Graph-Based Movie Genre Prediction: Unveiling Cinematic Patterns through Network Analysis

Sunishka Jain [*1] **Kate Spillane** [*1] **Atul Agarwal** [*1] **Angelic McPherson** [*1]

Abstract

The study of movies can have a major influence on society and can often be an enjoyable point of discussion. We investigate the intersection of cinematic storytelling and network analysis by conducting an examination of film narratives using the IMDb-MULTI (Rossi & Ahmed, 2015), IMDb-Binary (Yanardag & Vishwanathan, 2015) and Moviegalaxies (Kaminski et al., 2018) datasets. Our investigation prioritizes the extraction of pivotal features from the intricate web of character interactions, recognizing these elements as fundamental to understanding a movie's genre. Utilizing GraphSage, GIN (Graph Isomorphism Network), and GAT (Graph Attention Networks), we tune models for graph classification to determine movie genre for all three datasets. We then follow this with training and testing loops that enable us to benchmark our results against existing research. Central to our methodology is a focus on techniques including dropout for regularization, pooling to distill graph features, skip connections to enhance information flow, and custom layers tailored to capture the unique nuances of film narrative networks. Bridging the gap between graph theory and film analysis, our work contributes to the discourse on cinema's societal impact and showcases the potential of graph-based analyses in revealing storytelling's core dynamics.

1. Introduction

Movies are complex modes of cultural and artistic expressions with nuanced interactions. Unlike the relatively straightforward structures observed in fields such as chemistry or biology (e.g., molecule structures), movies present a complex challenge for classification due to their subjective nature and the common occurrence of multiple genres within a single film. Movie genre plays a crucial role in driving audience interest and thus, accurate genre classification is highly valuable.

The interconnected nature of movies, with their extensive networks of actors and characters, makes them particularly well-suited for graph-based analysis. Our research embarked on an iterative development process, meticulously fine-tuning models to handle graph classification tasks across the datasets: IMDb-MULTI with 1500 graphs and three genres; Moviegalaxies, with over 700 graphs and over 20 genres; and IMDb-BINARY with 1000 graphs and 2 genres.

We leverage GraphSage, Graph Isomorphism Network (GIN), and Graph Attention Networks (GAT) to conduct graph classification for genre prediction. Although there is various research regarding movie network analysis to examine popular actors, gendered interactions, or text analysis, these approaches center on node or edge classification. Moreover, there is a dearth of research regarding graph classification specifically for genre.

Through this approach, we aim to answer questions including which features are most indicative of genre, how do we navigate genre prediction in the highly subjective realm of cinema, and which models exhibit the highest efficacy for this application. Additionally, we investigate the potential of model combination techniques to enhance prediction accuracy. We ground our analysis by comparing our results to benchmarks from previous studies by a myriad of researchers (pap).

2. Related Work

We ground our approach in a range of scholarly papers. By identifying both strengths and prevailing gaps within the literature we are able to conduct a more rigorous study.

Chen et. al (Chen et al., 2019) identify the growing potential of graph neural networks and the concerning lack

[*]Equal contribution [1]Department of Computer Science, Dartmouth College, Hanover, NH, USA. Correspondence to: Sunishka Jain <sunishka.jain.gr@dartmouth.edu>, Kate Spillane <kate.m.spillane.24@dartmouth.edu>, Atul Agarwal <atul.r.agarwal.gr@dartmouth.edu>, Angelic McPherson <angelic.mcpherson.25@dartmouth.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

of understanding regarding these structures. The authors dissect GNNs and develop a framework which can surpass state of the art (SOTA) models. Chen et. al test their model on a variety of datasets including IMDb-MULTI.

Mourchid et. al (Mourchid et al., 2019) propose a multi-layer network model to delve into movie character interactions to provide a broader picture of film content. The authors extract information and perform a comparative analysis which creates future opportunities to characterize movie genres.

Xu et. al (Xu et al., 2018) delve into the strength of GNNs and their variants, discussing models GraphSage and GCN. The authors propose a powerful GIN model and compare it to several benchmarks, including those for IMDb-MULTI.

Nguyen et. al (Nguyen et al., 2022) introduces a novel approach to graph neural networks by incorporating a transformer self-attention mechanism and a recurrent transition, addressing the need for advanced aggregation functions in graph representation learning. Its application yields state-of-the-art accuracies across benchmark datasets like COLLAB, IMDB-M, and IMDB-B for graph classification.

