Graph-based Drug Recommendation: A Comprehensive Analysis from Representation Embedding to Rating Prediction

Heyuan Huang
Health Informatics
Yale University
New Haven, CT 06511
heyuan.huang@yale.edu

1 Introduction

Most objects involved in recommendation systems are intrinsically graph-structured data. Recommendation systems can be viewed as link prediction tasks between users and products. This project is to explore the implications of different neighbor aggregation methods and attention mechanisms on different heterogeneous graphs' link prediction tasks, with explainability analysis, which is important in medical applications.

We define Amazon Review and Drug Review datasets as heterogeneous bipartite (user-item) graphs. Rather than using off-the-shelf GNNs, we customize modular GNNs with encoder and decoder to predict the rating from user to item. We compare tf-idf and GloVe for text review embedding initialization methods and use CaptumExplainer for heterogeneous node saliency visualization. Our results show that after the same training epochs, GraphSAGE and GAT reach similar performance on the two graphs according to the RMSE metric; GloVe can accelerate training convergence by better initialization than tf-idf; drugs in the Drug dataset play a more crucial role in influencing prediction than products in the Amazon dataset.

2 Related Work

The previous work on recommendation systems mostly uses collaborative filtering, content-based filtering, and hybrid filtering[2], which can not capture graph structure information very well. However, most of the data in recommendation systems have intrinsic graph structures. Graph learning models can better utilize node relationships to automatically generate embedding representations for each node, hence benefit rating prediction and ranking for recommendations.

For rating prediction, which is also regarded as link label prediction or user-item rating matrix completion, Shi et al.[3] first applied a meta-path based random walk method to create node sequences for Heterogeneous Information Network (HIN) node embedding generation. Then they fused the node embeddings by several fusion functions to integrate all information in meta-paths to generate final user and item node embeddings. However, their HIN-based method is dependent on the path similarity, which can not represent all the structural information in the user-item graphs.

PinSage[4] used node visual and text content as initial node features for generating node embeddings, which improves the generalizability of previous work by its inductive feature. However, it needs to rely on the content information, which is not always available or accessible for all kinds of graphs. Zhang et al. [5] extracted 1-hop subgraphs around user-item pairs and mapped these subgraphs to pairs' ratings for GNN training, achieving comparable performance of state-of-the-art methods without using any side information. Instead of generating node-level embeddings, their method is distinguished by generating graph-level representations for subgraphs.

Preprint. Under review.

3 Method

3.1 Graph Construction

We construct the Amazon and Drug datasets as heterogeneous bipartite graphs with (user/condition-item/drug) pairs. We initialize the Amazon user node features as an identity matrix as each user ID doesn't contain too much feature information. The Amazon item and Drug nodes are initialized by their text review features with TF-IDF and GloVe representations. If one item has multiple reviews, we use the mean value of their review embeddings. Also, the condition nodes are initialized by their disease information written in text.

3.2 Graph Sampling

As the 2 datasets are too large to fit in memory size at once, we use the LinkNeighborLoader to load graphs for mini-batch training. It first samples predefined batch size edges from the input edges and then samples a predefined number of neighbors for the nodes in the sampled edges. With the sampled link and sampled neighbors, it can generate subgraphs with neighborhood information for mini-batch training.

3.3 GNN Layers

Similar to the Graphormer structure, we define an encoder to process multi-hop graph structure information and a decoder to give the final scalar prediction. For the encoder, we tried different message computation and aggregation methods proposed by GraphSAGE and GAT. Encoder1 consists of 2 SAGEConv layers with a ReLU activation function between the 2 layers and the aggregation method is mean aggregation. Encoder2 consists of 2 GATConv layers and 2 linear layers to add residual connections for preserving self-information. The activation function between layers is also ReLU for comparable consistency.

For the decoder, we constructed 2 linear layers with a ReLU activation function to gradually condense the final information.

3.4 Objective Function

We choose the most commonly used objective function in rating prediction papers, Mean Squared Error (MSE) between the predicted ratings and actual ratings:

$$\mathcal{L}_{\mathcal{MSE}} = \frac{1}{|\{(u,i)|\Omega_{u,i}=1\}|} \sum_{(u,i):\Omega_{u,i}=1} (R_{u,i} - \hat{R}_{u,i})^2, \tag{1}$$

where $R_{u,i}, \hat{R}_{u,i}$ are the labeled rating and predicted rating of user-item pair (u, i), and Ω is the mask matrix indicating the training entries of the input graph.

