SelfGNN with Hard Negative Sampling

Project by Faith Cramer, Jeffrey Yuan, Zihan Zhao, Sharon Lin

Table of contents

01

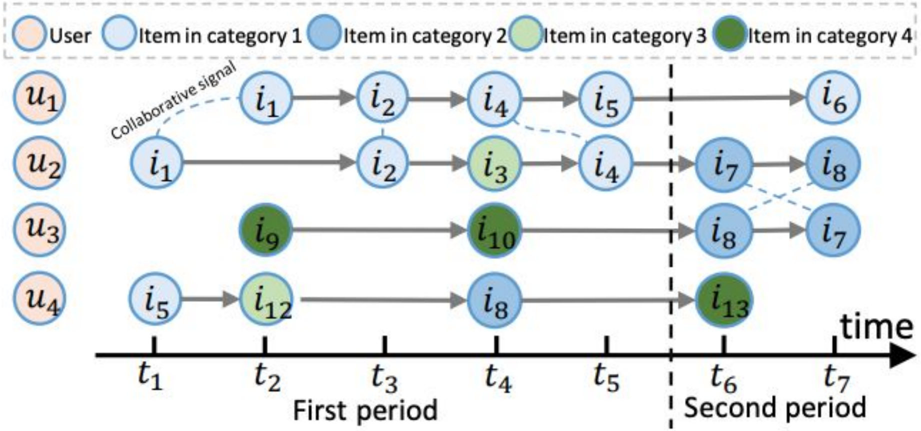
Introduction

Motivation

- **Sequential recommendation** aims to predict users' future behavior based on their historical interaction patterns.
- Traditional sequential recommenders focus on **single user sequences**, overlooking **collaborative patterns** between users.
- **Conventional** GNNs operate on **static data**, and are incapable of distinguishing between short-term and long-term user behaviors
- Existing self-supervised learning (SSL) methods **depends heavily on high-quality data**, while real-world data is often noisy
- The need to integrate **temporal dynamics**, **collaborative signals**, and **self-supervised learning** under noisy conditions.

What is SelfGNN?

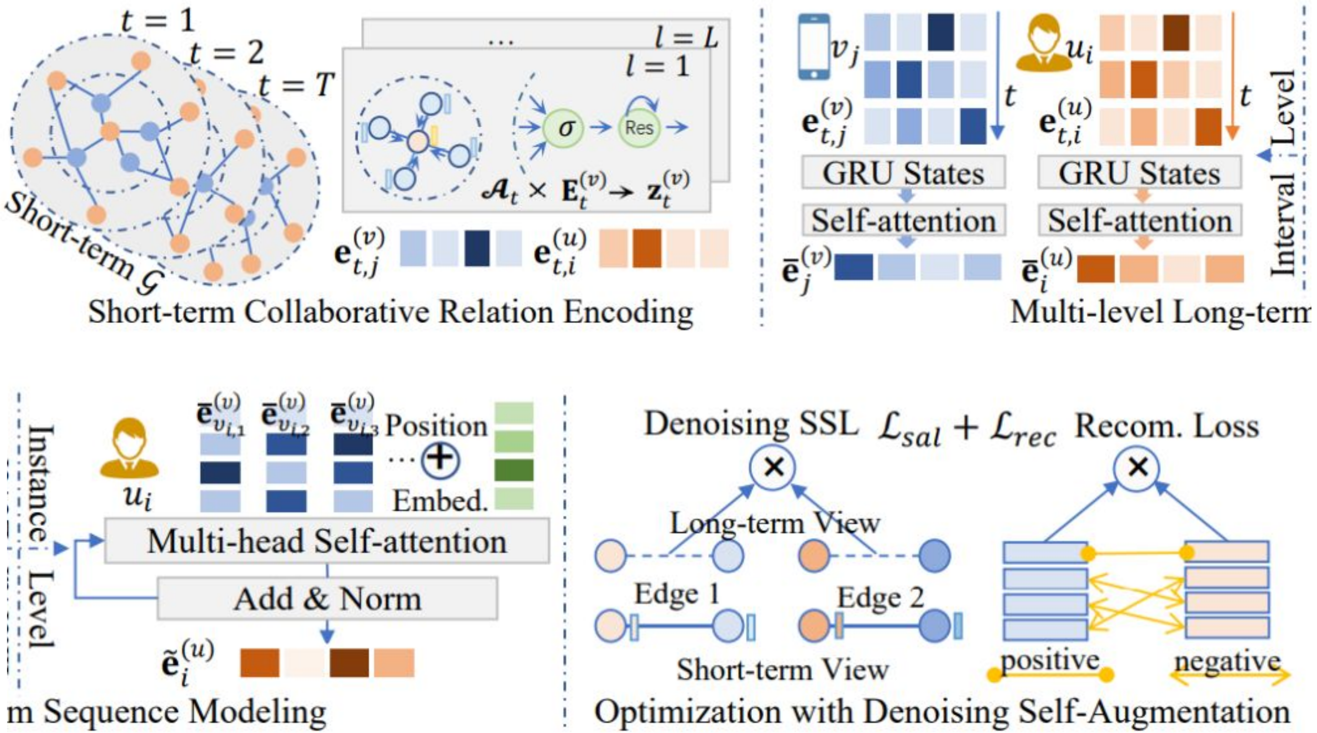
- Novel framework* that captures dynamic user interests
- Combine instance-level **collaborative graph learning** with interval-level **sequential attention modeling**
- Demonstrate consistent and robust **performance gains** over SOTA baselines across multiple datasets



