., weighted loss for different feedback categories).

Negative Sampling Categories:

1. Random Sampling: selects random items from those that have not interacted. A simple method, but often noisy.
2. Popularity-based Sampling: selects popular items as negative examples, as they are highly likely to have been seen but have not interested the user.
3. Hard Negative Mining: identifies the most difficult examples (e.g. items that the user could have potentially selected but did not).
4. Contrastive Negative Sampling: selects elements to minimize the similarity between positive and negative examples in the embedding space.

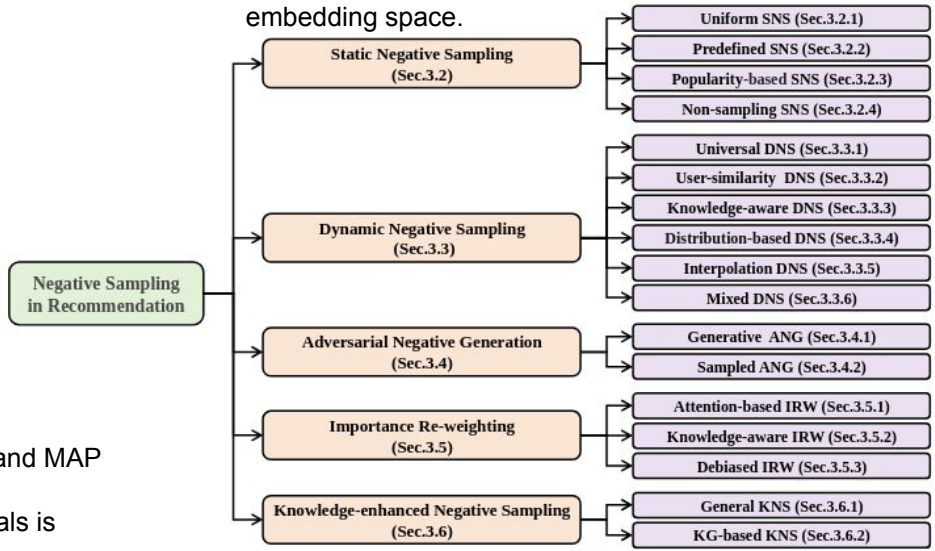


Fig. 3. Overall Ontology of Negative Sampling Strategies in Recommendation.

[Negative Sampling in Recommendation: A Survey and Future Directions - 2024]: Positive and negative aspects regarding negative feedback

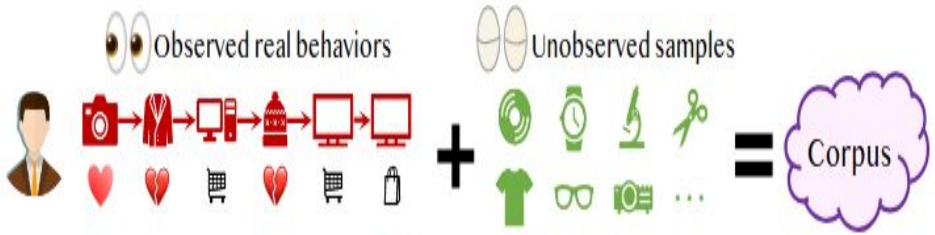


Fig. 4. An example of the real user behaviors, where denotes "like", denotes "dislike", denotes "add to cart", denotes "purchase", denotes "observed real behaviors" and denotes "unobserved samples".

Positive points:

- The article emphasizes the differences between explicit and implicit negative feedback, which is important for systems working with different data sources. The value of explicit negative feedback is especially emphasized, which allows for a more accurate understanding of user preferences.
- Systematization of approaches to negative sampling, including random, popularity-based, hard negative mining, allows for adapting the method to different scenarios.
- The usefulness of hard negatives is discussed, which are of interest to models, as they help to train subtle differences between relevant and irrelevant elements.

Negative points:

- The article mentions that explicit negative feedback is less common and often depends on the user context, which may limit its use in practical scenarios.
- Explicit and implicit feedback are described separately, but there is no detailed elaboration of their combined use within a single model.
- Negative sampling methods are mainly for tabular and sequential data, without an emphasis on the use of graph structures, which are especially important in the context of GNN.

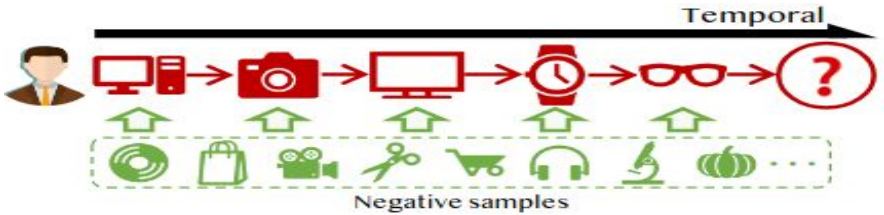


Fig. 6. An illustration of sequential recommendation.

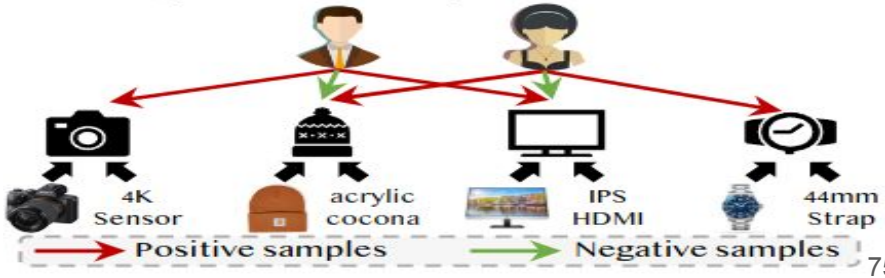


Fig. 7. An illustration of multi-modal recommendation.

Negative Sampling in Recommendation: A Survey and Future Directions - 2024: Limitations & Ideas to improve

Useful ideas:

- Using hard negatives in graphs can help to better identify irrelevant relationships (e.g. users who have not explicitly interacted with certain nodes but have high similarity on other parameters).
- Treating explicit negative feedback as labels on graph edges (e.g. "rejected item") can improve the representation of user preferences. Possibility of using weighted edges to account for explicit and implicit negative feedback.
- Contrastive Negative Sampling can be adapted for graph structures by minimizing the similarity between positive and negative nodes in the embedding space.

What can be taken from the methodology:

- Using weighted loss functions so that explicit negative feedback has a greater impact on model training.
- Adaptive selection of negative examples based on the graph structure and current node embeddings.

Ideas to improve:

- It is necessary to work out how existing strategies (hard negative mining, popularity-based sampling) can be applied to graph nodes and edges.
- Using both types of signals to train graph neural networks, for example, through separate embeddings or attention layers.

Towards Automated Negative Sampling in Implicit Recommendation - 2023: **Architecture**

The **scoring function** is $S(u, i \mid W)$

Loss Function:

$$\mathcal{L}(\mathbf{D}^p, \mathbf{D}^n, W) = -\sum_{(u,i)\in\mathbf{D}^p}\sum_{(u,j)\in\mathbf{D}_u^n} \ln\sigma\left(S(u,i|W) - S(u,j|W)\right),$$

PROPOSITION 1. *Given the set of mutually independent samplers* $\{\pi_t(u,k)|t=1,2,\ldots,T\}$, *we have*

$$\mathbb{E}_{\{\pi_t(u,\alpha_t k)|\forall u,t\}}\mathcal{L}(\mathbf{D}^p, \mathbf{D}^n, W) = \sum_t \alpha_t \mathbb{E}_{\{\pi_t(u,k)|\forall u\}}\mathcal{L}(\mathbf{D}^p, \hat{\mathbf{D}}_t^n, W) \quad (6)$$

Negative Sampler:

$$\mathbf{p}_u(j) > 0, \quad \sum_{j\in\mathbf{I}-\mathbf{I}_u} \mathbf{p}_u(j) = 1.$$

Automated Negative Sampling:

$$\min_{W,\boldsymbol{\alpha}} \quad \mathcal{L}(\mathbf{D}^p, \mathbf{D}^n, W), \quad (4)$$

$$\text{s.t.} \quad \mathbf{D}^n = \cup_u \mathbf{D}_u^n, \quad \mathbf{D}_u^n = \cup_t \mathbf{D}_{u,t}^n, \quad \forall u,$$
$$\mathbf{D}_{u,t}^n \text{ is generated by } \pi_t(u, \alpha_t k), \quad \forall u, t \quad (5)$$
$$\sum_{t=1}^{T} \alpha_t = 1 \quad \text{and} \quad \alpha_t \geq 0, \ \forall t.$$

AutoSample:

$$\mathcal{L}(\mathbf{D}^p, \mathbf{D}^n, W) = \sum_t \alpha_t \mathcal{L}(\mathbf{D}^p, \hat{\mathbf{D}}_t^n, W) \approx \sum_{t=1} p_t \mathcal{L}(\mathbf{D}^p, \hat{\mathbf{D}}_t^n, W), \quad (12)$$

Search Process Optimization.
In AutoSample, two sets of parameters need to be optimized:
the model parameter W contained in scoring function $S(\cdot, \cdot)$ and the search parameter α . Note that pt is directly generated by the Gumbel-Softmax operation. In the naive DARTS setting, two parameters are iteratively trained over different batches. However, in the implicit recommendation, the dataset only contains positive interactions, and it is hard to directly apply such a method. Inspired by the previous work, we jointly train both parameters over the same training batch.

Algorithm 1 The Optimization of Search Process

Require: Implicit dataset $\mathbf{D}^p$
Ensure: the well-trained search parameter $\boldsymbol{\alpha}^*$ and model parameter W'

```
while not converged do
    Sample a mini-batch of training data B^p from the implicit dataset D^p
    Generate negative mini-batch B^n given the current search parameter α and candidate samplers Π
    Calculate the loss L(B^p, B^n, W) given Equation 9
    Update model parameter W by descending gradient ∇_W L(B^p, B^n, W)
    Update search parameter α by descending gradient ∇_α L(B^p, B^n, W)
end while
```

Algorithm 2 The Re-training Process

Require: Implicit dataset $\mathbf{D}^p$, optimal search parameter $\boldsymbol{\alpha}^*$, initialization model parameter W'
Ensure: the well-trained model parameter W^*

```
select the optimal negative sampler π* given the optimal search parameter α* and candidate samplers Π
Re-initialize the model with parameter W'
while not converged do
    Sample a mini-batch of training data B^p from the implicit dataset D^p
    Generate negative mini-batch B^n given sampler π*
    Calculate the loss L(B^p, B^n, W) given Equation 3
    Update model parameter W by descending gradient ∇_W L(B^p, B^n, W)
end while
```

Retraining Process. After obtaining the optimal probability for each sampler $\alpha*$, we retrain the model with the best-performing
parameter W 'as initialization and freeze $\alpha*$ as the search result for the negative sampler. The idea for keeping the model parameter instead of dropping them as typical neural architecture search methods would is that as α gradually converge to the optimal, the model parameter W is also adaptively trained from easy negative instances to hard ones, similar to curriculum learning. It has also been proven that through careful selection of the initialization parameter, a deep recommender system can achieve better performance than random initialized

Towards Automated Negative Sampling in Implicit Recommendation - 2023: **Dataset & Results**

Datasets: Taobao2014, Taobao2015, Amazon; Eval metrics: Recall@K(R@K), NDCG@K(N@K), Precision@K(P@K) and Hit Ratio@K(H@K).

