Revolutionizing Movie Recommendations for Reviewers: Unleashing the Power of Graph-Based Recommender Systems

Gallos Konstantinos
Department of Informatics
Aristotle University of Thessaloniki

Chirtoglou Marios
Department of Informatics
Aristotle University of Thessaloniki

Polychronidis Nikolaos
Department of Informatics
Aristotle University of Thessaloniki

January 2024

Abstract

This survey investigates the effectiveness of three distinct algorithms in the realm of movie recommendation systems tailored for reviewers. Addressing the challenge of providing targeted movie suggestions to reviewers for review purposes, we analyze and compare the performance of the selected algorithms. We make usage of movie and review features and project their relations between entities into a heterogeneous graph. Our findings show that there are numerous ways of approaching this problem and solving it efficiently. Moreover, they offer valuable insights for the continued improvement of recommendation strategies in the context of user-generated movie reviews. Finally, we point out some important future extensions for each of our methods.

1 Introduction

Graph-based recommendation systems are crucial in today's digital landscape and leverage interconnected data to offer precise and context-aware content suggestions. By modeling the relationships between entities in a graph, these systems specialize in understanding user preferences and providing tailored recommendations, such as movie recommendations.

Many studies have been done on recommending movies to users and it is a well-studied problem. We extend this problem to a very similar one, but now we want to recommend movies to critics so they can rate them based on their preferences. Recommending a related movie to a reviewer increases the likelihood that the reviewer will watch and rate the movie and thus help future users choose a well-rated movie.

Our approach to the problem is to create a bipartite graph containing of reviewers and movies. Our goal is to predict future edges that connect a critic to a movie. In order to achieve this, we employ a preprocessing methodology that involves extracting and transforming data to construct a

meaningful representation of the relationships between reviewers and the movies they evaluate. We consider the assumption that two critics do not connect with each other and the same applies for the movies. Following this stage, we present and analyze three algorithms designed to predict links within the reviewer - movie network. We use content-based techniques as well as state-of-the-art Graph Neural Networks for learning graph representation.

This study is structured as follows:

- In section 2 we give some useful insights about our dataset and how we process the raw data to convert them into a graph structure. Moreover, we explain the metrics we use to evaluate the performance of our models.
- In sections 3, 4 and 5 we present three different algorithms used for movie recommendations. We explain how they work, point out their strengths and weaknesses, and also refer to some related work related to our models. Furthermore, we discuss some possible extensions of them.
- In section 6 we show the results of our experimentation on our dataset, and compare our methods.
- Finally, in section 7 we summarize our work, present some conclusions and give directions for future work.

2 Preprocessing and Data

2.1 Dataset

The dataset we used for our problem is taken from rotten tommatoes movies[1]. This dataset includes movie features, critics' reviews of any of them, and related information about reviews. From this data we create a directed bipartite graph, in which one set consists of the reviewers and the other of the movies. An edge between disjoint sets indicates the review which a reviewer made of a movie. Therefore, our graph is heterogeneous and it has two node types and one edge type.

We inspect movie features and keep these that are relative to our task, and we also do the same for review features. We do not associate any extra feature with the reviewers as we do not have such information. We consider the rating of a review to be the weight of the edge. In our analysis we do not consider the time factor (when was the review done). After that, we scale and preprocess our features so that they are ready to be applied to our algorithms. The result of our preprocessing is one movie feature matrix that contains movie features with numeric representations, and one list of node pairs (edges between reviewer and movie nodes) associated with edge features. Note that there are some movies that we have missing features for them. Each of the algorithms we implement handles this situation differently. Our dataset contains weights (reviewers' ratings) and 2 features for each review and 31 movie features. Movie features include movie genres, audience rating, categories based on the movie suitability for audience etc. The review features are whether a reviewer is considered as a top critic, and the type of review.

