Recommendation Systems via Graph-based Neural Networks

Tianqi Chen (tc34927) [1] **Yilin He (yh9767)** [2] **Qiujiang Jin (qj499)** [3]

Abstract

Empowered by the recent development of artificial intelligence, online recommendation market is growing rapidly. Among all the current competitors, Graph Neural Networks (GNNs) have shown state-of-the-art performance in providing highly-accurate and personalized recommendations. Based on this observation, we study relevant literature and decide to mainly focus on the application of GNN-based algorithms on movie recommendation problem. More specifically, we identify two such deep collaborative filtering algorithms as potential solutions to our problem: Neural Graph Collaborative Filtering (NGCF) and Light Graph Convolution Network (LightGCN). On top of that, we propose two negative sampling strategies to adapt the aforementioned algorithms to a multiscale movie rating dataset. We evaluate and analyze the model performances in terms of precision, recall and Normalized Discounted Culmulative Gain (NDCG). The empirical results demonstrate the efficacy of GNN-based collaborative filtering methods in recommendation, which can leads to huge business success.

1. Introduction

Personalized recommendation have been applied to many practical services such as E-commerce, advertising, and social media. In this study, we mainly focus on the movie recommendation scenario and utilize the movie rating data collected from websites like IMDb or Rotten Tomatoes. Each rating expresses a user's attitude towards a movie and contains some amount of information reflecting his/her personal preference. However, as the rating data are typically sparse with regard to the total numbers of users and movies, it is impossible to fully recover a user's preference and make precise and personalized recommendation solely based on his/her own ratings. Therefore, we need to take into account some additional information, such as the similarities between users and items. One corresponding solution is collaborative filtering (CF). The assumption of CF is: on one hand, users with similar behaviors (*e.g.*, watch history, purchase and ratings) would exhibit similar preference on items, and, on the other hand, items with similar user base would possess similar characteristics. Thus, the taste of one user can be inferred by combining the information from other users and items.

In the most common CF paradigm (Cao et al., 2019; He et al., 2017; Cheng et al., 2018; Kabbur et al., 2013), each user or item is assigned a learnable embedding vector, the prediction of user u's response[1] to item i is typically made through interaction such as inner product between user embedding $\mathbf{e}_u$ and item embedding $\mathbf{e}_i$. These embeddings are collectively trained to fit the observed responses, after that, they could be applied to predict unobserved response through the same numerical interaction. The design space of a CF algorithm usually includes two dimensions, the embedding generation (Wang et al., 2019; 2020) and modeling of user-item interaction (Koren, 2008; He et al., 2018). In this study, we fix the embedding interaction method to the efficient and widely adopted inner-product, and mainly focus on exploring the efficacy of different embedding generation process on movie recommendation.

Without any modification to the original user and item embeddings, the process of collaborative filtering is equivalent to rank-d factorization of the rating matrix (Koren et al., 2009), where d is the shared dimensionality of user embeddings and item embeddings. Though the model is parsimonious and usually perform well in practice, it has been criticized for not including collaboration in the embedding generation (Wang et al., 2019), therefore limits the ultimate potential of collaborative filtering efficacy. Since the inter-

[1]Department of Information, Risk, and Operations Management, The University of Texas at Austin, Austin, Texas, U.S.A [2]Department of Information, Risk, and Operations Management, The University of Texas at Austin, Austin, Texas, U.S.A [3]Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, Texas, U.S.A. Correspondence to: Tianqi Chen <tqch@utexas.edu>, Yilin He <yilin.he@mccombs.utexas.edu>, Qiujiang Jin <qiujiang@austin.utexas.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

[1]The response can either be explicit, *e.g.*, rating or like/dislike indicator, or be implicit, *i.e.*, if certain action has ever been observed.

actions between users and items naturally form a bipartite graph, the collaboration of user and item information could be enhanced by leveraging the message passing techniques introduced by the graph neural networks (GNNs) literature (Kipf & Welling, 2017; Hamilton et al., 2017; Yang et al., 2019). Namely, aside from the latent collaboration enforced by the user-item interactions, the GNNs explicitly propagate embeddings through the links between users and items, hence users interacting with the same items, or items possessing the same user base, would share embeddings with each other through high-order connectivity (as visualized in Figure 1). As a result, for the users or items that CF presumes to be similar, the correlation among their embeddings would naturally be enhanced.