	Sampler	Taobao2014					Taobao2015				
		R@20	N@20	P@20	H@20	Rank	R@20	N@20	P@20	H@20	Rank
MF	RNS	0.0238(2)	0.0247(1)	0.0155(1)	0.2403(1)	1.25	0.0627(2)	0.0412(2)	0.0094(2)	0.1701(2)	2.25
	PNS	0.0220(4)	0.0212(4)	0.0128(4)	0.2013(4)	4	0.0421(6)	0.0248(6)	0.0065(6)	0.1170(6)	6
	DNS	0.0093(6)	0.0093(6)	0.0059(6)	0.1044(6)	6	0.0556(4)	0.0406(4)	0.0082(4)	0.1516(4)	4
	AOBPR	0.0111(5)	0.0115(5)	0.0073(5)	0.1261(5)	5	0.0516(5)	0.0375(5)	0.0077(5)	0.1417(5)	5
	SRNS	0.0221(3)	0.0221(3)	0.0142(3)	0.2207(3)	3	0.0615(3)	0.0417(2)	0.0092(3)	0.1669(3)	2.75
	AutoSample	0.0243*(1)	0.0242(2)	0.0153(2)	0.2355(2)	1.75	0.0688*(1)	0.0449*(1)	0.0103*(1)	0.1854*(1)	1
LightGCN	RNS	0.0327(1)	0.0328(1)	0.0205(2)	0.2959(2)	1.5	0.0696(4)	0.0479(4)	0.0105(4)	0.1873(4)	4
	PNS	0.0267(5)	0.0264(5)	0.0170(5)	0.2535(5)	5	0.0652(5)	0.0454(5)	0.0097(5)	0.1757(5)	5
	DNS	0.0090(7)	0.0091(7)	0.0057(7)	0.1018(7)	6	0.0554(7)	0.0405(7)	0.0081(7)	0.1510(7)	7
	AOBPR	0.0133(6)	0.0144(6)	0.0099(6)	0.1628(6)	6	0.0591(6)	0.0426(6)	0.0088(6)	0.1602(6)	6
	SRNS	0.0316(3)	0.0307(3)	0.0192(3)	0.2847(3)	3	0.0716(3)	0.0489(3)	0.0107(3)	0.1917(3)	3
	MixGCF	0.0288(4)	0.0297(4)	0.0186(4)	0.2755(4)	4	0.0755(1)	0.0504(1)	0.0113(1)	0.2010(1)	1
	AutoSample	0.0321(2)	0.0323(2)	0.0206(1)	0.2982*(1)	1.5	0.0725(2)	0.0493(2)	0.0108(2)	0.1932(2)	2
NGCF	RNS	0.0270(3)	0.0263(2)	0.0173(2)	0.2609(2)	2	0.0572(4)	0.0356(4)	0.0087(3)	0.1557(2)	3
	PNS	0.0268(4)	0.0245(4)	0.0153(4)	0.2378(4)	4	0.0395(7)	0.0225(6)	0.0062(6)	0.1118(5)	5.75
	DNS	0.0088(7)	0.0089(7)	0.0056(7)	0.1025(7)	7	0.0556(5)	0.0407(3)	0.0082(4)	0.1517(3)	3.5
	AOBPR	0.0100(6)	0.0108(6)	0.0076(6)	0.1305(6)	6	0.0452(6)	0.0273(5)	0.0068(5)	0.1260(4)	4.75
	SRNS	0.0227(5)	0.0206(5)	0.0129(5)	0.2023(5)	5	0.0594(3)	0.0390(3)	0.0089(3)	0.1615(3)	3
	MixGCF	0.0274(2)	0.0259(3)	0.0166(3)	0.2501(3)	3	0.0614(2)	0.0429(2)	0.0092(2)	0.1671(2)	2
	AutoSample	0.0308*(1)	0.0293*(1)	0.0187*(1)	0.2747*(1)	1	0.0706*(1)	0.0481*(1)	0.0106*(1)	0.1904*(1)	1

	Sampler	Alibaba					Amazon				
		R@20	N@20	P@20	H@20	Rank	R@20	N@20	P@20	H@20	Rank
MF	RNS	0.0398(3)	0.0178(3)	0.0024(2)	0.0462(3)	3	0.0310(3)	0.0140(3)	0.0018(2)	0.0355(3)	2.75
	PNS	0.0410(2)	0.0187(2)	0.0024(2)	0.0475(2)	2	0.0317(2)	0.0147(2)	0.0018(2)	0.0358(2)	2
	DNS	0.0201(4)	0.0084(4)	0.0013(4)	0.0253(4)	4	0.0217(4)	0.0073(5)	0.0012(4)	0.0244(4)	4.25
	AOBPR	0.0115(6)	0.0047(6)	0.0007(5)	0.0144(5)	5.5	0.0161(6)	0.0066(6)	0.0009(6)	0.0177(6)	6
	SRNS	0.0124(5)	0.0056(5)	0.0007(5)	0.0142(6)	5.25	0.0198(5)	0.0098(4)	0.0011(5)	0.0222(5)	4.75
	AutoSample	0.0465*(1)	0.0211*(1)	0.0028*(1)	0.0543*(1)	1	0.0354*(1)	0.0155*(1)	0.0020(1)	0.0391*(1)	1
LightGCN	RNS	0.0630(2)	0.0284(3)	0.0038(1)	0.0735(2)	2	0.0416(2)	0.0182(3)	0.0023(2)	0.0465(2)	2
	PNS	0.0587(4)	0.0270(4)	0.0035(4)	0.0688(4)	4	0.0303(5)	0.0141(4)	0.0017(5)	0.0341(5)	4.75
	DNS	0.0234(7)	0.0090(7)	0.0015(7)	0.0290(7)	7	0.0219(6)	0.0072(6)	0.0012(6)	0.0246(6)	6
	AOBPR	0.0336(6)	0.0149(6)	0.0022(6)	0.0423(6)	6	0.0309(4)	0.0131(5)	0.0018(4)	0.0352(4)	4.25
	SRNS	0.0479(5)	0.0222(5)	0.0030(5)	0.0575(5)	5	0.0173(7)	0.0069(7)	0.0010(7)	0.0202(7)	7
	MixGCF	0.0630(2)	0.0305(1)	0.0037(3)	0.0721(3)	2.25	0.0403(3)	0.0189(2)	0.0022(3)	0.0444(3)	2.75
	AutoSample	0.0636*(1)	0.0289(2)	0.0038(1)	0.0745*(1)	1.25	0.0447*(1)	0.0203*(1)	0.0025*(1)	0.0499*(1)	1
NGCF	RNS	0.0464(3)	0.0204(3)	0.0028(2)	0.0548(3)	2.75	0.0269(5)	0.0114(5)	0.0015(5)	0.0304(5)	5
	PNS	0.0369(4)	0.0160(4)	0.0021(4)	0.0422(4)	4	0.0289(4)	0.0128(4)	0.0016(4)	0.0324(4)	4
	DNS	0.0209(5)	0.0090(5)	0.0013(5)	0.0259(5)	5	0.0220(7)	0.0070(7)	0.0012(7)	0.0247(7)	7
	AOBPR	0.0120(7)	0.0051(7)	0.0008(7)	0.0150(7)	7	0.0222(6)	0.0091(6)	0.0013(6)	0.0258(6)	6
	SRNS	0.0188(6)	0.0086(6)	0.0011(6)	0.0220(6)	6	0.0316(3)	0.0132(3)	0.0018(3)	0.0349(3)	3
	MixGCF	0.0473(2)	0.0234(2)	0.0028(2)	0.0550(2)	2	0.0331(1)	0.0148(1)	0.0026(1)	0.0507(1)	1
	AutoSample	0.0485*(1)	0.0246*(1)	0.0029(1)	0.0566*(1)	1	0.0329(2)	0.0137(2)	0.0019(2)	0.0367(2)	2

Table 4: Ablation over the Number of Negative Samples on Taobao2015 Dataset

	Sampler	T = 2					T = 3				
		R@20	N@20	P@20	H@20	Rank	R@20	N@20	P@20	H@20	Rank
MF	RNS	0.0629(2)	0.0419(2)	0.0094(2)	0.1706(2)	2	0.0647(2)	0.0437(2)	0.0097(2)	0.1753(2)	2
	PNS	0.0366(5)	0.0214(5)	0.0057(5)	0.1031(5)	5	0.0388(5)	0.0229(5)	0.0060(5)	0.1079(5)	5
	DNS	0.0554(3)	0.0405(3)	0.0081(3)	0.1510(3)	3	0.0555(3)	0.0405(3)	0.0081(3)	0.1511(3)	3
	AOBPR	0.0538(4)	0.0389(4)	0.0079(4)	0.1469(4)	4	0.0525(4)	0.0387(4)	0.0078(4)	0.1445(4)	4
	AutoSample	0.0688*(1)	0.0449*(1)	0.0103*(1)	0.1854*(1)	1	0.0663*(1)	0.0455*(1)	0.0099(1)	0.1793*(1)	1
LightGCN	RNS	0.0692(2)	0.0478(2)	0.0103(2)	0.1856(2)	2	0.0695(1)	0.0479(1)	0.0104(1)	0.1867(1)	1
	PNS	0.0536(5)	0.0343(5)	0.0082(4)	0.1461(5)	4.75	0.0499(5)	0.0307(5)	0.0077(5)	0.1365(5)	5
	DNS	0.0554(4)	0.0405(4)	0.0081(5)	0.1509(4)	4.25	0.0553(4)	0.0405(4)	0.0081(4)	0.1509(4)	4
	AOBPR	0.0643(3)	0.0449(3)	0.0095(3)	0.1736(3)	3	0.0661(2)	0.0456(2)	0.0098(2)	0.1780(2)	2
	AutoSample	0.0725*(1)	0.0493*(1)	0.0108*(1)	0.1932*(1)	1	0.0644(3)	0.0449(3)	0.0095(3)	0.1732(3)	3
NGCF	RNS	0.0573(2)	0.0362(3)	0.0086(2)	0.1555(2)	2.25	0.0571(2)	0.0360(4)	0.0087(2)	0.1554(2)	2.5
	PNS	0.0399(5)	0.0229(5)	0.0063(5)	0.1115(5)	5	0.0427(5)	0.0248(5)	0.0067(5)	0.1188(5)	5
	DNS	0.0553(3)	0.0405(2)	0.0081(4)	0.1506(3)	3	0.0554(4)	0.0406(2)	0.0081(4)	0.1512(3)	3.25
	AOBPR	0.0545(4)	0.0350(4)	0.0082(3)	0.1494(4)	3.75	0.0548(3)	0.0363(3)	0.0082(3)	0.1502(4)	3.75
	AutoSample	0.0706*(1)	0.0481*(1)	0.0106*(1)	0.1904*(1)	1	0.0638*(1)	0.0437*(1)	0.0094*(1)	0.1725*(1)	1

* denotes statistically significant improvements over baselines with $p < 0.05$.