Nouranizadeh et. al (Nouranizadeh et al., 2021) introduces MEWISPool, a novel graph pooling layer leveraging the Shannon capacity of a graph to maximize mutual information with the input, effectively framing graph pooling as optimizing information transmission across a noisy communication channel. MEWISPool outperforms existing methods.

3. Proposed Method

We first look at IMDb-MULTI. The node-level features extracted in this process included:

1. Clustering Coefficient: describes the community potential of a node. Nodes with high clustering coefficients may form a community within the graph.
2. Eigenvector centrality: measures the importance of a node in a network. Nodes with a higher eigenvector centrality score are well-connected in general, but also well-connected to other important nodes.

In addition to node-level features, we also consider graph-level attributes. The Graph Level attributes extracted in this step included:

1. Number of Triangles: We looked at the number of triangles and then normalized these values to gauge the presence and intensity of tightly-knit clusters within the graph.
2. Number of Communities: a reflection of the distinct character groups driving the plot forward. This aspect of our analysis sought to uncover how genres differ in their storytelling approaches.
3. Other Graph Features: We also analyze connectivity by looking at the average shortest path length. This offered clues about the narrative's flow and the ease with which characters influence each other. Additionally, the ratio of nodes to edges served as an indicator of the graph's sparseness and connectivity.

Next, we focus on using the existing node features to further refine them and get more such features. We do this by using the **GraphSage Model**. We believe that this provides better features to describe our nodes, which could be helpful when we use Message Passing Algorithms to classify our graphs. While the ideal use case for GraphSage does not fit our use case, we use it more as a pre-processing function to get more refined node feature embeddings. The architecture used for Graph Classification needs to learn both the Graph Features and the Node Features. The message-passing layers in the GAT and the GIN models make use of these features to learn the graph structures. We use GraphSage to get better node attributes for this purpose. The proposed GraphSage architecture consists of two layers that collect information from the immediate neighbors of a node. Hence, by using two layers, we essentially aim to collect information from the two-hop neighborhood of a given graph. The GraphSage architecture used consists of two layers, following which we take the mean of the embeddings, to capture the overall embeddings from the given node's embeddings. We believe, that by using GraphSage to get node embeddings, we can improve the performance of the Graph Classification model as it provides a better set of node embeddings to learn the overall graph structure. The node features evaluated by GraphSage (four) are added to the existing node features (two) hence giving six features for every node.

Furthermore, we calculate the edge weights for the given graphs. The datasets do not provide any refined information for this, hence, we experimented with **Pearson Correlation Coefficient** and **Cosine Similarity** to quantify the relation between two connected nodes based on the node features we calculated and the ones we obtained from GraphSage. Based on the experiments we conducted, Cosine Similarity provided more refined results, hence, that is chosen as the edge weight metric.

We propose a **Hybrid GIN-GAT Model** that combines both GIN and GAT Models. These message-passing-based architectures provide an improved performance for the task of Graph Classification. The model structure first passes the input mode and graph features through the GIN layer, giving refined features that contain information about a

given node's neighborhood. This is given by,

$$H_v^{l+1} = MLP^{(l)}((1+\epsilon^{(l)}.H_v^l) + \sum_u H_u^l) \quad (1)$$

Where, H_v represents the features for the given node, ϵ is a learnable weight and H_u represents the features of all neighbors of node v at layer l. Next, these features are passed through the GAT Layers that use an attention mechanism to learn weights to identify the importance of each node feature. The attention coefficient for a node pair is given by:

$$\alpha_{uv}^{(l)} = \frac{\exp\left(\text{LReLU}\left(\mathbf{a}^\top \left[W^{(l)}H_u^{(l)} \parallel W^{(l)}H_v^{(l)}\right]\right)\right)}{\sum_k \exp\left(\text{LReLU}\left(\mathbf{a}^\top \left[W^{(l)}H_u^{(l)} \parallel W^{(l)}H_k^{(l)}\right]\right)\right)} \quad (2)$$

Where, LReLU, is the Leaky Relu Function and W^l represents the learnable weight vector and k represents the Neighborhood of node u. Next, this learned attention coefficient is used to update the node features. This is given by,

$$H_u^{(l+1)} = \sigma\left(\sum_v \alpha_{uv}^{(l)} W^{(l)} H_u^{(l)}\right), \quad (3)$$

Where σ is the activation function ELU. Finally, these learned features are all aggregated using summation, where the node features and graph features are combined. This gives the final graph feature vector.

$$h' = \text{ELU}(\text{Batch Norm}(W_{\text{fc1}}[h_G \parallel g_G] + b_{\text{fc1}})) \quad (4)$$

$$\hat{y} = W_{\text{fc2}}h' + b_{\text{fc2}} \quad (5)$$

This graph representation vector is passed through the **Log Softmax** function to get the final classification prediction. Finally, we use **Accuracy** to evaluate our model's performance.

