**Review-enhanced Heterogeneous Graph Neural Networks for Item Recommendation**

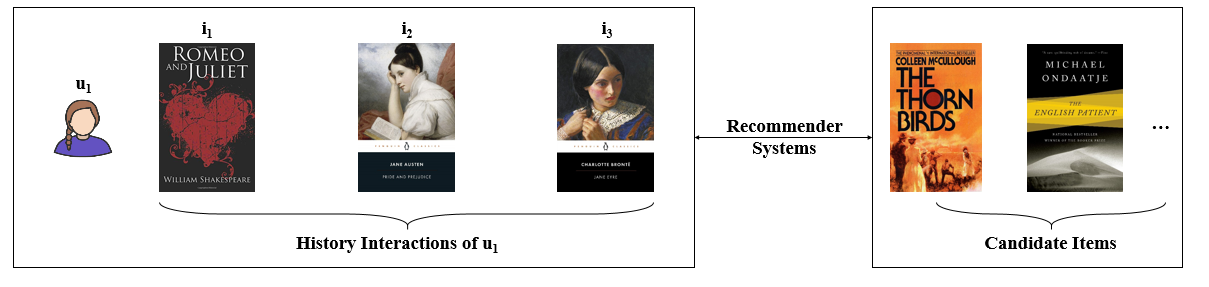
**Abstract**

With the rapid development of the internet and its related technologies, the information system has become a major component of our daily lives. Deep learning approaches to recommendation, especially Graph Neural Networks, have become one of the most popular methods employed in information systems. However, current GNN models are mostly homogeneous, exploiting only the relationships between users and items, and oftentimes model interactions using only rating information high in data sparsity. Therefore, we propose a review-enhanced heterogeneous graph neural framework to jointly model ratings and reviews. We have conducted multiple experiments to demonstrate the effectiveness of our model in terms of prediction accuracy when compared with the baseline models.

**1. Introduction**

With the information explosion in modern society, recommender systems have been widely used in shopping platforms and e-commerce websites (e.g., Taobao and Amazon). Recommender systems generally model users' personalized interest/preference based on the interactions (e.g., clicks, ratings, reviews, etc) between users and items, and recommend candidate items that users have not interacted with and may be interested in to users.

For example, a user u1 has rated several books on the Amazon website (i.e., i1, i2, i3), most of which are romantic novels, as shown in Figure 1. The recommender systems model historical interactions of user u1 to capture his preference and infer that the user may be interested in romantic topic. Consequently, some candidate items concerning romantic topic will be recommended to the user, considering his preference.



**Figure 1. Recommender systems**

Traditional recommendation methods, e.g., Matrix Factorization (MF) (Koren et al., 2009), BPR-MF (Rendle et al., 2009), and FISM (Kabbur et al., 2013), mainly design latent factor models to characterize users and items. MF maps user preferences and item characteristics into the same latent space to generate factor vectors inferred from the user-item interactions.

With the rise of deep learning in recent years, many recommendation methods, such as DeepMF (Xue et al., 2017), NCF (He et al., 2017) , and ONCF (He et al., 2018), have incorporated neural network structures into traditional models and achieved remarkable results in the field of recommendation. DeepMF adopts two neural networks to learn users’ and items’ latent representations, respectively. NCF models user–item interaction by multi-layer perceptron to capture non-linearities between users and items.

