ARS - Amazon Recommendation System using GNNs

Abdul Mannan Kanji
akanji@usc.edu

Aiden Chang
aidencha@usc.edu

Michael Kingsley
mkingsle@usc.edu

Prathamesh Koranne
koranne@usc.edu

Shubh Khandhadia
khandhad@usc.edu

1 Introduction

1.1 Project Importance

In the evolving landscape of machine learning, the challenge of handling unstructured datasets has emerged as a concern. Graph Neural Networks (GNNs) have risen to prominence in this domain, offering innovative solutions where traditional data processing methods falter. This paper explores the application of GNNs in the specific context of recommendation systems—a field where accurately predicting user preferences is both a challenge and a necessity.

At the core of our project is the deployment of GNNs for link regression to predict user ratings for Amazon products. This approach represents a significant stride in the realm of personalized recommendation systems, which have become integral to a wide array of online services, from e-commerce to content streaming. The success of these systems depends on their ability to discern and adapt to the intricate and often subtle patterns of user behavior and preferences.

Unlike traditional methods such as collaborative filtering, GNNs can capture both linear and nonlinear relationships between users and products. GNNs can also extract information even from sparse user-product interactions and unseen data or new users/products by learning from the graph structure. This mitigates the cold-start and sparsity problems that are present in modern solutions such as collaborative filtering.

Our project distinguishes itself by extending beyond the conventional use of homogeneous graphs typically employed in recommendation systems. We introduce a more complex graph structure, incorporating not just the standard user-to-product connections but also establishing product-to-product relationships. This multi-dimensional approach allows for a more nuanced understanding of both user preferences and product interrelations, promising to enhance the predictive accuracy of recommendation systems substantially.

2 Related Work

Recommendation systems have become a crucial component in the digital era, fundamentally shaping the way we interact with content online. By analyzing user data and preferences, these systems suggest products, services, and content that are likely to be of interest to the user.

2.1 Modern Recommendation Systems

Modern recommendation systems are designed to offer personalized suggestions for online products or services. These systems employ a range of techniques including content-based, collaborative filtering-based, knowledge-based, and hybrid approaches to cater to diverse scenarios (Peng, 2022). We examine a variety of related work for recommendation systems below.

One implementation uses K-Nearest Neighbors (K-NN) for recommendation systems and focuses on optimizing the computational efficiency by calculating distances between training and testing users only once and then using these results for all pending recommendations, significantly streamlining the process (Sagdic et al., 2020). This approach has certain constraints, such as its inability to effectively learn the unique representations of each node. This limitation stems from the method's reliance on a singular computation of distance, rather than a more dynamic or iterative process that could capture the nuances of each node's characteristics.

Another implementation of collaborative filtering uses user-based filtering and item-based filtering to predict movies for users (Wu et al., 2018). We used this approach when considering user-to-user and user-to-product edges for our graph.

Another paper enhances collaborative filtering in recommendation systems by integrating social network technology, proposing an improved algorithm and two community detection methods to efficiently identify user communities, thus reducing computational time and enhancing recommendation speed and accuracy (Li & Li, 2019). We want to use a similar method, with an emphasis on aggregating insights from neighboring nodes in our network.

A different paper improves collaborative filtering by employing an item-based algorithm that focuses on relationships between items instead of users, thereby avoiding the bottleneck of searching for neighbors among a large user population, and potentially maintaining recommendation quality due to the static nature of item relationships(Sarwar et al., 2001). We implemented a similar strategy, establishing direct relationships between products to better understand their interconnected dynamics.

Overall, our methods differ from the above given that we are using GNNs to investigate and model user-product relations. Nonetheless, we have integrated some of these techniques and principles into our GNN framework for enhanced performance.

2.2 Graph Neural Networks

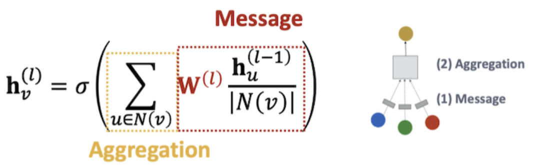
Message
$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u\in N(v)} \mathbf{W}^{(l)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}\right)$
Aggregation
(2) Aggregation
(1) Message

Figure 1: Sample GNN
(Leskovec, 2023b)

A graph neural network can be expressed as a forward pass of messages as shown above in Figure 1, where the messages are then aggregated with different functions. The GNNs we chose to implement after investigation were GraphSage and GAT (Hamilton et al., 2017; Veličković et al., 2017).

