UTRECHT UNIVERSITY

Department of Information and Computing Science

Applied Data Science Master Thesis

A Comparative Study: Graph Neural Network versus Gradient Boosting for Edge Sign Prediction in Social Networks

First examiner:

Javier Garcia-Bernardo

Candidate:

Filip Chrzuszcz

Second examiner:

Eva Jaspers

July 5, 2023

Abstract

Nowadays more and more social network data can be represented as graphs. The availability of such structures, especially in a form of a signed and directed networks bring new challenges that can be analyzed. One of the most common challenges in this field is the problem of sign prediction of the link. The main difficulty is the fact that negative links carry different meaning than positive ones. This research focuses on comparing two methods of predicting the sign of the links. The first method utilizes feature engineering approach, where node and graph specific characteristics are extracted and fed as an input to the gradient boosting model. The second method utilizes Signed Graph Convolutional Network, which focuses on extracting node representations in a low dimensional space, which are used to predict a sign of the link. In the end both methods were compared on the left out test set of randomly chosen edges using accuracy, AUC score, precision and recall. The whole experiment was carried out on the publicly available dataset, which is commonly used as the benchmark for signed network algorithms. The final scores obtained by the models were of high quality. However, the performance differ significantly among tested classes, with positive edges being more easier to predict for the models than the negative ones.

Contents

1. Introduction

1.1 Research motivation

The recent growth in the popularity of social networks has contributed significantly to the increased availability of such data, especially in the form of graphs. In these graphs, nodes usually represent users, while the edge between a pair of nodes indicates some type of relationship between them. Typically, these graphs are characterized by a high sparsity of connections and a large variety of node degrees in the graph. [20]

A vast part of the recent research is focused on the negative and positive sign prediction problem. This task is highly dependent on the network on which it has to be carried out and objectives to fulfill. In the sense of social networks, positive signs may implicate friendship or support, while negative ones - disapproval or disagreement with others. Detecting and predicting these links is important, as it allows us to better grasp human behavior: for example, it could tailor the user experience by suggesting new relationships to other users based on common friends or interests. In addition, such methods can be also utilized in various other domains, such as creating e-commerce recommendations [14] or to identify spurious links in the protein interaction networks [9].

Sources of the sign prediction task originate from social psychology, especially from social balance and status theory. In general, a social balance theory specifies relationships between triads (groups of three nodes), stating that "the enemy of my friend is my enemy," and "the friend of my enemy is my enemy,". The status theory is based on the hidden ordering of nodes, where a positive link between nodes states that one of them possesses higher status, while a negative link indicates a lower status. Each of these theories can be independently used for predicting signs of the edges,

however here they only serve as a basis for both of the methods utilized in this research. [10]

Detection and sign prediction of the links using machine learning methods (e.g., gradient boosting models or graph neural networks) might positively contribute to solving the problem. It might be challenging due to the fact that negative links tend to behave differently than positive ones as they carry different types of information, so the created predictors or embeddings have to take this into account. There exists research that focuses on the sign prediction task such as Leskovec et al.[16] or Chiang et al.[5], where the emphasis was put on creating features, which are in line with social balance theory and training the linear models on such predictors. Available research also includes an approach focused on the graph neural networks [13] [6], where the breakthrough discoveries in the field of neural networks, such as embeddings or attention mechanism, are employed in order to enhance performance on the sign prediction task. However, there is a research gap regarding the comparison of these approaches. As they differ significantly, this study will attempt to compare their performance on a publicly available network.

The present study aims to answer the following question: how well can a machine learning model trained on the dataset with features regarding nodes and edges perform at the sign link prediction task in comparison to the Graph Neural Network model? For this particular problem, the LightGBM (LGBM) model and Signed Graph Convolutional Network (SGCN) model were developed and their performances were compared.