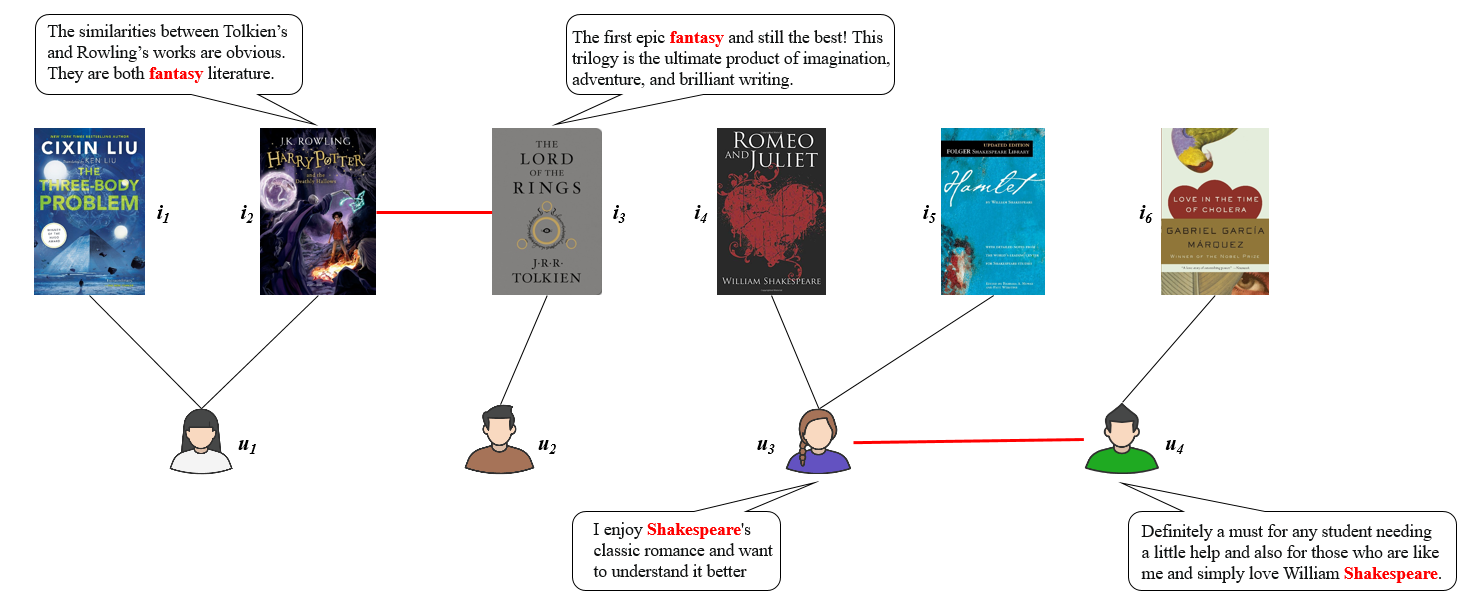
However, these DL-based methods treat each user-item interaction as an independent instance, which leads to ignoring the relations among instances.

Nowadays, Graph Neural Network (GNN) models transform user–item interactions into a user-item bipartite graph and explore high-order connectivity between users and items. GNN generates node (user or item) representation by aggregating the representations of neighbor nodes iteratively.

LightGCN (He et al., 2020), a popular paradigm of GNN, reveals that neighbor aggregation is the most essential part of GNN. Such neighbor aggregation techniques make it possible for GNN models to effectively learn high-order representations for both users and items by considering relations between different instances.

Though GNN has yielded excellent results in the field of item recommendation, they still have some limitations:

1. Existing recommendation models only utilize user-item rating interactions. While these interactions are relatively sparse in real-world scenarios, modeling only rating data cannot obtain a good recommendation model. Therefore, it is necessary to introduce additional information to alleviate data sparsity. When a user has rated an item, this user often generates corresponding reviews. Reviews contain rich semantic knowledge and we can better model user preferences and item characteristics by mining semantic information.
2. Additionally, traditional GNNs are homogeneous and henceforth exploits rating data, focusing only on user-item interactions, but not usually considering user-user and item-item interactions. Therefore, a homogeneous graph might not be adequate enough to learn users’ and items’ representations.