*Liu, Y., Xia, L. and Huang, C., 2024, July. Selfgnn: Self-supervised graph neural networks for sequential recommendation.

Key Components of SelfGNN

- **Short-term Collaborative Graphs Encoding**: constructs time-based user-item graphs and uses GNNs (LightGCN) to capture high-order collaborative patterns in recent user behavior
- **Multi-level Long-term Sequential Learning**: models long-term user preferences at interval and instance levels using GRU and self-attention, enabling temporally-aware user representations
- **Personalized Self-Augmented Learning for Denoising**: Aligns short- and long-term predictions using a user-specific denoising to suppress noisy short-term behaviors and enhance robustness



t = 1
t = 2
t = T
Short-term G
l = L
l = 1
σ
Res
A_t × E_t^(v) → z_t^(v)
e_t,j^(v)
e_t,i^(u)
Short-term Collaborative Relation Encoding
v_j
u_i
t
GRU States
Self-attention
ē_j^(v)
ē_i^(u)
Interval Level
Multi-level Long-term
Instance Level
ē_v_i,1^(v)
ē_v_i,2^(v)
ē_v_i,3^(v)
Position
Embed.
Multi-head Self-attention
Add & Norm
ẽ_i^(u)
m Sequence Modeling
Denoising SSL
L_sal + L_rec
Recom. Loss
Long-term View
Edge 1
Edge 2
Short-term View
positive
negative
Optimization with Denoising Self-Augmentation

02

objectives

Improvements to SelfGNN

- Liu et al. highlights that future extensions of SelfGNN should aim to more accurately capture **short-term behavioral characteristics** to improve recommendation performance
- Current framework reduces user-specific noise but does not explicitly distinguish **similar but non-interacted items** from items that users interact with
- Use **contrastive learning**, specifically **hard negative sampling**, to train model to distinguish between relevant / similar but irrelevant items, enhancing precision of short-term user representations

Hard Negative Sampling

- Selects top-K non-interacted items most similar to a user's short-term embedding, using **cosine similarity** to prioritize **informative negatives** over random ones
- Incorporate **InfoNCE contrastive loss** into model's loss function to train the model to pull the user's short-term embedding closer to the true next item (positive), while **pushing it away from highly similar but non-interacted items (hard negatives)**

Project Objectives

- Familiarize ourselves with SelfGNN codebase and **replicate results** of Liu et al. on Amazon, Gowalla, Movielens, and Yelp datasets
- Develop **HardGNN** by integrating **hard negative sampling** into the SelfGNN framework to strengthen the discriminative power of short-term user representations
- **Tune hyperparameters** related to hard negative sampling and **evaluate performance** of HardGNN on all four datasets
- Conduct **ablation studies** to analyze the impact of individual hyperparameter settings on model performance

03

Methodology

Hard Negative Sampling Strategy

- Given user ***u*** and positive items ***i*** find **negative items *j***
- Step 1: Obtain Relevant Embeddings
  - Learned preference embeddings $\boldsymbol{e}_u$ and item embeddings $\boldsymbol{e}_j$
- Step 2: Calculate Similarity
  - For user preferences, calculate similarity with negative items

$$\mathrm{sim}(\boldsymbol{e}_u, \boldsymbol{e}_j) = \frac{\boldsymbol{e}_u \cdot \boldsymbol{e}_j}{|\boldsymbol{e}_u||\boldsymbol{e}_j|}$$