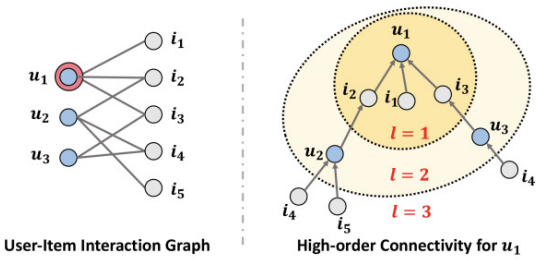


Figure 1. An illustration of the user-item interaction graph and the high-order connectivity. The node u_1 is the target user to provide recommendations for.

Based on this observation, NGCF (Wang et al., 2019) is proposed to adapt the embedding propagation rule of GNNs to the collaborative filtering context, followed by LightGCN (Wang et al., 2020) that further simplifies NGCF by removing some forward computation steps that are proved unnecessary for the recommendation tasks. Given the state-of-the-art performances achieved by these two methods, we select them for our movie recommendation task. Our major contributions are summarized as follows:

- we propose two negative sampling strategies to adapt NGCF and LightGCN to the multiscale movie rating dataset, we also test and heuristically explain the training efficacy of both strategies;
- we evaluate vanilla matrix factorization, NGCF, LightGCN on the movie rating dataset, and empirically analyze the cause to the difference in their performances.

The remaining content is organized as follows: in Section 2, we present the details of NGCF and LightGCN respectively; in Section 3, we describe the movie dataset used in our experiments; in Section 4, we report the empirical results and corresponding discussions; finally in Section 5, we end this report by some concluding remarks as well as a proposal of future work.

2. GNN-Based CF Methods

2.1. Neural Graph Collaborative Filtering

First, we briefly introduce the Neural Graph Collaborative Filtering. Let $\mathbf{e}_u^{(0)}$ denote the ID embedding of user u and $\mathbf{e}_i^{(0)}$ denote the ID embedding of item i. Then NGCF leverages the user-item interaction graph to propagate embeddings as:

$$\mathbf{e}_u^{(k+1)} = \sigma\left(\mathbf{W}_1^{(k)}\mathbf{e}_u^{(k)} + \sum_{i\in N_u} \frac{\mathbf{W}_1^{(k)}\mathbf{e}_i^{(k)} + \mathbf{W}_2^{(k)}(\mathbf{e}_i^{(k)} \odot \mathbf{e}_u^{(k)})}{\sqrt{|N_u||N_i|}}\right) \quad (1)$$

$$\mathbf{e}_i^{(k+1)} = \sigma\left(\mathbf{W}_1^{(k)}\mathbf{e}_i^{(k)} + \sum_{u\in N_i} \frac{\mathbf{W}_1^{(k)}\mathbf{e}_u^{k} + \mathbf{W}_2^{(k)}(\mathbf{e}_u^{(k)} \odot \mathbf{e}_i^{(k)})}{\sqrt{|N_u||N_i|}}\right) \quad (2)$$

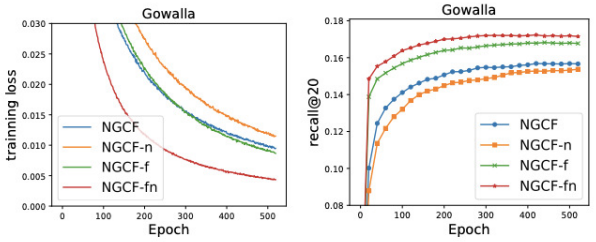
where $\mathbf{e}_u^{(k)}$ and $\mathbf{e}_i^{(k)}$ respectively denote the refined embedding of user u and item i after k layers propagation, σ is the nonlinear activation function[2], N_u denotes the set of items that are interacted by user u, N_i denotes the set of users that interact with item i, and $\mathbf{W}_1^{(k)}$, $\mathbf{W}_2^{(k)}$ are trainable weight matrix to perform feature transformation in layer k. By propagating K layers, NGCF obtains $K+1$ embeddings to describe a user u and an item i. It then concatenates these $K+1$ embeddings to obtain the final user embedding and item embedding, using inner product to generate the prediction score.