Other implementations include LightGCN which simply removes feature transformation and non-linear activation during graph convolution, making it more efficient and straightforward for collaborative filtering tasks(He et al., 2020). We did not implement LightGCN because we wanted a more complex structure with the freedom to experiment with different aggregation functions as well as attention layers.

Similarly, work on Graph Convolutional Networks for recommendation systems includes a study titled "Graph Convolutional Neural Networks for Web-Scale Recommender Systems." The paper presents PinSage, an innovative GCN algorithm developed for Pinterest's recommendation engine. PinSage combines graph convolutions and random walks to generate node embeddings, integrating graph structure and node features for enhanced recommendations (Ying et al., 2018). There are countless different variations on the traditional Graph Convolutional Network, including but not limited to Attention-Based Graph Convolutional

Neural Networks (Feng et al., 2019). Our implementation differs as it does not consider random walks.

In conclusion, our methodology differs from the above as we are employing modified versions (explained in Methodology) of their implementations on recommendation systems. In our project, we employ heterogeneous equivalences of GraphSage as well as GAT, unlike the original implementations which were done using homogeneous graphs.

3 Problem Formulation

In this study, our primary objective is to construct a heterogeneous graph utilizing the Amazon 2018 dataset (Ni et al., 2019). From this graph, we aim to develop a model specifically designed to perform link regression between users and products. Specifically, our goal is to accurately predict a user's rating for a given product. Given the massive size of the dataset and our limited computational resources, we will utilize a subset of the data employing the 5-kcore selection. This approach ensures that only datapoints with five or more connections are included, serving two critical purposes. By focusing on nodes with rich information (those with multiple edges), we expect to develop a more robust and effective model and also to address the challenge of managing a large dataset, allowing us to save significant time and computational resources. Our dataset will comprise of two distinct node types: user nodes and product nodes. It will also include two types of edges; 'Reviews' Edges: These connect user nodes to product nodes, with an attribute indicating the rating provided by the user. 'Also Bought' Edges: These connect product nodes, representing products frequently purchased together. Figure 2 shows a sample visualization of our constructed heterogeneous graph, not accounting for the 5-kcore.

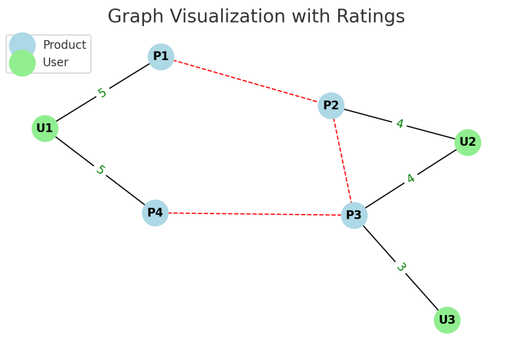

Figure 2: Sample Graph Visualization

Leveraging GNNs within a homogeneous graph, our approach involves an encoder-decoder model to improve our initial node embeddings. These embeddings enable us to execute link prediction post-training, allowing the decoder to accurately infer the ratings a user might assign to a product based on their respective embeddings.

Our training approach begins with a focused dataset comprising approximately 300 data points across 12 categories, specifically targeting gift card purchases. This initial phase is designed to finely tune our model with fast training time. We will increase the complexity of our training by using a more extensive dataset derived from Amazon's Software purchases. This larger dataset encompasses a vast array of 6,956 categories, 1,826 user nodes, and 21,639 product nodes. We divided our data into training, testing, and validation segments, adhering to an 80-10-10 split (Leskovec, 2023a; Fey & Lenssen, 2019). Like many other recommendation systems, we will employ the RMSE (root mean squared error) for our metrics. This choice will provide us with a solid baseline to evaluate and benchmark our recommendation system compared to others in the field.