- Step 3: Select Top K Hardest Negatives
  - Rank negatives by similarity to positive items
  - Select Top-K items with highest similarity score

InfoNCE Contrastive Loss

- Contrastive learning is a self-supervised learning technique that aims to pull semantically similar (positive) samples together and push dissimilar (negative) samples apart in the embedding space
- InfoNCE is a popular contrastive loss function that distinguishes positive samples from a set of negative samples

$$\mathcal{L}_{\mathrm{InfoNCE}}(\boldsymbol{q},\boldsymbol{k}^+,\{\boldsymbol{k}_i^-\}) = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{q},\boldsymbol{k}^+)/\tau)}{\exp(\mathrm{sim}(\boldsymbol{q},\boldsymbol{k}^+)/\tau)+\sum_{j=1}^{N-1}\exp(\mathrm{sim}(\boldsymbol{q},\boldsymbol{k}_j^-)/\tau)}$$

- Anchor (q): User preference embedding from their sequence (***e***$_u$)
- Positive Key (k^+): Embedding of the target positive item ($\boldsymbol{e}_i$)
- Negative Keys $\{\mathrm{k}_i^-\}$: Embeddings of negative items ($\boldsymbol{e}_j$)
- Tau (τ): Temperature Hyperparameter

Integrating HNS and InfoNCE

- The original SelfGNN loss is augmented with the InfoNCE loss

$$L_{\mathrm{Total}} = L_{\mathrm{Rec}} + \lambda \cdot L_{\mathrm{InfoNCE}}$$

- Lambda (λ): Contrastive loss weight parameter

Research Questions

- How does HardGNN perform *w.r.t* top-*k* recommendation as compared with SelfGNN?
- What are the benefits of the components proposed in HardGNN ?
- How do contrastive loss weight and number of hard negative samples impact HardGNN?

04

Experiment Settings

Datasets

- **Amazon-book**: users' ratings of Amazon books in 2014
- **Gowalla**: user geolocation check-in dataset from Gowala in 2009-10
- **Movielens**: users' ratings for movies from 2002 to 2009
- Unable to process **Yelp** dataset from source code given
- Split each dataset into test set (most recent interaction of each user), validation set (second most recent interaction), and training set (remaining interactions)

Evaluation Metrics

- **Hit Rate (HR)**: measures whether the true next item appears in the top-N predicted results
- **Normalized Discounted Cumulative Gain (NDCG)**: measures how high true item ranks in top-N predicted results, rewarding higher placements more
- Use N = 10 and 20

05

Results

Experimental Setup

- Perform **grid search** over number of hard negatives (3, 5, 7) and contrastive loss weights (0.2, 0.1, 0.01)
- Train each configuration for 25 epochs to efficiently **identify best-performing combination** for each dataset
- **Retrain the model** using selected hyperparameters for 100 epochs
- Evaluate **final model performance** on both top-10 and top-20 recommendations and compare with SelfGNN

Results

100 Training Epochs

Dataset	Amazon				Gowalla				Movielens			
Top N	Top 10		Top 20		Top 10		Top 20		Top 10		Top 20	
Metric	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
SelfGNN	**0.371**	**0.218**	**0.486**	**0.250**	0.568	0.370	0.703	0.407	**0.224**	**0.121**	**0.326**	**0.145**
HardGNN	0.223	0.120	0.401	0.203	**0.624**	**0.408**	**0.736**	**0.434**	0.206	0.111	0.308	0.137