NGCF largely follows the standard GCN, including the use of nonlinear activation function σ and feature transformation matrices $\mathbf{W}_1^{(k)}$ and $\mathbf{W}_2^{(k)}$. However, we suspect that these two operations are not as useful for collaborative filtering. In supervised graph node classification, each node has a lot of meaningful features as input. Thus performing multiple layers of nonlinear transformation is beneficial to feature learning. Nevertheless, in collaborative filtering, each node of user-item interaction graph only has an ID as input which has no concrete interpretations. In this case, performing multiple nonlinear transformations will not contribute to learn better features; even worse, it may add the difficulties to train well.

The authors conduct empirical experiments to verify this suspection. They conduct three simplified NGCF: (1) NGCF-f which removes the feature transformation matrices $\mathbf{W}_1^{(k)}$ and $\mathbf{W}_2^{(k)}$, (2) NGCF-n which removes the non-linear activation function σ, (3) NGCF-fn which removes both the feature transformation matrices and non-linear activation function. They compared the standard NGCF with these three simplified versions of NGCF on two different dataset: the Gowalla dataset and the Amazon-Book dataset. They

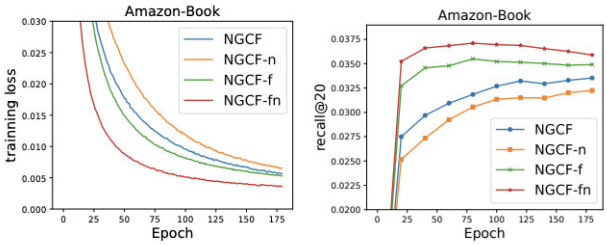
[2] In Wang et al. (2019), the activation is selected as LeakyReLU.

presented both the training loss and the test error of these different algorithms. See Figure 2 and Figure 3 for details. We observe that the NGCF-fn without the feature transformation matrices and non-linear activation function performs best in terms of the training error and test error on both different datasets.



(a) Training loss on Gowalla (b) Testing recall on Gowalla

Figure 2. Training curves (training loss and testing recall) of NGCF and its three simplified variants on the Gowalla dataset.



(c) Training loss on Amazon-Book (d) Testing recall on Amazon-Book

Figure 3. Training curves (training loss and testing recall) of NGCF and its three simplified variants on the Amazon-Book dataset.

2.2. Light Graph Collaborative Filtering

Based on the observation of Section 2.1, it's necessary to design simplified version of the NGCF to improve the performance. Therefore, Light Graph Convolution Network (LightGCN) model was proposed, as illustrated in Figure 4. In LightGCN, we adopt a simple weighted sum aggregator and abandon the use of feature transformation and nonlinear activation. The graph convolution operation (a.k.a., propagation rule) in LightGCN is defined as:

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u||N_i|}} \mathbf{e}_i^{(k)}, \tag{3}$$

$$\mathbf{e}_i^{(k+1)} = \sum_{u \in N_i} \frac{1}{\sqrt{|N_u||N_i|}} \mathbf{e}_u^{(k)}. \tag{4}$$

If we compare the Equations (3) and (4) with the previous Equations (1) and (2) in NGCF, we observe that the update rules are much concise and simple. In LightGCN, the only trainable model parameters are the embeddings at the 0-th layer, *i.e.*, $\mathbf{e}_u^{(0)}$ for all users and $\mathbf{e}_i^{(0)}$ for all items. When they are given, the embeddings at higher layers can be computed via Equations (3) and (4). After propagating embeddings through K LightGCN layers, we further combine the embeddings obtained at each layer to form the final representation of a user or an item:

$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)}, \qquad \mathbf{e}_i = \sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)}, \tag{5}$$

where $\alpha_k \geq 0$ denotes the importance of the k-th layer embedding in constituting the final embedding. It can be treated as a hyperparameter to be tuned manually. There are three reasons that we perform layer combination to get final representations. First with the increasing of the number of layers, the embeddings will be over-smoothed (Li et al., 2018). Thus simply using the last layer is problematic. Second the embeddings at different layers capture different meanings. The first layer enforces smoothness on users and items that have interactions, the second layer smooths users and items that have overlap on interacted items and users, and higher-layers capture higher-order proximity (Wang et al., 2019).Thus combining them will make the representation more comprehensive. Last combining embeddings at different layers with weighted sum captures the effect of graph convolution with self-connections, an important trick in GCNs.

The model prediction is defined as the inner product of user and item final representations:

$$y_{ui} = \mathbf{e}_u^\top \mathbf{e}_i, \tag{6}$$

which is used as the ranking score for the recommendation system.

We employ the Bayesian Personalized Ranking (BPR) loss (Rendle et al., 2009), which is a pairwise loss that encourages the prediction of an observed entry to be higher than its unobserved counterparts:

$$L = -\sum_{u=1}^{M} \sum_{i \in N_u} \sum_{j \notin N_u} \ln \sigma(y_{ui} - y_{uj}) + \lambda \|\mathbf{E}^0\|^2, \tag{7}$$

where λ controls the l_2 regularization strength and $\mathbf{E}^0$ are the embeddings of the 0-th layer which are the only trainable parameters of LightGCN. Note that we do not introduce

dropout mechanisms, which are commonly used in GCNs and NGCF. The reason is that we do not have feature transformation weight matrices in LightGCN, thus enforcing l_2 regularization on the embedding layer is sufficient to prevent overfitting. This showcases that LightGCN is easier to train and tune than NGCF.

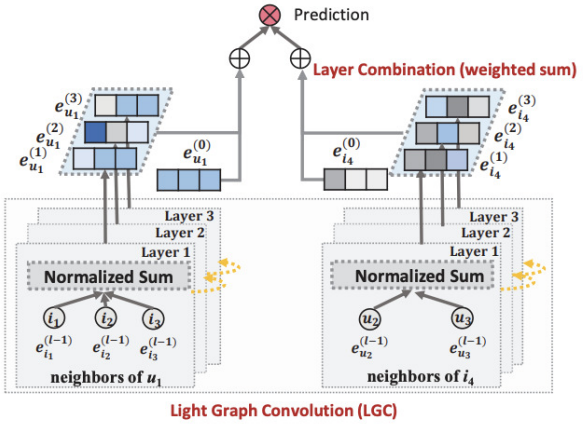


Figure 4. An illustration of LightGCN model architecture. In LightGCN, only the normalized sum of neighbor embeddings is performed towards next layer; other operations like self-connection, feature transformation, and nonlinear activation are all removed, which largely simplifies GCNs. In Layer Combination, we sum over the embeddings at each layer to obtain the final representations.

3. Datasets

In our experiment, we used a light-weight movie dataset, which is adapted form the original MovieLens dataset[3].

3.1. Data description

The dataset contains ∼1M ratings between 671 users and 9066 movies. The ratings have 9 scales from 1.0 to 5.0 with 0.5 as a unit. It is guaranteed that each user will have at least 20 ratings and each movie is rated at least once. Using a stratified sampling strategy, we split the dataset into training set and testing set at a radio of $4:1$ and ensured that the numbers of samples associated with each individual user in the two sets also follows the same ratio.

3.2. Preprocessing

Since our main purpose is to design an effective ranking algorithm to recommend potential (unseen) movies of interest to the users, we did not formulate our task into a multiclass classification problem[4] (*i.e.*, rating prediction).

