SI 608

Networks

SI 608 Final Report

2021 Fall

Name:

Junjie Zhang (junjayz)

Tianyi Yang (jackyty)

Xun Zhou (xunzhou)

Yang Yu (ipanema)

Instructor:

Dr. Ceren Budak

December 24, 2021

1 Motivation

Nowadays, the application scenarios of recommendation systems are ubiquitous in our lives. For instance, after opening a shopping software like Amazon, customized items will automatically come to our screen. The general task of such system is to recommend proper items adjusting to users' preferences. As the system will face a large number of audience with high variety, designing the recommending strategies becomes a interesting and challenging thing.

In this system, we notice that the data of interaction between users and items is critically important, which can be mathematically abstracted as a bipartite graph. In this bipartite graph, nodes on the two sides represent users and items respectively, and the edges can be created and weighted by the user's historical ratings on the items. This graph represents the previous preferences of users and may imply their future ones. Therefore, to make recommendation, our task is just to find those edges with potential high weights that have not been connected in this bipartite graph, indicating a candidate preference by a user to an item.

To achieve this goal, our team will first conduct a classical algorithm, collaborative filtering. This algorithm is also related to network. For example, in user-based collaborative filtering, it can be regarded as first applying user projection on the bipartite graph according to similarity. And then we recommend those items highly rated by users who connect to the target user in the user projection graph. Furthermore, we will also perform other algorithms such as random walk to explore this bipartite network and finally create satisfying strategies for recommendation.

2 Research Questions

The idea of collaborative filtering is to find a certain similarity (the similarity between users or the similarity between the subject matter) through group behavior, and make decisions and recommendations for users through the similarity. And our task is to predicted the rating for unseen movie from users.

Generally speaking, collaborative filtering recommendations are divided into three types.

- The first is user-based collaborative filtering
- The second is item-based collaborative filtering
- The third type is model based collaborative filtering

User-based collaborative filtering mainly considers the similarity between users and users. Item-based collaborative filtering is similar to user-based collaborative filtering, except that we turn to finding the similarity between items and items. Only when the target user's rating of certain items is found, then we It can predict similar items with high similarity, and recommend several similar items with the highest scores to the user.

Our problem is the data of m items and n users. Only some users and some data have scoring data, and the scores of other parts are blank. At this time, we need to use the existing sparse data to predict

those The rating relationship between the blank items and the data, find the highest-scoring item and recommend it to the user.

We want to design a model-based item-oriented prediction algorithms. We first build a item graph representing the item similarity matrix. For the item graph P of size m we build, the weight of the edge between nodes i and j is equal to the probability of passing from item i to item j of a user Then we computes the rank value by random walk algorithm.

Random walk applied here is to infer transition probabilities between items based on their similarities and models finite length random walks on the item space to compute predictions. Finally, we interprets and scales the ranks scores as final ratings. We conduct NMSE evaluation of our model-based method with item-based top-K method. With this random walk model-based method, We can do better deal with the case when the data is not plentiful and the data martrix is sparse.

3 Related Work

Collaborative filtering, literally understood, includes two operations: collaboration and filtering. The so-called collaboration is the use of group behavior to make decisions (recommendations). Biologically, there is a saying of co-evolution. Through the effect of synergy, the group can gradually evolve to a better state. For the recommendation system, through the continuous synergy of users, the final recommendation to the user will become more and more accurate. And filtering is to find (filter) the schemes (subjects) that users like from feasible decision-making (recommended) schemes (subjects).

Specifically, the idea of collaborative filtering is to find a certain similarity (the similarity between users or the similarity between the subject matter) through group behavior, and make decisions and recommendations for users through the similarity.

3.1 Movie Recommendation with Neural Network

As a classic task for recommender system, the movie recommendation has been the benchmark and main objective for many works in the recommendation system area. With the development of deep learning, many scholars have adapted neural network (NN) with traditional methods to gain better performance on movie recommendation and improve the performance on the MovieLens dataset [2]. He et al. leverage the non-linearity of multi-layer perceptron (MLP) to model the interaction between user and item features [3]. Also, Sedhain et al. use auto encoder to model the interaction [6]. Then, different variation of autoencoder has been used for modeling the similarities. Liang et al. introduce the variational autoencoder to get better statistically interpretability [4]. Vančura and Kordík further introduce a much deeper autoencoder FLVAE to improve the performance [10].