4 Methodology

4.1 Embeddings

Node embeddings in our model are representations influenced by both the node's inherent attributes and its neighboring nodes. As the encoder-decoder model evolves, these embeddings increasingly capture better representations of itself. Figure 4 illustrates the concept of updating the embeddings for layer i: the embedding of a specific node (node 3, for example) at layer i is a synthesis of its previous layer $(i-1)$ embedding and the aggregated embeddings of its adjacent nodes (nodes 1 and 2).

Figure 3 illustrates the initial embeddings (at layer $i = 0$) for both product and user nodes. For products, embeddings are formed by concatenating the description embedding with the one-hot encoded categories embedding. We employ the widely recognized sentence-transformer/all-MiniLM-L6-v2 model from Hugging Face for generating description embeddings (Reimers & Gurevych, 2019). In contrast, user embeddings are initialized using an identity matrix, reflecting the absence of prior information about the users.

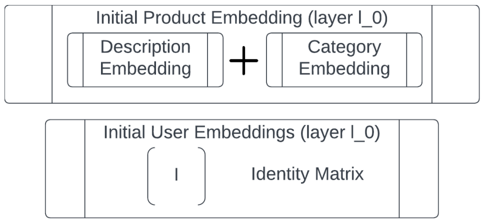
Initial Product Embedding (layer l_0)
Description Embedding
+
Category Embedding
Initial User Embeddings (layer l_0)
I
Identity Matrix

Figure 3: Generating Initial Embeddings

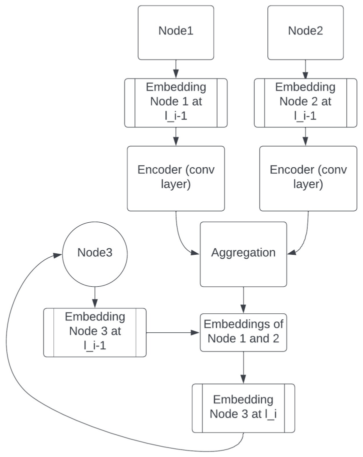
Node1
Node2
Embedding Node 1 at l_i-1
Embedding Node 2 at l_i-1
Encoder (conv layer)
Encoder (conv layer)
Node3
Aggregation
Embedding Node 3 at l_i-1
Embeddings of Node 1 and 2
Embedding Node 3 at l_i

Figure 4: Calculating Embedding 3 at Layer i

4.2 Convolutional Layers

Our first convolutional layer that we will use is GraphSAGE (Hamilton et al., 2017). There are four key steps to SAGE (GraphSAGE). The first step is SAGE sampling a fixed number of its neighbors. This sampling process considers different "hops" or layers of neighbors, enabling the model to capture information from both immediate and extended local neighborhoods. Next is aggregation: once neighbors are sampled, SAGE aggregates their features using a chosen aggregator function (like sum, mean, LSTM, and pooling). This aggregation is designed to combine and compress the high-dimensional feature information from the neighborhood into a single vector that represents the neighborhood's collective attributes. The next step is embedding generation, the aggregated neighborhood features are combined with the target node's embedding from the previous layer. This combination is processed through neural network layers, generating a low-dimensional vector embedding for the target node for the current layer. SAGE learns inductively, meaning it learns a generalizable function for generating node embeddings. Once trained, this function can be applied to nodes not seen during training, making SAGE suitable for dynamic graphs where new nodes are continually added.

The second type of convolutional layer we use is Graph Attention Network (GAT) (Veličković et al., 2017). The key idea behind GAT is that it assigns attention scores to each neighbor within a node, extracts a "message" from each neighbor node, and does a weighted aggregation using the attention scores with the messages.

Attention: This step is crucial for determining the importance or relevance of each neighbor node in the graph. As shown in figure 5, the GAT layer adds up the features or characteristics of the neighbor nodes in a certain way to determine their relative importance.

$$\textbf{Attention}_{GAT}(s,t) = \underset{\forall s \in N(t)}{\text{Softmax}}\left(\vec{a}\left(WH^{l-1}[t] \parallel WH^{l-1}[s]\right)\right)$$

Figure 5: GAT: Attention

Message: Once the model has identified which neighbor nodes are important, the next step is to extract a "message" from each important node. The message is essentially a collection of information or features from these nodes. In the GAT layer, the same weight is used across all nodes for calculating this message. This implies that the process for extracting information from each node is uniform, maintaining consistency in how the model interprets each node's data.

