ENGR-E 536 - HPGA - Project Report
Movie Recommendation System using GNN and PyTorch Geometric

Akash Bhapkar
Luddy school
abhapkar@iu.edu

Anurag Hambir
Luddy school
ahambir@iu.edu

Roopank Kohli
Luddy school
rookohli@iu.edu

Ruchir Tatar
Luddy school
rtatar@iu.edu

1 ABSTRACT

The project is on developing and implementing Recommendation Systems for movies using Graph Neural Networks. With the development of online movie viewing, it is important to create a system that can provide viewers multiple suggestions or recommendations. It's quite basic and clear. When you read the news, view a Netflix movie, or just purchase something on Amazon, you will see notifications such as:

- You will also have an interest in this item
- Products linked to this item
- Customers who purchased this item also bought
- Because you've watched movie A, you might also want to watch Movie B

There are several recommendations that might be displayed to you, but they are all essentially the same in that they are just promoting an item for consumption. That's because these complex systems have a profile of the users and know what kinds of things they like to eat. As a result, they just try to come up with goods that you are more likely to appreciate, rather than the most popular or a random group of items that may or may not be to your liking.

2 INTRODUCTION

Recommendation Systems include things like information retrieval, data mining, and machine learning. Recommender systems are very crucial in today's ecommerce company. Using recommender systems, users are directed to books, movies, videos, technological devices, and a range of other items. Users can get personalized recommendations from recommender systems, which can help them make better online transaction decisions, increase sales, redefine their web browsing experience, retain customers, and improve their shopping experience. Although search engines address the issue of information overload, they do not provide data personalization. Recommendation engines are used to enable personalization. Several types of recommender systems exist, including content-based, collaborative filtering, hybrid recommender systems, demographic, and keyword-based recommender systems.

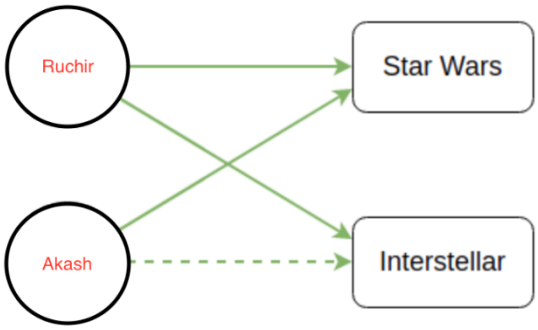
Recommender systems are used in a wide range of online businesses, including e-commerce for product recommendations and streaming services for movies and music. One of the most important characteristics of such systems is that they should deliver reliable recommendations that are similar to the preferences of clients. Based on reviews, product information, and demographic characteristics, customers' preferences are evaluated using a range of statistical and machine learning approaches. Graph Neural Networks can predict ratings by simulating the relationship between customers and things using a bipartite graph. In a computational study employing numerous benchmark data sets, we examine how Graph Convolutional Matrix Completion, a kind of Graph Neural Network, performs comparable to and even outperforms state-of-the-art machine learning models.

In this situation, graph neural networks were used as a technique to recommend a movie. Graph data structures employ graph theory to connect data, which may then be used to store, query, and present data by linking bits of data, referred to as nodes, with a link between them.
A recommendation system is usually a rating or prediction system that predicts a person's favorite options or choices based on their previous encounters. These technologies may be used to a wide range of fields, including social media,

streaming platforms, and e-commerce. Because of its immediate application value in everyday chores and extensive research potential, this has gotten a lot of attention lately.

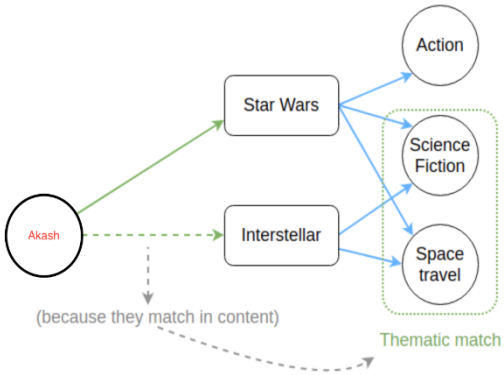
In general, there are three ways to achieve this: collaborative filtering, content-based filtering, or a blend of the two. A collaborative filtering recommender will decide what you want based on the interactions of users who are similar to you.