1.2 Related work

Limited research has been conducted regarding the comparative analysis of the graph feature approaches and the graph neural networks. However, both of these approaches have been developed separately and there have been plenty of significant research in recent years. The former approach primarily focuses on computing diverse predictors based on the nodes, edges, and overall graph traits. Some examples of such features can be social bal-

ance theory attributes, status theory attributes, counts of positive directed paths, and in-degree or out-degree. Notably, Leskovec et al. [16] presented an important instance of this approach. They created features per node representing the number of positive and negative edges incoming and outcoming from a node, as well as the number of common neighbors. In the end, the logistic classifier was trained on such a created set of features. This approach was extended by Chiang et al. [5] where predictors responsible for cycles of greater length were also added. As neither of these researches was using more complicated models than linear models it may be of great use to assess the performance of the gradient boosting model on a similar set of features.

The second approach has been mostly focused on utilizing graph neural networks, either using attention mechanism [13] or convolutional layers in order to present nodes in low dimensional space and based on that predicted sign of the edges.[6]. Both of the aforementioned approaches utilize social balance theory as a source of inspiration to create complex representations of the nodes and edges, based on which the graph neural networks operate. These methods lack the simplicity and explainability of the methods based on explicitly created predictors, however, due to their optimized work and ability to squeeze the characteristics of the nodes and edges into embeddings of small size, they also perform well on the task of edge sign prediction.

2. Data

2.1 Description of the data

The dataset utilized in this experiment is based on the collection of Wikipedia users. The network corresponds to votes cast by the users in the elections for the admin position. When a link is positive it signifies a recommendation for a particular user for the admin role, while the negative link represents an opposing vote. There also exist neutral votes, however, these were discarded in the data preprocessing part, as they do not carry the information that the research seeks. In addition, the previous research [16] [5] [2] also relied purely on the positive and negative edges from the input datasets as they bring the crucial information that is searched for in the task of the sign edge prediction. Besides that, some of the links were duplicated. The duplication occurred when the user ran for election several times, the same voter/votee pair may contribute several votes to the dataset. Due to problems with further processing of such edges, the duplicates were dropped, while keeping only the earliest vote in terms of date.

The final network contains 178096 edges and 11259 unique voters and votees. Approximately 78 percent of the edges represent positive edges, with the rest being negative. 10284 users have stated their opinion about another user's admin admission, 3494 unique users have received a vote about their admission, and 2519 have both received and given opinions. The time span of the votes in the dataset ranges from 2004 to 2013. In addition, each link contains a short text which serves as support for the given vote. This research did not take into account these texts, due to a limited way of employing them in the GNN approach.

Because only positive and negative edges were considered during the data processing stage, and the majority of nodes in the dataset lacked an

edge connecting them, it was essential to artificially create and add nonexistent links to the training dataset. In the dataset, they were considered as a third class. The models can then explicitly identify whether the edge exists and, if so, what its sign is.

2.2 Preperation of the data

The data preparation part varied greatly for both of the used methods. The approach based on creating features and traits from the network for the LGBM model required a lot of data preprocessing and preparation, while the SGCN approach hardly required any preparation. [6].

Predictors which were created for the approach relying on the feature dataset can be divided into two parts. The first set of predictors consists of features associated with the social balance theory and status theory. These features were added because Leskovec et al. [16] proved that they can bring a lot of information about the edge sign for the model.

common neighbors - number of common neighbors shared by nodes that form the edge

power of the adjacency matrix - entry from the adjacency matrix raised to power 2 and 3, corresponding to the nodes forming the edge. This power refers to the number of ways in which it is possible to travel between pairs of points creating the edge in a network in exactly k moves.

indegree and outdegree The overall number of incoming and outgoing edges from each node

positive and negative degrees The overall number of incoming and outgoing positive and negative edges for each node

node power This predictor is calculated by counting the number of times each user has been voting in the dataset. It can be associated with the status theory, as the nodes which have been voting more often are having higher status within the network than the ones which have been voting more rarely.

The second group of features contains commonly used graph metrics, which aim at creating the overall picture of the nodes that can be later utilized by the model. As shown by Alzubaidi [2] these features can enhance the performance of the model significantly.

clustering coefficient The clustering coefficient of each node. It is a measure of the degree to which nodes in a graph tend to group together. The coefficient is defined for every single node as the fraction of the number of links between the vertices which are its neighbors divided by the number of edges that could possibly exist between them.

page rank coefficient PageRank algorithm which measures the importance of each node within the graph, based on the number of incoming relationships and the importance of the corresponding source nodes.[11].