Table 6: Ablation over Retraining Scheme on Taobao2015 and Amazon Datasets.

	Metric	Taobao2015		Amazon	
		random	*custom*	*random*	*custom*
MF	R@20	0.0556	0.0688	0.0281	0.0354
	N@20	0.0406	0.0449	0.0123	0.0155
	P@20	0.0082	0.0103	0.0016	0.0020
	H@20	0.1516	0.1854	0.0323	0.0391
LightGCN	R@20	0.0696	0.0725	0.0416	0.0447
	N@20	0.0479	0.0493	0.0182	0.0203
	P@20	0.0105	0.0108	0.0023	0.0025
	H@20	0.1873	0.1932	0.0465	0.0499
NGCF	R@20	0.0553	0.0706	0.0269	0.0329
	N@20	0.0405	0.0481	0.0114	0.0137
	P@20	0.0081	0.0106	0.0015	0.0019
	H@20	0.1509	0.1904	0.0304	0.0367

Towards Automated Negative Sampling in Implicit Recommendation - 2023: Limitations & Ideas to improve

Limitations:

- AutoSample relies on a predefined set of candidate negative samplers, limiting its ability to explore novel or more adaptive sampling strategies that could better match specific dataset characteristics or dynamic user behaviors.
- While AutoSample improves performance, the iterative search process for optimal sampling strategies introduces significant computational overhead, especially on large-scale datasets like Amazon or Taobao2015. This limits scalability for real-time or resource-constrained environments.

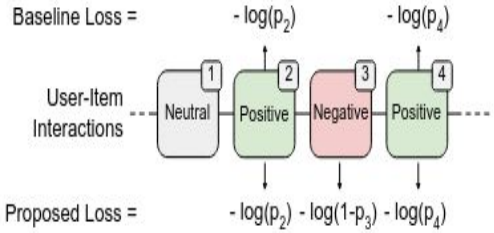
Ideas to improve:

- Replace the static set of negative samplers with a dynamically evolving search space that can adapt to the data distribution and user interaction patterns. Incorporating newer sampling techniques like contrastive sampling or using generative approaches (e.g., GANs) to create "harder" negative samples in real time.
- Approximate search methods (e.g., Bayesian Optimization or Reinforcement Learning) to reduce the cost of exploring sampling strategies.
- Parallelization or distributed training to handle large-scale datasets more efficiently.
-

Learning from Negative User Feedback and Measuring Responsiveness for Sequential Recommenders - 2023:

Architecture

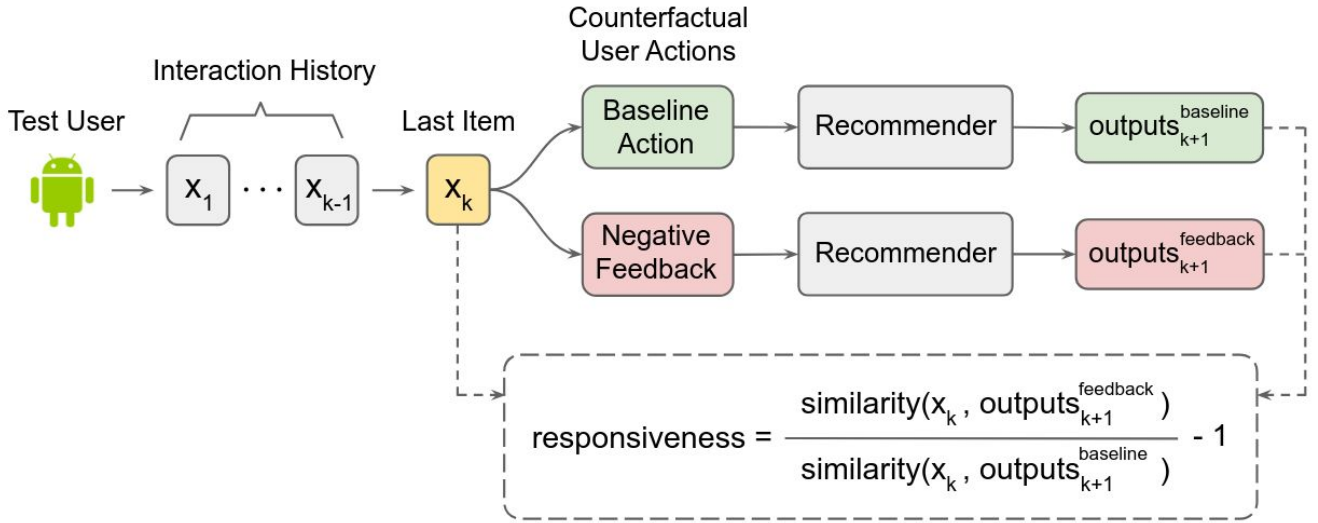
The **scoring function** is S(u,i)
Loss Function:



Proposed model loss:

$$\mathcal{L} = -\sum_{i \in D_{pos}} \left(r_i \cdot \log(p(y_i|s_i)) \right) - \sum_{i \in D_{neg}} \left(w_i \cdot \log(1 - p(y_i|s_i)) \right), \quad (1)$$

Negative Training Labels and Input Features. Both explicit and implicit negative user feedback can serve as negative training labels. The label weights can be tuned according to the feedback strength, signal density, and the relative loss value magnitude compared to positive labels.



The "not-to-recommend" loss function addresses several practical issues of modeling negative user feedback in the training objective. For example, using cross-entropy loss Li = −log(p (yi |si)) with negative-valued label weights could also reduce the probability of recommending unwanted items, but this leads to gradient blow-up when p (yi |si) → 0. In principle, reinforcement learning objectives support assigning negative reward values to negative training labels. In fact, we could replace the loss term for positive labels by a REINFORCE objective [5]. However, for negative labels, using negative rewards in REINFORCE recommenders faces a practical challenge, where gradient blow-up may still occur when p (yi |si) → 0 even after off-policy correction due to the extremely large item spaces in industry settings. The "not-to-recommend" loss function circumvents these issues as the gradient is guaranteed to be finite when p (yi |si) → 0. An alternative approach is to include and upweight negative feedback items among the softmax negative samples of adjacent positive labels.

Results

Live Experiments: Results

Explicit Negative Feedback: homepage

1.1. Incorporating dislikes as both input features and training labels:

1. **Dislike rate:** Reduced by **2.44%** (baseline comparison).
2. **Repeated dislikes:** Decreased by **9.60%** for the same creator.
3. **Dismissals:** Reduced by **2.05%**, while viewers only dropped by **0.14%**.

1.2. euristic-based solution (excluding disliked items from input sequence): Only **-0.84%** dislike reduction, far less effective than the proposed model.

1.3. Significant user experience improvement observed.

Implicit Negative Feedback.Skip signals were used as negative labels for the immersive feed.

- **Overall user enjoyment:** Increased by **0.40%** (Fig 2b).
- **Daily active users (1+ hour of activity):** Increased by **0.61%**.
- **Repeated skips:** Reduced by **1.44%** for the same creator.
- **Content diversity:** Improved by **0.88%**.

Measuring Responsiveness by Counterfactual Simulation

Objective: Measure how well the model responds to explicit negative feedback by reducing similar recommendations after a user provides negative feedback.

Key Components:

Simulated users with controlled counterfactual actions (e.g., disliking vs. not disliking an item).

Measures:

Content-based similarity:

- Proportion of recommendations from the same content cluster.

Creator-based similarity:

- Proportion of recommendations from the same creator.

Results:

1. **Baseline Model: No responsiveness** to dislikes. Relied on downstream system components for handling dislikes.
2. **Using Dislikes as Input Feature Only:** Moderate reduction in similar recommendations:
   - **Content similarity:** -22.7%.
   - **Creator similarity:** -22.8%.
3. **Using Dislikes in Training Labels + Input Features:** Significant reduction in similar recommendations:
   - **Content similarity:** -60.8%.
   - **Creator similarity:** -64.1% (Fig 3b).

Key Observations:

- Explicit negative feedback reduces undesirable interactions (e.g., dislikes, skips).
- Implicit feedback signals enhance engagement and enjoyment through broader coverage of negative user interests.

Learning from Negative User Feedback and Measuring Responsiveness for Sequential Recommenders - 2023:

Limitations & Ideas to improve

Limitations:

- Explicit negative feedback, such as dislikes, is highly sparse. In live experiments, reliance on such signals led to improvements in dislike reduction (e.g., 2.44% on the homepage), but the limited density of this data may reduce model generalizability across diverse user bases and scenarios.
- much of the user experience still relied on downstream ranking components and heuristics. For example, similarity-based improvements (-60.8% content similarity, -64.1% creator similarity) depended on integration with the entire pipeline, diluting the standalone contribution of retrieval.

Ideas to improve:

- Use implicit actions like skips, short dwell time, or rapid scrolls as proxies for explicit negative feedback to supplement sparse data. This was effective in experiments (+0.40% enjoyment, -1.44% repeated skips)
- Introduce diversity penalties or rewards in the training objective to encourage broader content exposure. For example, optimizing for diversity metrics alongside relevance could enhance the 0.88% diversity gain observed in implicit feedback experiments.
- Incorporate ranking-stage optimizations into the retrieval stage to address dependencies on downstream components. Jointly train retrieval and ranking models to better align user interests throughout the recommendation pipeline.

Enhancing Sequential Music Recommendation with Negative Feedback-informed Contrastive Learning - 2024:

Architecture

There are 4 SOTA models in experiment:

1. SASRec (Self-Attentive Sequential Recommendation):

1. Uses self-attention mechanisms to capture sequential dependencies between user-item interactions.
2. Highlights temporal and session-based relationships within user interactions.

2. BERT4Rec:

1. Adapts the BERT transformer architecture for sequential recommendation.
2. Captures bidirectional dependencies within user sessions for more accurate predictions.

3. GRU4Rec:

1. A recurrent neural network (RNN)-based approach using gated recurrent units (GRUs).
2. Models sequential dependencies between items in a user session.

4. CASER (Convolutional Sequence Embedding Recommendation):

1. A convolutional neural network (CNN)-based model for sequence-aware recommendations.
2. Learns both high-order and low-order patterns in user-item interactions.