**Figure 2. User and item interactions**

In this paper, we aim to address these limitations and propose a heterogeneous graph neural network to jointly model ratings and reviews. In addition to the user-item relation extracted from ratings, we mine semantic information from reviews and transform such information into user-user or item-item relations. This enables us to not only generate rich semantic relations to relieve data sparsity, but also to construct a heterogeneous graph to retain user-item, user-user, and item-item relations, in which high-order node representations of users and items could be learned by graph neural network. For example, by extracting sematic information from the reviews of two users (i.e., u3 and u4), as shown in Figure 2, we can infer that both users display a similar interest in reading Shakespearean literature. We connect the two users and establish the user-user relation in the heterogeneous graph. Consequently, this relation identifies the items i4 and i5 (both Shakespearean literature and interacted by u3) as the two-hop neighbors of u4. This neighbor relations allow our model to conveniently recommend both i4 and i5 to u4, a result not as easily achieved in the state-of-the-art GNN methods, in which of the relation between u3 and u4 is ignored. Similarly, we establish the item-item relation (i.e., i2 and i3) and recommend the items that u1 has interacted with to u2.

Specifically, our model is divided into three main parts.

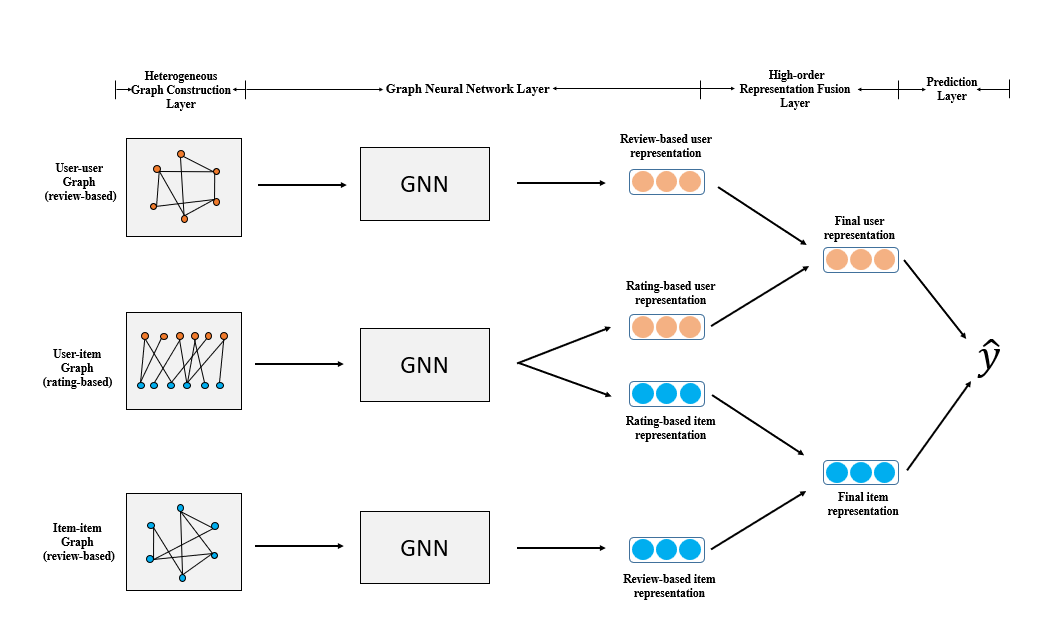
(1) We first model the user-item rating interaction to establish the user-item relation, and then employ a BERT model (Devlin et al., 2019) to extract the user-user and item-item relations from reviews. The user-user and item-item relations, together with user-item relation, construct a heterogeneous graph.

(2) We introduce GNNs to learn the high-order representations of users and items along with the heterogeneous graph. Considering the difference between ratings and reviews, we separately model rating-based representations and review-based representations to extend the performance of our model.

(3) Finally, rating-based representations are fused with review-based representations to form the high-order representations (users or items). The users’ and items’ representations are then employed to generate the final predictions using matrix factorization.

**2. Proposed Model**

In this section, we propose a heterogeneous graph neural network framework, which uses GNN to learn the high-order representations of users and items from rating and review information. Our framework contains four main components: the heterogeneous graph construction layer, the graph neural network layer, the high-order representation fusion layer, and the prediction layer, as shown in Figure 3.