For the next set of features, $\Gamma(v)$ denotes the set of neighbors of node v.

resource allocation index The resource allocation index which is defined for a pair of nodes (u,v) as $\sum_{w \varepsilon \Gamma(u) \bigcap \Gamma(v)} \frac{1}{|\Gamma(w)|}$.

preferential attachment score The preferential attachment score is defined for a pair of nodes (u,v) as $|\Gamma(u) * \Gamma(v)|$.

jaccard similarity The Jaccard similarity score is defined for a pair of nodes (u,v) as $\frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u)| \cup |\Gamma(v)|}$.

The SGCN model can build its internal embeddings for the nodes purely on the input set of positive and negative edges, without any precalculated predictors. As a result of that, no extra preprocessing was needed for that step, apart from the basic cleaning, such as removing duplicates as mentioned before.

2.3 Train test split

Because the approach based on the LGBM model requires nonexistent edges for the training phase, in order to be able to predict them on the test set, there was a need to add these to the training dataset. It was done via negative sampling, which works by selecting randomly pairs of nodes that do not

have a link between them and adding such a link to the training set. The number of added edges was chosen to be equal to the number of all existent edges prior to this operation. Due to this, the model should be able to learn the necessary traits of existent and nonexistent edges in order to be able to distinguish them. For the edges added in the process of negative sampling, their features were calculated in the same way as for the rest of the dataset.

The SGCN model works in a similar fashion, however, the negative sampling is done internally in the model. In each training epoch, an equal number of nonexistent edges are generated as the total number present in the entire dataset. The epoch loss is calculated using both the existent and nonexistent ones. Because it is done internally, it is not needed for the user to create new edges for the training phase.

The test dataset is common for both approaches. It is done in order to enable the possibility of comparing both methods. The positive and negative edges are selected from the whole set of edges at random. The size of the test set was set to be 10 percent of the initial dataset. To evaluate model performance on all edge types, including those which do not exist in the original network; similarly, as in the training phase, the nonexistent edges were added via negative sampling.

2.4 Data and code availability

With regards to data availability and ethical considerations associated with the utilization of this dataset, it is shared publicly on the `https://snap.stanford.edu/data/` website [17]. It is available under the BSD license, which means that it is free for both academic and commercial use. The dataset contains usernames, which cannot be connected to any personal data of the users, so it is safe to use from an ethical point of view. Code written for purposes of this research is shared under an MIT license and it is available on the Github repository with the link to it in the Appendix A2.

3. Methods

3.1 Description of the methods used

After collecting and preprocessing the data, two separate datasets and models were built. As stated in the research question, this research aimed to compare two approaches. For a feature based approach the LightGBM [15] model was selected and for the graph neural network approach the SGCN architecture was employed.

3.2 LightGBM and SGCN models

For the approach based on traits of the graph, there was a need to use a machine learning algorithm that is fast, efficient, and provides great results out of the box. Most types of regressors based on the idea of boosting satisfy these criteria, however, the LightGBM excels in terms of speed of efficiency so it was an algorithm of choice for this task.

The algorithm works by sequentially creating multiple weak learners, in this case regression trees which are built on the residuals from the previous trees. In the end, the final prediction is a combination of all built trees. This procedure allows to take advantage of the ability of decision trees to reduce bias, while keeping the variance of the final prediction low, due to the combination of multiple regressors. LightGBM algorithm improves over other boosting methods due to the usage of the Gradient-based One Side Sampling (GOSS) method for tree building and Exclusive Feature Bundling (EFB) for feature selection. GOSS works by attracting the algorithm's attention to the instances with high gradients (e.g., larger than a predefined threshold) and dropping randomly the instances with small gradients. EFB works by utilising the fact that usually, many features are mutually exclusive (e.g., they do not have zero value at the same time), and thanks to that

these features can be combined into a single feature, thus enhancing the speed of the training framework.