4. Experiments and Analyses

4.1. Datasets

4.1.1. IMDB-BINARY

IMDB-BINARY is a movie dataset of 1,000 actors/actresses who played roles in movies in IMDB. In each graph, nodes represent actors/actress, and there is an edge between them if they appear in the same movie. These graphs are derived from the Action and Romance genres. We have three files, IMBD-BINARY.edges, IMDB-BINARY.graph_idx, and IMDB-BINARY.graph_labels. IMDB-BINARY.edges contains a list of pairs in the format [node1, node2]. There are 386,124 pairs in this file. IMDB-BINARY.graph_idx labels which node belongs to which graph. There are 19,773 nodes. IMDB-MULTI.graph_labels gives each graph a binary genre either 1 or 2 corresponding to the values of Action or Romance. The data is balanced.

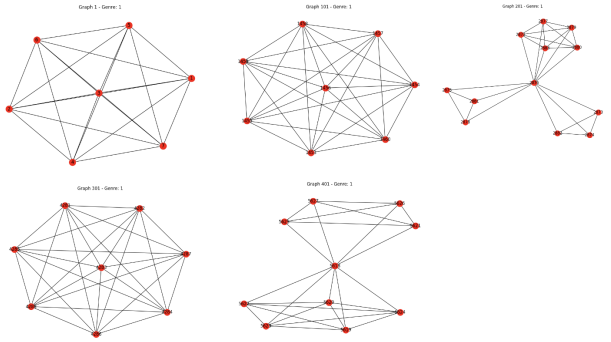

Figure 1. Genre 1 Graphs

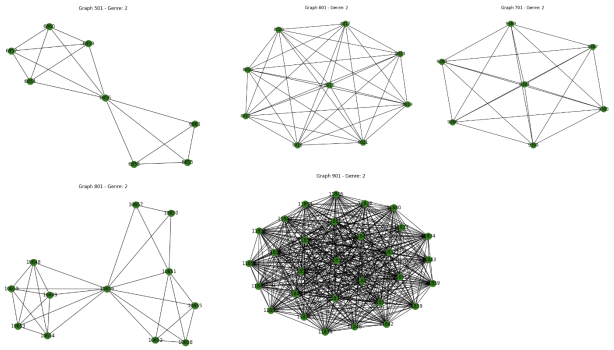

Figure 2. Genre 2 Graphs

4.1.2. IMDB-MULTI

IMDb-MULTI is a relational dataset that consists of a network of 1000 actors/actresses who played roles in movies in IMDb. A node represents the actor/actress and an edge connects two nodes when they appear in the same movie. There are 1500 graphs, each representing a movie network. There are 395,612 edges, 19,502 nodes, and genres between 1 and 3 corresponding to the values of Comedy, Romance, or Sci−Fi. The data is balanced with 500 graphs in each genre. We provide visualizations for a sample of the diverse graphs as show in Figures 1, 2, and 3. We obtain several graph statistics, including average clustering coefficient, node degree, number of triangles, path length, density, and diameter for each genre as seen in Table 1. We also obtain the general distribution of these metrics as show in the violin plots in Figures 8, 9. Additionally, for each genre we plot degree distribution.

4.1.3. MOVIEGALAXIES

In the Moviegalaxies (Kaminski et al., 2018) dataset graphs, nodes represent characters while edges represent interactions and connections between them. Each graph represents a movie and there are 773 graphs. To load this data, we have a gexf folder containing files in the format [gexfID].gexf. Each gexf file is a graph and contains infor-