3.2 Graph Based Recommender System

To solve the sparse and cold start problems, the knowledge graph (KG), which has the extra information and the connectivity information among users and items, is employed by many works. Some scholars make use of KG by embedding the knowledge. Zhang et al. embed items' features from visual, texual content as well as the its structure information to do joint learning [14]. Wang et al. use a cross and compress unit to share knowledge between the KB embedding module and the recommandation module to do multi-task learning [11]. But these work may underutilize information about connectivity, thus, some scholars focus the connection information and try to employ structure in the KG. Shi et al. propose a method to weight the semantic meta path [8]. Sun et al. use recurrent networks to mine and embed the path between users and items [9]. Also, as the development of graph neural network, lit is possible to utilize the KG in propogation manner. Wang et al. use graph attention network to grab the high-order relations to refine the nodes' embedding [12]. Qu et al. further proposed Knowledge-enhanced Neighborhood Interaction model which stresses the interaction between item-side neighbors and user-side neighbors [5].

4 Dataset

We chose MovieLens as our main dataset. MovieLens is a research platform for recommend system created by University of Minnesota and constantly collects ratings on movies from users [2]. Now it has become one of the most important databases in the field of recommend system. The dataset includes the main form of ratings on movies by users and some metadata on movies and users. The metadata of movies includes the title, release date and genre information. And the metadata of users tell demographic information, namely age, occupation and gender.

This dataset is pretty suitable for recommend system research as well as network analysis. The most important table in MovieLens, on ratings, has a simple form which can be regarded as a matrix, whose x axis is the user, y axis is the item, and each point represents the ratings on a movie by a user. From another perspective, this matrix can be seen as a bipartite graph between users and items. Our task is like to find the hidden edge in this graph and thus we are able to apply knowledge in network.

5 Data Collection

MovieLens is a friendly database providing easy access to downloading. We just found the links for downloading on the website GroupLens [1]. We started by downloading the basic version MovieLens 100K Dataset which has a relatively small size. This dataset has totally 943 users, 1682 items and 100000 ratings.

Afterwards, if time allows, we will further move on datasets with larger sizes such as MovieLens 1M Dataset or MovieLens 10M Dataset.

6 Methods

In this section, we are going to first introduce three baseline models of collaborative filtering. Then we have tried to improve the model from three perspectives, which are improving random walk algorithm, adding diffusion-based similarity and applying the multilayer perceptron to learn the corresponding relation.

6.1 Baseline Model

In our project, we have applied three baseline models which are user-based collaborative filtering, item-based collaborative filtering and random walk on the MovieLens data. The random walk method [13] is built in consideration of the impact of sparse matrix on collaborative filtering , so the following will first introduce collaborative filtering and then random walk.

For collaborative filtering, we first calculate the user similarities matrix and item similarities matrix using cosine method.

The idea of user-based cf is that when calculating user u's rating for item i, all similar users' ratings for item i should be considered. Therefore, the simple idea is to use the user similarity matrix dot product rating matrix. To calculate user-based prediction, we also normalize the similarities matrix and adjust users' rating by subtracting the average to ensure that we can find users' who have similar rating system.

$$prediction_{user\ based} = R_{mean} + \frac{Sim \cdot (R - R_{mean})}{\sum_u Sim_u} \quad (1)$$

For the item-based cf, the idea is that when calculating user u's rating fo r item i, we will take all similar items rating by user u into account. Therefore, we use the rating matrix to dot product the item similarities matrix. The feature of item-based cf is that each prediction is only based on each user's own historical rating, which means it will not take other users' rating into consideration.

$$prediction_{item\ based} = \frac{R \cdot Sim}{\sum_i Sim_i} \quad (2)$$

However, sparse matrix is very common in real world. But whether it is user-based cf or item-based cf, they cannot consider the ratings of three or more users at the same time, which lead to their poor performance in the sparse matrix. Suppose item i and item j do not appear at the same time, but they have a high probability of appearing at the same time as the third item k. For user-based cf, since only the similarity between two users can be considered, it is impossible to learn that all items i, j, and k have a high similarity to each other. And since item-based cf only consider the rating history of one user, it can only draw the conclusion that item i and j are not related.

Hilmi Yıldırım and Mukkai S. Krishnamoorthy[13] proposed that random walk can be applied to alleviate the sparse matrix problem. When the item similarities matri x is regarded as a transition matrix, it is entirely possible for item i to walk to item k and then to item j, which can easily solve the sparse problem. What's more random walk can also introduce a damping factor which control jumping probability, which also increases the generalization of the model ability.