The second model was the SGCN model. It was chosen due to its optimzed work and good results on various publicly available datasets. [12]. The principles responsible for its work differ greatly from the LightGBM. The SGCN model focuses purely on operating on signed and directed graphs. It utilizes social balance theory principles to create node embeddings. These embeddings are core elements of the algorithm as they are responsible for creating the final predictions. They are learned via an iterative process of training using the gradient descent method. To capture both positive and negative ties present in the network, for each node there is trained a pair of embeddings, one for positive ties and one for negative ones. Thanks to this approach, the network can keep track of the relationships in the neighborhood of each node. The positives and negatives embeddings are learned separately and after the training is done, they are combined. This combined output is used as an input for the final linear layer which predicts edge classification into one out of three classes (positive, negative, or nonexistent).

3.3 Used metrics

To compare LighgtGBM and SGCN models multiple performance metrics were used. The formulas used for these metrics are abbreviated as follows: TP - True positive, FP -False positive, TN - True negative, FN - False negative. The list of the metrics includes:

1. Accuracy = (TP + TN)/(TP + TN + FP + FN)
2. Precision = TP / (TP + FP)
3. Recall = TP / (TP + FN)
4. AUC - area under the ROC curve. That curve is created by calculating True Positive Ratio (TP / (TP + FN)) and False Positive Ratio (FP/(FP + TN)) and every possible classification threshold.

Both accuracy and AUC can be used as an overall source of information regarding the performance of the models, however, since research apart from

assessing the overall performance also aims at examining the ratios of correctly predicted examples for individual classes, there was a need to use other metrics. Precision and recall can explain and measure the performance of the model on negative, nonexisting, and positive edges separately. Recall explains how many of the true examples have been found by the model, while precision measures how many of the observations which model classified to one of the classes truthfully belong to this class.

3.4 Models setup

Both LightGBM and SGCN models have a plethora of hyperparameters, which can be altered to enhance performance. For the LightGBM model, the search of parameters was run for parameters responsible for the number of estimators and max depth of each tree, because as described on the model website `https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html` these parameters are one of the most important ones in regards of the model building speed and accuracy. For the SGCN model, the parameter search was done for the size of the hidden layers and number of them, because these parameters define the complexity of the model and can drastically change its performance [3].

There are multiple ways in which a search for the best parameters can be done. Starting from the simplest approach such as random search, and ending at the very sophisticated ones, which can explore the parameter space using Bayesian optimization [7]. For the needs of this research, the moderately time consuming approach was chosen in the form of a grid search. It works by exhaustively searching through a defined grid of possible parameter values and evaluating the model's performance for each combination. Thanks to this, it is possible to obtain a good solution within a few rounds of model training, simply by checking every possible option and choosing the best one in the end.

During the learning procedure of both models, the procedure of early stopping [19] was employed. It aims at stopping the training of the model when for a considerable number of epochs there was no improvement on

the loss metric. Thanks to this, the model is less prone to overfitting and the training procedure is much faster without significant loss of the performance.

4. Results

All code for the models, as well as the output files with the results of the grid search, can be found on Github, with the link present in the Appendix A2. All the combinations of the hyperparameters from the grid search were evaluated on the set-aside test set. Values of checked hyperparameters can be found in Appendix A1.

4.1 Overview of the results

Since multiple metrics were computed for each run of the model, there were plenty of variables based on which the best model could have been chosen. Due to its versatility, the area under the ROC curve (AUC) was chosen to be the parameter responsible for choosing the best model.

For the LGBM model, searched parameters were 'max depth' and 'n estimators'. They are responsible for the maximal depth of the grown tree and the number of created trees. The chosen values for these parameters were

- max depth: 4
- n estimators: 600

For the SGCN model, searched parameters were the number of layers and several neurons for each layer. Both parameters are responsible for the complexity of created embeddings. The chosen values for these parameters were

- number of layers: 2
- number of neurons: 512

Referring to figure 4.1, it can be observed that the feature based LGBM model outperforms the SGCN model in terms of both accuracy and AUC. The LGBM model achieves slightly higher scores, with an AUC of 0.956 and an accuracy of 0.94, compared to the SGCN model's AUC of 0.844 and ac-

curacy of 0.807. However, it is important to note that these metrics provide a general overview of the model's performance, so it is worth looking at the recall and precision metrics for all of the classes, especially the negative one, which is the hardest one to predict, due to the low number of examples in the training set.

