Vietnam National University

University of Science - HCM city

Department of Information Technology

Computer Science

Course seminar report

Topic: Graph-less Collaborative Filtering

Course: Big Data Application

Students:
Nguyễn Thế Thiện (21127170)
Nguyễn Trần Trung Kiên (21127327)
Châu Tấn Kiệt (21127329)

Guidance instructors:
Dr. Nguyễn Ngọc Thảo
Dr. Bùi Duy Đăng

27th April 2025

Contents

1 Abstract

Society has existed for as long as human does, or even long before, from the need of hording around to combine strengths and divide the work for survival. Since humans do not tend to share expertise, recommendation exists in communication as a way for humans to get suggestions for making the best of their own decisions.

In modern world, automatic recommender system has seen major breakthroughs, one of which is the birth of machine learning and its application in solving the problem. The latest trends have seen graph neural networks applied into collaborative filtering approach - one of the two major approaches of recommender system. Introduced by Franco Scarselli et. al. in 2008, GNN [15] exists from the needs for analyzing data in the form of graphs that mainly represents the relationships between each pair of objects in a set of many. This type of information could be easily seen in networks of communications, data organization, information systems, linguistics, etc. Such representation of relations between objects (nodes) is how GNN could leverage the given information into solving the original problems, and potentially give better solutions.

But everything has its own pros and cons, GNN is no stranger to this universal rule. Due to how GNN works at its core: recursive propagation back and forth between neighboring nodes, it is proven to be prone to over-smoothing [24] and noise effects [5], as well as its very high computational cost, meaning poor scalability in practical applications.

We are unlikely to leave GNN out of the equation with its undoubted importance among recent research trends, but we have to deal with its limitations. The paper suggests to make a half-breed of GNN and another class of neural network that can avoid such limitations easily, in which the authors chose MLP, using a median system to transfer knowledge from GNN to MLP. Such a system is called **SimRec**.

2 Problem

In this section, we shall dissect the existing problems of GNN: what causes those problems, the symptoms and how they impact the results to the recommender problem.

Over-smoothing. It is a phenomenon where aggregating messages back and forth between nodes in a graph, which is how GNN training works at its core, which tends to make node representations more similar and cause unrecoverable accuracy loss. Many papers throughout the recent years working with GNN have shown that over-smoothing is inevitable as the number of layers is extended [14, 11]. In fact over-smoothing tends to happen as soon as the number of layers reaches $O\left(\frac{\log N}{\log \log N}\right)$ for a sufficiently dense graph with N nodes [24], which is somewhat on par with $O\left(N^2\right)$ according to the following graphs:

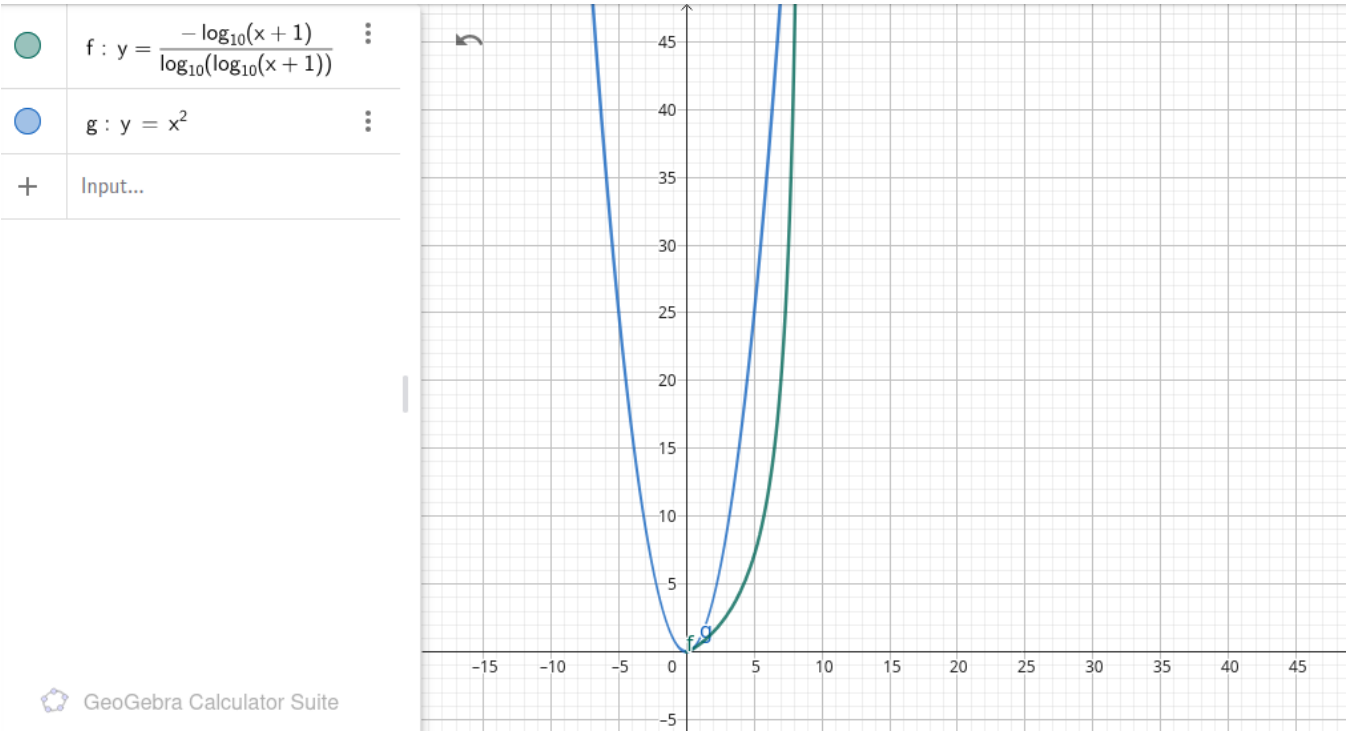

Figure 1: Graph scaling (demonstration using GeoGebra)

where $y = \frac{-\log(x+1)}{\log(\log(x+1))}$ is used to demonstrate $O\left(\frac{\log N}{\log \log N}\right)$. Compared to $O(N^2)$, it starts slower from 0, but quickly increases its gradient to infinity in a very short constant period:

$$\lim_{x \to 9^-} \frac{-\log\left(x+1\right)}{\log\left(\log\left(x+1\right)\right)} = \frac{-\log(10^-)}{\log(\log(10^-))} = \frac{-1}{\log(1^-)} = \frac{-1}{0^-} = +\infty$$

Meanwhile, some CNNs used for image-processing have far fewer layers than GNNs, e.g. Kaiming He et. al. [7] CNN provides experiment with 152 layers on ImageNet dataset, it shows why over-smoothing matters so much as a challenge in holding down potentials of GNN.