3.5 Evaluation Metrics

Similarly, we consider the most commonly used evaluation metric, Root Mean Square Error (RMSE).

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(r_i - \hat{r}_i)^2} \tag{2}$$

where $r_i, \hat{r}_i$ are the actual rating and predicted rating for a user-item pair, and N is the number of user-item links in the test set.

3.6 Explainability Analysis

After training the models, we used CaptumExplainer for heterogeneous data explainability saliency analysis on the model and phenomenon level as heterogeneous data are not yet supported by GNNExplainer.

3.7 Difference and Contribution

For learning purposes, instead of using off-the-shelf GNNs, we build customized modular GNNs referring to Colab3, Colab4, and PyG tutorials and give comparisons of feature initialization methods, hyper-parameters, and GNN layers.

Compared with prior work done on the Drug Review dataset, we are the first one trying to use graph structure information and GNN on the rating prediction task. We defined our method to change this classic dataset for non-graph method tasks into a graph.

Explainability is sometimes neglected by prior work. We add an explainability analysis section to find what drug or condition influences the model behavior most as it is important for medical applications.

4 Experiments

4.1 Dataset

We use the Amazon Fashion category dataset in Amazon Review Data by Ni et al.[1]. Each row contains one user ID, product ID, overall rating of the product, text of the review, etc. The *training objective* on the Amazon dataset is to minimize the difference between predicted ratings and actual ratings. To save computational time for the following drug dataset, here we use the 5-core dataset for the Amazon Fashion category in experiments, which is the largest connected component of the sparse Amazon Fashion graph. Each user or item has at least 5 reviews, making it a dense subgraph. Our code can run on both datasets by simply changing the file name as AMAZON_FASHION.json and AMAZON_FASHION_5.json are in the same format.

Besides the commercial data, we convert the Drug Review Datase from Drugs.com to a graph-based rating prediction task and further find some profound insights in drug recommendations, which hasn't been studied in this dataset before. This dataset contains 215063 rows with columns of patient condition (i.e., diagnosed disease), purchased drug name, patient rating (0 to 10), text review, date and useful votes. Our *goal* is to construct the condition-drug bipartite graph to predict the rating of a drug, given a disease (condition).

Table 1 is the basic statistics of the constructed bipartite graphs. Set A corresponds to user/condition and B is product/drug. The Drugs dataset is sparser than the Amazon 5-core dataset, according to their average degrees.

Table 1: Statistics of bipartite graphs

Dataset	Number of A	Number of B	Ave. degrees of A	Ave. degrees of B	Ave. Rating
Amazon Fashion	749,233	186,189	1.168	4.700	$3.91 \in [1, 5]$
Amazon Fashion 5-core	406	31	7.49	98.13	$4.40 \in [1, 5]$
Drugs.com	916	3,671	10.62	2.65	$6.99 \in [1, 10]$

Figure 1 is the subgraph of the Amazon Fashion dataset, where the green node is the most connected product, the red nodes are users linked to this product, and the blue nodes are other products linked to these users.

Figure 2 is the subgraph of Drugs.com dataset, in which we select the Depression condition and plot all linked drugs.

4.2 Training Details

4.2.1 Data Preprocessing

We drop all rows with missing values and there are duplicated rows to drop.

In the Drug datasets, every user (row) only has one purchase entry, so we use Condition to replace User for a denser graph. There are multiple links between one condition-drug pair, so we construct two kinds of graphs to see if they have different impacts on the model performance. The first

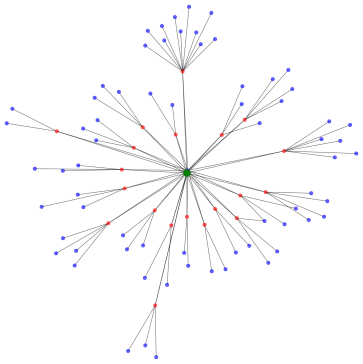

Figure 1: Amazon product-user-product subgraph

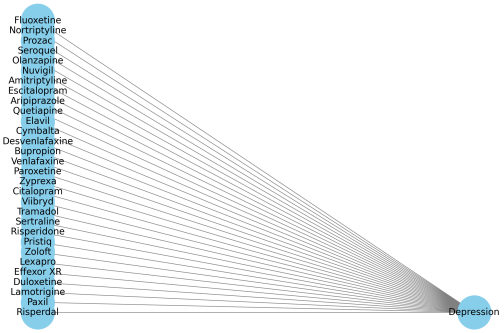

Figure 2: Depression-related drug subgraph

graph is the multi-link graph and in the second one, we use the mean rating of the multiple links of one pair as its single link label and remove other redundant links. The final heterogeneous graph objects are undirected and have 2 node types (user, item) and 2 edge types (user_rates_item and item_reverse_user). But only (user_rates_item) edges have labels with ratings.