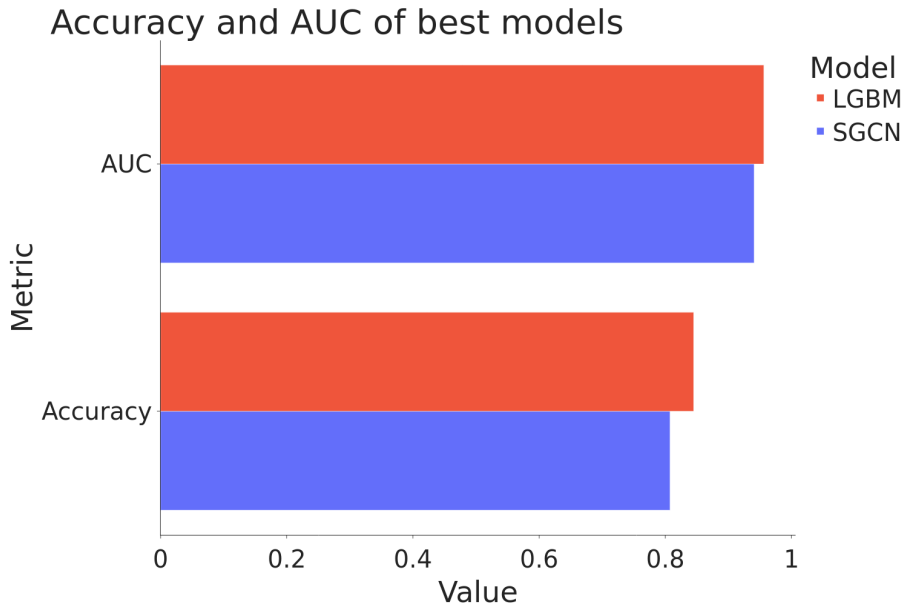

Figure 4.1: Accuracy and AUC of models chosen via grid search

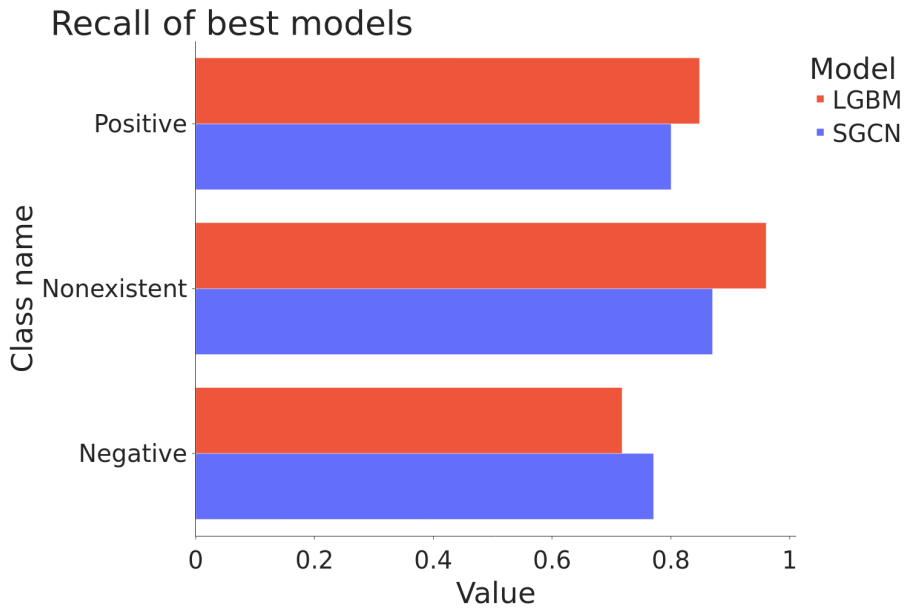

Figure 4.2: Recall of best models chosen via grid search. Scores differ significantly between classes, with the nonexistent class having the best results and negative the worst ones.