2.2 Evaluation metrics

The metrics we use in order to evaluate the perform of our algorithms which we present are:

[1] https://www.kaggle.com/datasets/stefanoleone992/rotten-tomatoes-movies-and-critic-reviews-dataset/data?select=rotten_tomatoes_critic_reviews.csv

- **precision**
- **recall**
- **f1**
- **accuracy**
- **area under ROC curve**

Precision is a commonly used metric in recommendation systems to evaluate the accuracy of the recommended items. Precision is the ratio of relevant items selected by the recommendation system to the total number of items recommended. It helps measure the accuracy of the system in terms of how many of the recommended items are actually relevant to the user. In summary, precision helps you understand the accuracy of your recommendation system by measuring the proportion of relevant items among the recommended items.

$$\text{Precision} = \frac{\text{Number of Relevant Items Recommended}}{\text{Total Number of Items Recommended}}$$

Recall is another important metric in recommendation systems, especially when evaluating the ability of a system to retrieve all relevant items for a user. Recall is the ratio of relevant items that have been recommended to the total number of relevant items available. It helps measure the coverage of the recommendation system in terms of capturing all relevant items.

$$\text{Recall} = \frac{\text{Number of Relevant Items Recommended}}{\text{Total Number of Relevant Items}}$$

The F1 score is a metric that combines precision and recall into a single value, providing a balanced measure of a recommendation system's performance. It is particularly useful when there is a need to balance the trade-off between precision and recall. The F1 score is particularly useful when precision and recall are equally important, and there's a need to balance the system's ability to recommend relevant items (recall) with its ability to avoid recommending irrelevant items (precision).

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Accuracy in the context of a recommendation system is typically calculated by measuring the proportion of correctly recommended items to the total number of recommendations. if you want to calculate accuracy, you can use the following formula:

$$\text{Accuracy} = \frac{\text{Number of Correctly Recommended Items}}{\text{Total Number of Recommended Items}}$$

ROC (Receiver Operating Characteristic) curves plot the true positive rate against the false positive rate at various thresholds. AUC-ROC (Area Under ROC Curve) measures the area under the ROC curve, providing a single scalar value summarizing the performance across different thresholds.

3 Content-Based Movie Recommendation System Using Genre Correlation

For our first algorithm we use the data after the reprocessing we described in previous chapter. Our algorithm based on the mechanism proposed by [Reddy et al., 2019] it is a mechanism that finds the similarities among users over the genres of the movies and recommend to the target user all the movies from the top k-nearest neighbours (meaning the most similar users).

3.1 Our mechanism

As we already said our mechanism is based on the existing algorithm by [Reddy et al., 2019] and keeping the baseline of this algorithm. Although our algorithm has slightly changes overall. The steps we follow in our algorithm are the following:

Step 1: Input by the user. We ask from the user to give us the id for the target user and the total number of users that he wants to influence the final result.

Step 2: Storage the necessary data. We define an 2-dimensional array (genre matrix) in which we storage the data from the file with the movies (movies.csv). This array includes the information about the movies and the genres that the movies belong to using 1 if the genre exists and 0 if not.
we define another second 2-dimensional array. (ratings matrix) to storage the data from our reviewers file (reviewers.csv). This array includes the information about the reviewers and the movie they rated in scale 1-10.

Step 3: Convert the ratings matrix. We convert the ratings matrix using the following rule. Only for the movies that have been rated with a non zero rating, we covert them to 1 or -1 depending on its value its greater than 6 or not, accordingly.

Step 4: Dot product. We calculate the dot product between the transposed genre matrix and ratings matrix and we keep that info in a 2-dimensional array (result matrix).

Step 5: Covert the result matrix. We convert the result matrix into a binary matrix. If a value is negative we convert it to 0, otherwise to 1.

Step 6: Calculate Distances. We calculate the euclidean distances between the target user and the other ones.

Step 7: Find similar reviewers. Sorting that matrix by ascending we obtain the top k users that are have the most similar taste in genres on movies with the target user.

Step 8: Recommend movies. We create a list with the recommendation movies for the target user. For this procedure, in our final list for the recommended movies we keep all the distinct movies that the similar users liked excluding the ones that our target user already watched.

3.2 Negative aspects

Lets discuss some of our negatives aspects of this approach.

