Unveiling the Black Box: Exploring Explainable AI through Graph Neural Networks on the IMDB-Binary Dataset

Lunay Cicek and Michael Reißmüller

University of Paderborn, Germany

Abstract. This report explores Explainable AI (XAI) through the use of Graph Neural Networks (GNNs) for sentiment analysis on the IMDB-BINARY dataset. Our objective is to build a model that achieves high classification accuracy while providing interpretable explanations for its predictions. We outline the methodology for data preprocessing, GNN model architecture, and training procedures. Experimental results demonstrate the model's performance and the quality of its explanations.

1 Introduction

1.1 Motivation

In recent years, Artificial Intelligence (AI) has achieved remarkable advancements in various fields, ranging from healthcare to finance, revolutionizing the way we live and work. As AI systems become increasingly complex, their decision-making processes often become difficult to interpret, leading to a lack of transparency and trust. This increases the demand for Explainable AI (XAI), which aims to provide human-understandable explanations for AI system outputs. [1]

One promising approach to XAI is the utilization of Graph Neural Networks (GNNs), which have emerged as powerful tools for processing and analyzing structured data. GNNs model relationships between entities represented as nodes in a graph, enabling them to capture intricate dependencies and reveal hidden patterns within the data. By using GNNs, we can shed light on the decision-making process of AI systems and offer insights into the reasoning behind their predictions. [2]

1.2 Goals

In this report, we delve into the realm of XAI by creating and testing a Graph Neural Network on the IMDB-BINARY dataset. The IMDB-BINARY dataset, derived from the Internet Movie Database (IMDB), contains 1000 graphs each representing the ego-network of one actor/actress, with additional edges between actors if they appear in the same movie. Each graph is classified as being either in the Romance or Action genre.

Our primary objective is to build a GNN-based model that not only achieves a high classification accuracy but also provides interpretable explanations for its predictions. Through this project, our main goal is to improve the transparency of GNN models by understanding the model's predictions and defining features it recognizes in the data.

1.3 Structure

The report is structured as follows: In Section 2, we analyze the IMDB-BINARY data set. Section 3 describes the methodology for training and evaluating the GNN model, including data preprocessing, model architecture, and training procedures. The explanation of model predictions, graph features and possible interpretation are done in Section 4. Section 5 includes a summary of our key findings and further outlook.

2 Data analysis

The following section focuses on exploring the data set and gaining more insight into it in order to deliver reasonable, probable and understandable explanations for the GNN model's predictions. We start with an in-depth analysis of the general structure, before diving deeper into more intricate statistics.

2.1 General structure

Reiterating, the IMDB-BINARY data set, which we loaded with PyTorch Geometric [3], represents 1000 graphs each showing the ego-network of one actor/actress. Nodes in the graph represent actors and edges in the graph indicate that two actors appear in the same movie. Each graph in the dataset belongs to either the Action or the Romance genre, with movies belonging to both genres having been removed. Graphs contain ground truth labels of either 0 or 1, even though based on our research, it is unclear which label corresponds to which of the two genres. The dataset contains an equal amount of 0 and 1 graphs, does not contain any node or edge features and each graph in the dataset includes at least one movie. There is also no information about which actors or movies were used to generate the dataset, which is why little information other than the connections of and overlap between actors can be used to extract information from a given graph.

2.2 Statistics

Intuitively, a graph does not contain a lot of information that the model can use. Given such simple graphs, one could assume that computing various statistics could produce different results for 0 and 1 graphs. Calculating the mean, median and standard deviation for the number of nodes and edges shows little difference between 0 and 1-graphs however. Notable statistical results include a slightly

lower median number of nodes and edges for 1-graphs and a slightly higher standard deviation for 0-graphs. This leads to the assumption that the network learns mainly based on the specific configuration of nodes and edges in the graph.

3 GNN Training and Evaluation

3.1 Training Pipeline

Because the 1000-graph long dataset does not contain any node features, we first initialize the node features with the one hot encoding of their degrees. We then divide the 1000-graph long dataset into a training set of 700 graphs and a testing set of 300 graphs. Each set contains an equal amount of 0 and 1 graphs. The model is trained on the data for 50 epochs with a batch size of 50 and a learning rate of 0.02. The training set is shuffled at the start of each epoch. For each batch, we simply calculate the binary cross-entropy loss of the model's prediction and the ground truth label and then perform standard backpropagation and update the network's parameters.

3.2 Network Structure

We opted for a simple Graph Convolutional Network [4], where we use two of PyTorch Geometric's graph convolutional layers followed by one ReLU-activation layer each. We then apply a pooling layer (which is especially important due to varying numbers of nodes for each batch of graphs) followed by a dropout layer with a value of 0.5. We then use a linear layer to reduce the dimensionality of the data to one and finally calculate the network prediction with a sigmoid layer.