Aggregate: The final step involves aggregating, or combining, all the messages gathered from the neighborhood of a particular node. The GAT model employs an attention-weighted average with a nonlinear activation function, whereas SAGE utilizes a uniformly weighted average across neighbors but differs in its aggregation functions.

5 Results and Discussion

To evaluate the performance of our proposed model, we compare it with both baseline and state-of-the-art recommender systems. We chose K-Nearest Neighbors (KNN) and Singular Value Decomposition (SVD) as baseline methods due to their widespread adoption and established performance.

Unfortunately, we were unable to find readily available RMSE scores for state-of-the-art models specifically trained on the Amazon product review dataset. Therefore, we present the performance of the Graph-based Hybrid Recommendation System (GHRS) (Darban & Valipour, 2022) on the MovieLens dataset, a closely related benchmark dataset for recommender systems. The KNN and SVD metrics were obtained from the resources for the Amazon Product dataset found on Kaggle (Deo, 2019).

Model	RMSE
KNN	1.237
SVD	1.196
SAGE	1.080
GAT	1.210
GHRS (MovieLens)	0.833

Table 1: Comparison of RMSE values for different recommender systems
(Deo, 2019; Darban & Valipour, 2022)

Our initial experiments with a smaller dataset of about 200 entries resulted in a RMSE of approximately 0.3. However, we identified a significant bias in both the dataset and model, predominantly favoring reviews of 5 stars. This bias led to our model consistently predicting higher scores, and with a dataset skewed towards higher reviews, our RMSE appeared artificially low. As we worked with a more balanced dataset and tested our model with more data, we observed improved accuracy in predicting lower scores, although this came with an increased error rate.

As shown in Table 5, our proposed model (SAGE) achieves an RMSE of 1.080, outperforming both KNN and SVD. While direct comparison with GHRS is difficult due to the different datasets, its significantly lower RMSE on MovieLens suggests the potential for strong performance on the Amazon dataset as well.

Further research and experimentation are necessary to compare our model directly with state-of-the-art models on the Amazon dataset. This would involve implementing and evaluating these models under the same conditions and metrics as our proposed model.

To gain insights into the reasoning behind our model's predictions, we utilized graph explainability techniques (Amara et al., 2022) provided by Captum explainer, specifically the integrated gradient method introduced in the paper (Sundararajan et al., 2017). This technique helps identify the neighboring nodes that have the most significant influence on the model's prediction for a specific user-product interaction.

For example, the model pinpointed products such as 'TurboTax Home & Business 2014 Fed + State ...' and 'TurboTax Home and Business Fed + Efile + State...' as highly influential in predicting ratings for 'Fundamentals for Home and Small Business,' it's clear that the system's recommendations are logically consistent.

6 Conclusion and Future Work

In this research, our team navigated the challenges of limited computing resources to execute link regression prediction on user ratings for Amazon products. A notable challenge we faced was the dataset's inherent bias, characterized by high average product ratings, which influenced our regression model towards predicting higher ratings. To counteract this, we strategically undersampled 5-star reviews and explored diverse models to address the bias. While removing data might hinder the model's ability to reflect general user rating trends, we aimed to ensure our model generates accurate embeddings, particularly for users who tend to rate products highly. Our proposed model, SAGE, achieved an RMSE of 1.080, showcasing its potential and surpassing baseline methods like KNN and SVD. However, it falls short of the state-of-the-art GHRS model trained on the MovieLens dataset. Several factors contribute to this performance gap, which we must address for further improvement.

Firstly, our model currently lacks user features. While it successfully leverages product features, incorporating user information like demographics, purchase history, and browsing behavior could provide valuable insights for recommendation and potentially lead to improved performance.