The first main problem of this approach which is being inherited by the [Reddy et al., 2019] is **scalability**. Our mechanism is not scalable which means that takes more and more time as we have more data tp proceed. From our simulations our algorithm takes about 5 minutes from start to finish for total 17255 movies and total of 6000 reviewers. This arises probably from the usage of matrices for storaging the data with the majority of the info being zeros. This can be solved by using hashmaps or lists and keeping only the useful info for the processing in order to reduce the running time.

Another thing that is not performing so well with this mechanism is **the total amount of the recommended movies** that being recommend in the end. The number of that movies can be arbitrary large or small or even zero. We have to mention here that this problem arises more in small datasets like ours and that this is not going to happen with such a propability with larger datasets. That behavior of our algorithm is still bad so.

In the **first case** if the target user has seen only a few movies and the similar users to him have seen a large amount of movies, that could be an occasion in which we obtain a large set of recommended movies.

On the **other hand**, if the target user has already seen a large amount of movies and the similar users just a few of them, its possible to not be recommended movies.

A possible solution for this could be to specify the total number of movies we are going to recommend or perform more analysis in our algorithm in order to reduce the total amount of the movies in the final stage of recommendation.

3.3 Positive aspects

Lets discuss some of our positives aspects of this approach.

First of all our algorithm is an easy approach who solves the problem with basic techniques such as dot product and finds the similar users to a target user based on similarity on the genres of the movies the like the most.

In our method, the construction of a graph is not necessary for the implementation of our algorithm gaining that time of constructing it, especially if that was about a large network.

In our algorithm we use the genre of the movie to find similarities between users. That can be extend into more factors such (as e.g actors, production, director etc).

3.4 Extensions

Some interesting extensions of our mechanism in order to obtain possibly better results are the following.

- **Don't convert matrices to binary:** Converting matrices in step 3 and step 5 to our algorithm is a tactic in order to say which movies has rated good or bad (in step 3) and if a genre is being liked by a user or not (in step 5). This procedure separates the two states, meaning we keep the qualitative information (likes, dislikes) but lose the quantitative information about it (how much it is liked). For example two movies rated 6.5 and 10 will both have a 1 at the end.
  It would be interesting to see if by keeping this information we obtain better results.

- **Recommend only the movies that the target user like their genres may be more evaluate.** In our mechanism the final movies who are being recommended are all the distinct movies that the target user haven't already see, that were being saw by the similar to him users. That means that movies with genres that the user target seems to not prefer are still being recommended.
  Its more rational to restrict that and recommend only the movie with genres that like by target user. This is probably going to have even better results

For the evaluation of our model we use the metrics we described in section 2. As we can see in Table 1, our algorithm has a very good ratio when it comes to recall, but it lacks precision.

4 Graph Neural Network Approach

Graph Neural Networks (GNNs) are state-of-the-art models in graph representation learning. The survey done in [Gao et al., 2022] lists some GNNs techniques used in recommendation systems and in link prediction scenarios. Also, more GNNs are used for representation learning in heterogeneous graphs as discussed in [Zhang et al., 2019], [Dong et al., 2017]. In our study we use *GraphSAGE* framework, proposed in [Hamilton et al., 2017], to learn node embeddings for both movie and review nodes by using local neighborhood information. GraphSAGE was originally designed for homogeneous graphs, but there are some extensions that can handle heterogeneous graphs[2]. GraphSAGE samples some neighbors of the target node and uses an aggregation function (such as median, maximum, or LSTM networks) to combine information from the neighbors. It also works well in unseen data and it can produce embeddings for nodes not seen during training.