Loss Function: Multi-objective optimization combining **ranking loss** and **feedback modeling loss**

$$\mathcal{L}_{NCE} = -\mathbb{E}_X \left[\log \frac{f_k\left(\mathbf{i}_m, \hat{\mathbf{i}}_{t+1}\right)}{f_k\left(\mathbf{i}_m, \hat{\mathbf{i}}_{t+1}\right) + \sum_{n_j \in N} f_k\left(n_j, \hat{\mathbf{i}}_{t+1}\right)} \right] \quad (1)$$

Enhancing Sequential Music Recommendation with Negative Feedback-informed Contrastive Learning - 2024:

Dataset & Results

Datasets: Spotify, Last.fm; Eval metrics: Hit Rate @{1, 5, 10, 20}, Precision @10

Data	Metric	SASRec		BERT4Rec		GRU4Rec		CASER		WRMF			BPR		
		orig.	*ours*	*orig.*	*ours*	*orig.*	*ours*	*orig.*	*ours*	*orig.*	*-BL*	*-NR*	*orig.*	*-BL*	*-NR*
MSSD	HR@1	.377	**.410 (9%)**	.204	.355 **(74%)**	.210	.235 **(12%)**	.223	.251 **(13%)**	.203	.207	.211	.208	.213	.216
	HR@5	.615	**.628 (2%)**	.450	.608 **(35%)**	.398	.431 **(8%)**	.412	.455 **(10%)**	.400	.406	.411	.406	.413	.421
	HR@10	.696	**.706 (1%)**	.553	.693 **(25%)**	.491	.534 **(9%)**	.518	.530 **(2%)**	.488	.498	.505	.502	.510	.517
	HR@20	.767	**.774 (1%)**	.648	.767 **(18%)**	.600	.627 **(5%)**	.616	.649 **(5%)**	.597	.603	.608	.605	.609	.612
	MAP@10	.397	**.417 (5%)**	.185	.369 **(99%)**	.176	.193 **(10%)**	.195	.221 **(13%)**	.149	.156	.164	.160	.168	.173
LFM-2B	HR@1	.190	**.221 (16%)**	.101	.117 **(16%)**	.096	.102 **(6%)**	.102	.112 **(10%)**	.097	.098	.102	.098	.105	.107
	HR@5	.371	**.400 (8%)**	.227	.248 **(9%)**	.203	.221 **(9%)**	.197	.208 **(6%)**	.138	.142	.147	.143	.152	.169
	HR@10	.452	**.477 (6%)**	.292	.320 **(10%)**	.273	.291 **(7%)**	.281	.302 **(7%)**	.269	.273	.279	.276	.286	.294
	HR@20	.532	**.553 (4%)**	.366	.394 **(8%)**	.311	.342 **(10%)**	.326	.354 **(9%)**	.305	.311	.316	.314	.324	.348
	MAP@10	.188	**.219 (16%)**	.098	.110 **(12%)**	.078	.085 **(9%)**	.081	.088 **(9%)**	.062	.065	.067	.066	.072	.078
LFM-1K	HR@1	.152	**.181 (19%)**	.069	.086 **(25%)**	.048	.059 **(23%)**	.052	.071 **(37%)**	.042	.044	.047	.043	.050	.052
	HR@5	.301	**.330 (10%)**	.207	.230 **(11%)**	.182	.200 **(10%)**	.188	.198 **(5%)**	.139	.146	.177	.150	.153	.159
	HR@10	.392	**.421 (7%)**	.299	.320 **(7%)**	.261	.289 **(11%)**	.269	.293 **(9%)**	.270	.285	.294	.292	.298	.301
	HR@20	.478	**.491 (3%)**	.413	.433 **(5%)**	.388	.397 **(2%)**	.390	.404 **(4%)**	.346	.368	.375	.369	.374	.376
	MAP@10	.092	**.107 (16%)**	.049	.064 **(31%)**	.034	.042 **(24%)**	.037	.040 **(8%)**	.028	.031	.034	.032	.036	.038

Table 1: ***Hit Ratio @ [1, 5, 10, 20]*** **and** ***Mean Average Precision @ 10*** **for the sequential models (SASRec, BERT4Rec, GRU4Rec, CASER) and the non-sequential baselines (WRMF, BPR) on the three datasets. Non** ***"orig."*** **models incorporate negative feedback. They are compared to their** ***"orig."*** **baselines which do not model negative feedback. Numbers in parentheses show the relative increase in the percentage of the approach over the respective baseline; bold entries mark the better performing approach between the baseline and negative feedback-informed approach. The overall best performance is also highlighted in bold (i.e., SASRec-ours).**

Metric		SASRec		BERT4Rec	
		orig.	*ours*	*orig.*	*ours*
MRR@10	MSSD	.960	.950 **(-1%)**	.840	.950 **(13%)**
MRR@10	LFM-2B	.969	.911 **(-6%)**	.953	.950 **(-0.3%)**
MRR@10	LFM-1K	.731	.731 **(0%)**	.540	.460 **(-15%)**

Table 2: ***Mean Reciprocal Rank @ 10*** **on skip targets for the best sequential models (SASRec, BERT4Rec) on the three datasets. Lower values are better.**

Self-attentive architectures (SASRec, BERT4Rec) outperform recurrent and convolutional models (GRU4Rec, CASER), particularly in streaming-based datasets.

Dataset Characteristics Impact Performance - algorithmic content in MSSD aligns well with the feedback-informed loss, while user-curated datasets (LFM-1K) show less improvement.

For MSSD, BERT4Rec may unintentionally **model biases in the underlying algorithm, leading to overfitting.**

The results emphasize the effectiveness of incorporating explicit negative feedback for sequential recommendation models. SASRec consistently achieves the best performance, while bidirectional approaches like BERT4Rec face challenges in certain scenarios. Dataset characteristics (e.g., skip rates, curation styles) play a crucial role in determining model success.

Limitations & Ideas to improve

Limitations:

- BERT4Rec, which employs bidirectional training objectives, exhibits inconsistent performance gains. For instance, it struggles on LFM datasets compared to MSSD, where its results are exceptional. This indicates potential misalignment between bidirectional objectives and sequential recommendation tasks.
- In datasets like MSSD with high proportions of algorithmically recommended content, models like BERT4Rec may inadvertently learn biases from the underlying recommendation algorithms, leading to unintended feedback loops.

Ideas to improve:

- Develop methods to better interpret negative feedback signals. For example, incorporate context (e.g., track genre, session type) to distinguish between dissatisfaction and exploratory skips.
- Pretrain models on diverse datasets with varying characteristics (e.g., streaming-based, user-curated) to improve generalization across datasets with different proportions of negative feedback.

Negative Feedback for Music Personalization - 2024: **Architecture**

There are 2 SOTA models in experiment:

1. SASRec (Self-Attentive Sequential Recommendation):

1. Uses self-attention mechanisms to capture sequential dependencies between user-item interactions.
2. Highlights temporal and session-based relationships within user interactions.

2. BERT4Rec:

1. Adapts the BERT transformer architecture for sequential recommendation.
2. Captures bidirectional dependencies within user sessions for more accurate predictions.

Loss Function: Multi-objective optimization combining **ranking loss** and **feedback modeling loss**

$$\mathcal{L}_{NCE} = -\mathbb{E}_X \left[\log \frac{f_k\left(\mathbf{i}_m, \hat{\mathbf{i}}_{t+1}\right)}{f_k\left(\mathbf{i}_m, \hat{\mathbf{i}}_{t+1}\right) + \sum_{n_j \in N} f_k\left(n_j, \hat{\mathbf{i}}_{t+1}\right)} \right] \quad (1)$$

The baseline model uses only positive feedback as inputs (i.e.no feedback embedding) and uses $phard$ = 0 (randomly-sampled instead of user-provided hard negatives during training). Generally, these random samples are likely to be from an unrelated genre and thus are too easy for the model to rank (leading to the overconfidence issue identified by Petrov & Macdonald (2023)

Negative Feedback for Music Personalization - 2024: **Dataset & Results**

Datasets: Spotify, Piki, Pandora; Eval metrics: accuracy

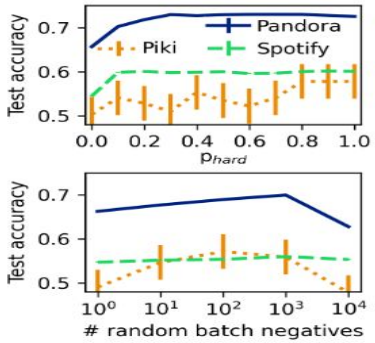


Figure 2: Top: The test accuracy generally increases with higher p_{hard}. Bottom: Test accuracy increases with more batched random negatives, but using too many hurts performance. The maximum lift is also smaller than using a true hard negative sample.

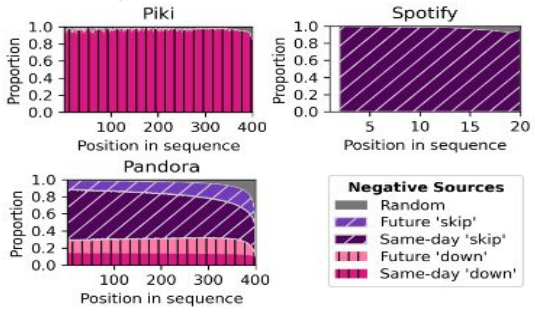


Figure 1: Proportion of sources for hard negative samples used at each position. More recent positions (higher x) are more likely to use randomly-selected negatives as there is less future feedback by definition.

Table 2: Ablation study with percentage point changes in accuracy. The Piki data is too small for any significant results and is omitted.

Method	Δ Accuracy Spotify	Pandora
No positional emb.	−0.6%	−0.7%
No hard negatives	−5.4%	−6.3%
Only positive inputs	−0.6%	−0.5%
Half max. seq. length	+0.1%	−0.2%

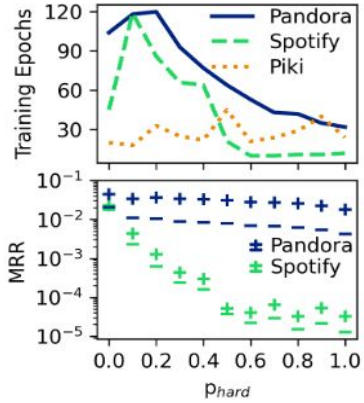


Figure 3: For higher p_{hard}, training time tends to decrease (top), while the mean reciprocal ranks of the positive (+) and negative (−) samples generally both decrease (bottom), even as the test accuracy increases (Fig. 2). The Piki dataset is too small/noisy to analyze and is omitted.