Consider two fictional users, Ruchir and Akash, who both enjoy science fiction films and Star Wars. Mike like Interstellar as well, although Akash has yet to see it. Because Mike, who enjoys the same things as Akash, like Interstellar, the collaborative filtering recommender would suggest it to him.



Substance-based filtering recommenders, on the other hand, would examine the content of both films to see if the similarity in content supports a suggestion. That is, similar things will appeal to individuals who have similar tastes. This is a more successful and transparent technique than collaborative filtering since we perceive similarity through more concrete qualities such as genres, actors, and so on. It's worth noting that, in our case, we may still deduce user preferences even if no one has given Interstellar a review. This is not possible with collaborative filtering. In the case of a cold-start item, content-based filtering may truly shine. This happens when the system adds a new item to the system that no one has rated yet.

We're going to create a content-based recommender that leverages a knowledge graph to provide product recommendations to users using Graph Neural Networks.



3 RELATED WORK

1. Movie Recommendations powered by Knowledge Graphs and Neo4j.

   The article shows how, independent of the application domain, knowledge graphs and graph databases may be used to generate very effective product recommendations. It also shows how the dataset is built using Neo4j. The article also urges us to examine how using graphs to model an issue can provide new and effective tools for quickly solving complicated problems. Using Neo4j, the author was able to design a recommendation system that allows people to collaborate to create a dataset unlike any other. The article compares the results with traditional database technologies such as SQL and concludes that obtaining the same objective using traditional database systems would have been far more difficult.

2. Graph Neural Networks for Efficient Recommender Systems || Oweys Momenzada, Michael Palk, Stefan Voß

   This paper has solved the problem of recommendation engine using Graph Neural Networks. The paper has formulated the problem as a link prediction problem between users and items and modeled the relation between them as a bipartite graph. Embeddings are created for user and item features. The paper compares the conventional methods with Graph neural network architectures for generating recommendations with the latter

outperforming the conventional models in terms of accuracy for predicting ratings. Performance of the models has been evaluated on benchmark recommender system datasets. RMSE (Root Mean Squared Error) has been used as the evaluation function. The paper mainly focuses on the implementation of GCMC model which is a GCN (Graph Convolutional Network) architecture but have also experimented with several Graph Architectures such as GraphSAGE and GAT (Graph Attention Network) with all the models giving similar accuracies.

3. Integrate Neo4j with PyTorch Geometric to create recommendations.

   The article states how PyTorch Geometr ic library is used for creating and training Graph Neural Networks applications.
   The article talks about creating movie embeddings
   to capture similarities between movies b ased on performers and directors. The problem has been formulated as a link prediction problem between the users and movies and the graph constructed is a bipartite graph. The author has created a heterogeneous graph using PyTorch Geometric by using the Neo4j Graph. The model used for training is GraphSAGE and the loss function used is Weighted Mean Squared Error. Finally, the predictions are exported back to Neo4j and visualized.

3 METHOD

For our method, we have performed the following steps,

- Data Collection and preprocessing
- Generating embeddings for movies and users
- Creating a GNN model
- Performance tuning and evaluation

We have used MovieLens dataset which is readily available on the internet. For preprocessing the data, we have converted the movie features such as genre in the format that we wanted by removing the '|' separator and storing each genre in a separate column.

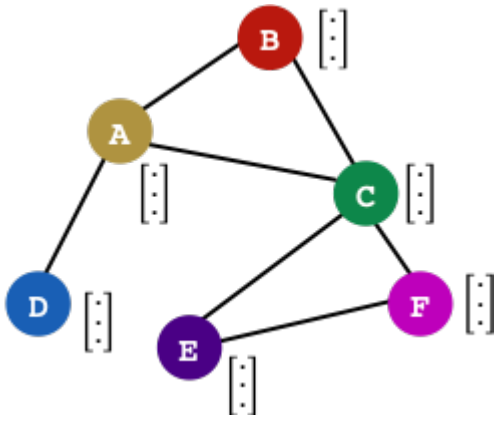
Next step is to generate embeddings for movies and users based on their features. For movies, the features we used were the movie title and genres. Following is how we embedded the movie features,

- For movie titles, we used Sentence Transformers which gave us a 384-dimensional vector for each movie title.
- For genres, we used One Hot Encoding. Since we have 20 genres, the resulting vector was a 20-dimensional vector for each movie.