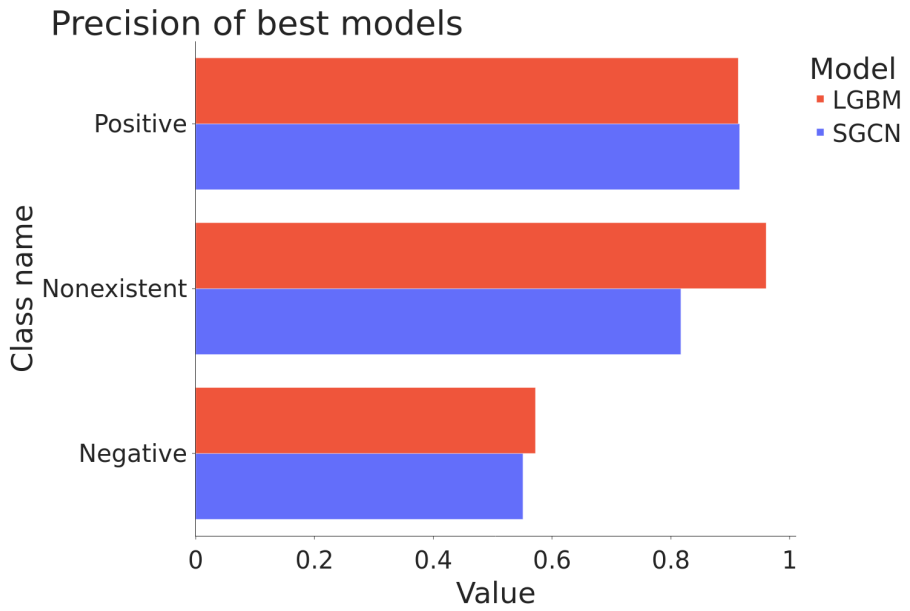

Figure 4.3: Precision of best models chosen via grid search. Precision for the negative class is much lower than for others, while positive and nonexistent classes have scored almost the same for both models.

The LGBM model outperforms the SGCN model when considering precision and recall metrics, however, it is worth noting that on the negative class, the SGCN model has managed to obtain similar results for precision and even slightly better one for recall. Neither of the models had any problems with the nonexistent class, as both of them had high recall and precision, meaning that models have no problems with finding and correctly labeling examples from this class. On the other hand, negative edges pose much more trouble, having recall equal to 0.77 and a precision of 0.55 for the SGCN model and 0.717 and 0.57 for the LGBM model. It means that both models are doing quite well with finding examples from this class, as indicated by high recall. However, the models also show a tendency to predict plenty of false positive examples, which results in low precision. Both models did relatively well in the positive class, having both recall around 0.8 and precision around 0.9.

The ROC curves for the both best models chosen via grid search confirm good results. They can be seen in figure 4.4 and 4.5. These graphs show the performance of the models of all 3 classes at every possible classification threshold. Both models exhibit similar performance on this metric for

positive and negative classes. However, the LGBM model has managed to obtain slightly better results for the nonexistent class in comparison to the SGCN model.