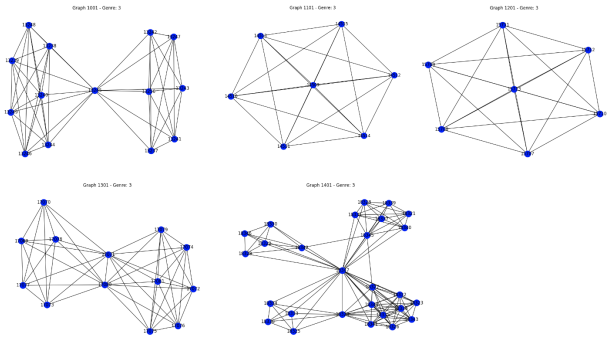

Figure 3. Genre 3 Graphs

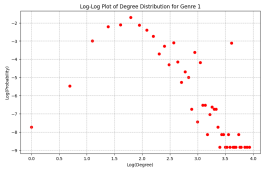

Figure 4. Genre 1 Distribution

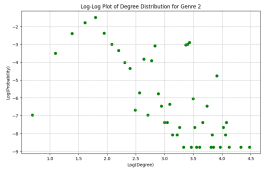

Figure 5. Genre 2 Distribution

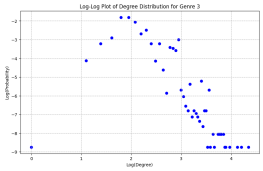

Figure 6. Genre 2 Distribution

Figure 7. Degree Distributions

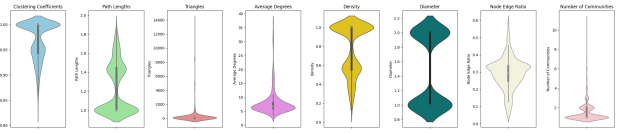

Figure 8. Violin Plots

Table 1. Average Graph Metrics by Genre for IMDb-MULTI

Metric	Genre 1	Genre 2	Genre 3
Avg Clustering Coefficient	0.963	0.971	0.973
Avg Node Degree	7.771	8.049	8.483
Avg Number of Triangles	275.3	430.276	212.12
Avg Path Length	1.283	1.234	1.166
Avg Density	0.717	0.766	0.834
Avg Diameter	1.56	1.478	1.384
Avg Node Edge Ratio	0.302	0.309	0.265
Avg Number of Communities	1.716	1.556	1.262

mation regarding node centrality attributes, character labels, edge weights, visualization attributes, degree measures, eccentricity, and modularity. The dataset also contains a network_metadata.tab with various information regarding ID, movie name, release date, and so forth. The data is balanced. The Moviegalaxies dataset does not contain the movie genres but it does contain the IMDb ID. We utilize the Cinemagoer package which enables users to access more in-depth information including genres based on the ID. We gather this data to label our graphs.

4.2. Baselines and Evaluation Metrics

As a baseline, we utilize the PapersWithCode website which displays graph classification results on the dataset from over 30 models. We utilize accuracy because it is a straightforward, simple calculation that provides a clear picture of how often we are achieving the correct prediction. Moreover, our baselines utilize accuracy as a metric and thus this makes comparison easy. The datasets are balanced so this also makes accuracy an appropriate metric.

4.3. Models

We start with a standard GAT. We consider node embeddings and concatenate with graph level features to then pool the information. Since node embeddings are a substantial part of our model, we build upon our foundation by utilizing GraphSage with our GAT and also combining the model with GIN. We implement a standard train/test split and pipeline to evaluate our model. Working with the information we have gained from our exploration of IMDb-MULTI, we then process the Moviegalaxies data. We first run the same IMDb-MULTI pipeline on Moviegalaxies. Then, we adapt a model to incorporate the Moviegalaxies edge weights by creating custom module and layers to normalize the edge weights. We also have a custom GAT convolutional layer to handle edges as well. We run some of our models on IMDb-BINARY for testing purposes.

4.3.1. Graph Attention Network

Our first set of experiments involved using the simple GAT Network to classify our graphs. Graph Attention Networks (GATs) are suited for graph classification tasks because of their innovative use of attention mechanisms, which enable the model to dynamically assign significance to the nodes based on their relevance to the task at hand. The model architecture involved passing the node features through GAT Convolutional Layers, which are essentially message-passing layers. These layers learn attention coefficients for the node features across different heads and combine these learned coefficients by concatenating them for the first, or intermediate layers and summing them for the final layer. The idea is to use these features to learn unique graph structures. This model only uses the Node Features, which are Eigen-

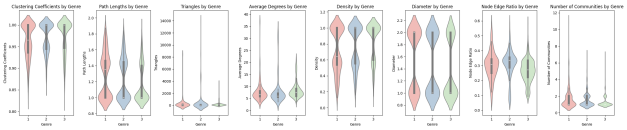

Figure 9. Violin Plots

vector Centrality and Clustering Coefficients. The input layer has 7 output features and 7 heads. The intermediate layers have the same number of input and output features (49) and the final layer has 49 input features and 16 output features. We also utilize dropout, which is a regularization technique to prevent overfitting. In dropout, we randomly drop a certain percentage of nodes at each iteration. We also use pooling methods to aggregate the information for graph classification.

We start by using a GAT Network as it is a simple yet powerful message-passing-based model that can capture unique graph representations and provide a meaningful starting point.