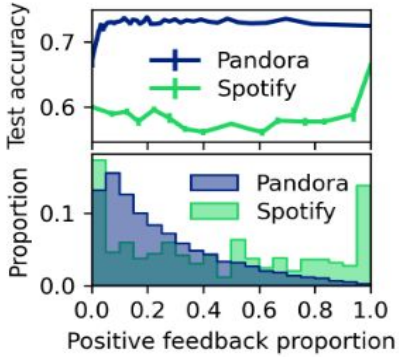


Figure 4: Test accuracy is fairly consistent across users with different proportion of feedback types (top). The distribution of feedback proportions differs by dataset (bottom).

The incorporation of negative feedback, particularly hard negatives, significantly enhances recommendation accuracy. However, the tradeoff between true positives and false positives, as well as variability in user feedback behaviors across datasets, requires tailored approaches. Differences in feedback embeddings and the effectiveness of negative sampling thresholds suggest further optimization opportunities for specific datasets and use cases.

Negative Feedback for Music Personalization - 2024: Limitations & Ideas to improve

Limitations:

- increasing phard improves test accuracy but reduces metrics like MRR and recall for positive samples, potentially leading to over-optimization on reducing false positives while sacrificing true positives.
- Models struggle with tail-end songs, as these items are less frequently encountered as random negatives, resulting in weaker embeddings and poorer recommendation quality for niche or rarely played tracks.
- Smaller datasets (e.g., Piki) suffer from insufficient explicit feedback, limiting the model's ability to fully utilize negative signals and reducing the statistical significance of observed improvements.

Ideas to improve:

- Incorporate tail-focused regularization techniques, such as oversampling rare tracks during training or using attention mechanisms to enhance embeddings for less frequently played items.
- Cluster users into segments (e.g., "mostly-skip" or "mostly-play") and apply segment-specific models or loss adjustments to account for distinct behaviors, improving personalized recommendations.

Comprehensive Analysis of Negative Sampling in Knowledge Graph Representation Learning - 2022:

Architecture

There are 2 SOTA models in experiment:

1. SASRec (Self-Attentive Sequential Recommendation):

1. Uses self-attention mechanisms to capture sequential dependencies between user-item interactions.
2. Highlights temporal and session-based relationships within user interactions.

2. BERT4Rec:

1. Adapts the BERT transformer architecture for sequential recommendation.
2. Captures bidirectional dependencies within user sessions for more accurate predictions.

Loss Function: Multi-objective optimization combining **ranking loss** and **feedback modeling loss**

$$\mathcal{L}_{NCE} = -\mathbb{E}_X \left[\log \frac{f_k\left(\mathbf{i}_m, \hat{\mathbf{i}}_{t+1}\right)}{f_k\left(\mathbf{i}_m, \hat{\mathbf{i}}_{t+1}\right) + \sum_{n_j \in N} f_k\left(n_j, \hat{\mathbf{i}}_{t+1}\right)} \right] \quad (1)$$

The baseline model uses only positive feedback as inputs (i.e.no feedback embedding) and uses $phard$ = 0 (randomly-sampled instead of user-provided hard negatives during training). Generally, these random samples are likely to be from an unrelated genre and thus are too easy for the model to rank (leading to the overconfidence issue identified by Petrov & Macdonald (2023)

Comprehensive Analysis of Negative Sampling in Knowledge Graph Representation Learning - 2022: **Dataset & Results**

Datasets: FB15k-237, WN18RR, YAGO3-10; Eval metrics: Hits@1, Hits@3, Hits@10, MRR

WN18RR

Model	Sub.	MRR	Hits@1	Hits@3	Hits@10
RESCAL	None	41.5	39.0	42.3	45.9
	Base	43.3	40.7	44.5	48.2
	Freq	**44.1**	41.1	**45.6**	**49.5**
	Uniq	**44.1**	**41.4**	45.5	**49.5**
ComplEx	None	45.0	40.9	46.6	53.4
	Base	47.1	42.8	48.9	55.7
	Freq	**47.6**	**43.3**	49.3	**56.3**
	Uniq	**47.6**	43.2	**49.5**	**56.3**
DistMult	None	42.4	38.3	43.6	51.0
	Base	43.9	39.4	45.2	53.3
	Freq	**44.6**	**40.0**	45.9	**54.4**
	Uniq	**44.6**	39.9	**46.2**	54.3
TransE	None	22.6	1.8	40.1	52.3
	Base	22.4	1.3	40.1	53.0
	Freq	23.0	1.9	40.7	**53.7**
	Uniq	**23.2**	**2.2**	**41.0**	53.4
RotatE	None	47.3	42.6	49.1	56.7
	Base	47.6	43.1	49.5	56.6
	Freq	47.8	42.9	**49.8**	**57.4**
	Uniq	**47.9**	**43.5**	49.6	56.7
HAKE	None	49.1	44.5	51.1	57.8
	Base	**49.8**	45.3	**51.6**	58.2
	Freq	49.7	45.2	51.4	**58.5**
	Uniq	**49.8**	**45.4**	51.5	58.3

FB15k-237

Model	Sub.	MRR	Hits@1	Hits@3	Hits@10
RESCAL	None	17.2	9.9	18.1	31.8
	Base	22.3	13.9	24.2	39.8
	Freq	**26.6**	17.4	**29.4**	**45.1**
	Uniq	**26.6**	**17.6**	29.3	44.9
ComplEx	None	22.4	14.0	24.2	39.5
	Base	32.2	23.0	35.1	51.0
	Freq	**32.8**	**23.6**	**36.1**	51.2
	Uniq	32.7	23.5	35.8	**51.3**
DistMult	None	22.2	14.0	24.0	39.4
	Base	**30.8**	**22.1**	**33.6**	**48.4**
	Freq	29.9	21.2	32.7	47.5
	Uniq	29.1	20.3	31.8	46.6
TransE	None	33.0	22.8	37.2	**53.0**
	Base	32.9	23.0	36.8	52.7
	Freq	**33.6**	**24.0**	**37.3**	52.9
	Uniq	33.5	23.9	**37.3**	52.8
RotatE	None	33.1	23.1	37.1	53.1
	Base	33.6	23.9	37.4	**53.2**
	Freq	**34.0**	**24.5**	**37.6**	**53.2**
	Uniq	**34.0**	**24.5**	**37.6**	53.0
HAKE	None	32.3	21.6	36.9	53.2
	Base	34.5	24.7	38.2	54.3
	Freq	34.9	25.2	38.6	54.2
	Uniq	**35.4**	**25.8**	**38.9**	**54.7**

YAGO3-10

Model	Sub.	MRR	Hits@1	Hits@3	Hits@10
TransE	None	50.6	40.9	56.6	67.7
	Base	51.2	41.5	**57.6**	**68.3**
	Freq	51.3	41.9	57.2	68.1
	Uniq	**51.4**	**42.0**	**57.6**	67.9
RotatE	None	50.6	41.1	56.5	67.8
	Base	50.8	41.8	56.5	67.6
	Freq	51.0	41.9	56.5	67.8
	Uniq	**51.5**	**42.5**	**56.8**	**68.3**
HAKE	None	53.4	44.9	58.7	68.4
	Base	54.3	46.1	59.5	69.2
	Freq	54.0	45.5	59.4	69.1
	Uniq	**55.0**	**46.6**	**60.1**	**69.8**

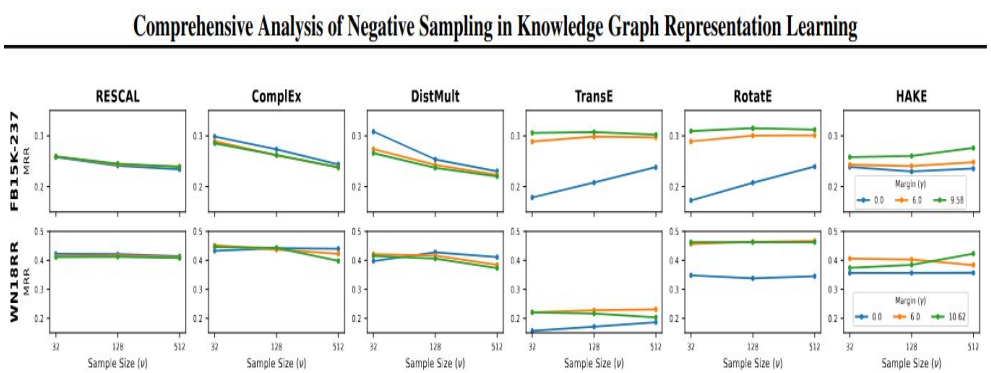


Figure 3: MRR scores for each model of each dataset when we vary negative sample size ν and margin term γ.

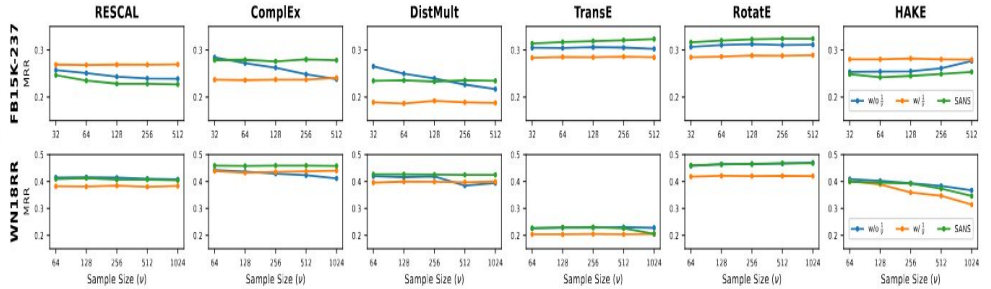


Figure 4: MRR scores for each model of each dataset when we vary negative sample size ν. w/o $\frac{1}{\nu}$ and w/ $\frac{1}{\nu}$ represent NS loss in Eqs. (2) and (3), respectively.

Comprehensive Analysis of Negative Sampling in Knowledge Graph Representation Learning - 2022:

Limitations & Ideas to improve

Limitations:

- The approach inherently relies on user-item interaction history and provides no explicit mechanism to handle new users or items without sufficient prior data.
- The architecture focuses predominantly on graph structure for augmentations and learning. This may overlook richer semantic information, such as temporal dynamics or auxiliary user/item metadata.