Calculating the random walk algorithm first requires a transition matrix. Since we recommend item, we use item similarities matrix as the transfer matrix here. Moreover, jumping probability is also included. P refers to the transition matrix, and β refers to the jumping probability.

$$P_{ij} = (1-\beta) * \frac{Sim}{\sum_j Sim_{ij}} + \frac{\beta}{\#\ of\ items}$$

Through random walk and related calculations, Hilmi and Mukkai obtained the following formula. For detailed mathematical derivation, please refer to their paper.[13] Here, R refers to the rating matrix, α means the probability of item continuing its transition, and P refers to the transition matrix. A little change we made here is that, it is almost impossible to calculate inverse matrix of $(I - \alpha P)^{-1}$, so we use pseudo-inverse matrix instead.

$$prediction = \alpha RP(I - \alpha P)^{-1}$$

6.2 Random Walk Method Improvement

In the baseline model of random walk, we simply use item-item similarity to fill the transition matrix, which we noticed afterwards that can be optimized. There are at least two other ways that can be considered. The first is that since the user-item matrix can be regarded as a bipartite graph, we can project it into an item-item graph. For this project, we define the edge weight of the transition matrix as the number of users who have rated the same movie by no less than 3 points. The second way is that we can utilize the genre information of movies. The metadata of MovieLens dataset records genre information for every movie in 19 binary dimensions. By calculating the cosine distance between movie genre vectors we can acquire another kind of transition matrix. Now after we get 3 kinds of transition matrix, the item similarity matrix P_{item_sim}, the bipartite graph projection matrix P_{bi_proj} and the genre similarity matrix P_{gen_sim}, we can linearly combine them to acquire a final matrix

$$P_{mix} = w_1 P_{item\_sim} + w_2 P_{bi\_proj} + (1 - w_1 - w_2) P_{item\_sim}, (w_1 \geq 0, w_2 \geq 0, w_1 + w_2 \leq 1).$$

Correspondingly, the final transition matrix will also include jumping factor, and the formula become

$$P_{final} = \beta P_{mix} + (1-\beta) P_{uniform},$$

and our prediction of ratings becomes

$$prediction_{mix} = \alpha RP_{mix}(I - \alpha P_{mix})^{-1}.$$

6.3 Diffusion-based Similarity

A simple idea of user-based similarities is that we are trying to find users who have similar tastes. When we implement the normal user-based collaborative filtering, we only take ratings into the consideration of finding similarities. Besides the ratings data, since MovieLens 10M Dataset was released, we started to have the tags that users put on movies. By using tag data, Shang[7] proposes that maybe a new user similarity matrix can be built by integrating users' preference of items and tags.

Tags data consists of UserID, MovieID, Tag and Timestamp, and thus we can build a item-user-tag tripartite network. When calculating the diffusion-based similarity matrix, we need to first split the tripartite into two bipartite graphs, which are user-item bipartite and user-tag bipartite.

Firstly, considering the user-item bipartite, and our goal is to get a user similarity matrix. Shang[7] assumes that we can distribute a unit of recommend resource to each user at the initial step. Then Shang[7] process the graph in two steps. At the first step, each user contributes the resources equally to the connected items. Below is the equation of distribution

$$r_{\alpha v} = \frac{a_{v\alpha}}{k(v)}$$

where $r_{\alpha v}$ represents the resource of an item a which delivered from v, $a_{v\alpha}$ represents whether there is a connection between user v and item α, $k(v)$ represents the degree of v in the user-item bipartite network. After all the items have been distributed by resources, the second step is to distribute the resource back to the user. Therefore, we can get the similarity between user u and user v.

$$S_{uv} = \sum_{\alpha \in item} \frac{a_{u\alpha} \cdot r_{\alpha v}}{k(\alpha)} = \frac{1}{k(v)} \sum_{\alpha \in item} \frac{a_{u\alpha} \cdot a_{\alpha v}}{k(\alpha)}$$

Shang[7] where S_{uv} represents similarity of user u and user v, $a_{u\alpha}$ represents whether there is a connection between user u and item α. Since at first we divide our item-user-tag tripartite into 2 bipartite, we can get two similarity matrix. By combining the two matrix linearly, we can finally get a new similarity matrix which contains information from the tag data set.

$$s^{\star} = \lambda s_{uv} + (1 - \lambda) s^{'}_{uv}$$

where λ represents the hyperparameter of the linear combination.