4.3.2. Graph Isomorphism Network

The reason for incorporating GIN into our architecture is to improve its ability to distinguish between different graph structures. The GIN layer works by using the Multi-Layer Perceptron model for updating node embeddings. The Multilayer Perceptron takes in information from the neighborhood of the node. The GIN layer aggregates messages by summing them instead of concatenating them as the GAT model. The GIN layer takes the node features as its input and passes them through its MLP-based layer architecture to get refined node features, which are then passed to the GAT layers. This helps ensure that the model learns how to differentiate between similar graph structures, a task at which GAT alone may not perform at the same level.

4.3.3. Other Layers

We also experimented with other different kinds of message-passing layers with the objective of capturing meaningful information about the Graph Structures. These layers included:

1. Transformer Layers: We believed this layer failed to help improve the performance of the model because it is more suited to learning long-range similarities and relations between nodes of a graph which does not apply to our given dataset. Our datasets consist of smaller, simpler graphs and it is much more important for us to learn about the immediate neighbors of a given node rather than its extended neighborhood.

2. GATv2Conv Layer: The GATv2Conv Layer aims to improve the performance of the existing GAT Convolutional layer in the PyTorch Library, by introducing dynamic attention. However, we believe that this failed to help improve performance in our case as our graph data consists of small, simple graph structures. GATv2Conv usually improves performance over large graphs.

3. AGNNConv Layer: The Attention-based GNN Layer helps learn attention weights for more heterogenous nodes. However, we believe that our graphs exhibit homophily. Hence this layer failed to improve the performance of our model.

Finally we have five different models that we experimented with. All of these models either used just GAT layers or a combination of GAT and GIN Layers. The final model, which provided the best results is described above in Section 3. The other models are given below:

4.4. Simple GAT Model: GAT

We used a simple GAT Model to start off with. The model architecture is described above in subsection 4.3.1. The model provided a good baseline to start off with. However, it is worth noting that the best accuracy score for this model did not cross 30% for classification over three labels, which is worse than randomly predicting labels for the graphs. This model only uses the Node Features such as the Eigenvector Centralities, Clustering Coefficients and the features evaluated using the GraphSage Model and the Graph Features such as the number of triangles, etc. It does not consider the edge weights.

4.5. GAT with Edges

The next obvious step for us was to incorporate the edge weights into the GAT model. Hence, this model makes use of all the available features, node, graph, and edge. This improved the performance of the model slightly.

4.6. Hybrid GIN GAT Node Features Only

We also want to consider the combination of GIN and GAT which accounts for different graph structures and provides a more nuanced approach to evaluate the graph labels. By using the benefits of a GIN Model, which improves the ability of our model to differentiate between different kinds of graph structures, by employing an MLP-based architecture, we aim to improve the performance of the GAT Model. Hence, our Hyrid model now has a GIN layer as it first layer, followed by a set of GAT Convolutional layers, which keep concatenating node features from the neighborhood. This hybrid approach improved the performance of the model slightly over the previous approach.

4.7. Hybrid GIN GAT Node Features and Graph Features: GIN GAT Plus G

Next, we begin experimenting with passing the graph features into the model too. This helps improve the performance as we are working on a Graph Classification problem and learning good features that describe the graph structure is important. To incorporate the graph features into the model, we concatenate them with the learned graph description feature vector by the model and calculate the final logits based on this updated feature vector. Doing this greatly improves the performance of the model, and helps us cross the baseline.

4.8. Hybrid GIN GAT with all features: GIN GAT Plus GE

Our final approach combines features for all, nodes, graphs, and edges to learn the final graph representation feature vector. The edge weights improve the quantification of the relations between the nodes, which is very important for such a dataset. We evaluate the cosine similarities between the nodes as the edge weights. The edge weights are incorporated into the hybrid model, by using them in the GAT Convolutional layer. This layer learns attention coefficients for every node pair, hence we add the edge weights to this calculation which is the objective of using this weight to improve the coefficient. This further improves the performance of the model and occasionally provides over 50% accuracy for multi-label classification.

4.9. Labeling of Moviegalaxies Data

We gather genre information for each movie in Moviegalaxies in a way that keeps the dataset balanced. The goal is to ensure an equal representation of three selected genres ('Comedy', 'Romance', and 'Sci-Fi') in the dataset. We drop necessary genres and then calculate the minimum count by using the .sum function to calculate the sum of the values in each of these columns, and then using .min to get the minimum among these sums. For each genre, we select a random subset of instances from the dataset, ensuring that the size of the subset is equal to the minimum genre count. Finally, we assign the chosen genre to each movie.