Ideas to improve:

- Instead of generic node/edge dropout, employ dynamic augmentations that mimic real-world interaction patterns, such as time-based edge reweighting or subgraph sampling to simulate dynamic user behaviors.
- Extend the architecture to include temporal embeddings or content-based features (e.g., user demographics or item attributes) alongside topological information

Article	Link
NFARec	github
TGT	github
KGUF	github
RNS over KG	github
RevGNN	github
SBNR	github
SIGformer	github
AGL-SC	github
MacGNN	github

DataSet	Link	Count	DataSet	Link	Count	DataSet	Link	Count	DataSet	Link	Count
Amazon-Book	link	3	MovieLens-20M	link	2	RetailRocket	link	1	Diginetica	link	1
Amazon-CDs	link	1	Last-FM	link	2	Zhihu	link	1	Spotify	link	3
Amazon-Music	link	1	MIND	link	1	WeChat	link	1	Piki	link	2
Amazon Electronics	link	1	Taobao	link	3	Frontiers-4k	link	1	FB15k-237	link	1
Yelp	link	8	Beibei	link	3	Frontiers-8k	link	1	WN18RR	link	1
citeUlike	link	1	Tmall	link	2	Adressa	link	1			
Beauty	link	1	IJCAI-Contest Data	link	2	Globo	link	1			
Recipes	link	1	Yahoo! Movies	link	1	YAGO3-10	link	1			
MovieLens	link	6	Facebook Books	link	1	Epinions	link	1			
KuaiRec	link	3	QK-article	link	1	KuaiRand	link	1			
Collection with datasets	link	-	Yoochoose	link	2	PRM public	link	1			

Online training with GNN & choosing of domain

тут нада: пара статей про реализацию онлайн, суммаризация минусов в конце, выдвижение гипотез по улучшению + ссылка на документ с датасетами + на слайде подведение доменов и перечисление выбранных датасетов

Macro Graph Neural Networks for Online Billion-Scale Recommender Systems - 2024: **Architecture**

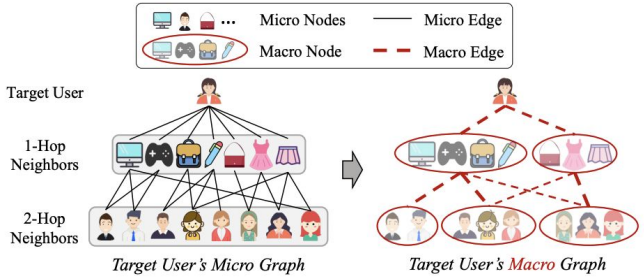


Figure 2: Sketch map of the construction of the macro graph.

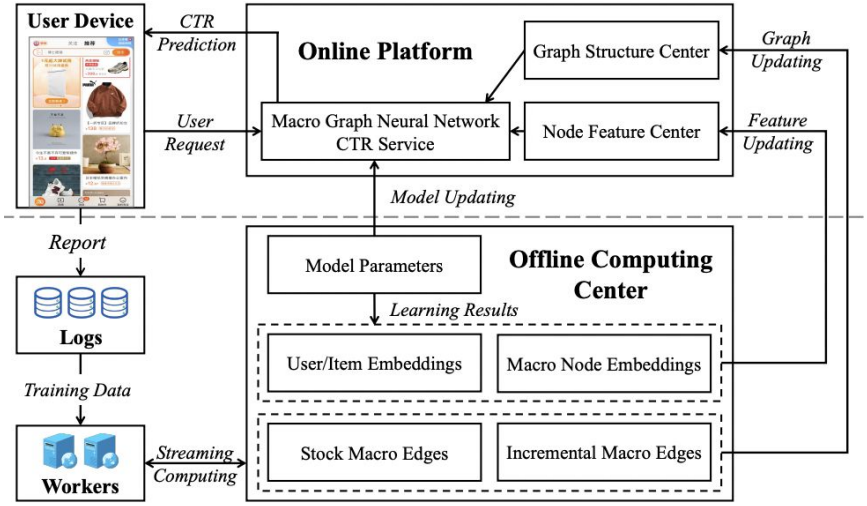


Figure 4: The system architecture for online deployment.

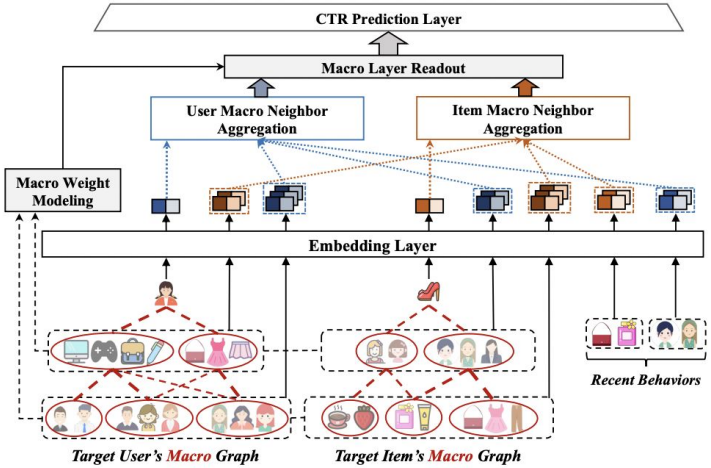


Figure 3: The model architecture of the proposed MacGNN.

MIG - MIcro Recommendation Graph: **Micro Node** - connect users with their interacted items, under this setting, users and items are treated as micro nodes; **Micro Edge** -

$$y_{ui} = \begin{cases} 1, & \text{if } u \text{ exhibits positive behavior towards } i; \\ 0, & \text{if } u \text{ exhibits negative behavior towards } i. \end{cases}$$

MAcro Recommendation Graph (MAG). Our proposed MAG can be defined as $\widetilde{\mathcal{G}} = (\widetilde{\mathcal{U}}, \widetilde{\mathcal{I}}, \widetilde{\mathcal{R}})$, where $\widetilde{\mathcal{U}}$, $\widetilde{\mathcal{I}}$, and $\widetilde{\mathcal{R}}$ are the macro user nodes, macro item nodes, and macro edges respectively, and $\widetilde{\mathcal{N}}_v^{(k)}$ represents the k^{th}-hop macro neighbors of node v. Specifically, each macro node v is associated with a trainable embedding $\widetilde{E}_v \in \mathbb{R}^d$. With MAG, MacGNN only needs to aggregate hundreds of macro nodes, significantly reducing computational complexity.

Macro Graph Neural Networks for Online Billion-Scale Recommender Systems - 2024: **Dataset & Results**

Datasets: MovieLens, Amazon Electronics, Kuaishou;

Table 2: CTR prediction comparison results over *five* trial runs (↑: the higher, the better; ↓: the lower, the better). The best baseline(s) are highlighted with underlining.

Model	MovieLens			Electronics			Kuaishou		
	AUC (↑)	GAUC (↑)	Logloss (↓)	AUC (↑)	GAUC (↑)	Logloss (↓)	AUC (↑)	GAUC (↑)	Logloss (↓)
Wide&Deep	0.7237±0.0008	0.6922±0.0009	0.6072±0.0020	0.8242±0.0009	0.8247±0.0008	0.5132±0.0033	0.8202±0.0023	0.7761±0.0006	0.4922±0.0025
DeepFM	0.7215±0.0015	0.6910±0.0011	0.6080±0.0026	0.8064±0.0028	0.8066±0.0028	0.5352±0.0081	0.8207±0.0014	0.7753±0.0007	0.4922±0.0023
AFM	0.7199±0.0008	0.6884±0.0007	0.6091±0.0013	0.7995±0.0008	0.7999±0.0009	0.5330±0.0008	0.8184±0.0034	0.7731±0.0049	0.4969±0.0041
NFM	0.7156±0.0039	0.6850±0.0042	0.6171±0.0078	0.8044±0.0009	0.8049±0.0009	0.5372±0.0033	0.8186±0.0045	0.7717±0.0022	0.4951±0.0040
DIN	0.7248±0.0010	0.6974±0.0005	0.6143±0.0043	0.8295±0.0026	0.8307±0.0030	0.5186±0.0028	0.8208±0.0019	0.7792±0.0005	0.4978±0.0031
DIEN	0.7262±0.0010	0.6958±0.0009	0.6112±0.0020	0.8313±0.0031	0.8323±0.0027	0.5167±0.0056	0.8273±0.0016	0.7783±0.0009	0.4943±0.0054
UBR4CTR	0.7245±0.0002	0.6943±0.0010	0.6233±0.0076	0.8300±0.0005	0.8299±0.0006	0.5056±0.0007	0.8266±0.0005	0.7799±0.0006	0.4907±0.0020
SIM	0.7255±0.0014	0.6950±0.0012	0.6254±0.0094	0.8296±0.0033	0.8305±0.0031	0.5186±0.0062	0.8273±0.0005	0.7800±0.0005	0.4906±0.0021
PinSage	0.7298±0.0017	0.7069±0.0017	0.6121±0.0039	0.8136±0.0027	0.8133±0.0027	0.5269±0.0078	0.8163±0.0019	0.7810±0.0006	0.5037±0.0041
LightGCN	0.7305±0.0009	0.7077±0.0012	0.6122±0.0061	0.8329±0.0011	0.8333±0.0010	0.5101±0.0049	0.8139±0.0019	0.7803±0.0014	0.5068±0.0041
GLSM	0.7320±0.0003	0.7096±0.0007	0.6088±0.0035	0.8318±0.0026	0.8324±0.0026	0.5112±0.0066	0.8170±0.0012	0.7811±0.0004	0.5031±0.0059
GMT	0.7353±0.0014	0.7097±0.0010	0.6003±0.0023	0.8313±0.0020	0.8322±0.0024	0.5110±0.0083	0.8215±0.0018	0.7803±0.0017	0.4981±0.0020
MacGNN	**0.7458**±0.0006	**0.7198**±0.0007	**0.5886**±0.0027	**0.8444**±0.0009	**0.8458**±0.0008	**0.4892**±0.0040	**0.8306**±0.0013	**0.7813**±0.0010	**0.4872**±0.0026

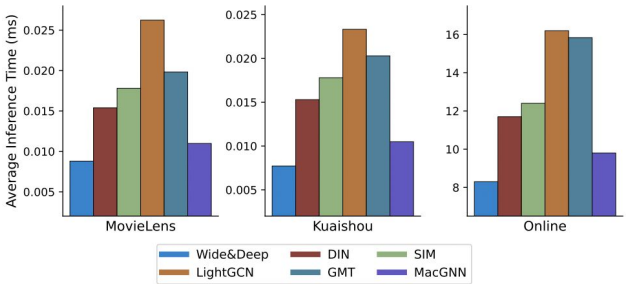


Figure 5: Efficiency study of the model inference time.