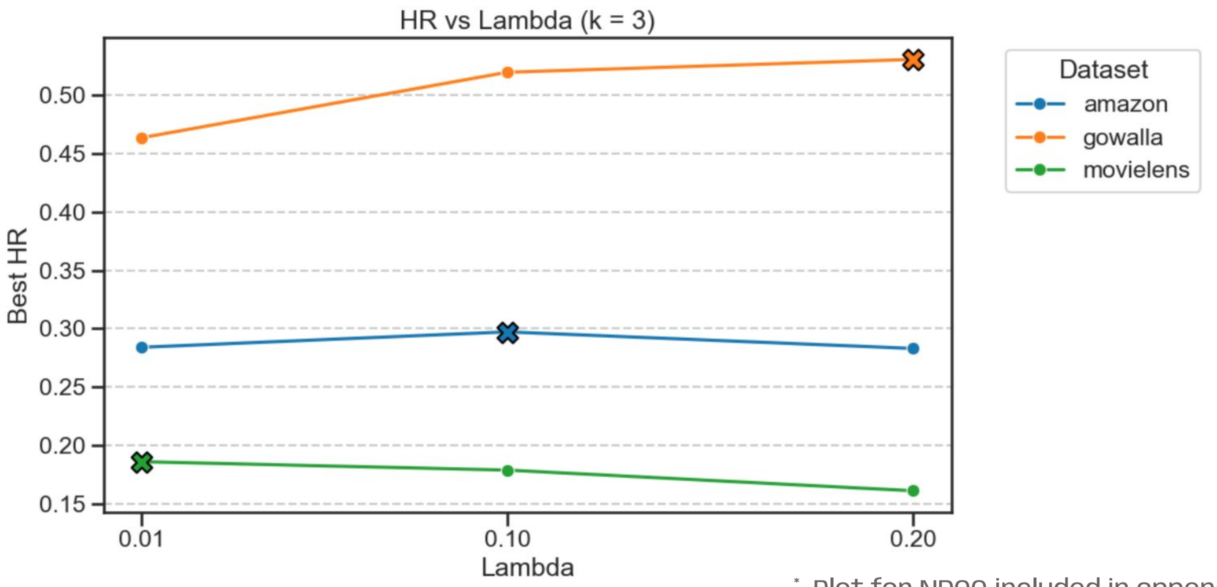
- HardGNN trained with best hyperparameters for each dataset (from gridsearch)
  - Amazon: $K = 3$, $\lambda = 0.1$
  - Gowalla: $K = 3$, $\lambda = 0.2$
  - Movielens: $K = 3$, $\lambda = 0.01$

Ablation Studies

- Use the results from the grid search to investigate model performance across **varying values of the two hard negative sampling parameters**
  - The number of hard negatives k = (3, 5, 7) and the contrastive loss weight λ = (0.01, 0.1, 0.2)
- Examine how performance changes as K varies while holding λ constant
- Examine how performance changes as λ values while holding K constant