**Figure 3. Overall model structure**

**2.1 Definition**

Our dataset consists of users, items, and their interactions in terms of both ratings and reviews. We denote the set of all users as with a total number of and the set of all items as with a total number of . Each interaction record between any user and item can be defined as , in which denotes the user had rated item and denotes the user ’s review about item . is the set of observed interactions that all users have rated all items.

**2.2 Heterogeneous Graph Construction Layer**

**2.1.1 User-item Rating Relation**

We first construct the user-item interaction matrix based on user-item ratings. Each matrix entry in matrix is 1 if there exists a rating interaction between user and item and otherwise 0. We define user-item matrix as follows:

,

where .

**2.1.2 User-user Review Relation**

For each user , we concat all of his reviews (, ) to generate a user document . A pretrained BERT model then encodes the user document to learn a semantic representation for each user based on all of his reviews:

= .

To establish the user-user semantic relation, we introduce cosine similarity to compute the similarity between two user representations. For two users and , the similarity of these semantic representations could be calculated using the following formula:

).

By calculating the similarity between all users , we then construct the user-user interaction matrix . We establish a link relation between two users and if is greater than a certain value . Otherwise, two users and do not have exist a link relation. Therefore, we define user-user review matrix as follows:

,

where .

**2.1.3 Item-item Review Relation**

We construct the item-item interaction matrix in a similar way as we did for , only that when assembling the item document , we collected the reviews directed towards a particular item instead of the reviews made by a user Similarity, we define item- item review matrix as follows:

,

where .

**2.1.4 Rating and Review Graph Construction**

We incorporate , , and to construct our rating and review graphs as follows:

= ,

= .

where and are adjacency matrices.

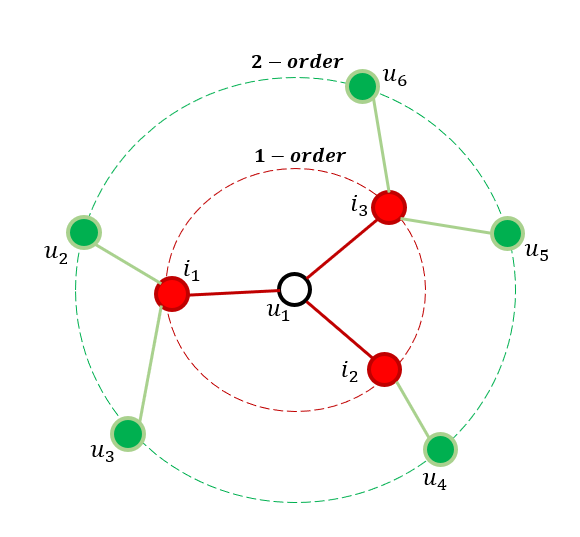
It is worth noting that we have decidedly constructed two independent adjacency matrices, and , to represent our interaction graph, with being assembled into , and and being assembled into , instead of being incorporated together into a single adjacency matrix. This is done out of the consideration that only features interactions based on ratings, and that and only feature interactions based on reviews. We surmise that ratings and reviews, differing so drastically from each other in terms of both data structure and data specificity, might not yield satisfying results when directly processed together in a single adjacency matrix. Hence, we have separated them into and in our design.

**2.1.5 Task definition**

The task is defined as follows: given a rating-based graph and a review-based semantic graph , we aim at learning a prediction function ƒ = (𝑢, 𝑣; **𝛩**, , ) that can accurately predict the possibility of user click item and generate a top-N candidate item rank list.

**2.3 Graph Neural Network Layer**

**2.3.1 Graph Neural Network**

****

**Figure 4. Structure of GNN models**

We employ GNNs to learn rating-based and review-based representations for users and items. The GNNs employ the neighborhood aggregation mechanism to learn high-order representations for users or items by aggregating the representations of its neighboring nodes.

In our method, we first define each user as an initial embedding and each item with an initial embedding .