Graph4Rec: A Universal Toolkit with Graph Neural Networks for Recommender Systems - 2021:

framework for training boosting

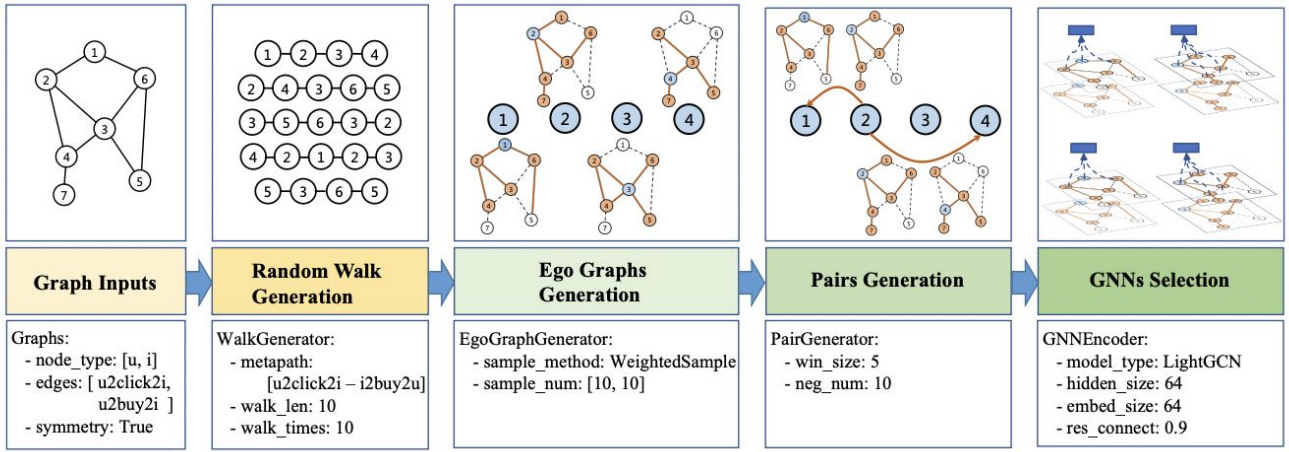


Figure 1: Graph4Rec unifies the paradigm to train GNN models into the following parts: graphs input, random walk generation, ego graphs generation, pairs generation and GNNs selection.

Distributed Graph Engine Meanwhile, Graph4Rec designs a distributed graph engine to deal with large-scale heterogeneous graph data. Nodes are partitioned uniformly into several machines. And the adjacency list of each node is stored in the corresponding server. To lower the communication cost between processes and machines, we optimize the data generation pipeline which will be discussed later.

3.4 Pairs Generation

Pairs inside a random path are usually used to define proximity for contrastive learning or self-supervised learning on graphs (Qiu et al. 2020; Wu et al. 2021). Besides, items or users in the same path like "u-i-u-i" can be a potential recommendation result. Because the path implies that the users have the same behaviors, and the items have the same user groups. In this component, as shown in 1, win_size is used for the user to control the definition of proximity in a meta path. Then we generate ego graph pairs as positive training samples for the next procedure.

3.2 Random Walk Generation

Defining proximity using nodes within the same random path is the most essential method for graph representation learning (Perozzi, Al-Rfou, and Skiena 2014; Qiu et al. 2020). As for heterogeneous graph mining, meta path random walk (Dong, Chawla, and Swami 2017) is adopted as a basic operation in Graph4Rec with its completeness satisfying most situations. Metapath can be simply defined as shown in Figure 1 that a sequence of edge types assembled head-to-tail with a hyphen. Inspired by the metapath2vec model, we develop a multi-metapaths random walk strategy that can sample multiple meta-paths from the heterogeneous graph. For example, given a heterogeneous graph described in Section 3.1, we can specify the two metapaths, "u2click2i - i2click2u" and "u2buy2i - i2buy2u", to generate different types of paths to learn more structure information from the graph. As for the homogeneous graph, we can set the metapath to "u2u - u2u", which is equal to a random walk.

3.3 Ego Graphs Generation

For every node in the training samples, neighborhood sampling is required to reduce the computation cost for the later multi-hop neighbor aggregation in GNNs. In this work, we use an ego graph to represent a training sample of a central node. The definition of ego graph is that its node is composed of a central node and its neighbors. For a node v, its neighbors of type r are defined as $S_{v,r} = \{u : d(u, r, v) \leq K\}$, where $d(u, r, v)$ is the shortest path distance between u and v of type r. Thus, an ego graph of node v in type r, denoted as $G_{v,r}$, is the subgraph induced by $S_{v,r}$.

Since there are multiple edge types, we then develop a relation-wise neighbor sampling method to allow relation-wise aggregation as described in Section 3.5. Formally, a relation-wise ego graph is denoted as G_v, where $G_v = \{G_{v,r} : r \in \mathcal{R}\}$. Therefore, each node inside the path received from random walk generation becomes a central node. And thus nodes in the same paths batch will form their disjoint ego graphs with relation-wise neighborhood sampling, as shown in Figure 1. Besides, ego graphs generation can be skipped if one only wants to train a walk-based model.

Graph4Rec: A Universal Toolkit with Graph Neural Networks for Recommender Systems - 2021:

framework for training boosting

In-batch Negative Sampling. The main idea of our framework is to learn the representation of each node, which can pull similar points (positive pairs) together while pushing away dissimilar points (negative pairs). Here, the positive pairs can be the observed interactions while the negative pairs are randomly selected from $\mathcal{V}$. However, random selection of negative samples is time consuming, especially in distributed training mode where nodes and their side information are saved in different machines. Therefore, we implement an improved version that uses in-batch negative sampling. We maximize the scores for linked nodes while minimizing the scores of other nodes in a batch. This method can reduce additional data input, thereby increasing the training speed.

Pre-training and Parameters Warm Start. The traditional walk-based models are fast and effective. We first pre-train the sparse embeddings from walk-based models. Then we train a GNN-based model and inherit the parameters for fast convergence and performance improvement.

Datasets	Negatives	Speed (Sec.)	ICF	UCF	U2I
Rec15	random	5788	0.4477	0.4647	0.4406
	in-batch	1516	0.4443	0.4680	0.4415
Tmall	random	10779	0.1708	0.1222	0.1434
	in-batch	2675	0.1696	0.1255	0.1448

Table 6: The training speed and performance of metapath2vec model between random and in-batch negative sampling. We recall the most similar top-100 items for Rec15 and top-1k items for Tmall.

Sample Generation Order	Speed (Sec.)	ICF	UCF	U2I
Walk, Pair, Ego	10025	0.1392	0.1179	0.1068
Walk, Ego, Pair	6195	0.1341	0.1175	0.1059

Table 7: The training speed and performance of LightGCN model. We recall the most similar top-1k items on Tmall dataset.

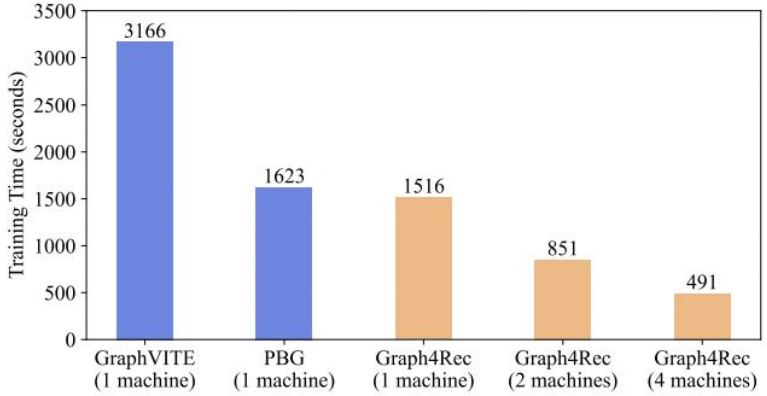


Figure 2: The runtime on Rec15 with 2 billion pairs.

Graph4Rec: A Universal Toolkit with Graph Neural Networks for Recommender Systems - 2021: **Dataset & Results**

Datasets: RetailRocket, Rec15, Tmall, UB;

	RetailRocket			Rec15			Tmall			UB		
	ICF@100	UCF@100	U2I@100	ICF@100	UCF@100	U2I@100	ICF@1k	UCF@1k	U2I@1k	ICF@1k	UCF@1k	U2I@1k
DistMult (PBG)†	0.0987	0.1106	0.0673	0.3830	0.4549	0.3258	0.1480	0.0829	0.1270	0.1122	0.0415	0.0441
DeepWalk (GraphVite)†	0.1343	0.1551	0.1034	**0.4809**	0.4448	0.3504	0.1765	0.1351	0.1281	**0.1304**	**0.0769**	0.0756
DeepWalk (ours)	0.1314	0.1639	0.1316	0.4527	0.4768	0.4348	0.1643	0.1162	0.1398	0.1028	0.0638	0.0519
metapath2vec (ours)	0.1392	0.1650	0.1354	0.4563	**0.4828**	0.4362	0.1654	0.1165	0.1407	0.1021	0.0638	0.0523
LightGCN (ours)	**0.1625**	**0.1684**	**0.1602**	0.4517	0.4730	**0.4473**	**0.1870**	**0.1361**	**0.1752**	0.1252	0.0751	**0.0774**

† Results obtained running code provided by the original authors.

Table 3: The performance of different models implemented with our Graph4Rec and other existing systems on four datasets.

	RetailRocket			Rec15			Tmall			UB		
	ICF@100	UCF@100	U2I@100	ICF@100	UCF@100	U2I@100	ICF@1k	UCF@1k	U2I@1k	ICF@1k	UCF@1k	U2I@1k
DeepWalk	0.1314	0.1639	0.1316	0.4527	0.4768	0.4348	0.1643	0.1162	0.1398	0.1028	0.0638	0.0519
metapath2vec	0.1392	**0.1650**	0.1354	0.4563	0.4828	0.4362	0.1654	0.1165	0.1407	0.1021	0.0638	0.0523
GraphSAGE (mean)	0.1369	0.1598	0.1337	0.4575	0.4774	0.4448	0.1708	0.1242	0.1420	0.1050	0.0659	0.0490
GraphSAGE (sum)	0.1349	0.1561	0.1299	0.4547	0.4705	0.4474	0.1677	0.1238	0.1496	0.1002	0.0652	0.0528
LightGCN	**0.1543**	0.1597	**0.1451**	**0.4666**	**0.4834**	0.4439	**0.1848**	**0.1404**	0.1705	**0.1232**	**0.0770**	**0.0825**
GAT	0.1347	0.1589	0.1273	0.4617	0.4743	**0.4562**	0.1680	0.1230	0.1410	0.1071	0.0672	0.0597
GIN	0.1529	0.1587	0.1210	0.4546	0.4568	0.4259	0.1829	0.1371	0.1593	0.1205	0.0710	0.0721
NGCF	0.1239	0.1520	0.1063	0.4503	0.4765	0.4464	0.1620	0.1154	0.1331	0.0983	0.0621	0.0463
GATNE	0.1387	0.1609	0.1342	0.4601	0.4817	0.4321	0.1826	0.1359	**0.1708**	0.1142	0.0713	0.0678

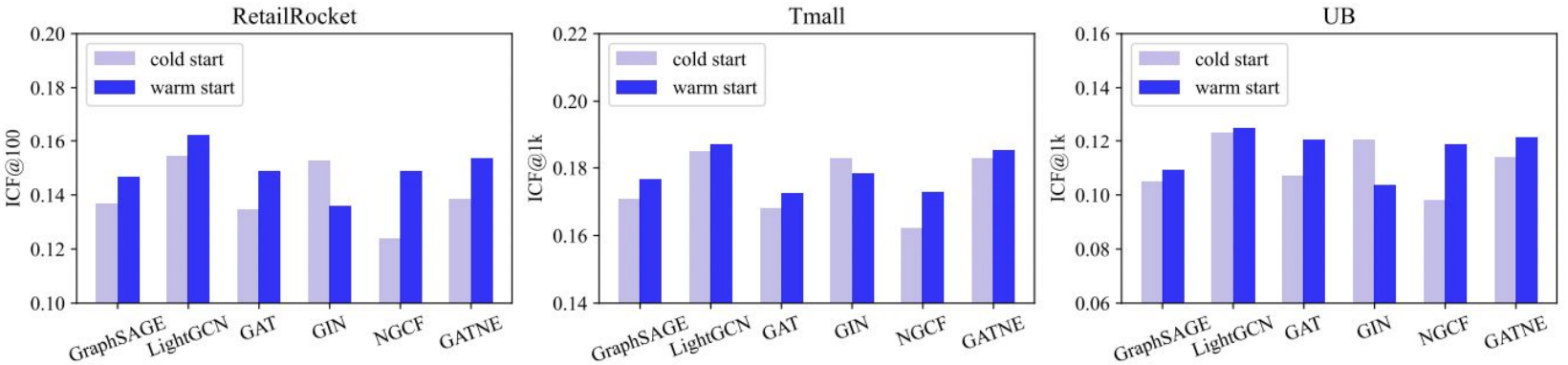


Figure 3: The influence of pre-training metapath2vec and warm start for GNN models.