Noise issues. This shall be a common problem when working with neural networks, since noise is passed between nodes and layers. But due to how GNN aggregates messages back and forth multiple times between nodes, there is a good chance, even though seemingly not thoroughly researched by any author, could be hypothesized that GNN amplifies noisy signals even more than some other types of neural networks.

Scalability. Multiple researches on GNN yield expensive computational cost [6, 26]. For a general GNN model with a directed graph consisting of N nodes reprenting items or users, one for one each, means there are N^2 vertices, or N^2 messages sent between the nodes at max in one iteration for one layer. Even though computational cost could be linearly cut down since real-life graphs are sparse, $O(N^2)$ is still a nightmare for any program or algorithm to deal with.

3 Solution

With the problems presented by the authors on GNN, this section is the in-depth explanation on their perspective of the problems and how they address the issues.

3.1 Abstract & core ideas

In summary, the authors suggest the SimRec system with the following core ideas to alleviate these issues:

- Knowledge transfering from a complicated GNN model to a simpler MLP model, in order to reduce the computational cost.
- Contrastive regularization in knowledge transfer by weighing factor adjustments, to distance the knowledge and alleviate the over-smoothing issues.
- Additional adjustments on loss function gradients to address the noise issues based on recent works.

3.2 Component models

As the solution proposed in this paper is a process to transform a more complex GNN-based CF model into a simpler MLP-based CF model, we shall recap the basics of their architectures and how they commonly work.

CF recap. CF is a main approach in recommendation systems, basically gives results based on similar users. Suppose that we have a set of I users $U = \{u_1, u_2, ..., u_I\}$ and J items $V = \{v_1, v_2, ..., v_J\}$. A good way to represent how each user interacts with each item is by using Cartesian product and assigning each pair of the results a real number to indicate how much they interact, usually it is user's rating towards that specific item - an interaction matrix $A \in \mathbb{R}^{I \times J}$. A CF shall predict an interaction $y_{i,j}$ between an unobserved user-item pair (u_i, v_j) from the other observed interactions in A.

$$y_{i,j} = Predict(u_i, v_j, A)$$

Matrix factorization method in the RS problem is predicting an interaction by simply decomposing the interaction matrix A into a product of matrices: $A \approx H_I^\top H_J = Y$ with $H_I \in \mathbb{R}^{d \times I}, H_J \in \mathbb{R}^{d \times J}, d < min(I, J)$, where d represents the d-dimensional space for latent features, and one column, size $d(\times 1)$, is a vector representation for the corresponding user/item, or shortly: its embedding. Overall, we can look into CF approach consisting of two stages:

$$y_{i,j} = Predict(h_i, h_j); h_i = Embed1(u_i, v_j, A), h_j = Embed2(u_i, v_j, A)$$

Linear algebra does allow the CF approach in RS problem to be solved using known algorithms, such as SVD, however we come to deep learning because real life is not as such a dreamland: missing data, underlying noise, etc. The paper examines two types of NNs:

MLP-based CF. A basic ML approach to the CF problem that divides neurons into multiple fully-connected layers, simplifies implementations and computations through linear regression and activation function between each pair of layers. MLP feed-forwards the input matrix through a network where neurons are organized in layers with non-linear activation functions, then uses the feedback to update weights (and biases) through back-propagation. Here, MLP is used to re-embed the initial embeddings gained from matrix factorization.

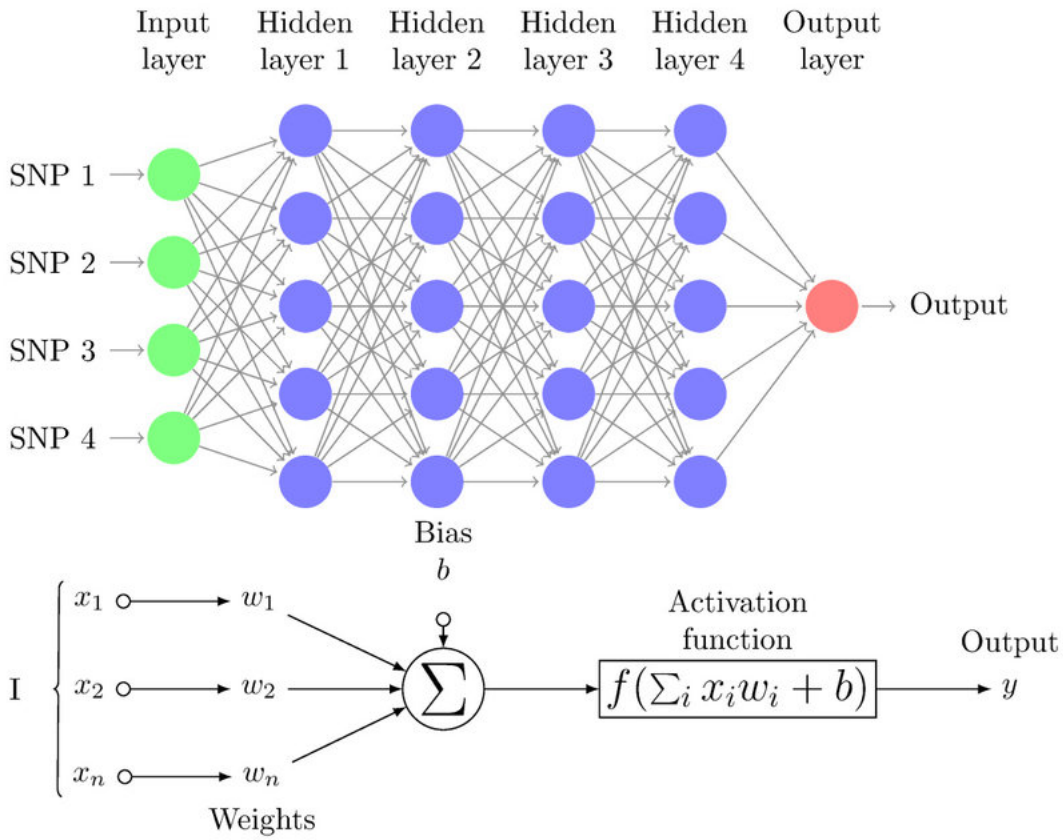

Figure 2: MLP example (by CreativeCommons)

Such simplicity allows MLP-based CF models to avoid learning over-smoothed embeddings while being efficient [3]. But in return, MLP is well-known to be prone to over-fitting when we scale it up to DNN when dealing with DL problems.

GNN-based CF. As the name suggests, it works around graphs as an input, including the interaction matrix that we usually have from observing user-item relations in the RS problem, generally by pairwise message passing: graph nodes iteratively update their representations from exchanging information with neighboring nodes[15].