Considering that the collaborative filtering model typically takes implicit feedback, such as indicators of whether a user once clicked an advertisement, we chose to binarize the rating data using a threshold of 3.0, *i.e.*, we labeled the ratings greater than or equal to 3.0 as positive examples and the ones less than 3.0 as negative examples. The thresholding here is informed by the histogram of user ratings (see Figure 5) and a commonsense that a recommendation with a rating of 3 out of 5, which is average of 1 (min) and 5 (max), can be considered acceptable to the user.

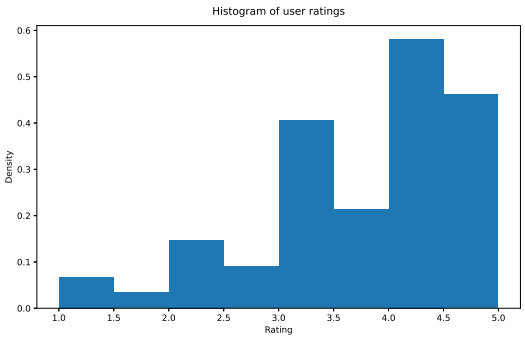


Figure 5. Histogram of user ratings

4. Experiments

4.1. BPR Loss and Negative Sampling Strategies

In our experiment, we choose to use a ranking loss, Bayesian Personalized Ranking (BPR) loss (see Equation 7) to train our model. To recap, BPR is a pairwise loss that encourages the difference between positive and negative samples. In order to incorporate the rating score into our model training process, we have tested two negative sampling strategies.

Specifically, for any user, a sample is drawn randomly from the movies rated by the user. We use $r_o^{(+)}$, $r_o^{(-)}$ to denote random draws of a viewer's positive and negative ratings, and r_u denotes the potential rating from the same viewer on a randomly selected movie that he/she does not review. $>$ indicates the order relation in terms of ranking. Here we propose two assumptions of the inherent order relation among the ratings, first, for a specific reviewer, the potential rating of an unrated movie is expected to be between an observed low rating and an observed high rating submitted, we formalzie this assumption by $r_o^{(+)} > r_u > r_o^{(-)}$; second, the potential rating of an unrated movie is expected to be

[3]https://grouplens.org/datasets/movielens/

[4]In that case, we can only evaluate the accuracy of our model on movies with ratings in the testing set, which fails to take account into the movies outside it.

lower than an observed rating by the same reviewer, no matter the observed rating is high or low, we formalized this assumption by $r_o > r_u$.

We express the two negative sampling strategies under each assumption as follows:

- ($r_o^{(+)} > r_u > r_o^{(-)}$) If it has a positive rating, draw a negative sample uniformly from the union of unrated ones and negatively-rated ones; if it has a negative rating, draw a positive sample uniformly from the union of unrated ones and positively-rated ones (**Note**: in this case the first sample becomes a negative sample in the BPR loss)
- ($r_o > r_u$) Pair it with another movie uniformly sampled from all the movies the user has not rated yet.

Intuitively speaking, the first negative sampling strategy will give more weights to positively-rated observations than unobserved ones than negatively-rated ones whereas the second strategy generally give more weights to observed ones than unobserved ones.

4.2. Hyperparameter Settings

In this experiment, for NGCF and LightGCN we chose the number of hidden layers to be 3. For all three collaborative filtering models, we used a embedding with 64 dimensions. In training, we used Adam optimizer and set learning rate to be 0.001, l2 regularization to be 0.0001.

4.3. Empirical results

High-rating movie recommendation: The first set of experiments are intended to evaluate the movie recommendation efficacy of three collaborative filtering methods under the two negative sampling strategies as specified in Section 4.1. The three models are: (i) vanilla matrix factorization (MF), (ii) NGCF (Wang et al., 2019), and (iii) LightGCN (Wang et al., 2020). We follow the evaluation protocol of LightGCN (Wang et al., 2020) and report the performance of each model by three metrics[5]: (i) Precision@20, (ii) Recall@20, and (iii) Normalized discounted culmulative gain (NDCG@20). For all the three evaluation metrics, a higher value indicates a better model.