4.10. Loss and Optimization

We consider both cross-entropy loss as well as negative log-likelihood loss. Cross Entropy Loss penalizes predictions that are confident but wrong, encouraging the model to output probabilities that are closer to the actual distribution. NLL loss is closely related to cross-entropy loss and it directly penalizes the model for being uncertain about the correct classification. We use the Adam optimizer and experiment with the learning rate to obtain optimal results. Additionally, we looked at the amsgrad Adam parameter. AMSGrad improves convergence problems in Adam's original implementation.

4.11. Dropout

Dropout addresses overfitting by randomly deactivating a subset of neurons in the network during training, effectively thinning the network temporarily. By preventing complex co-adaptations on training data, dropout encourages the model to become more generalized, improving its performance on unseen data.

Choosing the right dropout rate is crucial for balancing between underfitting and overfitting. After extensive testing, we determined that a dropout rate of 0.6 was optimal for our task.

4.12. Attention Heads

We also delve into selecting the number of attention heads. Each attention head can be thought of as an independent feature extractor that focuses on different parts of the input data. By aggregating the outputs of multiple attention heads, the model can integrate diverse perspectives, leading to a more comprehensive understanding of the data. The choice of the number of attention heads involves a trade-off between predictive capacity and computational complexity. The number of heads in each model was determined through robust testing and is specific to each model.

4.13. Pooling and Batch Norm

Pooling mechanism significantly impacts the model's ability to capture and summarize the essential features of the graph. We experiment with mean, add, and max pool. By normalizing the inputs of each layer to have a mean of zero and a variance of one, BatchNorm ensures a consistent distribution of inputs across layers throughout training.

4.14. Findings

From our work, we extract several key findings that shed light on the interactions between movies and their genres.

4.14.1. Features

Our analysis revealed that node-level features, such as eigenvector centrality and clustering coefficients, significantly contribute to genre classification. However, the incorporation of additional graph and edge features plays a pivotal role in enhancing a model's performance on classification tasks.

Models solely relying on node features, such as GAT and HybridGINGAT, underperformed by approximately 10 percent compared to our best-performing model, HybridGIN-

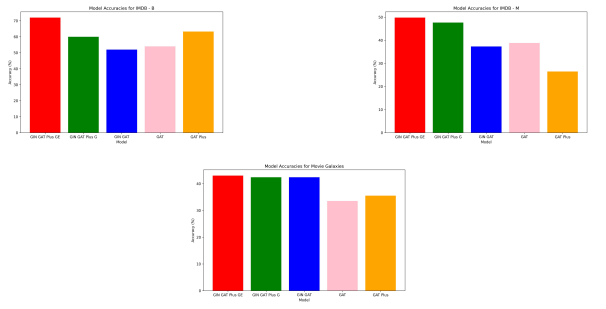

Figure 10. Accuracies

GATPlusGE, which leveraged both graph and edge features.

Eigenvector centrality highlights key characters within the movie's network, while clustering coefficients identify tightly-knit communities of characters, proving valuable in genre differentiation. These node features underscore the significance of character dynamics and interactions in defining a movie's genre. However, when not complemented by other features, these models fail to grasp a clear understanding of the graph structure, which hinders genre prediction accuracy.

The inclusion of edge features demonstrates the need for extra features. As reflected in majority of our results, when edges are added to most models, the accuracy improves. The determination of the edges by computing the cosine similarity of the nodes allows the model to pick up on the similarity measures, revealing further insights into the relevant structural information of the graph.

The inclusion of graph features, as detailed in the proposed methods section, proved indispensable in augmenting our model's ability to accurately classify genres by providing pertinent structural insights about the graph. These features encompass normalized triangles, number of communities, node-to-edge ratio, and average path length-to-diameter ratio, offering insights into clustering potential, density, sparseness, and connectedness. This reveals potential information about subplots, character groupings, and main plotlines of the movie. Consequently, our model gains a more comprehensive understanding of the graphs and can predict genre more accurately.