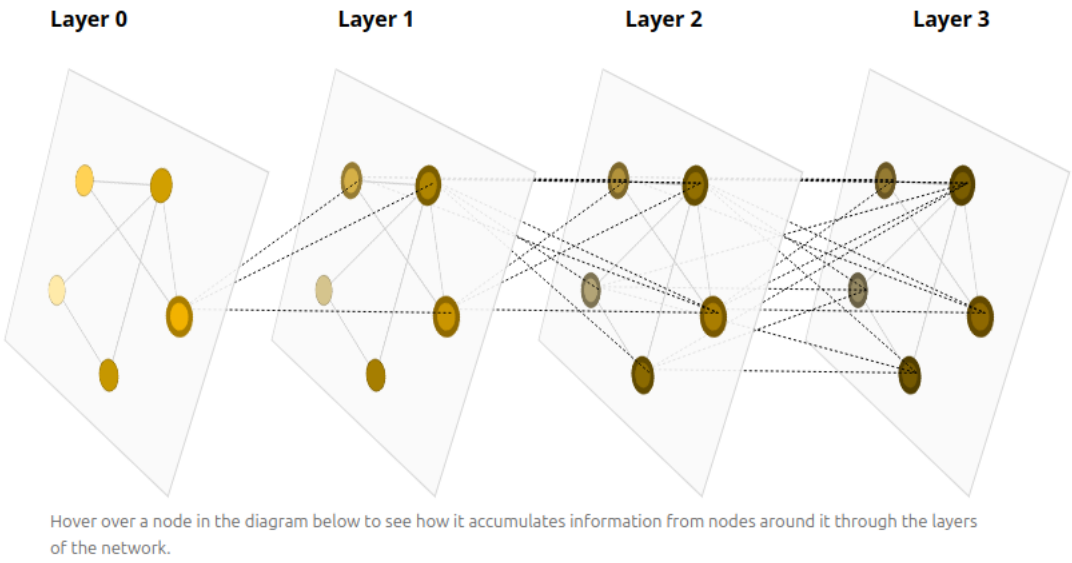

Figure 3: GNN example (by distill.pub)

Similar to MLP, here GNN is used to re-embed the initial embeddings through its core processes: (iteratively) propagating and aggregating along the input interaction graph G gained from encoding users' high-order interactive relations into embeddings. Such mechanism explains how GNN is able to reduce over-fitting, but being prone to over-smoothing issues as the network depth increases.

Hence, the authors of this paper choose to use GNN as the teacher model in order to capture the latent features but stop before it reaches the over-smoothing states, instead use proposed methods to distill and transfer such knowledge to MLP student model to boost the efficiency and effectiveness of the outcomes.

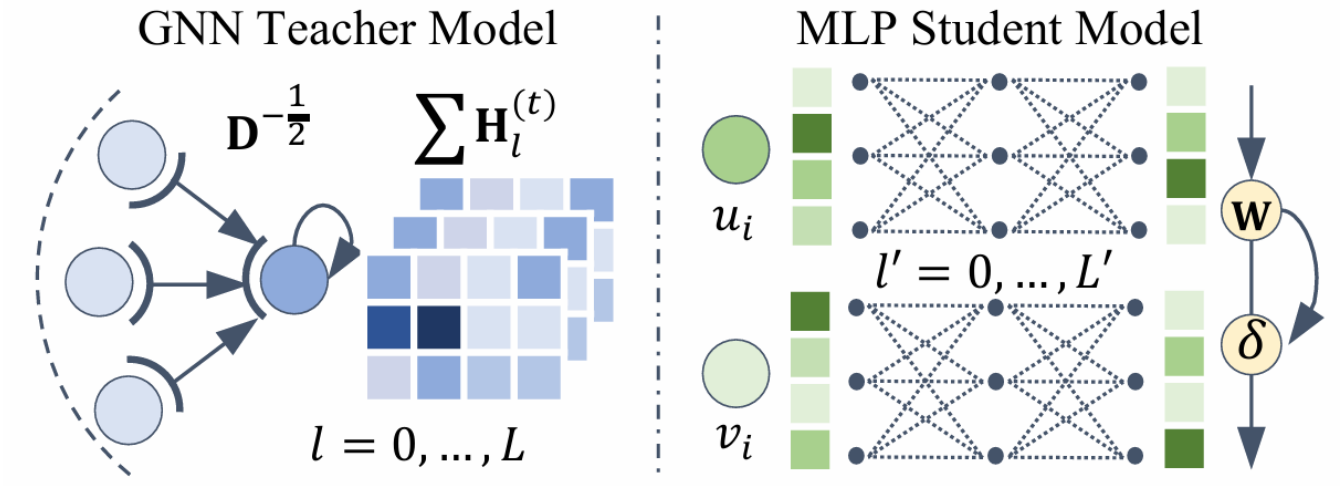

Figure 4: Teacher-Student Framework

3.3 SimRec

3.3.1 Contrastive Knowledge Distillation

SimRec is designed to overcome the limitations of traditional Graph Neural Network (GNN)-based recommender systems. It integrates knowledge distillation and contrastive learning to enhance recommendation performance while addressing issues like over-smoothing and scalability inherent in GNN-based models. It leverages a Teacher-Student Framework, which consists of:

- Teacher Model: A GNN-based model that captures intricate user-item interaction patterns.
- Student Model: A lightweight neural network (MLP) designed to learn from the teacher model's knowledge.

SimRec involves two primary components: **prediction-level distillation** and **embedding-level distillation**.
In **prediction-level distillation**, the student model is trained to replicate the output predictions of the teacher model. By aligning its predictions with those of the teacher, the student model learns to approximate the teacher's decision-making process. This alignment ensures that the student captures the high-level patterns and relationships identified by the teacher, leading to improved performance. The effectiveness of this distillation is controlled by a hyperparameter, often referred to as 'softreg', which determines the weight of the prediction-level distillation loss in the overall training objective.