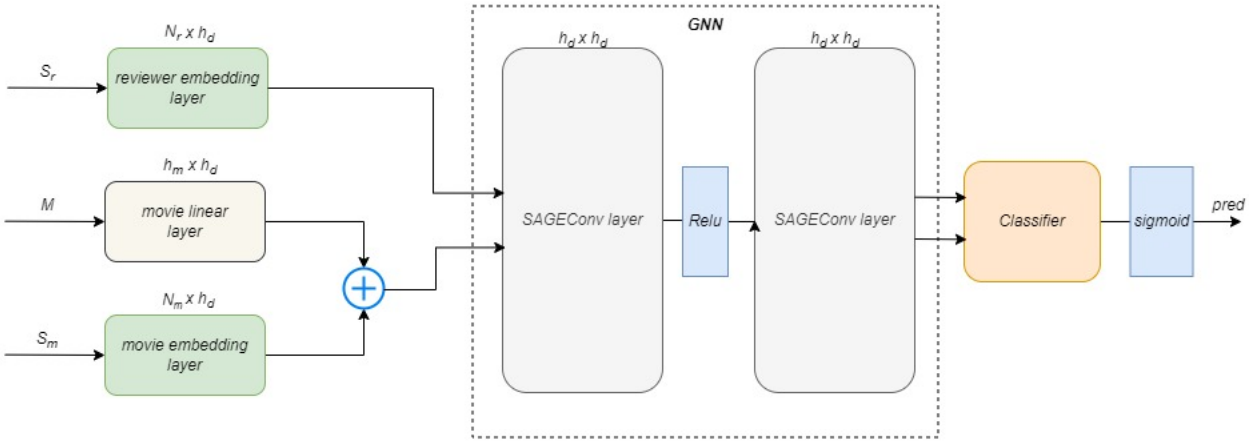

Figure 1: Architecture of our Neural Network model for the link prediction task. GNN layers are used to capture the structural information of the graph. S_r, S_m are the sets containing reviewers and movies node ids respectively (with $|S_r| = N_r$, $|S_m| = N_m$). M is the movie feature matrix, of size $N_m \times h_m$. h_m is the number of movie features and h_d is a hyperparameter of the model, the dimension of hidden layers. Finally, link prediction results are in *pred* vector.

[2]https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.conv.SAGEConv.html

4.1 Architecture of the model

In Figure 1 we present the architecture of our model. First, we pass reviewer node ids (set S_r) and movie node ids (set S_m) into our model. We define $|S_r| = N_r$ and $|S_m| = N_m$. Two embedding matrices are learned for their representation based on the loss function of the downstream task. Node embeddings are projected in a h_d-vector space, where h_d is a hyperparameter of our model. Moreover, we pass the movie feature matrix M through a linear layer which is responsible for transforming the input features of movies. After that, we add the movie feature matrix with the movie embedding matrix to obtain a compact representation for movies. We call this M', and R' the reviewer embedding matrix.

Our GNN consists of two SAGEConv layers (GraphSAGE). The first layer improves reviewer and movie emdeddings by using the graph structure (local neighborhood of nodes). We use ReLU activation function to introduce non-linearity into the model. The second layer is used for better results. These two layers combined help our model to learn a more global representation of the graph. Furthermore, since are our task is not very complicated and our graph is bipartite (which means that local neighborhood provides useful insights by itself), there is no need to stack more GNN layers. M' and R' are passed and being processed separately to GNN, so that the model can learn their representations. We also provide the selected edges of the processed nodes so that our model can exploit the structure of the graph.

Our model is being trained in edges (pairs of reviewer-movie nodes). While training the GNN layers we use split the training edges into edges for message passing and edges for supervision. Edges for message passing are utilized for the propagation of information through the graph. Edges for supervision are used for focusing on the link prediction task and calculating the loss. This method guides the model to focus on specific patterns in the graph that are relevant to the specific link prediction task. It allows for more flexibility and control over what the model learns from the graph structure. What is more, we generate some non existing edges (negative examples) in order to help the classifier to distinguish whether an edge really does exists or not.

Finally, a binary classifier is applied between source and destination node embeddings to derive edge-level predictions. We use binary cross entropy loss for training.

Except SAGEConv layer we also tried the approach mentioned in [Morris et al., 2019] [3], but the results were not as good as SAGEConv's.

4.2 Advantages and Drawbacks

The model described above uses all the information we have available. It uses movie features matrix, edge features and the topology of the graph. These results to achieve some really good scores in all metrics, as we can see in Table 1. The performance obtained by this model explains the capability of Graph Neural Networks in representation learning and (for our purpose) in link prediction task. This task was a relative easy one and it did not require a lot of fine tuning, but this is not the usual case.

However, one thing to consider is that GNNs usually require a huge amount of data in order to be trained and they are not suitable for problems where the graph is not large. Finally, their architecture and complexity can be complicated even for some easier tasks.