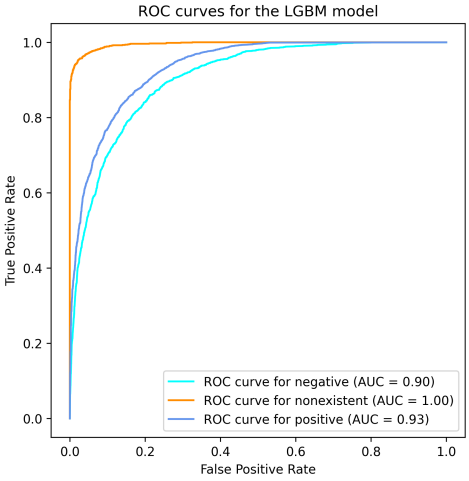

Figure 4.4: ROC curves for the LGBM model

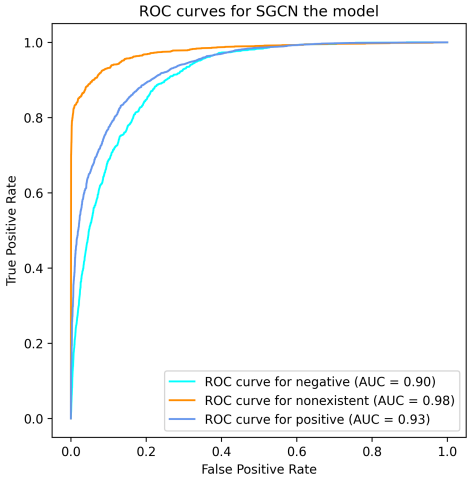

Figure 4.5: ROC curves for the SGCN model

The obtained results for the SGCN model match and even slightly outperform results obtained in the previous research. For example, He et al. [12] have been checking multiple Graph Neural Networks architectures on the task of the sign edge prediction. The exact result obtained for the SGCN model on the AUC metric was 0.847, while the model used in this research has managed to score 0.94. Such a difference may be caused by the fact that in the mentioned research, the nonexistent class has not been predicted, meaning the authors performed only binary classification on positive and negative edges. Because models utilized in this research performed well on the nonexistent class, this may explain the difference between these two approaches. In addition, He was using the logistic classifier on top of the trained nodes embeddings, while the implementation from the Pytorch Geometric package [8], which has been used in this research, utilizes the linear layer of the neural network, which also can increase the performance.

4.2 Feature importance of the LGBM model

Because the LGBM model is based on multiple stacked decision trees, it is possible to easily extract the importance of the features used in the model-

building process. There are 3 main ways in which such importance can be calculated. The first one is based on the number of times the feature was used in all trees, to create a new split. The second one calculates the total gain of the splits, which use a particular feature. These feature importance methods are widely used because they are simple and fast to compute. However there exists a third method called permutation importance [1], which was used in this research. The permutation feature importance refers to the reduction in a model's score when a single feature value is randomly shuffled. Because of the usage of random sampling, which breaks any linkage of the predictor with the target, it is possible to see how much the model depends on that particular predictor [18]. Due to that, it is possible to assess the features importance in a broader sense than simply counting the number of splits or summing the gains.

Table 4.1: Permutation feature importance from the LGBM model

Predictor	Score
degree_in_target	0.182
degree_out_source	0.081
resource_allocation_index	0.058
vote_power	0.047
page_rank_target	0.041
common_neighbors	0.034
clustering_source	0.024
clustering_target	0.020
degree_out_target	0.017
jaccard_coefficient	0.014
target_name	0.009
source_name	0.008
preferential_attachment	0.003
adj_matrix_2	0.000
adj_matrix_3	-0.000
degree_in_source	-0.001
page_rank_source	-0.002

Permutation importance shows that predictors related to status theory and social balance theory such as vote power, indegree, and outdegree of the node are key for the final model performance. Also important are features related to the overall graph structure such as resource allocation index and common neighbors. On the other hand, predictors revolving around the power of the adjacency matrix seem to not matter that much for the model behavior.

5. Conclusion

In the study, we took a technical and comparative approach to address the task of sign edge prediction in the graph structures. Due to the wide availability of such structures, it is important to understand them and have the ability to predict the existence and sign of the edges in such networks. This may allow us to find factors driving human behavior and provide insights into the mechanisms of showing trust and distrust among individuals. This research aimed at comparing two widely used machine learning approaches that can predict the sign of the edges in the graph structures.

Regarding the research question, the accuracy and AUC were high for both the SGCN and LGBM models. They were able to detect and classify the majority of the edges, from all of the classes, however, both models performed noticeably worse on the negative class in comparison to other classes. The majority of the results were slightly in favor of the LGBM, with recall on the negative edges being the only score that was better for the SGCN model. One possible solution for improving the performance of the models on all types of edges may be analyzing the text attached to every existing edge within the dataset. Each such text opinion may give viable information regarding each edge, however, it may also pose problems, as there are no such text entries for edges that do not exist, so negative sampling may be tough for these.

One of the limitations of this research was the use of only one medium-sized dataset. One possible solution to this problem could be the usage of a bigger network for training purposes. Having more edges and nodes would be beneficial for used models, as they would be able to better understand the relationships and dependencies between nodes inside the network. In addition, it may be of great use to compare developed methods on a greater number of datasets in order to check the performances of the models on various types and sizes of networks.