We stacked these vectors together to give 404-dimensional feature vector for each movie. In case of users, there were no features available, so we embedded the users as an identity matrix.

We then created a heterogeneous data structure that consisted of both movie and user features as well as the edges between them. The weight of each edge is the associated rating given by a user to a movie. The problem hence gets formulated as a link prediction problem and the graph being a bipartite graph. Next step is to model the data.

We modeled our data using GraphSAGE GNN framework. GraphSAGE is a known and proven framework for large-scale inductive representation learning. For our model, we tried different combinations of convolutional and linear layers with the best results coming from 2 convolutional and 2 linear layers. For optimization, we have used Adam Optimizer. We also experimented with GAT and GCN GNN frameworks, but we chose GraphSAGE as it was faster and yielded slightly better results in terms of the evaluation metric which is Root Mean Squared Error. We evaluated the loss using Weighted Mean Squared Error.



$$wmse = \frac{\sum_{i=0}^{n} w_{observedRating} \cdot (PredictedRating_i - ObservedRating_i)^2}{n}$$

A loss function is a measure of the model's ability to forecast the predicted outcome accurately. The reason we chose Weighted mean squared error is because it used for handling the skewness of predictions and give more weightage to some predictions over others. In our dataset, the number of movies rated 0 or 1 are significantly low and hence the error(weight) associated with them will be significantly high as a user is more likely to not rate a movie with 0 or 1. This has helped in improving the performance of the model and get better predictions. For calculating the weight associated with each rating, we first calculate the count of each rating. Then, we obtain the maximum count and divide count of each rating with the maximum count to get the weight of each rating. This result is multiplied with the squared difference between predicted and actual values. We then sum up the values across all ratings and divide it by the total number of ratings as mentioned in the above formula.

We analyzed our model's performance based on our loss function and evaluation metric and evaluated its performance by tuning our hyperparameters. We have discussed the different combinations of our hyperparameters further in the Experimental Setup and Results sections.

5 EXPERIMENTAL SETUP

Data: We used the MovieLens dataset. The dataset consists of interactions(ratings) between users and movies. There are 71,567 users and 10,682 movies in the dataset with total ratings summing up to 10,000,054. We did not have any features for users. For movies, we used the movie title and genres. For embedding the features, we used Sentence Transformers for movie titles and One Hot Encoding for genres and stacked them together. For training and testing purposes, we have used 80% of the data for training, 10% for validation and 10% for testing. We are using weighted mean squared error to handle the slight skewness present in the dataset.

Hyperparameters: There are quite a few hyperparameters that we had to tune to obtain the best results possible for this problem. They are the following,

- Learning rate
- Number of Convolutional and linear layers
- Epochs
- Number of channels

Models: We experimented on different convolutional models to generate embeddings for our dataset such as GraphSAGE, GAT, GCN.

Software Versions: We used Python Version 3.6.8.

Hardware Platforms: We experimented on different platforms. Initially we prototyped our code on Google Colab and later ran the code on Carbonate where we tried different combinations of the hyperparameters mentioned above.

6 RESULTS

The main objective of this project was to build a recommendation system which could give out the proper results taking into consideration the ratings a user has provided, genre and movie name. We ran some experiments with different hyper parameters to test our model and below are the results:

No. of Epochs	Learning Rate	No. of Channels	Loss	Train RMSE	Test RMSE
300	0.1	64	6.164	1.413	1.413
700	0.1	64	6.131	1.415	1.414
700	0.1	128	6.162	1.414	1.415
300	0.01	64	2.547	1.060	1.105
700	0.01	64	1.892	0.929	1.076
700	0.01	128	2.072	0.958	1.075

The performance of our model utilizing GraphSAGE GNN is shown in the table above. We also attempted GAT and GCN, however due to the size of our dataset, for GAT it became computationally costly, and the results were identical to GraphSAGE for initial setup. For GCN, the accuracy was slightly lesser as compared to GraphSAGE. We may deduce from the table above that the model with a learning rate of 0.01, 64 channels, and 700 epochs produces the best results.