[3] https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.conv.GraphConv.html

4.3 Extensions

Our model could also handle a feature matrix, with information about critics reviewers by simple adding a reviewer linear layer to transform the input features into the desired dimension. Additionally, we could experiment more with the architecture of the network as well as with the values of various hyperparameters (such as the dimension of hidden layers). By adding more and different type of GNN layers we could try to capture more global patterns of the graph.

5 Recommendation algorithm based on cliques

Our final approach draws inspiration from the work presented in [Vilakone et al., 2018]. However, we have introduced modifications to both the data representation and the overall process. The original algorithm, initially designed for movie recommendations to users, has been adapted to incorporate users' characteristics. In our approach, we place a strong emphasis on leveraging the features of movies to enhance the recommendation process.

The implementation follows a structured series of steps, combining elements from the original k-cliques algorithm with considerations for users' preferences based on the movies which have previously reviewed and movies' features.

5.1 Our mechanism

As previously mentioned, our methodology is rooted in the algorithm proposed by [Vilakone et al., 2018], maintaining its foundational structure. However, our algorithm introduces subtle modifications to enhance its overall efficacy. The procedural steps of our algorithm are outlined as follows:

Step 1: Data Input. Instead of relying on user input, our algorithm takes movies and their characteristics as input. Refer to Chapter 2 for details on data preprocessing.

Step 2: Create Adjacency Matrix. Utilizing cosine similarity measure with k=0.5, we construct a 2-dimensional adjacency matrix representing the relationships between movies. The matrix size is $n \times n$, where n is the number of movies and there is a 1 if the 2 movies are similar.

Step 3: Generate Cliques. Unlike the original algorithm, our approach employs a different algorithm to compute cliques. Despite the algorithmic difference, the output remains consistent with the original paper.

Step 4: New Movie Input. Once cliques are determined, we assign a new movie to the cluster that exhibits the highest cosine similarity with the new movie's features.

Step 5: Select Reviewers. Identify the reviewers associated with movies in the chosen cluster. The top-5 reviewers are then selected based on their top critic ratings, a binary field from the review.csv file. Reviewers with the highest count of top-critic reviews for cluster movies are recommended.

5.2 Pros and Cons

This methodology adopts a content-based approach, offering a straightforward and advantageous framework. Its primary strength lies in the unrestricted utilization of a diverse set of movie features.

Model	**Precision**	**Recall**	**F1**	**Accuracy**	**AUC**
Content-based using genre	0.15	0.95	0.2	0.65	-
GNN GraphConv	0.7200	0.6658	0.6918	0.8023	0.7682
GNN SAGEConv	0.8613	0.9082	0.8841	0.9207	0.9175

Table 1: Comparison of the described methods on different metrics.

This flexibility allows for the incorporation of an extensive range of movie attributes without any predefined limitations. In the process of selecting an appropriate reviewer, we have the flexibility to consider additional features beyond the standard ones, such as 'top-critics,' enhancing the precision of the recommendation process when it aligns more effectively with our specific case.

However, the previous approach exhibits increased time and resource complexities. The adjacency similarity matrix demands a considerable amount of memory space and it is really sparse. Additionally, the computational burden is significantly intensified during the cluster calculation phase, resulting in prolonged execution times and resource-intensive operations. In our specific case, we encountered limitations when attempting to scale the algorithm beyond 4000 movies. The persistent failure beyond this threshold was attributed to memory constraints, presenting a substantial obstacle to conducting a more comprehensive evaluation of our approach.

5.3 Extensions/ Improvements

Building upon the points discussed earlier, there is a crucial necessity for enhancing our approach. The primary challenge lies in the computation of movie clusters, urging the exploration of alternative algorithms or probabilistic methods to alleviate the resource demands. Additionally, incorporating reviewer characteristics could contribute to a more accurate and personalized outcome.

6 Results

In Table 1 we summarize the results of each model.

As we can see for the first algorithm we obtain a near perfect recall but we lack in precision. That arises naturally from our code. The final recommendation is a possibly large amount of movies from the top-k similar users and that means higher probability to get the movies we want but also some that we do not too.