SUM

What can be useful in the future (GNN on GPU optimization): link
A link with a study of the pros and cons of datasets for choosing a domain: link

Selected datasets					
Domain	Dataset	Domain	Dataset	Domain	Dataset
Music	Zvuk	e-Commerce	TaoBao	Video	KuaiRand
	Last.fm		Amazon (Electronics)		VKclip
	Piki		Megamarket		

SUM

Basic descriptive statistics: music

Dataset	#Users	#Items	#Interactions	#User features	#Item features	Feedback	Timestamp
Zvuk	382,790	1,506,950	244,673,551	3	4	1	+
Piki	8896	246,450	1,762,501	1	2	3	+
Last.fm	1,892	17,632	92,834,734	2	2	1	+

SUM

Basic descriptive statistics: e-commerce

Dataset	#Users	#Items	#Interactions	#User features	#Item features	Feedback	Timestamp
Amazon-Eletronics	4,201,696	476,002	**7,718,000,000**	4	2	1	+
TaoBao	987,994	4,162,024	100,150,807	1	1	4	+
Megamarket	**2,730,776**	**3,562,321**	**196,644,020**	1	2	4	+

SUM

Basic descriptive statistics: short-video

Dataset	Collection Policy	#Users	#Items	#Interactions	#User features	#Item features	Feedback	Timestamp
KuaiRand-27K	15 policies	27,285	32,038,725	322,278,385	30	62	12 signals	+
	Random policy	27,285	7,583	1,186,059	30	62	12 signals	+
VKclips	-	183,404	337,727	337,7270	2	3	3 types	+

SUM

Best evaluation in other research (Piki) только 1 исследование - гибрид SASRec and BERT4Rec

Model/ best evaluation (without n.f./ with n.f.)	**HR@10 (without/ with)**	**Precision@10 (without/ with)**	**MRR@10 (without/ with)**
SASRec	0.452-0.477	0.188-0.219	0.969-0.911
BERT4Rec	0.292-0.320	0.098-0.110	0.953-0.950
GRU4Rec	0.273-0.291	0.078-0.085	

SUM

Best evaluation in other research (Last.fm (2B))

Model/ best evaluation (without n.f./ with n.f.)	HR@1 (without/ with)	HR@5 (without/ with)	HR@10 (without/ with)	HR@20 (without/ with)	Precision@10 (without/ with)	MRR@10 (without/ with)
SASRec	0.190-0.221	0.371-0.400	0.452-0.477	0.532-0.553	0.188-0.219	0.969-0.911
BERT4Rec	0.101-0.117	0.227-0.248	0.292-0.320	0.366-0.394	0.098-0.110	0.953-0.950
GRU4Rec	0.096-0.102	0.203-0.221	0.273-0.291	0.311-0.342	0.078-0.085	
CASER	0.102-0.112	0.197-0.208	0.281-0.302	0.326-0.354	0.081-0.088	
WRMF(non seq)	0.097-0.102	0.138-0.147	0.269-0.279	0.305-0.316	0.062-0.067	
BPR	0.062-0.067	0.143-0.169	0.276-0.294	0.314-0.348	0.066-0.078	

SUM

Best evaluation in other research (Amazon(electronics_full 2018)

Model/ best evaluation (without n.f.)	Recall@20 (without)	NDCG@20 (without)	NDCG@40 (without)
DeepWalk	0.0725	0.0579	0.0654
Node2Vec	0.0799	0.0534	0.0724
NGCF	0.1286	0.0710	0.0830
LightGCN	0.1259	0.0715	0.0855
DGCF	0.1298	0.0714	0.0853
UltraGCN	0.1291	0.0791	0.0894
SVD–GCN	0.1318	0.0809	0.0913
UltraGCN	0.1362	0.0824	0.0935
AGL-SC(pytorch)	0.1413	0.0850	0.0967
AGL-SC(MindSpore)	0.1139	0.0459	0.0671

SUM

Best evaluation in other research (Amazon(electronics_full 2018)

Model/ best evaluation (without n.f./ with n.f.)	Recall@20 (without/ with)	NDCG@20 (without/ with)	Precision@20(with)	Hit@20(with)
LightGCN	0.0416-0.0447	0.0182-0.0203	0.0023-0.0025	0.0465-0.0499
NGCF	0.0331-0.0329	0.0148-0.0137	0.0026-0.0019	0.0507-0.0367
MF	0.0317-0.0354	0.0147-0.0155	0.0018-0.0020	0.0358-0.0391

SUM

Best evaluation in other research (TaoBao_2018) (1)

best evaluation (without n.f.) /Model	NGCF	LightGCN	NGCF_all	LightGCN_all	SGL	SimGCL	SGL_all	SimGCL_all
Recall@10 (without)	0.0135	0.0163	0.0144	0.0405	0.0431	0.0395	0.0395	0.0520
Recall@50 (without)	0.0330	0.0477	0.0391	0.0991	0.0775	0.0772	0.0772	0.1242
NDCG@10 (without)	0.0073	0.0085	0.0075	0.0223	0.0257	0.0226	0.0226	0.0274
NDCG@50 (without)	0.0115	0.0152	0.0127	0.0350	0.0332	0.0308	0.0308	0.0433

SUM

Best evaluation in other research (TaoBao _2018) (2)

Model/ best evaluation (without n.f.)	MATN	MBGMN	CML	MATN	EHCF	S-MBRec	GHCF
Recall@10 (without)	0.0520	0.0259	0.0485	0.0299	0.0672	0.0814	0.0807
Recall@50 (without)	0.1242	0.0760	0.1314	0.0914	0.1618	0.1878	0.1892
NDCG@10 (without)	0.0274	0.0132	0.0250	0.0150	0.0403	0.0446	0.0442
NDCG@50 (without)	0.0433	0.0239	0.0428	0.0282	0.0594	0.0677	0.0678

SUM

Best evaluation in other research (KuaiRand)

Model/ best evaluation (without n.f.)	Recall @20	NDCG@ 20	Model/ best evaluation (without n.f.)	Recall@20	NDCG@20	Model/ best evaluation (without n.f.)	Recall @20	NDCG@ 20
LightGCN	0.1197	0.0588	SiReN	0.1167	0.0571	SGFormer	0.1082	0.0520
LightGCL	0.1291	0.0628	SiGRec	0.1266	0.0699	SignGT	0.0883	0.0423
XSimGCL	0.1293	0.0641	PANE-GNN	0.1066	0.0522	SIGformer	0.0927	0.0439
GFormer	0.1083	0.0532	SBGNN	0.0750	0.0361			

SUM

Best evaluation in other research (Zvuk) (1)

Model/ best evaluation (without n.f.)	
NDCG@10 (without)	0.868

SUM

Best evaluation in other research (TaoBao _2018) (1)

Model/ best evaluation (without n.f.)	MATN	MBGMN	CML	MATN	EHCF	S-MBRec	GHCF
Recall@10 (without)	0.0520	0.0259	0.0485	0.0299	0.0672	0.0814	0.0807
Recall@50 (without)	0.1242	0.0760	0.1314	0.0914	0.1618	0.1878	0.1892
NDCG@10 (without)	0.0274	0.0132	0.0250	0.0150	0.0403	0.0446	0.0442
NDCG@50 (without)	0.0433	0.0239	0.0428	0.0282	0.0594	0.0677	0.0678

SUM

Briefly about the identified common disadvantages

High Computational Complexity

- Scalability Challenges for Real-World Applications
  a. Models such as **PANE-GNN** and **NFARec** use computationally intensive pairwise ranking and graph convolution mechanisms that struggle to scale efficiently for millions of users and items in industrial settings.
  b. Techniques like dual-frequency modeling in **DFGNN** further exacerbate memory consumption, limiting their deployment in low-resource environments or on-the-fly recommendation systems.
- Trade-Offs in Resource Allocation
  a. Approaches like **SIGformer** heavily rely on dense embeddings and complex attention mechanisms, which may achieve marginal gains in accuracy at the expense of prohibitive hardware requirements.
- Impracticality of Online Learning
  a. Models like **KGUF** and **RevGNN**, which use graph-based contrastive learning and negative sampling, are inherently batch-oriented and computationally expensive. This makes them unsuitable for real-time or incremental training, which is crucial for dynamic applications like e-commerce.

Prediction Smoothing

- GNN models, particularly deeper ones, are more likely to "smooth out" predictions where the node representations become indistinguishable. **(NFARec, DFGNN)**

Accounting for All Negative Interactions

- Imbalance-Induced Overfitting
  a. Models like **PANE-GNN** and **NFARec** consider all negative interactions, which creates a skewed loss landscape. This leads to the model disproportionately focusing on predicting negatives rather than learning nuanced patterns for positive feedback.
- Noise in Negative Interactions
- Lack of Hierarchical Handling of Negatives
  a. Few approaches implement strategies to categorize negative interactions based on severity or context, leading to simplistic modeling of negative signals. For instance, **SIGformer** treat all negative edges uniformly without contextual granularity.

Static Graph Assumption

- Inability to Handle Temporal Dynamics
  a. Most models assume a static graph representation. This ignores dynamic user preferences and item popularity trends, making recommendations less accurate over time.
- Over-Reliance on Offline Training
  a. While approaches like **KGUF** and **NFARec** excel in modeling relationships with rich offline data, they lack mechanisms to adapt in near-real-time scenarios, leading to stale and less relevant recommendations.

SUM - Making hypotheses