Link Prediction on Bitcoin OTC Network

Oliver Tanevski
Faculty of Computer Science and Engineering
Skopje, North Macedonia
oliver.tanevski@students.finki.ukim.mk

Igor Mishkovski
Faculty of Computer Science and Engineering
Skopje, North Macedonia
igor.mishkovski@finki.ukim.mk

Miroslav Mirchev
Faculty of Computer Science and Engineering
Skopje, North Macedonia
miroslav.mirchev@finki.ukim.mk

***Abstract*-Link prediction is a common problem in many types of social networks, including small Weighted Signed Networks (WSN) where the edges have positive and negative weights. In this paper, we predict transactions between users in Bitcoin OTC Network, where the links represent the ratings (trust) that the users give to each other after each transaction. Before predicting, we transform the network where we convert negative weights into positive so that the feature scores, calculated by existing algorithms (such as Common Neighbours, Adamic Adar etc.) would improve the models performance in our link prediction problem. We consider two methods that will help us in our link prediction: attributes estimation based on similarity scores link prediction and link prediction as supervised learning problem. The first method can be used more as a way to determine which of the attributes (feature scores) are more important in link prediction. The second method is used for estimating attributes importance, but even more for actual prediction using the calculated feature scores as input to the machine learning and deep learning models. The predicted links can be interpreted as possible transactions between certain users.**

***Index Terms*—link prediction, weighted signed directed graphs, network science, machine learning**

I. Introduction

In the past several years, the Bitcoin and other cryptocurrencies emerged into the digital world of transactions. The ability to transfer unlimited amount of money without paying fees to a third party, like banks, and the idea of mining Bitcoin using the blockchain technology, caught the users interest. Because of that, a lot of platforms were created where people can make these transactions based on certain market policy. When a user registers on such platform and it has been authenticated, he can later use it to buy or sell bitcoins. On some platforms, when a transaction has been completed, users can rate each other. That rating indicates how trustworthy the user is and helps other users to estimate this user whenever they want to make a transaction with him in the future.

In this paper, we use the Bitcoin OTC Network[1] in order to predict whenever a possible transaction will be made between two users in the network. In order to achieve this we use a plethora of methods and algorithms from network science and machine learning. We represent this network as a graph, where links indicate the user ratings. Since users usually rate each other after transaction or several transactions, the links also indicate that those users made a transaction, which comes down to link prediction in a graph and/or predicting future transactions in the network.

This research can be used on other similar trading markets which are based on ratings. Also it can be used in order to determine if there would be an interaction between users or companies and who would initiate the interaction.

II. Related Work

There have been many studies in the field of link prediction, most of them focused on big social networks like Facebook and Twitter [1], citation networks [2] and dynamic networks (networks were edges are created over time) [3]. The problem have been also addressed for multilayer online social networks [4]. Most of these studies use methods from machine learning, but on graphs where the edges are undirected and unweighted. There also has been work on link prediction on directed weighted graphs [5], where the weights are always positive.

In this work, we will present a plethora of algorithms and methods used on our sparse network with negative weights and evaluate them using ROC Curve and Precision-Recall Curve.

III. Dataset

We used the Bitcoin OTC network dataset [6, 7] that is available in the Standford Large Dataset Collection.[2]

The dataset contains source and target nodes, where the source node gives rating to the target node that is between -10, which means that the user did not hold his end of the bargain and that he is a fraud, and 10, which means that this user is very trustworthy.