After training the model, we ran several tests to see how accurate the predictions were. Assume that for a certain user, the following are the top 10

movies that he enjoyed and gave high ratings:

movieId	title	genres
50	Usual Suspects, The (1995)	Crime|Mystery|Thriller
214	Before the Rain (Pred dozhdot) (1994)	Drama|War
293	Léon: The Professional (a.k.a. The Professiona...	Action|Crime|Drama|Thriller
296	Pulp Fiction (1994)	Comedy|Crime|Drama|Thriller
541	Blade Runner (1982)	Action|Sci-Fi|Thriller
741	Ghost in the Shell (Kôkaku kidôtai) (1995)	Animation|Sci-Fi
745	Wallace & Gromit: A Close Shave (1995)	Animation|Children|Comedy
778	Trainspotting (1996)	Comedy|Crime|Drama
858	Godfather, The (1972)	Crime|Drama
924	2001: A Space Odyssey (1968)	Adventure|Drama|Sci-Fi

As per above movies, we can see the user likes movies from 1990's decade and more into drama, thriller, and action movies. Our model gives out below top 10 predictions for this user along with predicted ratings.

movieId	title	genres	predicted_rating
51	Guardian Angel (1994)	Action|Drama|Thriller	5
53	Lamerica (1994)	Adventure|Drama	5
59	Confessional, The (Confessionnal, Le) (1995)	Drama|Mystery	5
139	Target (1995)	Action|Drama	5
142	Shadows (Cienie) (1988)	Drama	5
143	Gospa (1995)	Drama	5
167	Feast of July (1995)	Drama	5
396	Fall Time (1995)	Drama	5
399	Girl in the Cadillac (1995)	Drama	5
403	Two Crimes (Dos crímenes) (1995)	Comedy|Crime|Drama	5

7 CONCLUSIONS

By implementing and testing GNN, particularly the GraphSAGE model, on MovieLens data set, we investigated the predictive quality of GNN and, in particular, the GraphSAGE model. We demonstrated that a simpler version of the GraphSAGE model may also provide good results, and that GNN in general can match or even exceed CF and matrix factorization-based recommender systems. Practitioners can use GNN in an online platform to achieve good rating predictions for reliable predictions on the one hand and can potentially visualize the results as a bipartite graph to draw further conclusions on the other hand. Thanks to the fact that it can be implemented efficiently with the help of existing programming libraries. In future, more data for users might be collected, and collaborative filtering based on user embeddings could be provided.

8 TEAM MEMBER CONTRIBUTIONS

Ruchir Tatar: Data collection, Preprocessing, and Analysis

Roopank Kohli: Embeddings generation (User and Movie) / Representation Learning

Anurag Hambir: Modeling and Tuning (Building model architecture using GraphSAGE and GAT)

Akash Bhapkar: Performance Evaluation and Experimentation

All the team members researched and analyzed different methodologies related to the project and contributed dedicatedly to documentation (report) and presentation preparation

9 REFERENCES

[1] R. Lavanya and B. Bharathi, "Systematic analysis of Movie Recommendation System through Sentiment Analysis," 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021,pp.614-620,doi: 10.1109/ICAIS50930.2021.9395854.

[2] S. Agrawal and P. Jain, "An improved approach for movie recommendation system," 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017, pp. 336-342, doi: 10.1109/I-SMAC.2017.8058367.

[3] P. Wang, X. Li, F. Du, H. Liu and S. Zhi, "A Personalized Recommendation System based on Knowledge Graph Embedding and Neural Network," 2019 3rd International Conference on Data Science and Business Analytics (ICDSBA), 2019, pp. 161-165, doi: 10.1109/ICDSBA48748.2019.00042.

[4] Y. Niu, X. Xing, M. Xin, Q. Han and Z. Jia, "Multi-preference Social Recommendation of Users Based on Graph Neural Network," 2021 International Conference on Intelligent Computing, Automation and Applications (ICAA), 2021, pp. 190-194, doi: 10.1109/ICAA53760.2021.00040.

[5] Momenzada, Oweys & Palk, Michael & Voss, Stefan. (2021). Graph Neural Networks for Efficient Recommender System.