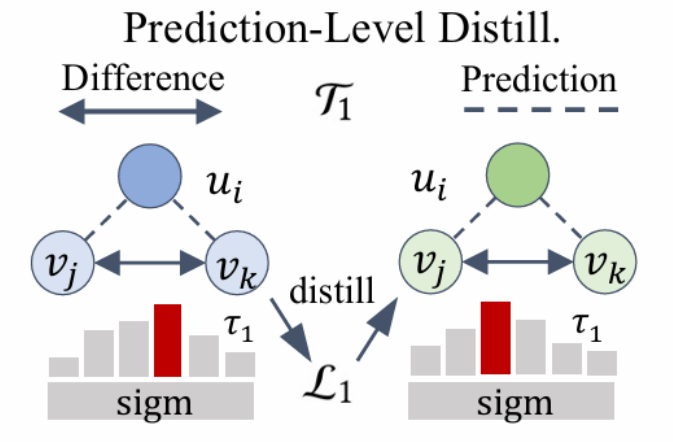

Figure 5: Prediction-Level Distillation

Beyond matching predictions, **embedding-level distillation** focuses on aligning the internal representations (embeddings) learned by the student model with those of the teacher. In doing so, the student model captures the nuanced features and intermediate representations that the teacher has learned, leading to a deeper understanding of the data. This process is regulated by a hyperparameter known as cdreg, which controls the weight of the embedding-level distillation loss during training.

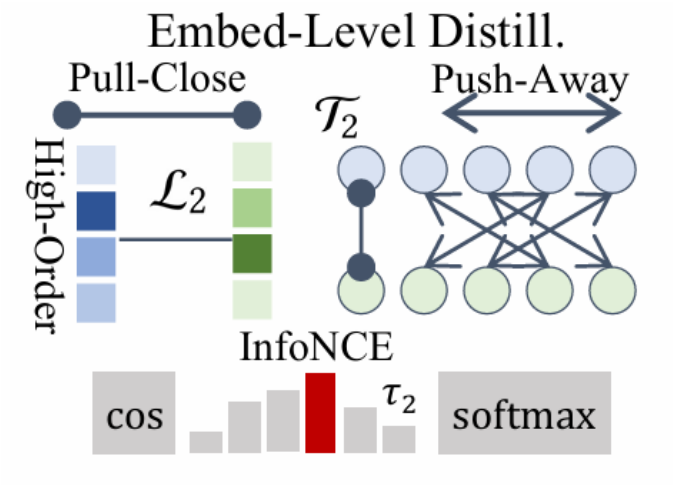

Figure 6: Embedding-Level Distillation

The combination of prediction-level and embedding-level distillation in SimRec serves to:

- Comprehensive Knowledge Transfer: By aligning both the outputs and internal representations, the student model effectively captures the holistic knowledge of the teacher model.
- Enhanced Performance: This dual distillation approach enables the student model to achieve performance levels comparable to the teacher model while maintaining a simpler and more efficient architecture.

- Mitigation of Over-Smoothing: Embedding-level distillation helps the student model retain distinct and informative representations, addressing the over-smoothing issue commonly observed in deep GNNs.

By integrating both prediction-level and embedding-level distillation, SimRec ensures that the student model inherits the strengths of the teacher model, resulting in an efficient and effective collaborative filtering system.

3.3.2 Adaptive Contrastive Regularization

The CKD process presents us how it transfers the knowledge in prediction using a simple log loss function, and distills the user-item embeddings through a more complex contrastive log loss function based on cosine differences, all between embeddings of the teacher and student models iteratively.

However, we still have yet to account for the over-smoothing problem that some signals from the GNN teacher model might have. A common approach is to sparsify properties of the nodes, and as proposed in this paper: to use another loss function that aims to minimize similarities, as well as regularization to avoid over-fitting, and finally customized weights to differentiate the impacts of (possibly) over-smoothed signals from the rest.

The authors aim to minimize similarities by pushing away the user-user, user-item, item-item distances since both were originally implemented into GNN nodes. The basic idea of the loss function for pushing away distances of each pair of user/item is regularizing dot-product similarity minimization on node embeddings of the student model, with a weighting factor ω_i:

$$\omega_i = \begin{cases} 1-\epsilon & \text{if } \left(\nabla_i^{1,2}\right)^\top \nabla_i^{\text{rec}} > \left(\nabla_i^1\right)^\top \nabla_i^2 \\ 1+\epsilon & \text{otherwise} \end{cases} \tag{1}$$

...where ω_i adjusts the weight of contrastive regularization for user u_i, and $\nabla_i = \frac{\partial L}{\partial h_i^{(s)}}$ denotes the gradient of a corresponding loss function L for the student model (such as ∇_i^1 for L_1, $\nabla_i^{1,2}$ for compound gradients of L_1, L_2). ω_i differentiates over-smoothed signals by a hyperparameter $\epsilon (0 < \epsilon < 1)$.

The recommendation task is defined as maximizing similarity between positive user-item pairs, which is achieved by minimizing the loss function:

$$L_{\text{rec}} = -\sum_{(u_i, v_j) \in T} y_{i,j}$$

where $y_{i,j}$ represents the similarity between the positive user-item pair (u_i, v_j) from observed interactions.

When such threshold occurs, we can assume that gradients of combined distillation tasks and recommendation task are close to each other, or specifically, on the same (negative) directions towards minimizing the losses for the node representing user u_i. Since the signal is close to optimal values of the problem goals, we reduce its weight w_i in order to decrease user u_i's effect on the loss function and avoid adjusting the node further when training. The same could also be applied to each node representing item v_j with weighting factor w_j.

3.3.3 Parameters learning

To summarize SimRec, the GNN-based teacher model is trained first, then later performs knowledge distillation from that model to the MLP-based student model. The teacher model is trained by iteratively feeding forward a batch of samples, where each sample has one user, two items - one negative, one positive:

$$\{(u_i, v_j, v_k)|a_{i,j} = 1, a_{i,k} = 0\}$$

to converge the Bayesian personalized ranking (BPR) loss function [17]:

$$L^{(t)} = -\sum_{(u_i,v_j,v_k)} logsigmoid\left(y_{i,j}^{(t)} - y_{i,k}^{(t)}\right) + \lambda^{(t)} \left|\left|\bar{H}^{(t)}\right|\right|^2$$

where in the last term, $\bar{H}^{(t)} \in \mathbb{R}^{(I+J)\times d}$ denotes the initial embedding vectors for users and items (mentioned in section **Component Models**).

The BPR loss function works by converting and inverting the original optimization function for personalized ranking criterion:

$$OPT = \ln p(\Theta| >_u) = \ln \frac{p(>_u |\Theta)p(\Theta)}{p(>_u)} \propto \ln p(>_u |\Theta)p(\Theta)$$

where Θ is the parameter vector of an arbitrary model class. Since $>_u \subset V^2$ is a "desired but latent" personalized total ranking between all items in V for user u, meaning $p(>_u)$ could be seen as a constant value that could only be changed if the original data (users and items) is changed.
Since each user's ranking is personalized and independent:

$$OPT = \ln\left(p(>_u |\Theta)p(\Theta)\right) = \ln\left(\prod_{u\in U} p(>_u |\Theta)\right) + \ln p(\Theta)$$

and also assuming the order in each pair of items is unrelated to the others:

$$OPT = \ln\left(\prod_{(u,i,j)\in U\times V^2} p(i >_u j|\Theta)\right) + \ln p(\Theta)$$

then rewritten into what we have:

$$OPT = \sum_{(u,i,j)\in U\times V^2} \ln p(i >_u j|\Theta) + \ln p(\Theta)$$

where $p(i >_u j|\Theta) := \sigma(y_{u,i} - y_{u,k})$ showing the relationship between the triplet, and $p(\Theta) := \lambda \left|\left|\Theta\right|\right|^2$ is weight-decay regularization just to avoid over-fitting.