Effect of Contrastive Loss Weight

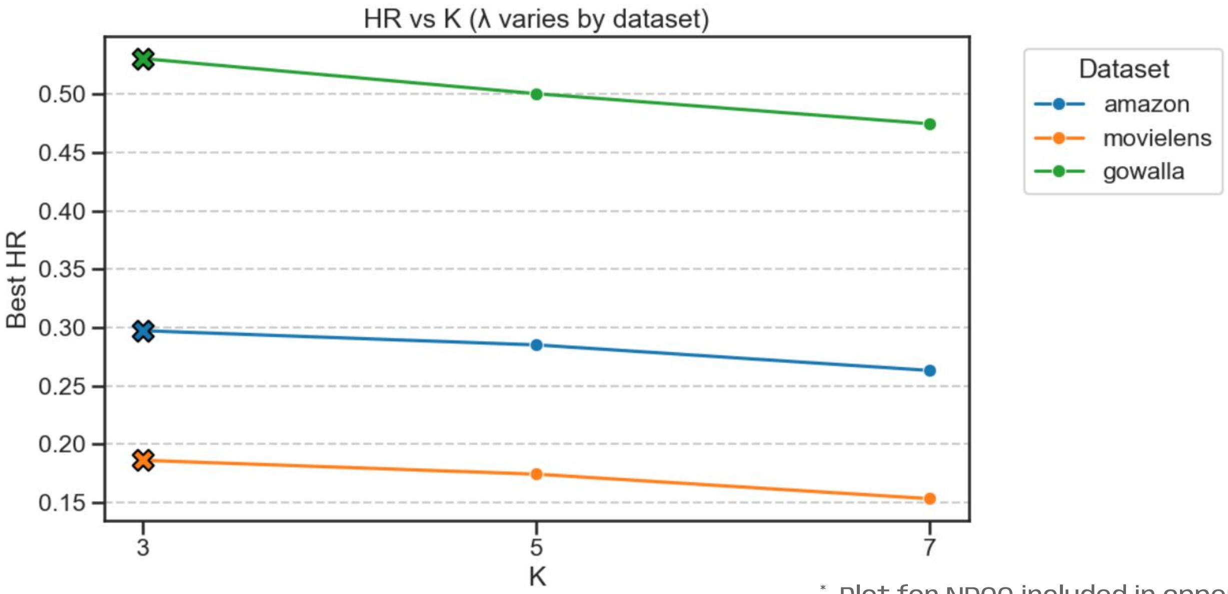
25 Training Epochs



* Plot for NDCG included in appendix

Effect of # Hard Negative Samples

25 Training Epochs



* Plot for NDCG included in appendix

06

conclusion

Summary of Findings

- HardGNN (our solution) **underperformed** SelfGNN (original work) for Amazon and Movielens datasets
  - These datasets rely more on long-term signals, so emphasizing short-term behavior via hard negatives may disrupt model balance and hurt performance
- HardGNN **outperformed** Self GNN on Gowalla dataset, likely reflecting the benefits of hard negative sampling for the specific characteristics of this dataset
  - Gowalla is the sparsest dataset, where random negatives (e.g., distant or irrelevant locations) rarely offer meaningful learning signals
  - Hard negatives, being contextually similar but behaviorally distinct, provide more informative information for training