Table 1 records the recommendation performances of the three CF models on our customized movie dataset and under the two negative sampling strategies. Two conclusions could be draw from Table 1. First, models that employ GNN technique (NGCF, LightGCN) generally perform better than vanilla MF, no matter which negative sampling strategy is adopted. Such observation is in consistent with the empirical results reported in Wang et al. (2019) and Wang et al. (2020) where some click-through rating (CTR) datasets are adopted to evaluate the recommendation performances of these three CF methods.

Second, models trained with negative sampling strategy $r_o > r_u$ obtain better recommendation results than models trained with $r_o^{(+)} > r_u > r_o^{(-)}$. Note that the two negative sampling strategies correspond to two assumptions of the viewers' general preference between movies where ratings are unobserved, and movies they submitted low ratings, we find that the inherent characteristic of ratings may be better reflected by assumption $r_o > r_u$, that the viewers would even like a low-rated movie than an unrated counterpart. The heuristic is that most people tend to watch movies that they believe they could enjoy, they would have a general preference for watched movies than those they did not watch. Since to submit a review, one usually watches the movie at the first place, it is reasonable to treat the action of "submitting rating" as a signal of preference.

	Models	Precision@20	Recall@20	NDCG@20
$r_o^{(+)} > r_u > r_o^{(-)}$	MF	5.67	5.69	6.88
	NGCF	6.92	7.87	9.10
	LightGCN	11.28	12.65	16.39
$r_o > r_u$	MF	7.24	7.14	9.20
	NGCF	16.72	20.95	25.01
	LightGCN	**16.33**	**20.13**	**24.27**

Table 1. Ranking evaluation metrics (in %). The upper group are performances under negative sampling strategy $r_o^{(+)} > r_u > r_o^{(-)}$, the lower group is under negative sampling strategy $r_o > r_u$.

Relationship between complexity and performances: From Table 1, we find that between the two GNN-based CF methods, LightGCN consistently outperforms NGCF no matter which negative sampling stratey is adopted. The second set of experiments are intended to look into this phenomenon. Inspired by the ablation studies in Wang et al. (2020), we create two variants of NGCF with different model complexities by gradually taking out user-item embedding interaction and embedding transformation from the original NGCF forward propagation function. For simplicity, we show the propagation rules of the three variants (including the original NGCF, represented by NGCF-full) in matrix format:

- (NGCF-full) $\mathbf{E}^{(k+1)} = \sigma\big((\mathbf{L}+\mathbf{I})\mathbf{E}^{(k)}\mathbf{W}_1^{(k+1)} + (\mathbf{L}\mathbf{E}^{(k)} \odot \mathbf{E}^{(k)})\mathbf{W}_2^{(k+1)}\big)$,
- (NGCF-ego-t) $\mathbf{E}^{(k+1)} = \sigma\big((\mathbf{L}+\mathbf{I})\mathbf{E}^{(k)}\mathbf{W}_1^{(k+1)}\big)$,
- (NGCF-ego-n) $\mathbf{E}^{(k+1)} = \sigma\big((\mathbf{L}+\mathbf{I})\mathbf{E}^{(k)}\big)$, where

$$\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}, \text{ and } \mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^\top & \mathbf{0} \end{bmatrix},$$

[5]The implementation of the three metrics follows the code https://github.com/gusye1234/LightGCN-PyTorch

$\mathbf{R}$ denotes the rating matrix, and $\mathbf{D}$ is the diagonal degree matrix. The matrix format propagation function of NGCF-full is equivalent to Equations (2) and (3).

To have a clearer view of the relationship between model complexity and recommendation performances, we tested vanilla matrix factorization, LightGCN and three variants of NGCF under the first negative sampling strategy[6]. We report the experiment results in Table 2: the most parsimonious model is MF, next is LightGCN, where performances reach peak, then as the model complexity increases, the performances gradually decrease. From our second set of experiments, we conclude that LightGCN benefits from a more appropriate model complexity than NGCF for the movie recommendation task.