3.3 Parameter and Network Tuning

To reduce the time it takes to train the model, we chose a high batch size, a fairly low number of epochs and higher (but not too high) learning rate for faster convergence. We observed that training on more epochs (e.g. 75) was less accurate due to overfitting, whereas training on less epochs (e.g. 25) did not give the model enough time to learn. The network structure was a lot more stable to change but we found that two graph convolutional layers and a dropout value of 0.5 created the best results.

3.4 Evaluation

Evaluating the trained model on the training set results in an accuracy of around 0.74. We calculate the accuracy by setting a probability threshold of 0.5 and casting predictions above that threshold to 1 and vice versa. We also verified that the majority of the network's predictions are close to 0 or 1, since casting predictions like 0.51 or 0.49 to the corresponding labels might not be as meaningful. It is worth noting that PyTorch's initialization of the network weights and/or how exactly the dataset is shuffled every epoch can decrease the model's accuracy by up to 10%.

4 Explanation of the GCN model

As mentioned before, the IMDB-BINARY dataset does not contain any node or edge features, which is why we use the one hot encoding of nodes' degrees as the sole feature to be able to perform training. But because the degree is the only feature and only contains information about the graph's connectivity (and no information about the nodes), feature importance seems to be just a small criterion for our explanation. Using PyTorch Geometric's GNNExplainer [5] on the data and model and then analyzing the feature importance supports this hypothesis: the feature importance returned by the explanation just sums up the degrees in the graph. Using the GNNExplainer's graph visualization function yielded more interesting results. While the dataset does contain information about the edges in a graph, which signify that two actors (nodes) appear in the same movie, it is difficult to see how many different movies there are in a given graph. The explainer's visualization makes it easier to see said movies, because it groups actors belonging to the same movie closer together, i.e. it clusters the nodes. The GNNExplainer's main function, in our case, is to present the graphs in a more readable and understandable way. Our strategy for model explanation is now to modify graphs, observe how the model's predictions change, discover graph features that the model uses and, finally, test our discovered features by approximating the predictions in a decision tree-like format.

4.1 Graph modification

In order to modify the data, we have created a custom graph class that, without changing the data, changes the shape of the data in order to perform operations such as edge removal or node removal and addition. This graph class translates the edges into movies (much like the explainer does when clustering nodes in its visualization), which proved essential for our observations and approximations. We also added a custom graph class that creates symmetric graphs, such as single movie graphs (all nodes in the graph are connected to all other nodes) or single central node graphs (SCNG) (a single node in the center is connected to all groups of nodes in the graph, i.e. the actor's ego network contains no overlap). We have experimented with both symmetric graphs as well as modified graphs from the dataset.

4.2 Graph Features

After testing how graph modifications change the model's prediction we were able to extract the following relevant graph features.

Single Movie Graphs (SMG). In single movie graphs, the graphs contain no information other than the size of the cast (i.e. size of the movie). We observed that given a fixed value for the cast, the model outputs a fixed prediction. In the cast range of 3-7, the model outputs 1. In the range 8-50, there is high variance

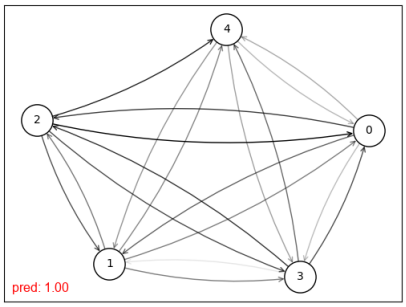

Fig. 1. A single movie graph with model prediction 1.00.

but the model has an ever so slightly greater tendency to output 1. In the range 50+, the model outputs values close to 0. Summarizing, a low cast seems to be a predictor for 1-Graphs and a high cast seems to be a predictor for 0-Graphs.

SCNGs with same cast sizes. SCNGs contain at least two movies (if they contained just one movie, they would be an SMG). In SCNGs where each movie cast is of the same size, we observed that removing any number of movies from the graph does not change the model's prediction. Therefore, every SCNG with same cast sizes across movies can be transformed into an SMG and the fixed prediction can be calculated easily.

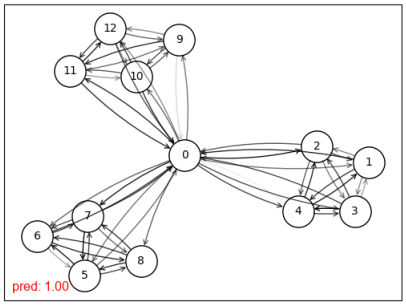

Fig. 2. A single central node graph with model prediction 1.00.