For example, user has an initial embedding, . In the first GNN layer, will aggregate information from the representations of all its one-hop neighbors (i.e., , , and , as shown in Figure 4), generating the new 1-order embedding . Similarly, in the second GNN layer, will furthermore aggregate information from the representations of both its one-hop and two-hop neighbors (i.e., , , …, , as shown in Figure 4), yielding the 2-order embedding . The process is then repeated across all GNN layers for all users and items until the final high-order representations are yielded. This is summed up in the following formulae for the classical user-item GCN graph:

),

),

where and each denotes the k-order representation for user and item , denotes the nonlinear activation function, and each denotes the set of items interacted by user and the set of users interacted by item , is a normalization coefficient, and and W denote the weight parameters.

**2.3.2 Rating-based Representation Learning**

These embeddings are then concatenated to form the 0-layer rating-based embedding matrix:

*.*

We then implement an iterative propagation of on the rating-based adjacency matrix across layers of GNN. The propagation process is directly inherited from the LightGCN model (He et al., 2020), which allows for a simple linear method for neighborhood aggregation and could be expressed as:

, ,

The matrix form of LightGCN can be described as follows:

,

where denotes a diagonal matrix, denotes the current number of GNN layers already propagated, and denotes the - layer rating-based embedding matrix. Every layer of propagation allows each embedding, or , to aggregate information from the embeddings of their neighboring nodes as shown in .

When the propagation is finished after layers, we implement a summation of the embedding matrices from every layer to form the final rating-based embedding matrix, :

.

**2.3.3 Review-based Representation Learning**

The review-based embedding matrix, , is learned in a like manner, only using a different initial embedding for update and the review-based adjacency matrix for propagation. The process is summed up as below:

*,*

,

.

After completing the procedures above, we have now generated rating-based and review-based representations ( or ) for every user and item .

**2.4 High-order Representation Fusion Layer**

After yielding the rating-based and review-based representations, we concatenate and to form a final high-order representation for every user and item :

,

where is the fusion function. Finally, we generate the fusion representation of user and the fusion representation of item We choose addition operation to conduct the fusion:

where is the ratio value.

**2.4 Prediction Layer**

After obtaining the fusion representation and , we adopt matrix factorization to generate the final prediction:

,

where is the prediction function.

After generating the probability value , we choose cross-entropy as our loss function which is defined as follows:

where is the set of observed interactions and is the set of negative samples that can be randomly sampled from unobserved interaction to prevent over-fitting. is the ground-truth.

**3 Experiments**

**3.1 Experimental Settings**

**3.1.1 Dataset**

To verify our model, we conduct experiments on a publicly available dataset, the Amazon dataset (Amazon Review Data, n.d.), which includes rich rating and review data and is widely used in recommender systems. We select the Amazon Instant Video to carry out the detailed experiments.

**3.1.2 Evaluation Metric**

To evaluate the performance of our model, we use Hit Ratio (HR), Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG) as evaluation metrics to measure recommendation accuracy.

**3.1.3 Comparison Methods**

To demonstrate the effectiveness, we compare our model with the following baselines which include: MF (Koren et al., 2009), NCF (He et al., 2017), GC-MC (van et al., 2017), GCN (Kipf & Welling, 2016), and LightGCN (He et al., 2020). Table 1 compares our model to the baselines across different factors.

**Table 1. Compatison between models**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method | MLP | GNN | Ratings | Reviews |
| MF |  |  | ✓ |  |
| NCF | ✓ |  | ✓ |  |
| GNN-based |  | ✓ | ✓ |  |
| **OURS** |  | ✓ | ✓ | ✓ |

MF and NCF employ traditional and deep learning-based design to explore the user-item interactions for recommendation, respectively. MF represents users and items with factors learned from historical interactions, in which high correspondences between representations lead to possible recommendations. NCF, on the other hand, adds nonlinearities to the interaction between user and item factors by leveraging a multi-layer perceptron to replace the traditional matrix factorization.