4.14.2. Model Performance

When developing our project, we saw that GAT alone was not enough to achieve satisfactory results. While Graph Attention Networks (GAT) are powerful for leveraging node features and their dependencies, GAT may not fully capture the complexity of movie genre classification from character interaction networks. This limitation stems from GAT's primary focus on node features and pairwise node interactions, which might not sufficiently account for the broader and more intricate graph structures. Incorporating GraphSage and Graph Isomorphism Network (GIN), alongside models that consider edge weights with custom layers, can significantly enhance the model's capability. GraphSage, with its ability to aggregate features from a node's local neighborhood, helps in capturing local structural information. GIN distinguishes subtle differences in graph topology. Considering edge weights through custom layers allows the model to differentiate between the strength and significance of interactions among characters, an essential factor in understanding narrative dynamics. This combination offers a comprehensive approach by integrating local and global graph features, thereby providing a richer and more nuanced understanding of the graph that aligns more closely with the complex nature of movie genres. We compare the performance of our model with other baseline datasets in Table 3. The other baseline model performances are obtained from the paper (Zhao & Wang, 2019)

As the models under consideration are the first of their kind to be applied to the Moviegalaxies dataset, there are currently no established benchmarks for direct comparison.

4.14.3. Fine Tuning

We delved into the nuances of model optimization, focusing on aspects such as batch normalization, pooling mechanisms, learning rate adjustments, and the choice of optimizer. Our findings highlight the pivotal role these elements play in enhancing model performance and accuracy. Batch normalization emerged as a critical factor in stabilizing training across deep networks by standardizing the inputs to each layer, thereby allowing for higher learning rates and faster convergence without the risk of significant overfitting. The exploration of various pooling strategies, including mean, max, and add pooling, revealed their importance in effectively summarizing the graph's features into a comprehensive representation crucial for genre classification. Adjusting the learning rate and utilizing the Adam optimizer, with its adaptive learning rate capabilities, further refined our models, enabling nuanced control over the training process and optimization landscape. These adjustments, in concert, significantly contributed to our models' ability to discern complex patterns within movie character interaction networks.

4.15. Limitations

One notable limitation of the presented study is the inherent bias toward certain types of films, a consequence of relying on datasets like IMDb-MULTI and Moviegalaxies. These datasets are predominantly films that have gained more visibility and recognition on platforms like IMDb, potentially skewing towards Hollywood-centric cinema.

Table 2. Accuracies for Movie Galaxy on the discussed models

Model	Movie Galaxy Accuracy (%)
GIN GAT PlusGE	43.0±0.6
GIN GAT PlusG	42.4±2.1
GIN GAT	42.4±2.4
GAT	33.5±3.1
GAT Plus	35.4±1.3

Table 3. Comparison of accuracies on IMDB-BINARY and IMDB-MULTI datasets.

Model	IMDB-B Accuracy (%)	IMDB-M Accuracy (%)
RetGK	71.9±1.0	47.7±0.3
WL	70.8±0.5	49.8±0.5
PF	71.2±1.0	48.6±0.7
PSCN	71.0±2.3	45.2±2.8
WKPI-kM	70.7±1.1	46.4±0.5
WKPI-kC	75.1±1.1	49.5±0.4
GIN GAT PlusGE	72±0.3	49.83±0.2
GIN GAT PlusG	60±0.2	47.67±0.1
GIN GAT	52±0.1	37.33±0.1
GAT	54±0.2	38.33±0.1
GAT Plus	63.25±0.1	26.50±0.1

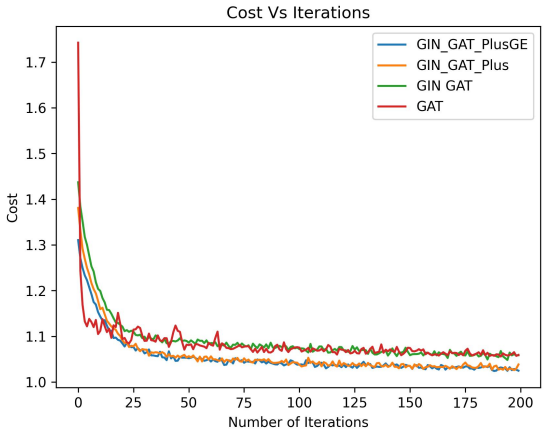

Figure 11. Cost History

Another challenge was a lack of information, particularly in the case of IMDb-MULTI which only had three genres. Since there was not a connection to an IMDb ID we were unable to extract richer features such as release year, rating, cast, or plot. Nonetheless, this dataset provided a basic foundation upon which we could expand with Moviegalaxies. Moviegalaxies had over 20 classes which also allowed more complex analysis.

5. Future Work

The subjective nature of movies also presents both a challenge but also an opportunity for future research. Genres often blend and overlap. In the future, we would like to expand upon our work by investigating multi-label classification, where a movie can simultaneously belong to multiple genres. This provides a new challenge that would necessitate more complex models and interactions.