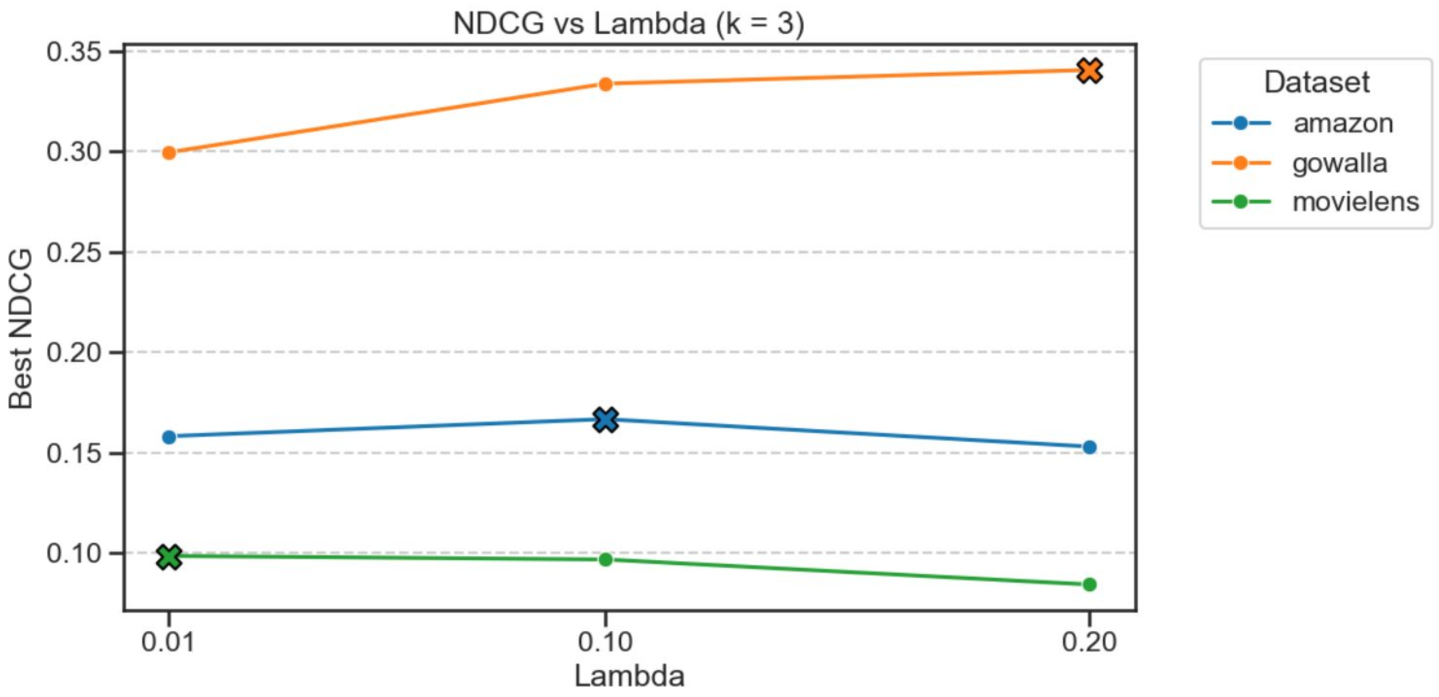
Future Work

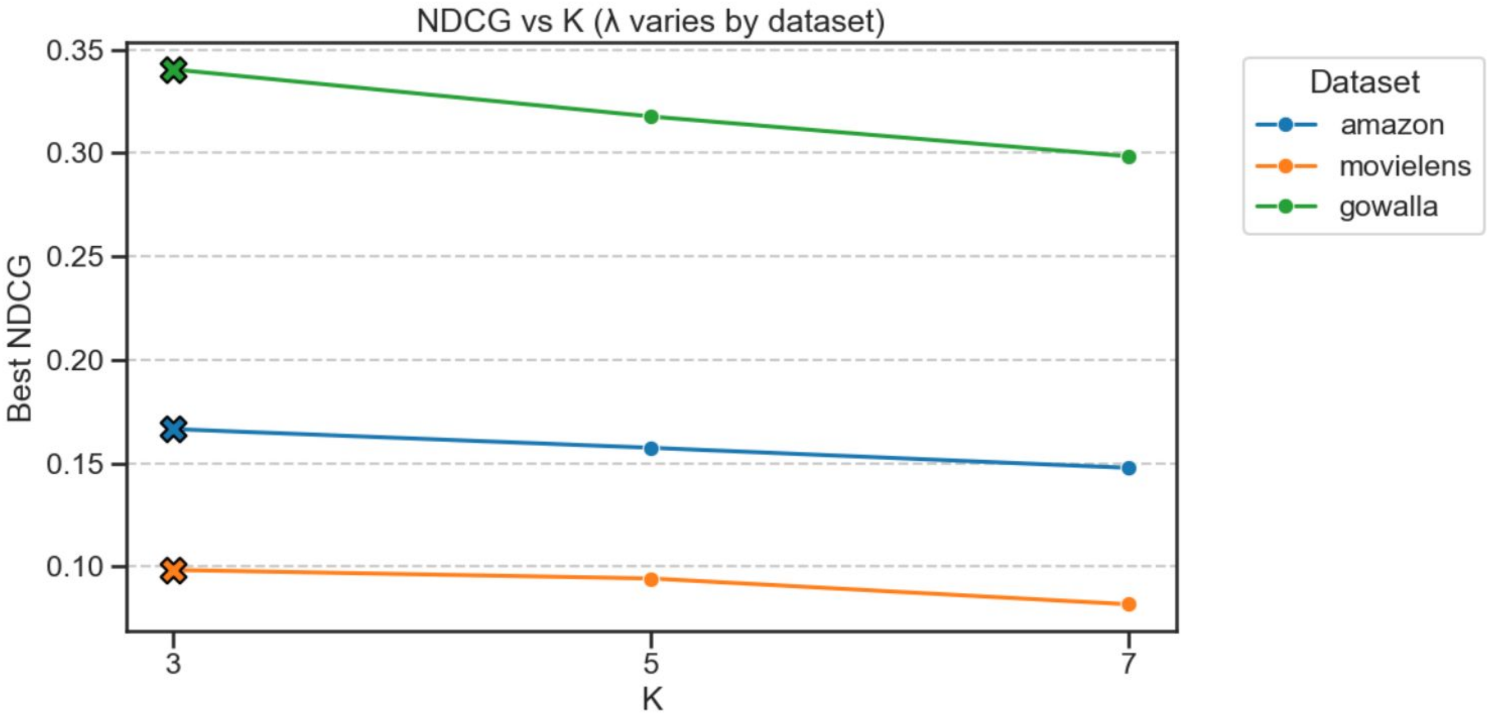
- Improve the time and space efficiency of HardGNN implementation
- Conduct more extensive hyperparameter tuning and run grid search for more training epochs
- Train HardGNN to 150 epochs (instead of 100) and retrain 5 times (instead of once) to obtain p-values to align with original SelfGNN experimental setup

Thanks!

Questions?

Appendix


NDCG vs Lambda (k = 3)
Best NDCG
0.35
0.30
0.25
0.20
0.15
0.10
0.01
0.10
0.20
Lambda
Dataset
amazon
gowalla
movielens


NDCG vs K (λ varies by dataset)
Best NDCG
0.35
0.30
0.25
0.20
0.15
0.10
3
5
7
K
Dataset
amazon
movielens
gowalla