4.2.2 Embedding Initialization

The node embedding initialization is done by TF-IDF and GloVE to extract lexicon and semantic features from the text in each node (condition name, drug, and product reviews). The input feature shape is (number of nodes, 300), where 300 is the predefined feature dimension. The user node embedding is a simple identity matrix as the user ID doesn't contain too much information.

4.2.3 Experiment Configuration

We use RandomLinkSplit to split links into train, validation, and test sets. The configuration parameters are num_test=0.1 for the test set and num_val=0.05 for the validation set. After the splitting, we use LinkNeighborLoader with num_neighbors=30 for each edge type and batch_size=512 to sample subgraphs for mini-batching the train, validation, and test sets.

The hidden_channels and out_channels for the 2 layer SAGEConv and GATConv are both 32 in the Encoder part, and they are followed by the same Decoder for comparison consistency. The learning rate is 0.01 and the optimizer is Adam. Epoch is 20.

4.3 Results

We run each model for 5 times and calculate the mean RMSE on the test set and standard deviation in the table 2. Drug(multi) means the multi-link between one (condition-drug) pair graph and Drug(single) is the mean rating single-link graph. Their node embedding initialization is done by GloVe.

Table 2: RMSE of SAGE and GAT model with GloVe

Dataset	SAGEModel RMSE (std)	GATModel RMSE (std)
Amazon Fashion 5-core	1.033 (0.05)	1.031 (0.05)
Drug(multi)	3.310 (0.008)	3.290 (0.003)
Drug(single)	2.676 (0.057)	2.809 (0.046)

Our models' performance is a little weaker (high RMSE) and less stable (high standard deviation) than state-of-the-art models. Possible reasons include that our model only uses the node-level embeddings to do the link prediction, but advanced models like IGMC[5] extract subgraph-level which contains more structure and neighborhood information for the link prediction task.

4.4 Analysis of the Results

4.4.1 Findings

For the node embedding initialization, we find GloVe and TF-IDF don't influence the model performance a lot if our model is trained after enough epochs. Figure 3 shows the Test RMSE curve on the Amazon dataset with SAGEModel, initialized by TF-IDF and GloVe. GloVe initialization can initialize embeddings better at the start as it contains more semantic and other information, rather than only lexicon information in the TF-IDF vectors. But after enough training, the embeddings are updated closer to ground truth so the importance of initialization decreases.

Figure 4 is the comparison of the impact on test set RMSE by different optimizer and learning rate. We can see that Adam and NAdam with high learning rate (0.001) helps the model to converge faster, than SGD and small learning rates.

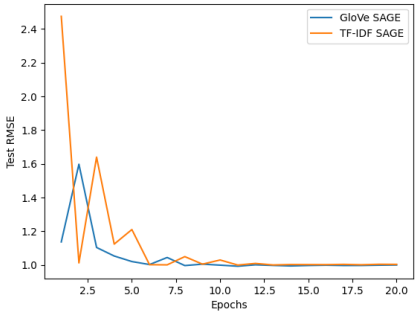

Figure 3: Amazon Test curve: GloVe v.s. TF-IDF

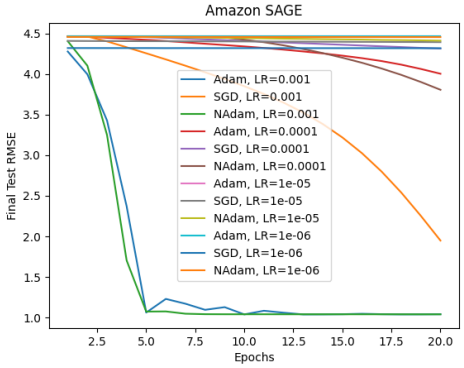

Figure 4: Hyperparameter and Configuration comparison

For the Drug dataset, the model performance on the single-link graph (RMSE=3.35) is better than multi-link graph (RMSE=3.83). This is reasonable because multi-link contains multiple ratings between one (condition-drug) pair, given by different patients. The individual difference will make the model more confused about the actual rating, which can be mitigated by the mean rating in the single-link graph.