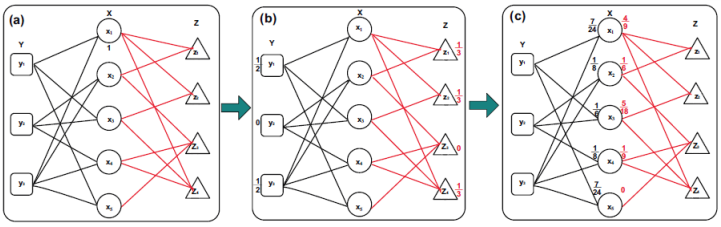

Figure 1: diffusion-based algorithm on tripartite graph of tags data [7]

6.4 Neural Collaborative Filtering

For the traditional matrix factorization method, it simply combines the latent of user and the item feature linearly, which may not be sufficient. For the neural collaborative filtering, the matrix factorization is replace by the multilayer perceptron.

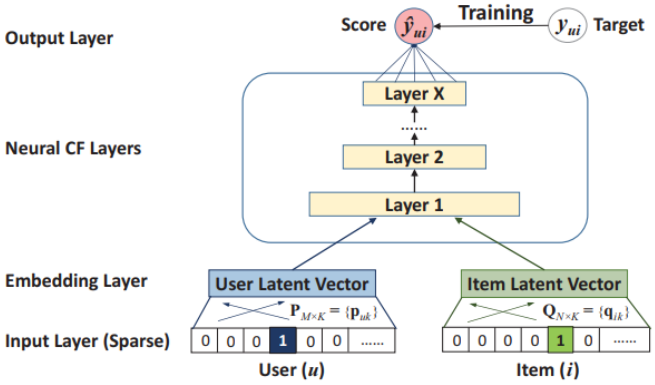

Figure 2: The overall framework of nueral collaborative filtering [3]

To make the structure simple, for the neural CF layer, we just calculate the dot product of user latent vector and item latent vector. The sigimoid function is used to map the product to the range of 0 to 1. To make the schema of the input data consistent with that of the output data, the input data should go through the minmax scaler.

7 Challenges

7.1 Random Walk Method Improvement

Since we use a linear combination of transition matrix, the parameters α, β, w_1 and w_2 need to be tuned. However, as it is not an end-to-end model, we cannot simply propagate gradient from the final score to tune these parameters. Alternatively, we decided to utilize grid search. We enumerate some reasonable choices of 4 parameters and get the experiment results. As we used Rooted Mean Squared Error (RMSE) as the metric, the figures of score vs. parameters are shown in Figure 1 & 2.

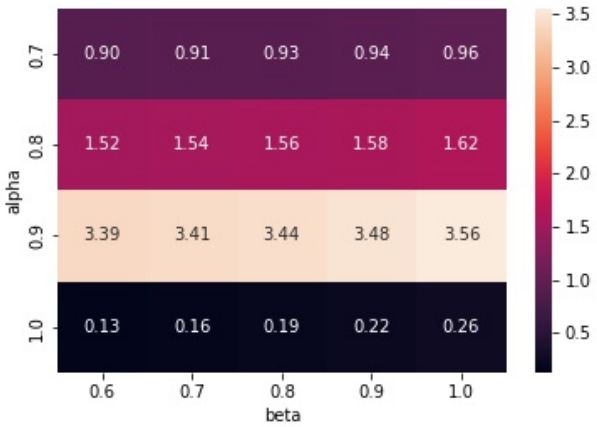

Figure 3: RMSE vs. α and β

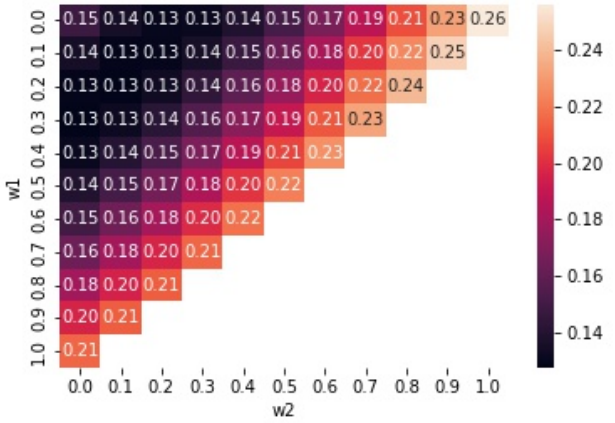

Figure 4: RMSE vs. w_1 and w_2

And we picked the parameters with the smallest RMSE, which are $\alpha = 1.0$, $\beta = 0.6$, $w_1 = 0.3$ and $w_2 = 0.0$ respectively.