Furthermore, we also compare our method with the state-of-the-art GNN models, such as Graph Convolutional Matrix Completion (GC-MC), graph convolutional networks (GCN), and LightGCN. GC-MC passes information of neighboring nodes on an interaction graph to yield one-order representations for users and items, which are then used for predicting further rating links. Correspondingly, GCN extends on the GC-MC model by conducting layer-wise propagations on the interaction graph, learning more complex, high-order representations for users and items. Similarly, LightGCN effectively simplifies GCN by only including neighborhood aggregation, the most essential component of the model, in its method, therefore yielding significant improvements in prediction accuracy.

**3.1.4 Parameter Settings**

We divide our data into the training dataset and the testing dataset by an 80% to 20% ratio. Specifically, we sample 4 negative samples for each observed interaction to generate the training data, while the testing data is obtained by sampling 99 negative samples for each observed interaction. All models are implemented under the TensorFlow framework and trained using the Adam Optimizer. We set the batch size to 128 and the learning rate to 0.005.

We select the ReLU function as the activation function for the NCF model and the Leaky ReLU function for the GC-MC and GCN model. The number of GNN layers is set to 3 for the GCN and LightGCN model as well as our proposed model. The size of the GNN embeddings for all three layers is 64. In addition, for the NCF model, there is only one MLP layer with an embedding size of 32. The dropout rate for all the experiments is set to 0 by default. Finally, the ratio of the review embedding against the rating embedding in the fusion function is set to 0.10.

**3.2 Performance Comparison**

**Table 2. Overall Performance Comparison**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Model** | **HR@10** | **MRR@10** | **NDCG@10** |
| 1 | MF | 0.1971 | 0.0544 | 0.0852 |
| 2 | NCF | 0.3884 | 0.1692 | 0.2205 |
| 3 | GC-MC | 0.4040 | 0.1525 | 0.2106 |
| 4 | GCN | 0.4387 | 0.1700 | 0.2327 |
| 5 | LightGCN | 0.3975 | 0.1838 | 0.2341 |
| 6 | OURS | **0.4896** | **0.2216** | **0.2847** |

Table 2 reports the overall performance of our proposed model compared with the baseline models in terms of HR@10, MRR@10, and NDCG@10. The best result for each evaluation metric is highlighted in bold, as shown in Table 2, while the second-best result is underlined.

By comparing the performance of GNN models with traditional, non-graph models, we see an overall improvement in results in the models in which GNN is employed. The average HR of traditional models (i.e., MF and NCF) is 0.2928, while the average HR of GNN models (GC-MC, GCN, LightGCN, and our proposed model) is as high as 0.4325. This significant improvement of 32.30% in prediction accuracy possibly owes to the fact that when compared to traditional methods, GNN models are able to consider the relations between different user-item instances by learning high-order representations for users and items, leading to a more global representation of historical interactions.

Furthermore, when evaluating the results of our proposed model against the baseline models, we also see a clear improvement. While the average HR for baseline models (i.e., MF, NCF, GC-MC, GCN, and LightGCN) is 0.3651, the average HR for our proposed model is as high as 0.4896, with a significant improvement of 24.42%. By mining semantic information from reviews and transforming such information into user-user or item-item relations, our model is able to reduce the data sparsity of baseline methods, which only model user ratings, and therefore leading to a drastic improvement in prediction accuracy.

**3.3.1 Effect of layer numbers**

**Table 3 Comparison Between Different Number of GNN Layers**

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer** | **HR@10** | **MRR@10** | **NDCG@10** |
| 1 | 0.2687 | 0.1058 | 0.1437 |
| 2 | 0.4274 | 0.1930 | 0.2479 |
| 3 | 0.4896 | 0.2216 | 0.2847 |
| 4 | 0.5017 | 0.2368 | 0.2991 |
| **5** | **0.5428** | **0.2658** | **0.3313** |

Table 3 reports the performance result of our model with different numbers of GNN layers. Down the row, as the number of layers increases, the prediction accuracy of our model likewise sees a gradual improvement. This clearly demonstrates the GNN’s ability to explore high-order connectivity within a graph. When the number of layers is set to 1 or 2, our model has not yet gathered enough information from its one-hop or two-hop neighbors. But with the increase of GNN layers, our model is able to explore higher-order relationships between users and items, so that at a number of 5 GNN layers, as highlighted in bold in Table 3, the model already outperforms our default model, set at a number of 3 GNN layers.