GNN SAGEConv based model uses all the available information (movie features matrix, edge features and graph structure). This, combined with GNN's ability to capture local relationships between nodes, leads to excellent results on all metric scores, outperforming the rest of the algorithms.

In the third algorithm, we utilize all the movie features to derive similarity measures between them. This implies that the algorithm can function effectively even with varying numbers of features, although computation time may increase. Therefore, it is crucial to conduct research to determine the optimal amount of data and features for obtaining valid results efficiently.

In Table 2 we can see some characteristics of each algorithm. Content-based using genre method uses reviewer-movie Adjacency matrix and movie (genres) features to generate the reviewer-reviewer correlation matrix. GNN SAGEConv learns reviewer and movie embedding matrices. Recommendation algorithm based on cliques creates movie-movie correlation matrix. Finally, the rest of the columns explain which algorithms use edge weights, movie features, and review features. Of course, all three algorithms use the reviewer-movie adjacency matrix.

	Matrix Created	**Movie Features**	**Review (Edge) Features**	**Edge Weights (Ratings)**
Content-based using genre	Reviewer-Reviewer Correlation matrix	Genres only	No	Yes
GNN SAGEConv	Reviewer and Movie Embedding matrices	All	Yes	Yes
Recommendation algorithm based on cliques	Movie-Movie Correlation matrix	All	Yes (Only Top Critic)	No

Table 2: Characteristics of each presented algorithm. All three algorithms use the reviewer-movie adjacency matrix.

7 Conclusions and Future Work

In this paper we tried to make recommendations in a network with reviewers and movies. We used three different approaches in order to achieve that, and we evaluated the mechanisms we used with the metrics we described in section 2. We described each algorithm, the way it performs, their positive and negative aspects.

Firstly, we used an Content-based Using genre correlation for recommendation systems which takes into account the rating of the reviewers upon the movies and the genre of the movies. Using this, we obtain a correlation between the reviewers based on the genres who prefer using euclidean distances. Having the top-k similar users to the target user, we recommend to him the movies that has been enjoyed by his similar to him reviewers excluding the ones that he already saw.

Afterwards, we explained a Graph Neural Network method in order to represent the different type of nodes and used a binary classifier to predict whether an review is likely to exist or not.

On our third algorithm, in summary, our movie recommendation system prioritizes simplicity by emphasizing movie features over intricate user data. However, a drawback lies in its resource-intensive nature, especially when dealing with a large number of movies, resulting in prolonged processing times. Streamlining this complexity while preserving simplicity is an ongoing challenge for optimization.

Our algorithms have performed well using movie and edge features. As for future extensions, we could incorporate features for reviewers as well, which is not the case in our case. Another extension could be adapting these algorithms to handle situations when it comes to heterogeneous graphs with more than one types of edges.

References

[Dong et al., 2017] Dong, Y., Chawla, N. V., and Swami, A. (2017). metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining, pages 135–144.

[Gao et al., 2022] Gao, C., Zheng, Y., Li, N., Li, Y., Qin, Y., Piao, J., Quan, Y., Chang, J., Jin, D., He, X., and Li, Y. (2022). A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Transactions on Recommender Systems*. Beijing National Research Center for Information Science and Technology (BNRist), Department of Electronic Engineering, Tsinghua University, China; School of Information Science and Technology, University of Science and Technology of China, China.

[Hamilton et al., 2017] Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[Morris et al., 2019] Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609.

[Reddy et al., 2019] Reddy, S. R. S., Nalluri, S., Kunisetti, S., Ashok, S., and Venkatesh, B. (2019). Content-based movie recommendation system using genre correlation. In *Smart Intelligent Computing and Applications: Proceedings of the Second International Conference on SCI 2018, Volume 2*, pages 391–397. Springer Singapore.

[Vilakone et al., 2018] Vilakone, P., Park, D.-S., Xinchang, K., and Hao, F. (2018). An efficient movie recommendation algorithm based on improved k-clique. In *Human-centric Computing and Information Sciences*, volume 8, pages 4602–4609.

[Zhang et al., 2019] Zhang, C., Song, D., Huang, C., Swami, A., and Chawla, N. V. (2019). Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 793–803.