7.2 Diffusion-based Similarity

As is mentioned in the result section, the major challenge is the sparse matrix problem. In the real world, it's very hard to collect most of the relationships for a bipartite relationship. Also, it is very common to have lots of repeated relationship. Therefore, it is necessary to find out a way dealing with the sparse problem.

For the further research, we really recommend to apply the random walk method which can fill most values with a ranking score on this diffusion-based method.

7.3 Neural Collaborative Filtering

In the prepossessing of data part, the embedding of the user latent vector and the movie latent vector are quite simple, namely map the id to the a contiguous array starts at zero. No meta data is embedded here, namely the model can not tell from the favourite artist of user A, or the favourite movie genre. The reason why we do not add the meta data is because we want to make a comparison with the matrix factorization method which can only work on rating data.

For the neural layer, here we simply conduct the dot product, due to the computing resource limit. By increasing one or two dense layer along with the drop out layer will absolutely make the NMSE performance better.

8 Results

8.1 Random Walk

We mainly divide the MovieLens 10M Dataset into 3 parts, namely 75% for train, 10% for validation and 25% for test. we use rooted mean square error (RMSE) to evaluate the quality of the model. The score for baseline random walk is 1.50. After mixing 3 kinds of transition matrix, the score becomes 1.25. More details are shown in parameter tuning in the chapter of Challenge.

8.2 Diffusion-based Similarity

The data we used to apply diffusion-based similarity is the tag data set in MovieLens 10M Dataset. The major problem when processing the data is the sparse matrix. The total number of samples in the tag data set is 95,580. However tags data set has about 4,000 unique users, and there are many repeated user-movie relationship. Therefore, in our experiment, we get a sparse matrix when processing the user-movie bipartite, which means it doesn't contain any information. Here, I made some modifications of the application on MovieLens 10M Dataset. Since the major goal of our experiment is to check whether we can get benefit by making use of tag information, we combine the user-tag similarity matrix with the rating similarity matrix which calculated by the cosine similarity score of each user.

$$s^{\star} = \lambda s_{ratings} + (1 - \lambda) s^{'}_{tag}$$

In this section, our evaluation metric is also rmse. In the below graph, as the λ goes smaller, the rmse score also decreases correspondingly, which means that assigning more weights on the tag similarity do have some improvements on the performance. We can also conclude that tag data is a meaningful reference to the recommendation predictions.

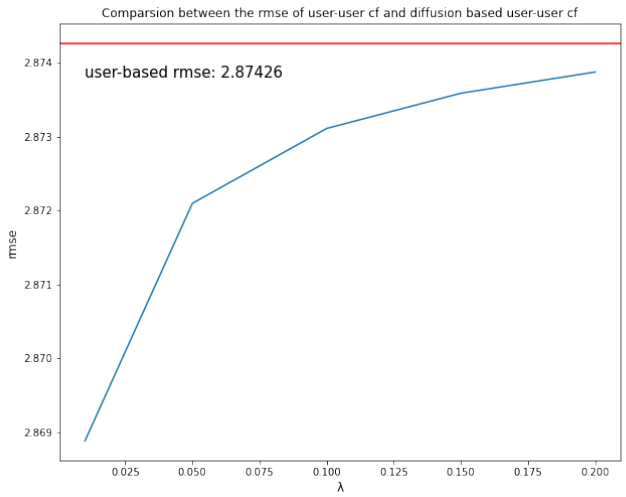

Figure 5: Comparsion between the rmse of user-user cf and diffusion based user-user cf

RMSE Scores	Baseline		Improved version	
Collaborative Filtering	User-based	2.96	User-based with Diffusion	2.87
	Item-item	3.17	Neural collaborative filtering	0.25
Random Walk	Item similarity transition	1.50	Mixed transition	1.25

Table 1: RMSE Value for different Methods

8.3 Neural Collaborative Filtering

Below is the overall NMSE loss of 8 epochs of the training.