6. Conclusions

Overall, we show that by combining several simple GNN models in an organized method can be used to achieve excellent results that can be compared with other more complex architectures. This requires a thorough understanding of the distribution of the structures in the given dataset and how to extract the most ideal information from them. Our exploration into graph-based movie genre prediction has revealed the rich potential of applying network analysis techniques to cinematic storytelling. By combining GraphSage, GIN, and GAT, as well as feature extraction and model architecture, we have navigated the complex landscape of film narratives. Our findings underscore the significance of interactions and the structural nuances of movie networks in genre classification, revealing key insights into how movies communicate their genres through the web of dynamics.

This research not only advances our understanding of graph-

based models for genre prediction but also contributes to the broader discourse on the intersection of artificial intelligence and film studies. The challenges encountered, particularly the limitations of existing datasets and the complexity of genre as a concept, pave the way for future investigations. Expanding this work to include multi-label classification and exploring the integration of richer feature sets and more diverse datasets stand out as promising directions.

Moreover, the implications of our findings for recommender systems and content discovery platforms highlight the practical value of this research. By enhancing the accuracy and personalization of movie recommendations, this work has the potential to transform viewers' engagement with cinematic content, offering a more nuanced and satisfying exploration of the vast landscape of film.

As we continue to refine our models and expand our datasets, the future of movie genre classification looks promising, with the potential to deepen our appreciation for movies as complex, narrative-driven art forms.

7. Division of Work and Individual Contribution

In our collaborative effort to explore graph-based movie genre prediction, each team member played a pivotal role, contributing equally to the research. Atul focused on experimenting with GraphSage and exploring its applicability to the IMDb-Binary dataset. He also integrated GIN with GAT to enhance model performance. Angelic played a crucial role in synthesizing our efforts into a cohesive pipeline by fine-tuning the IMDb-MULTI models, integrating GraphSage, and running experiments. Angelic's contribution also extended to meticulously labeling the Moviegalaxies datasets and conducting preliminary data preparation and visualization. Kate experimented with the initial GAT structure and refined this approach to the IMDb-MULTI and Moviegalaxies models through the use of more complex GATs and custom layers tailored for edge attributes. She tuned models and explored various parameters to optimize the results. She also contributed to data preparation of both IMDb-MULTI and Moviegalaxies as well as the production of visualizations. Sunishka's primary focus encompassed experimenting with diverse models for IMDb-MULTI and Moviegalaxies. Employing GAT and GraphSage, she delved into the datasets, conducting temporal analysis and exploring the One-vs-Rest (OVR) method. She also experimented with GraphSage layers. Beyond that, Sunishka actively sought additional datasets to broaden our analysis scope. Her dedicated efforts were directed at fine-tuning the pipeline, optimizing our overall research methodology.

References

Papers with code: The latest in machine learning. https://paperswithcode.com/.

Chen, T., Bian, S., and Sun, Y. Are powerful graph neural nets necessary? A dissection on graph classification. *CoRR*, abs/1905.04579, 2019. URL http://arxiv.org/abs/1905.04579.

Kaminski, J., Schober, M., Albaladejo, R., Zastupailo, O., and Hidalgo, C. Moviegalaxies - Social Networks in Movies, 2018. URL https://doi.org/10.7910/DVN/T4HBA3.

Mourchid, Y., Renoust, B., Roupin, O., Van, L., Cherifi, H., and Hassouni, M. E. Movienet: A movie multilayer network model using visual and textual semantic cues. *CoRR*, abs/1910.09368, 2019. URL http://arxiv.org/abs/1910.09368.

Nguyen, D. Q., Nguyen, T. D., and Phung, D. Universal graph transformer self-attention networks. In *Companion Proceedings of the Web Conference 2022*, WWW '22, pp. 193–196, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391306. doi: 10.1145/3487553.3524258. URL https://doi.org/10.1145/3487553.3524258.

Nouranizadeh, A., Matinkia, M., Rahmati, M., and Safabakhsh, R. Maximum entropy weighted independent set pooling for graph neural networks. *ArXiv*, abs/2107.01410, 2021. URL https://api.semanticscholar.org/CorpusID:235731922.

Rossi, R. A. and Ahmed, N. K. The network data repository with interactive graph analytics and visualization, 2015. URL https://networkrepository.com.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *CoRR*, abs/1810.00826, 2018. URL http://arxiv.org/abs/1810.00826.

Yanardag, P. and Vishwanathan, S. V. N. Deep graph kernels. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015. URL https://api.semanticscholar.org/CorpusID:207227372.

Zhao, Q. and Wang, Y. Learning metrics for persistence-based summaries and applications for graph classification. *Advances in Neural Information Processing Systems*, 32, 2019.