SCNGs with differing cast sizes. Unlike the previous case, removing movies from these types of SCNGs can alter the prediction. We hypothesize that the

model calculates the prediction p with a formula as such:

$$p(graph) = c_i * f(cast_i) + c_{i+1} * f(cast_{i+1}) + ... + c_n * f(cast_n) \quad (1)$$

where n is the number of movies in the graph, c is some constant that weighs each movie, *cast* is the size of the movie and f is a function that given an input cast returns a fixed output prediction. We believe this to be the case because weighing each movie equally (by setting c to 1/n) and calculating the above function yields a correct approximation of the model's prediction in a lot of cases. The predictions of same cast SCNGs could also be explained this way:

$$\begin{aligned} p(SCNG) &= c_i * f(cast_i) + c_{i+1} * f(cast_{i+1}) + ... + c_n * f(cast_n) \\ &= (1/n) * f(cast_i) + (1/n) * f(cast_{i+1}) + ... + (1/n) * f(cast_n) \\ &= (1/n) * (f(cast_i) + f(cast_{i+1}) + ... + f(cast_n)) \\ &= (1/n) * (n * f(cast)) \\ &= f(cast) \end{aligned}$$

Messiness. We consider a graph to be messy if it has many movies and a large amount of nodes that are entangled in various ways with each other, i.e. the graph doesn't have clear clusters. A graph can be transformed into a messy graph by removing edges. That is because removing an edge from a graph signifies that two actors are no longer in the same movie. This means that for each movie that both of the actors are in, the movie has to be divided into two new movies: one where the first actor is not in and one where the second actor is not in. Increasing the number of movies spreads apart the clusters the explainer visualizes and makes the graph look more entangled and less readable. While observing the model's predictions we realized that messiness seems to be a predictor for 0-Graphs and cleanness seems to be a predictor for 1-Graphs. This feature is in line with the cast size feature illustrated before, because graphs with one movie or few movies and few nodes cannot be messy.

4.3 Approximation

To test the general validity of our feature hypotheses, we approximated the model's prediction in a decision-tree like way (Alg. 1).

Using this algorithm, we could approximate the model's label prediction with an accuracy of 0.84. Approximating the prediction itself was more difficult, but allowing for a difference of 0.25 between approximation and prediction still yielded an accuracy of 0.75. Allowing only a difference of 0.15 reduced the accuracy to around 0.65. It is also worth noting that running this approximation on the IMDB-BINARY dataset itself (without any model involvement or tuning of the approximation) yields an accuracy of 0.71, which is just a little bit smaller than the model accuracy itself.

Algorithm 1 Approximation algorithm

```
graph ← dataset
if graph = SMG then
    pred = movie(graph.cast)
else if graph = SCNG then
    pred = p(SCNG)
else if messiness ≥ 0.7 then
    pred = 0
else if messiness ≤ 0.3 then
    pred = 1
else
    ...
end if
```

4.4 Interpretation

We interpret these findings as follows. Assuming that the label 0 refers to action movies and label 1 refers to romance movies, we believe that in many cases, a high cast is more probable in an action movie, while a low cast is more probable in a romance movie. This could be rooted in the very nature of the genres (groups fighting vs. two people falling in love). It could also be that romance movies are more likely to be independent films with lower budgets (forcing a lower cast) while action movies tend to be backed by major film studios who do not have such a restriction. Graph messiness, i.e. a large overlap of actors between movies, could be due to the existence of action movie franchises (like Marvel) where a certain set of characters appear in each movie of the franchise. Bigger casts in the action genre might also imply that more actors get to be in movies with more other actors, resulting in more edges in graphs.

5 Conclusion

In the scope of this project, we trained a graph convolutional network that achieved an adequate accuracy of 0.74. To understand the predictions the model made and which features it deemed important, we analyzed dataset entries and created our own graph class to modify graphs freely. After experimenting with dataset graphs, we determined two important features (among others): cast size and messiness of graphs. To validate the hypothesized features, we crafted an algorithm that not only was able to approximate the model with an accuracy of up to 0.84, but it was also able to classify the entire dataset directly with an accuracy of 0.71. At the start of the project our goal was to understand and explain how the model predicted graphs of what is quite an information scarce dataset, and we have achieved that goal successfully. Further work on this project could include leveraging the information extracted from the graphs during training, tuning the training and algorithm even further or designing a more complex network architecture.

6 Contributions of team members

We both contributed to all aspects of this project about equally because we worked on it together in several group sessions over the duration of the course.

References

1. Prashant Gohel,Priyanka Singh and Manoranjan Mohanty.: Explainable AI current status and future directions. https://doi.org/10.48550/arXiv.2107.07045
2. Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, Maosong Sun: Graph neural networks: A review of methods and applications. AI Open **1**, 57–81 (2020)
3. PyTorch Documentation, https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.datasets.TUDataset.html. Last accesed 2 July 2023
4. Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).
5. PyTorch Geometric's Explainer and GNNExplainer class documentation, https://pytorch-geometric.readthedocs.io/en/latest/modules/explain.html