Model	Precision@20	Recall@20	NDCG@20
MF	5.67	5.69	6.88
LightGCN	**11.28**	**12.65**	**16.39**
NGCF-ego-n	8.76	10.58	11.23
NGCF-ego-t	7.23	8.33	10.48
NGCF-full	6.92	7.87	9.10

Table 2. Model performance with ascending complexity.

5. Conclusion and Future Work

In this project, we apply three collaborative filtering methods to the movie recommendation task, and analyze the empirical results. To adapt these CF methods to our unique movie rating dataset, we proposed two negative sampling strategies, which correspond to two assumptions on viewers' preference priority on unrated movies and low-rated movies. Through the empirical studies, we find that the negative sampling strategy derived from the assumption that viewers generally prefer rated movies over unrated movies leads to better model performances. We also find that GNN-based CF methods outperform vanilla matrix factorization by a great margin, which shows the efficacy of GNN's neighborhood feature propagation mechanism in producing high-quality user/item embeddings. Finally, the combination of "$r_o > r_u$" assumption + LighGCN already yields satisfactory movie recommendation performances (*e.g.*, the Recall is 20.13%, meaning that on average for each five recommendations we make, the viewer would accept one).

5.1. Hierarchical Recommendation Model

Based on our observation of the success of the "$r_o > r_u$" strategy and the heuristic behind, we think a hierarchical recommendation model can be beneficial in this case. Specifically, we will have a two two-stage model. In the first stage, we will train a coarse submodel to predict the probability of whether a user will watch a movie or not. In the second stage, we will train a fine submodel to predict the conditional probability of whether the user will give a positive rating given the fact that he/she has already watched the movie. By chaining the two models, we can obtain the probability of the event that a user will not only watch a certain movie but also give positive rating. For the first submodel, a simple solution is to use LightGCN trained with "$r_o > r_u$" strategy. For the second submodel, we can train a model solely based on the observed data and use the BPR loss with "$r_o^{(+)} > r_o^{(-)}$" strategy. In practice, we will use the coarse submodel to help us select a list of movies that a user is likely to watch and then use the fine model to pick out those models to which the user will actually give high rating scores (*i.e.*, like the movie). In this way, we expect the user would not only accept our movie recommendation for this time, but also build confidence in the quality of our recommendation.

[6]Because when the positive-negative pairs are created by the first way, the differences in performances are more significant between LightGCN and NGCF.

References

Cao, Y., Wang, X., He, X., Hu, Z., and Chua, T.-S. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *WWW*, 2019.

Cheng, Z., Ding, Y., Zhu, L., and Kankanhalli, M. S. Aspectaware latent factor model: Rating prediction with ratings and reviews. In *WWW*, pp. 639–648, 2018.

Hamilton, W. L., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NeurIPS*, pp. 1025–1035, 2017.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. Neural collaborative filtering. In *WWW*, pp. 173–182, 2017.

He, X., He, Z., Song, J., Liu, Z., Jiang, Y.-G., and Chua, T.-S. Nais: Neural attentive item similarity model for recommendation. In *TKDE 30, 12*, pp. 2354–2366, 2018.

Kabbur, S., Ning, X., and Karypis, G. Ism: factored item similarity models for top-n recommender systems. In *KDD*, pp. 659–667, 2013.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, pp. 1025–1035, 2017.

Koren, Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pp. 426–434, 2008.

Koren, Y., Bell, R. M., and Volinsky, C. Matrix factorization techniques for recommender systems. In *IEEE Computer 42, 8*, pp. 30–37, 2009.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, pp. 3538–3545, 2018.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pp. 452–461, 2009.

Wang, X., He, X., Wang, M., Feng, F., and Chua, T.-S. Neural graph collaborative filtering. In *SIGIR*, pp. 165–174, 2019.

Wang, X., He, X., Wang, M., Feng, F., and Chua, T.-S. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*, pp. 639–648, 2020.

Yang, K., Swanson, K., Jin, W., Coley, C., Eiden, P., Gao, H., Guzman-Perez, A., Hopper, T., Kelley, B., Mathea, M., et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388, 2019.