From then on, SimRec conducts knowledge distillation to the MLP-based student model by combining the loss functions of the techniques mentioned before. In summary, we have L_1 and L_2 as contrastive loss functions for the distillation process, L_3 to prevent transferring over-smoothed signals from the GNN-based teacher model.

Additionally, like the BPR loss function for the teacher model, we would also have $L_4 = \left|\left|\bar{H}^{(s)}\right|\right|^2$ as regularization to avoid over-fitting, and finally an overall objective recommendation loss $L_{rec} = -\sum_{(u_i,v_j)\in T_2} y_{i,j}$ based on results.
So the overall optimization objective for the MLP-based student distilled from the teacher model is conducted as:

$$L^{(s)} = L_{rec} + \lambda_1 \times L_1 + \lambda_2 \times L_2 + \lambda_3 \times L_3 + \lambda_4 \times L_4$$

where λ_1, λ_2, λ_3, λ_4 are weights for the optimization terms.

3.3.4 Further discussions

3.3.4.1 Addressing noise effects

The authors list previous works [21, 13, 25] that address the noise and sparsity problems of collaborative filtering using contrastive learning techniques, in which we could see the similarity between the InfoNCE-based contrastive learning (CL) loss, and the authors' KD loss L_1:

$$\frac{\partial L_{CL}}{\partial h_i} = -\frac{1}{\tau} \times \frac{\partial h_i'^{\top} h_i' / \left(||h_i'||_2 \, ||h_i''||_2\right)}{\partial h_i}$$

$$\frac{\partial L_1}{\partial h_i^{(s)}} = \sum_{v_j, v_k} -w_{i,j,k} \times \frac{\partial h_i^{(s)\top} h_j^{(s)}}{\partial h_i}$$

Aside from the weighing factor, we could see KD generates pull-close optimization terms with each item couple v_j, v_k for each node u_i which h_i represents the embeddings. This shall prove that KD does at least enough to deal with the noise effects in the data.

3.3.4.2 High-order Smoothing

Furthermore, from the BPR function used to minimize the loss when training the GNN teacher model using triplets sampling (section **Parameters learning**):

$$L^{(t)} = -\sum_{(u_i,v_j,v_k)} logsigmoid\left(y_{i,j}^{(t)} - y_{i,k}^{(t)}\right) + \lambda^{(t)} \left|\left|\bar{H}^{(t)}\right|\right|^2$$

... through sophisticated derivation in respect to embedding $\overline{h}_i$ for node n_i (item or user), when using two nodes n_i, n_j are smoothed using each other by using triplets sampling with other nodes represented by v_k:

$$\frac{\partial L_{i,j}}{\partial \overline{h}_i} = \sum_{v_k} -\sigma \times \left(\sum_{P_{i,j}^{2L}} \prod_{(n_a,n_b)\in P_{i,j}^{2L}} \frac{1}{\sqrt{d_a d_b}}\right) \times \frac{\partial \overline{h}_i^{\top} \overline{h}_j}{\partial \overline{h}_i}$$

where we could see the bracketed part represents how closely nodes are connected, for node couples in $P_{i,j}^{2L}$ representing a set of nodes that are connected through a path with maximum length $2L$, which we could easily think of as nodes being 2 nodes away (a possible path with one another node in between). Such phenomenon called *high-order smoothing* that KD in SimRec is capable of, and it is a studied strength in GNN as it is known to be able to "harness polyadic relations between vertices beyond plain edges" [1].

Not only is high-order smoothing applied to teacher GNN model, but also the student MLP model with L_1 loss function, where its derivative to represent similarity between two nodes:

$$\frac{\partial L^1_{i,j}}{\partial h^{(s)}_i} = \sum_{v_k} -\frac{1}{\tau_1} \times \left(\overline{z}^{(t)}_{i,j,k} - \overline{z}^{(s)}_{i,j,k}\right) \times \frac{\partial h^{(s)\top}_i h^{(s)}_j}{\partial h^{(s)}_i}$$

and thanks to the nature of the log loss function itself in L_1 between node pairs:

$$L_1 = \sum_{(u_i,v_j,v_k)\in T_1} -\left(\overline{z}^{(t)}_{i,j,k} \cdot \log \overline{z}^{(s)}_{i,j,k} + \left(1 - \overline{z}^{(t)}_{i,j,k}\right) \cdot \log\left(1 - \overline{z}^{(s)}_{i,j,k}\right)\right)$$

we should be able to alleviate the effects of noisy signals via this technique.

3.3.4.3 Model complexity

In short, the authors claim to have SimRec comparable to existing SSL collaborative filtering methods through a series of complex computations.

But it would seem for us that this analysis should be said further into the experimentation of SimRec and baseline models on chosen datasets later.

3.4 Experiments

In this section, we shall talk about the experiments on SimRec conducted by the authors on some public datasets, and its comparisons to the baseline models consisting of traditional, popular and/or state-of-the-art methods according to the paper.

3.4.1 Experimental Settings

3.4.1.1 Dataset

Three benchmark datasets collected from real-world online services are used to evaluate the performance of **SimRec**. The data statistics are shown in the image below, and the interaction data is divided into a training set, validation set, and test set with a ratio of 70%:5%:25%.
The details of the test datasets are as follows:

- **Gowalla:** This dataset is collected from Gowalla and includes check-in records of users at geographical locations, from January to June 2010.
- **Yelp:** This dataset contains user ratings of locations, collected from the Yelp platform, covering the period from January to June 2018.
- **Amazon:** This dataset includes user rating behavior for books collected from the Amazon platform in 2013.

Table 1: Statistics of the experimental datasets.

Dataset	# Users	# Items	# Interactions	Interaction Density
Gowalla	25,557	19,747	294,983	5.85×10^{-4}
Yelp	42,712	26,822	182,357	1.59×10^{-4}
Amazon	76,469	83,761	966,680	1.51×10^{-4}

Figure 7: Table 1. Experimental datasets

3.4.1.2 Evaluation Protocols

Model results evaluation is based on the testing dataset, where for each user, positive items are ranked with all items that user does not interact.