In terms of future work, further comparisons of various approaches and models can be done. Interesting examples of models that may be used are xGboost [4] for the feature-based approach and SiGAT [13] for the GNN approach. Both of these models differ from the ones used in this research, however, they still may obtain good or even better results.

A. Appendix

A.1 Grid search

In order to gain maximal performance of the models, grid search of the hyperparameters was performed. It was done for each model separately, because LGBM and SGCN models have different set of parameters which can be optimized.

For the LGBM model searched parameters were ‘max depth‘ and ‘n estimators‘. They are responsible for maximal depth of grown tree and number of created trees. For both parameters tested values were:

- max depth -> [4,8,10,15]
- n estimators -> [400,600,1000]

For the SGCN model searched parameters were number of layers and number of neurons for each layer. Both parameters are responsible for complexity of created embeddings. Tested values were:

- number of layers -> [2,3,4]
- number of neurons -> [64,128,256,512]

Scores of the grid search can be found in the csv file present in the github repository.

A.2 Github repository

The code with the whole preprocessing pipeline and explanatory notebooks can be found under `https://github.com/arctickey/Master_thesis`

Bibliography

[1] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: A corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 2010.

[2] A. M. N. Alzubaidi, "Lightgbm for link prediction based on graph structure attributes," *ICIC express letters. Part B, Applications: an international journal of research and surveys*, vol. 14, no. 3, pp. 303–311, 2023.

[3] R. Andonie and A.-C. Florea, "Weighted random search for cnn hyperparameter optimization," *arXiv preprint arXiv:2003.13300*, 2020.

[4] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, San Francisco, California, USA: ACM, 2016, pp. 785–794, ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785.

[5] K.-Y. Chiang, N. Natarajan, A. Tewari, and I. S. Dhillon, "Exploiting longer cycles for link prediction in signed networks," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, 2011, pp. 1157–1162.

[6] T. Derr, Y. Ma, and J. Tang, "Signed graph convolutional networks," in *2018 IEEE International Conference on Data Mining (ICDM)*, IEEE, 2018, pp. 929–934.

[7] M. Feurer and F. Hutter, "Hyperparameter optimization," *Automated machine learning: Methods, systems, challenges*, pp. 3–33, 2019.

[8] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.

[9] R. Guimerà and M. Sales-Pardo, "Missing and spurious interactions and the reconstruction of complex networks," *Proceedings of the National Academy of Sciences*, vol. 106, no. 52, pp. 22 073–22 078, 2009.

[10] F. Harary, "On the notion of balance of a signed graph.," *Michigan Mathematical Journal*, vol. 2, no. 2, pp. 143–146, 1953.

[11] T. Haveliwala, "Efficient computation of pagerank," Stanford, Tech. Rep., 1999.

[12] Y. He, X. Zhang, J. Huang, B. Rozemberczki, M. Cucuringu, and G. Reinert, "Pytorch geometric signed directed: A software package on graph neural networks for signed and directed graphs," *arXiv preprint arXiv:2202.10793*, 2022.

[13] J. Huang, H. Shen, L. Hou, and X. Cheng, "Signed graph attention networks," in *Artificial Neural Networks and Machine Learning–ICANN 2019: Workshop and Special Sessions: 28th International Confer-*

ence on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings 28, Springer, 2019, pp. 566–577.

[14] Z. Huang, X. Li, and H. Chen, "Link prediction approach to collaborative filtering," in *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, 2005, pp. 141–142.

[15] G. Ke, Q. Meng, T. Finley, *et al.*, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[16] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 641–650.

[17] J. Leskovec and A. Krevl, *SNAP Datasets: Stanford large network dataset collection*, `http://snap.stanford.edu/data`, Jun. 2014.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[19] L. Prechelt, "Early stopping-but when?" In *Neural Networks: Tricks of the trade*, Springer, 2002, pp. 55–69.

[20] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The anatomy of the facebook social graph," *arXiv preprint arXiv:1111.4503*, 2011.