$$NMSE = \frac{\Sigma(y_{predict} - y_{true})^2}{y_{true}^2}$$

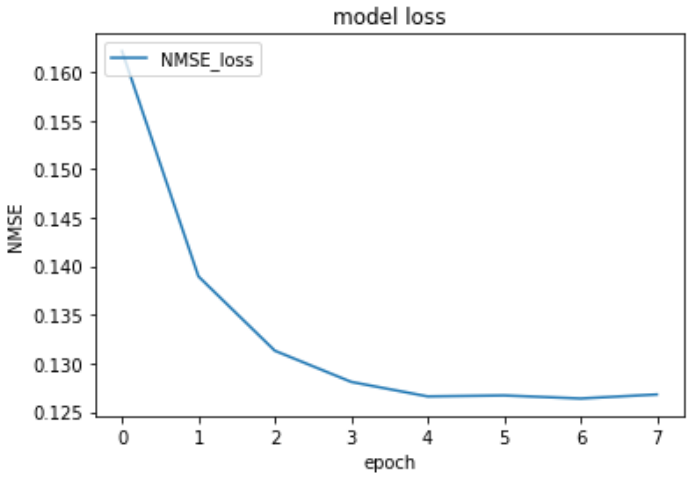

Figure 6: The overall framework of nueral collaborative filtering [3]

We can see that the NMSE value drops from 0.16 to around 0.125 and then converges. Compared with the NMSE value of traditional Matrix Factorization, neural collaborative filtering explored the nonlinear of the latent user vector and the latent movie vector and have a better performance.

8.4 Comparation

Finally, we can compare RMSE scores of all kinds of our models in table 1. We can see that Neural collaborative filtering outperforms other methods. The detailed code can refer to our github(https://github.com/SI608-final-project/SI608-project).

9 Conclusion

We further explore the possibility of model-based collaborative filtering methods.

With the assumption of the Markov process, we found that the employment of random walk has good performance if we model the transition matrix properly. To approximate the transition matrix, we make use of the item-similarity matrix, genre-similarity matrix, and edge weight of the projection of the

Bipartite graph constructed from user-item relation. After selecting hyper-parameter on the valid set, we claim that the performance for the random walk improves compared with only using the item-similarity matrix.

By applying diffusion-based method, we find out that combining the similarity matrix with external relationship between users and objects instead of just calculating cosine or jaccard similarity score can improve the performance of the collaborative filtering model. Although this method encounters the sparse matrix problem, we think random walk can solve this problem in the further research.

The use of neural networks and even deep learning for collaborative filtering should be a trend in the future. We substitute MLP for the similar matrix to see the performance, namely the neural collaborative filtering. By setting different activation function, the model can explore non-linear relationship between the user and item latent vector, therefore achieve a better performance than the naive MF. In the future, through embedding the meta data and increase the hidden layers, the learning ability of the model can be further strengthened.

References

[1] Movielens, Mar 2021. URL `https://grouplens.org/datasets/movielens/`.

[2] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[3] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[4] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*, pages 689–698, 2018.

[5] Yanru Qu, Ting Bai, Weinan Zhang, Jianyun Nie, and Jian Tang. An end-to-end neighborhood-based interaction model for knowledge-enhanced recommendation. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*, DLP-KDD '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367837. doi: 10.1145/3326937.3341257. URL `https://doi.org/10.1145/3326937.3341257`.

[6] 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*, pages 111–112, 2015.

[7] Ming-Sheng Shang, Zi-Ke Zhang, Tao Zhou, and Yi-Cheng Zhang. Collaborative filtering with diffusion-based similarity on tripartite graphs. *Physica A: Statistical Mechanics and its Applications*,

389(6):1259–1264, 2010. ISSN 0378-4371. doi: https://doi.org/10.1016/j.physa.2009.11.041. URL https://www.sciencedirect.com/science/article/pii/S0378437109009753.

[8] Chuan Shi, Zhiqiang Zhang, Ping Luo, Philip S Yu, Yading Yue, and Bin Wu. Semantic path based personalized recommendation on weighted heterogeneous information networks. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 453–462, 2015.

[9] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. Recurrent knowledge graph embedding for effective recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 297–305, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359016. doi: 10.1145/3240323.3240361. URL https://doi.org/10.1145/3240323.3240361.

[10] Vojtěch Vančura and Pavel Kordík. Deep variational autoencoder with shallow parallel path for top-n recommendation (vasp), 2021.

[11] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. Multi-task feature learning for knowledge graph enhanced recommendation. In *The World Wide Web Conference*, pages 2000–2010, 2019.

[12] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 950–958, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330989. URL https://doi.org/10.1145/3292500.3330989.

[13] Hilmi Yildirim and Mukkai S. Krishnamoorthy. A random walk method for alleviating the sparsity problem in collaborative filtering. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, page 131–138, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580937. doi: 10.1145/1454008.1454031. URL https://doi.org/10.1145/1454008.1454031.

[14] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362, 2016.