The main comparative metrics are Recall and Normalized Discounted Cumulative Gain (NDCG) on the ranking results. Recall is a common metric used in general machine learning to get the model's capability of identifying the positive relevant samples. NDCG is a common metric used for recommendation ranking list output on the relevance and quality of the top items. For both, the higher the better.

The range for evaluation on the model output recommendation ranking list is set at the top 20 items.

3.4.1.2 Baseline Models

In this paper, the authors compare **SimRec** with the following 14 baseline models from four research streams for comprehensive validation.

Traditional Collaborative Filtering Technique:

- **BiasMF** [10]: A classic matrix factorization approach that combines user/item biases with learnable embedding vectors.

Non-GNN Neural Collaborative Filtering:

- **NCF** [9]: An early deep learning collaborative filtering model that enhances user-item interaction modeling using MLP networks.
- **AutoR** [16]: Applies a three-layer autoencoder with fully connected layers to encode user interaction vectors.

Graph Neural Architectures for Collaborative Filtering:

- **PinSage** [27]: Combines random walks with graph convolutions for web-scale recommendation graphs.
- **STGCN** [28]: Augments GCN with autoencoding sub-networks on hidden features for improved inductive inference.
- **GCMC** [18]: A representative model introducing graph convolutional operations into the matrix completion task.
- **NGCF** [19]: A GNN-based CF method that applies graph convolutions on the user-item interaction graph to learn embeddings.
- **GCCF** [2] and **LightGCN** [8]: Simplify conventional GCN structures by removing transformations and activations to improve performance.

Disentangled GNN-based Collaborative Filtering:

- **DGCF** [20]: Disentangles user-item interactions into multiple latent factors during graph message passing.

Self-Supervised Learning Approaches for Recommendation:

- **SLRec** [23]: Applies contrastive learning with feature-level data augmentations to recommendation models.
- **NCL** [12]: Enhances graph-based CF models with enriched neighbor-wise contrastive learning.
- **SGL** [22]: Uses multiple types of graph and feature augmentations for self-supervised graph contrastive learning in CF.
- **HCCF** [3]: Augments GNN-based CF with a global hypergraph GNN and performs cross-view contrastive learning.

3.4.2 Overall Performance Comparison

This section of the paper focuses on comparing the performance of SimRec with baseline models on three experimental datasets (Gowalla, Yelp, Amazon).

Table 2: Performance comparison on Gowalla, Yelp, and Amazon datasets in terms of *Recall* and *NDCG*.

Data	Metric	BiasMF	NCF	AutoR	PinSage	STGCN	GCMC	NGCF	GCCF	LightGCN	DGCF	SLRec	NCL	SGL	HCCF	*SimRec*	p-val.
Gowalla	Recall@20	0.0867	0.1019	0.1477	0.1235	0.1574	0.1863	0.1757	0.2012	0.2230	0.2055	0.2001	0.2283	0.2332	0.2293	**0.2434**	$2.1e^{-8}$
	NDCG@20	0.0579	0.0674	0.0690	0.0809	0.1042	0.1151	0.1135	0.1282	0.1433	0.1312	0.1298	0.1478	0.1509	0.1482	**0.1592**	$1.2e^{-9}$
	Recall@40	0.1269	0.1563	0.2511	0.1882	0.2318	0.2627	0.2586	0.2903	0.3181	0.2929	0.2863	0.3232	0.3251	0.3258	**0.3399**	$2.4e^{-8}$
	NDCG@40	0.0695	0.0833	0.0985	0.0994	0.1252	0.1390	0.1367	0.1532	0.1670	0.1555	0.1540	0.1745	0.1780	0.1751	**0.1865**	$1.7e^{-9}$
Yelp	Recall@20	0.0198	0.0304	0.0491	0.0510	0.0562	0.0584	0.0681	0.0742	0.0761	0.0700	0.0665	0.0806	0.0803	0.0789	**0.0823**	$3.7e^{-4}$
	NDCG@20	0.0094	0.0143	0.0222	0.0245	0.0282	0.0280	0.0336	0.0365	0.0373	0.0347	0.0327	0.0402	0.0398	0.0391	**0.0414**	$3.8e^{-5}$
	Recall@40	0.0307	0.0487	0.0692	0.0743	0.0856	0.0891	0.1019	0.1151	0.1175	0.1072	0.1032	0.1230	0.1226	0.1210	**0.1251**	$4.8e^{-5}$
	NDCG@40	0.0120	0.0187	0.0268	0.0315	0.0355	0.0360	0.0419	0.0466	0.0474	0.0437	0.0418	0.0505	0.0502	0.0492	**0.0519**	$2.4e^{-4}$
Amazon	Recall@20	0.0324	0.0367	0.0525	0.0486	0.0583	0.0837	0.0551	0.0772	0.0868	0.0617	0.0742	0.0955	0.0874	0.0885	**0.1067**	$1.1e^{-10}$
	NDCG@20	0.0211	0.0234	0.0318	0.0317	0.0377	0.0579	0.0353	0.0501	0.0571	0.0372	0.0480	0.0623	0.5690	0.0578	**0.0734**	$7.0e^{-12}$
	Recall@40	0.0578	0.0600	0.0826	0.0773	0.0908	0.1196	0.0876	0.1175	0.1285	0.0912	0.1123	0.1409	0.1312	0.1335	**0.1535**	$6.6e^{-10}$
	NDCG@40	0.0293	0.0306	0.0415	0.0402	0.0478	0.0692	0.0454	0.0625	0.0697	0.0468	0.0598	0.0764	0.0704	0.0716	**0.0879**	$2.0e^{-12}$

Figure 8: Table 2. Performance of baseline models and SimRec on chosen datasets

From the results in Table 2, SimRec consistently achieves the best performance in terms of Recall@N and NDCG@N metrics compared to all baselines. The results have been statistically verified (with very small p-values), demonstrating that the improvement of SimRec is significant.
Comparison with GNN and SSL-based models:

- Although modern GNN models such as SGL, NCL, and HCCF have adopted self-supervision strategies to improve performance, they still cannot completely eliminate the over-smoothing phenomenon.
- For example, SGL can be affected by noise amplified through propagation steps, while NCL and HCCF tend to over-smooth irrelevant nodes.
- In contrast, removing the GNN structure in SimRec's inference process reduces the possibility of over-smoothing. SimRec spreads knowledge from the (well-trained) GNN model through distillation and regularization tasks, helping the model achieve higher performance.

Comparison with CF models that do not use GNN (e.g., NCF and AutoR):

- These models, although having similar MLP structures, yield results close to SimRec.
- This shows that simply using an MLP is not enough to capture complex interaction information in the graph.
- The addition of the KD mechanism in SimRec significantly improves the learning of high-level connections between users and items, thereby addressing the limitations of conventional MLP approaches.