4.5 Explainability Analysis

For the Drug dataset, we want to know what condition or drug is the most influential for the prediction result. Therefore, we use the CaptumExplainer's Saliency attribution method to pick the top 20 influential nodes.

Figure 5 is the influential nodes in the multi-link graph. The top 5 influential nodes are BuSpar(drug), Morphine(drug), Methadone(drug), Clostridial Infection(condition), and Hyperprolactinemia(condition).

Figure 6 is the influential nodes in the single-link graph. The top 5 influential nodes are Morphine(drug), NuvaRing(drug), Levora(drug), BuSpar(drug),and Anastrozole(drug).

There are some overlaps in the top 20 influential nodes, hence making them more important, such as BuSpar and Morphine drugs. Clinical professionals can make possible interpretations for the model behavior based on their expert knowledge.

Figure 7 is the influential nodes on the Amazon dataset, where we can see users play a more important role in determining the rating prediction than products. Possible reasons for this difference from the drug dataset include that drug quality and suitability is more important for the drug rating but user opinion and taste is more important for the product rating.

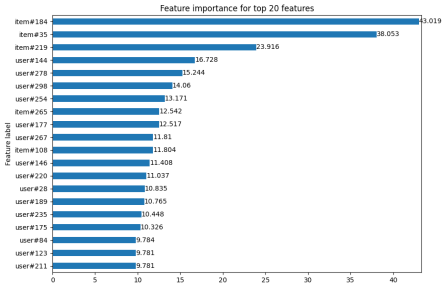

Figure 5: Influencial nodes on multi-link graph

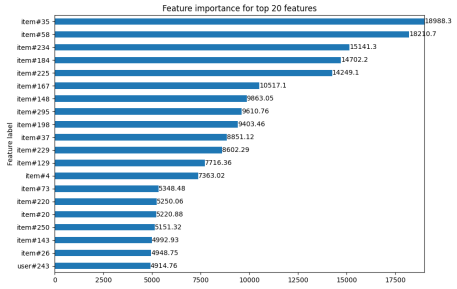

Figure 6: Influencial nodes on single-link graph

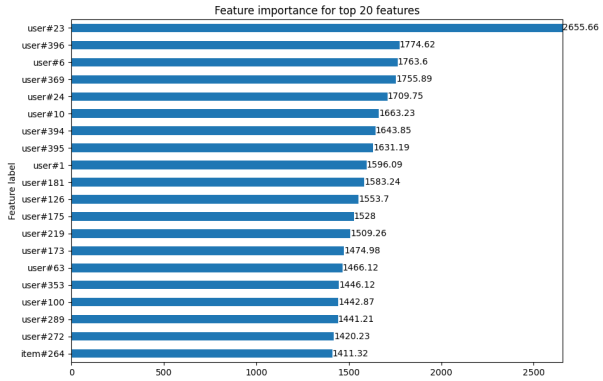

Figure 7: Influencial nodes on the Amazon dataset

5 Conclusion

Our SAGEModel and GATModel can reach similar performance after enough training epochs, with RMSE=1.03 on the Amazon dataset and RMSE=3.35 on the Drug dataset. GloVe can accelerate training convergence by better initialization than TF-IDF but after enough epochs, their influence decreases. Drugs in the Drug dataset play a more crucial role in influencing prediction than products in the Amazon dataset, which can reflect the user's taste or bias for the product and the drug suitability for the disease condition.

6 Reproducibility

The GitHub repository link is `https://github.com/HeYuan919/Graph_drug_rating` and the GitHub ID is HeYuan919 in case the link doesn't work. The code is run on Colab so there is only a jupyter notebook file. It can be directly uploaded to the Colab and Run all at once. There is no environment.yml or a script to run.

References

[1] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1018. URL `https://aclanthology.org/D19-1018`.

[2] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender Systems Handbook*, volume 1-35, pages 1–35. 10 2010. ISBN 978-0-387-85819-7. doi: 10.1007/978-0-387-85820-3_1.

[3] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and Philip S. Yu. Heterogeneous information network embedding for recommendation, 2017.

[4] 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*. ACM, jul 2018. doi: 10.1145/3219819.3219890. URL `https://doi.org/10.1145%2F3219819.3219890`.

[5] Muhan Zhang and Yixin Chen. Inductive matrix completion based on graph neural networks, 2020.