**3.3.2 Effect of ratio .**

**Table 4 Comparison Between Different Values of**

|  |  |  |  |
| --- | --- | --- | --- |
| **ratio** | **HR@10** | **MRR@10** | **NDCG@10** |
| 0.01 | 0.4092 | 0.1927 | 0.2435 |
| 0.05 | 0.4289 | 0.1968 | 0.2525 |
| 0.1 | 0.4896 | 0.2216 | 0.2847 |
| **0.2** | **0.4960** | **0.2272** | **0.2905** |
| 0.25 | 0.4919 | 0.2256 | 0.2882 |
| 0.5 | 0.4697 | 0.2109 | 0.2718 |
| 1 | 0.3933 | 0.1687 | 0.2211 |

To investigate how the ratio of review information and rating information would affect the performance of our model, we conducted our experiment with different values of . The number of GNN layers is set to 3 for all experiments. As shown in Table 4, we found that when we increase the ratio from 0.01 to 0.2, our model will see a significant improvement in its prediction accuracy, specifically achieving optimum performance when is set to 0.2, as highlighted in bold in Table 4. However, a further increase in the value of will decrease prediction accuracy.

**4. Conclusion**

In this paper, we have proposed a review-enhanced heterogeneous graph neural framework to jointly model ratings and reviews. By mining rich semantic information from user reviews and constructing them as user-user and item-item interactions, we are able to achieve better prediction results than a number of baseline models, demonstrating the effectiveness of our model in correcting data sparsity. This allows for more accurate recommendations in real-life scenarios in which the dataset procured is high in data sparsity and also provides a method in which the review information could be employed in the recommendation process to bring about more satisfying and personalized services for customers. In our future work, we hope to incorporate into our model more varied forms of information, such as photos, descriptions, clicks, likes, etc., so as to further explore the GNN’s potentials when placed in a more complicated recommendation situation.

**5. References**

Koren, Y., Bell, R. M., & Volinsky, C. (2009). Matrix factorization techniques for recommender Systems. IEEE Computer, 42(8), 30–37. https://doi.org/10.1109/mc.2009.263

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2009). BPR: Bayesian personalized ranking from implicit feedback. Uncertainty in Artificial Intelligence, 452–461. http://ants.iis.sinica.edu.tw/3BkMJ9lTeWXTSrrvNoKNFDxRm3zFwRR/80/Rendle\_et\_al2009-Bayesian\_Personalized\_Ranking.pdf

Kabbur, S., Ning, X., & Karypis, G. (2013). FISM. Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Xue, H., Dai, X., Zhang, J., Huang, S., & Chen, J. (2017). Deep Matrix

Factorization Models for Recommender Systems. International Joint Conference on Artificial Intelligence.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T.-S. (2017). Neural Collaborative Filtering. <https://doi.org/10.48550/arxiv.1708.05031>

He, X., Du, X., Wang, X., Tian, F., Tang, J., & Chua, T.-S. (2018). Outer Product-based Neural Collaborative Filtering. https://doi.org/10.48550/arxiv.1808.03912

‌He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. (2020). LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. ArXiv (Cornell University). https://doi.org/10.48550/arxiv.2002.02126

‌Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ArXiv, abs/1810.04805.

‌ Amazon review data. (n.d.). Jmcauley.ucsd.edu. http://jmcauley.ucsd.edu/data/amazon/links.html

‌van den Berg, R., Kipf, T., & Welling, M. (2017). Graph Convolutional Matrix Completion.

‌Kipf, T., & Welling, M. (2016). Semi-Supervised Classification with Graph Convolutional Networks.

‌