3.4.3 Model ablation study

Ablation study. In machine learning, it is about studying and experimenting with a model by removing its components and analyzing the alternate results to discuss its importance in the main subject. [4]

In this paper, the authors have conducted the experiments on the applied sub-modules in SimRec, namely Prediction-level and Embedding-level knowledge distillation (PKD, EKD), Contrastive Regularization (CR), from testing the model without each.

Comparison on overall results.

Table 3: Ablation study on key components of SimRec.

Data		Gowalla		Yelp		Amazon	
Variant		Recall	NDCG	Recall	NDCG	Recall	NDCG
$-\mathcal{L}_1$		0.2180	0.1415	0.0756	0.0377	0.1012	0.0692
$-\mathcal{L}_2$	User	0.2292	0.1493	0.0806	0.0405	0.0998	0.0667
	Item	0.2266	0.1477	0.0808	0.0406	0.0974	0.0649
	Both	0.2222	0.1451	0.0787	0.0399	0.0938	0.0626
$-\mathcal{L}_3$	U-I	0.2330	0.1496	0.0814	0.0410	0.0939	0.0607
	U-U	0.2349	0.1512	0.0811	0.0407	0.0965	0.0634
	I-I	0.2331	0.1514	0.0813	0.0409	0.1009	0.0674
	All	0.2282	0.1480	0.0810	0.0407	0.0933	0.0605
SimRec		**0.2434**	**0.1592**	**0.0823**	**0.0414**	**0.1067**	**0.0734**

Figure 9: Table 3. Results of ablation on key components of SimRec

In the first glance, we could see clearly how the differences each one makes, where PKD ablation has significantly lower results than the other two on Gowalla and Yelp, following by EKD and then CR. However, this does not hold true on the Amazon dataset, as PKD shall have the least difference in comparison to the others, also by a significant margin. And the differences between the results of the EKD and CR is far more unclear as well.

But even then, we could see a non-neglibible drop on all the datasets when any of the components is cut out, compared to SimRec as a whole. This shows the necessity of every component in this system, that SimRec should be used as is.

It is worth noting that from this ablation study, the authors choose to take closer inspections on Gowalla and Amazon in the later experiments, as Gowalla and Yelp shall yield similar properties and hence, accuracy metrics but Gowalla has clearer difference between EKD and CR.

Comparison on learning curves.

Since SimRec is a breed between the two models, the ablation study is basically comparison between SimRec and the two.

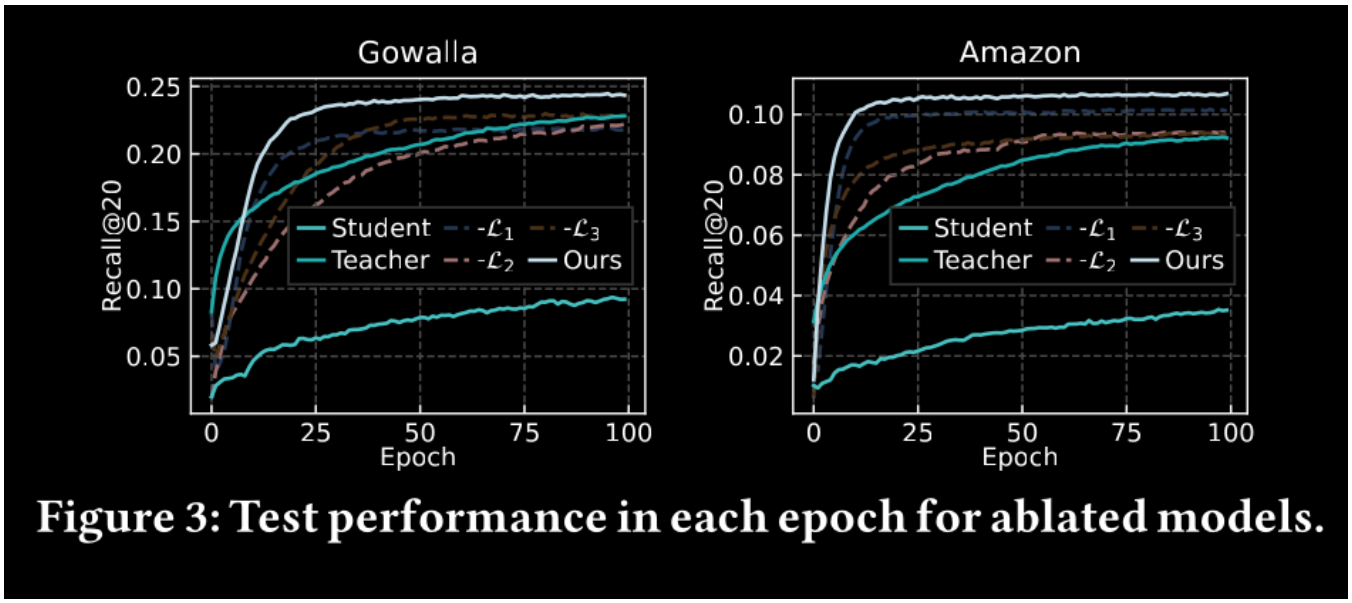

Figure 10: Figure 3. Learning curves on Gowalla and Amazon datasets

On Gowalla, SimRec has a lower start than the teacher GNN model, but tremendously improve its recall rate, to go past the GNN in the first 10 epochs, and already converges at around epoch 25, meanwhile GNN needs a longer time period to gradually increase its result effectiveness to epoch 100, but still nowhere near SimRec. The pattern can be seen even more clearly on the Amazon dataset. In conclusion: **SimRec achieves better results in far less time**.

Furthermore, the learning curves of component ablation also show that they consistently yield worse learning curves and worse converging results at every epoch compared to the original SimRec.

3.4.4 Model scalability study.

This experiment focuses on evaluating the scalability of the SimRec model on a large-scale dataset, specifically an e-commerce dataset from the Tmall platform with about 40 million user clicks. On the Tmall dataset, GNN models must apply graph sampling strategies to perform information propagation, with different scales (e.g., subgraphs with 1.6 million and 2.9 million edges).
In contrast, SimRec with MLP architecture does not require sampling because the inference process does not depend on the entire graph structure, thereby significantly reducing the computational cost.
The experimental goal is to verify SimRec's ability to handle large data compared to GNN-based models. At the same time, it evaluates both the recommendation performance (according to the Recall and NDCG metrics) as well as the inference time.
Better recommendation performance:

- SimRec shows higher Recall and NDCG scores than the tested baselines.
- This is explained by SimRec's ability to avoid over-smoothing on large graphs, thanks to the diffusion of knowledge from the GNN model to the MLP without having to directly propagate information across the entire graph.