Secondly, the product review dataset inherent sparsity presents challenges for learning accurate user-item relationships. This sparsity is more pronounced compared to the MovieLens dataset used for GHRS, contributing to lower performance. Addressing this issue could involve strategies like data augmentation or incorporating additional data sources to enrich the model's input and learning opportunities.

Thirdly, GHRS utilizes a more complex hybrid architecture, combining graph-based and matrix factorization techniques. While our simpler SAGE model demonstrates promising performance, exploring more complex models or ensemble approaches could potentially further improve accuracy and capture the intricate relationships within the data.

Finally, directly comparing performance across different datasets like Amazon and MovieLens presents challenges due to inherent differences. While GHRS results on MovieLens provide a valuable reference point, evaluating our model on Amazon-specific benchmarks is crucial for a definitive comparison and understanding its true performance within the target domain.

Moving forward, we will focus on addressing these limitations by incorporating user features, exploring denser data representations, investigating more complex models, and evaluating our model on relevant Amazon benchmarks. These efforts aim to bridge the gap between our model and state-of-the-art recommender systems, allowing us to unlock its full potential and deliver accurate and valuable product recommendations for Amazon users.

The code for this project is available at https://github.com/aiden200/ARS .

References

Kenza Amara, Rex Ying, Zitao Zhang, Zhihao Han, Yinan Shan, Ulrik Brandes, Sebastian Schemm, and Ce Zhang. Graphframex: Towards systematic evaluation of explainability methods for graph neural networks. *arXiv preprint arXiv:2206.09677*, 2022.

Zahra Zamanzadeh Darban and Mohammad Hadi Valipour. Ghrs: Graph-based hybrid recommendation system with application to movie recommendation. *Expert Systems with Applications*, 200:116850, 2022.

Saurav Deo. Amazon product reviews. `https://www.kaggle.com/datasets/saurav9786/amazon-product-reviews` , 2019.

Chenyuan Feng, Zuozhu Liu, Shaowei Lin, and Tony Q.S. Quek. Attention-based graph convolutional network for recommendation system, 2019.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017. URL `http://arxiv.org/abs/1706.02216`.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. *CoRR*, abs/2002.02126, 2020. URL `https://arxiv.org/abs/2002.02126`.

Jure Leskovec. Graph neural networks - part 5. `https://web.stanford.edu/class/cs224w/slides/05-GNN3.pdf`, 2023a.

Jure Leskovec. Graph neural networks - part 3. `https://web.stanford.edu/class/cs224w/slides/03-GNN1.pdf`, 2023b.

Xiaofeng Li and Dong Li. An improved collaborative filtering recommendation algorithm and recommendation strategy. *Mobile Information Systems*, 2019:11, 2019. doi: 10.1155/2019/3560968.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fined-grained aspects, 2019.

Yuanzhe Peng. A survey on modern recommendation system based on big data. *arXiv*, abs/2206.02631, 2022. URL `https://arxiv.org/abs/2206.02631`.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084, 2019. URL `http://arxiv.org/abs/1908.10084`.

A. Sagdic, C. Tekinbas, E. Arslan, and T. Kucukyilmaz. A scalable k-nearest neighbor algorithm for recommendation system problems. In *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, pp. 186–191, 2020. doi: 10.23919/MIPRO48935.2020.9245195.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, 2001.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pp. 3319–3328. PMLR, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Ching-Seh Mike Wu, Deepti Garg, and Unnathi Bhandary. Movie recommendation system using collaborative filtering. In *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, pp. 11–15, 2018. doi: 10.1109/ICSESS.2018.8663822.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems, 2018. URL `http://arxiv.org/abs/1806.01973`.

A Appendix

This section lists the contributions of each author.

Abdul	Research, Poster, Paper, Embedding Implementation, GNN Implementation, Experimentation
Aiden	Research, Poster, Paper, Embedding Implementation, GNN Implementation, Experimentation
Shubh	Research, Poster, Paper, GNN Implementation, Dataset Research
Michael	Research, Poster, Paper, Embedding Implementation, Dataset Research
Prathamesh	Research, Poster, Dataset Research

Table 2: Contributions