Higher computational efficiency:

- SimRec's inference time is significantly lower.
- SimRec's MLP architecture has lower computational costs than the complex computational layers in GNN, especially when processing huge graph datasets.
- Since there is no need to perform graph sampling, the processing cost does not increase with the size of the entire graph, making the model well adapted to large-scale data environments.

3.4.5 Hyperparameter study

The goal of the hyperparameter study here is to understand how hyperparameters such as loss weights (λ_1, λ_2, λ_3), temperature (τ_1, τ_2, τ_3), and batch size ($|T_1|$) in the prediction-level distillation task affect the model performance.

Strength of Prediction-Level Distillation (λ_1, τ_1):

- The model uses a KD loss at the prediction level (L1) to "train" the student model to learn from the predictions of the teacher model.
- The results show that when the value of λ_1 is too small, the model does not receive enough "dark knowledge" from the teacher, leading to poor performance. On the contrary, if λ_1 is too large, L1 "overwhelms" the main loss, reducing the efficiency of the optimization process.
- In addition, using a small temperature of τ_1 helps to create larger gradients, thereby improving the transfer of knowledge information (see Figure 5(a)).

Strength of Embedding-Level Distillation (λ_2, τ_2):

- At the embedding level, the L2 task is to limit the difference between the embeddings of the student model and the teacher model.
- The authors find that the combination of λ_2 and τ_2 must be tuned appropriately: too high or too low values cause ineffective regularization, which can lead to over- or under-constraining the embeddings.
- Figure 5(b) clearly illustrates the performance change when changing these values, showing the need for "moderate" tuning to both ensure knowledge diffusion and avoid over-regularization.

Strength of Contrastive Regularization (λ_3, τ_3):

- The contrastive regularization task (L3) is designed to "push" the embeddings of unrelated nodes away to counteract the over-smoothing phenomenon.
- The results show that if λ_3 is too small or τ_3 is too high, the regularization is not strong enough to "push" the embeddings away, leading to the model learning over-smoothed representations.
- On the other hand, if the value is too strong, it can harm the model's ability to learn useful relationships between nodes. This is illustrated in Figure 5(c).

Effect of batch size $|T_1|$:

- Based on the experimental results (see Table), the results show that when $|T_1|$ increases from $1e3$ to $5e5$, the model performance (according to the Recall and NDCG indices) increases until a certain saturation level.
- The reason for this is that when the batch size is larger, the low-frequency noise in the teacher model's prediction will be "filtered" better, thereby providing more stable and accurate learning signals for the student model.

3.4.6 Over-smoothing investigation

The goal of this experiment is to determine whether **SimRec**'s graph-less architecture helps the model avoid "blurring" user and item embeddings. At the same time, the experiment compares the *MAD (Mean Average Distance)* values of the embeddings between SimRec and baseline models, including GCN-based models (e.g., GCCF) and SSL-based models.
Evaluation method:

- MAD is used as a measure of the dispersion of embeddings, with higher MAD values indicating that embeddings are more evenly distributed, thereby having higher discriminative power among nodes.
- This metric is computed on datasets such as Gowalla and Yelp, with a focus on the most popular users and items.

Comment:

- The results in the corresponding Table show that SimRec achieves higher MAD values than not only GCN models such as GCCF but also other state-of-the-art SSL methods.
- This demonstrates that, by employing an MLP architecture for inference, combined with a knowledge diffusion strategy and adaptive contrastive regularization, SimRec retains a higher level of embedding diversity.
- This "distinction" prevents embeddings from becoming too fuzzy, thereby improving the ability to discriminate and to accurately describe the unique features of each user and item.

References

[1] Maciej Besta, Florian Scheidl, Lukas Gianinazzi, Grzegorz Kwasniewski, Shachar Klaiman, Jürgen Müller, and Torsten Hoefler. Demystifying higher-order graph neural networks, 2024.

[2] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):27–34, Apr. 2020.

[3] Mengru Chen, Chao Huang, Lianghao Xia, Wei Wei, Yong Xu, and Ronghua Luo. Augmenting gnn-based collaborative filtering with hypergraph contrastive learning. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, pages 3747–3756. ACM, 2022.

[4] Paul R. Cohen and Adele E. Howe. How evaluation guides ai research: The message still counts more than the medium. *AI Magazine*, 9(4):35, Dec. 1988.

[5] Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. Towards robust graph neural networks for noisy graphs with sparse labels. 2022.

[6] Claudio Gallicchio and Alessio Micheli. Fast and deep graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3898–3905, Apr. 2020.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[8] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 639–648, New York, NY, USA, 2020. Association for Computing Machinery.

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering, 2017.

[10] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[11] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning, 2018.

[12] Jiaxi Lin, Tianxiong Zhou, Zhiwei Liu, and Philip S. Yu. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1148–1157. ACM, 2022.

[13] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 2320–2329. ACM, April 2022.

[14] T. Konstantin Rusch, Michael M. Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks, 2023.

[15] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20:61–80, 2009.

[16] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, page 111–112, New York, NY, USA, 2015. Association for Computing Machinery.

[17] Zeno Gantner Lars Schmidt-Thieme Steffen Rendle, Christoph Freudenthaler. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 452–461. AUAI Press, 2009.

[18] Rianne van den Berg, Thomas N. Kipf, and Max Welling. Graph convolutional matrix completion, 2017.

[19] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '19, page 165–174. ACM, July 2019.

[20] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. Disentangled graph collaborative filtering. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1001–1010. ACM, July 2020.

[21] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. Self-supervised graph learning for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 726–735. ACM, July 2021.

[22] Le Wu, Tong Chen, Richang Hong, Meng Wang, and Hao Wang. Self-supervised graph learning for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 726–735. ACM, 2021.

[23] Le Wu, Tong Chen, Richang Hong, Meng Wang, and Hao Wang. Self-supervised learning for recommendation with pseudo user representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 912–921. ACM, 2021.

[24] Xinyi Wu, Zhengdao Chen, William Wang, and Ali Jadbabaie. A non-asymptotic analysis of oversmoothing in graph neural networks. 2023.

[25] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Huang. Hypergraph contrastive collaborative filtering. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 70–79. ACM, July 2022.

[26] Bencheng Yan, Chaokun Wang, Gaoyang Guo, and Yunkai Lou. Tinygnn: Learning efficient graph neural networks. page 1848–1856, 2020.

[27] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '18, page 974–983. ACM, July 2018.

[28] Jiani Zhang, Xingjian Shi, Shenglin Zhao, and Irwin King. Star-gcn: Stacked and reconstructed graph convolutional networks for recommender systems, 2019.