The Bitcoin OTC Network consists of 5881 users and 35592 links between users, which is a relatively small network. The *Average Clustering Coefficient* is 0.151, which shows that this network is clustered. We wanted to see the node degree distribution (Fig. 1) in our network in order to determine if the network is scale-free and follows a Power-Law distribution. Additionally, we calculated the *Power-law exponent:* $\gamma = 1.969$ and *Tail power-law exponent:* $\gamma_t = 2.051$. Comparing the two

[1]Bitcoin OTC Network: `https://www.bitcoin-otc.com/`

[2]`https://snap.stanford.edu/data/soc-sign-bitcoin-otc.html`

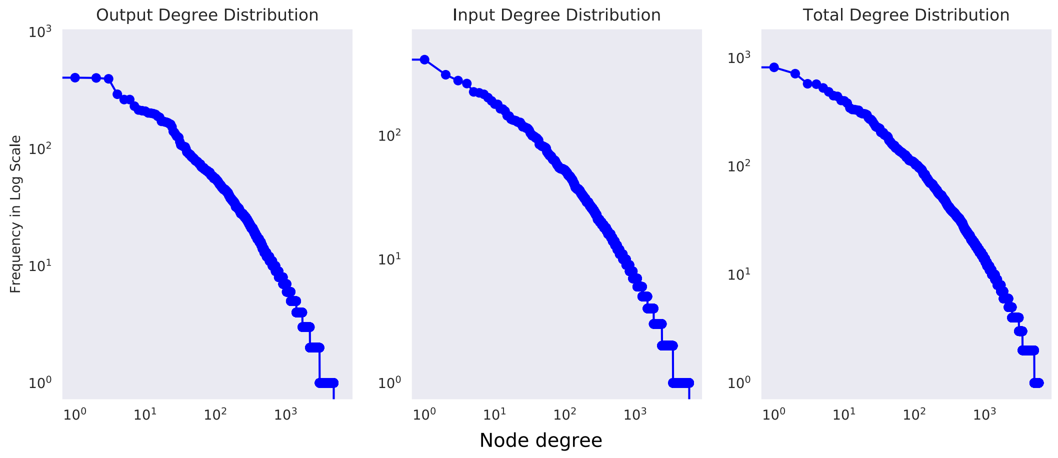

Fig. 1: Node degree distribution in the Bitcoin OTC Network.

exponents, we can see that the Bitcoin OTC Network is not a typical scale-free network, since they are not almost equal.

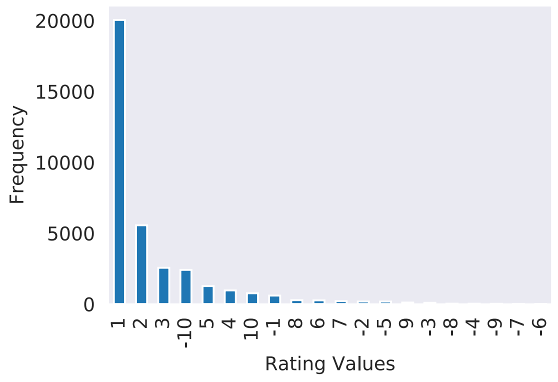

Fig. 2: Ratings given in the Bitcoin OTC Network.

We also examined what rating was most often given by the users (Fig. 2) in order to have a rough estimation of how much of the transactions were successful. We see that most of the ratings are positive, which probably indicates that most of the transactions were successful.

IV. Research Methodology

By definition, the term *Weighted Signed Network* means a directed, signed graph $G = (V, E, W)$ where V is the set of users, $E \subseteq V \times V$ is the set of edges in the graph and $W : E \rightarrow [-n, +n]$ is weight value between some -n and n assigned to an edge. In our graph, a $W(u, v)$ means how much the user u "likes" or "trusts" user v. In the following, we describe the methods used for link prediction and define the attributes used by these methods.

A. Methods

In our paper, we use two methods that will help us in our link prediction. In both of these methods we consider two graphs G_1 and G_2 where G_1 is used as a training graph and G_2 is the graph in which we predict links. The methods are:

- **Attribute estimation based on Similarity Scores link prediction**
- **Link Prediction as a Supervised Learning Problem**.

1) Attribute estimation based on Similarity Scores link prediction: Let u and v denote two nodes in G who are not connected in G_1, but are connected in G_2. Then we can assign similarity measurement $score(u, v)$ for the node pair (u, v). Thus, we assign score value $\forall e \notin G_1$, $S : score(e)$ where e is the edge (node pair) that does not exist in the graph G_1 and create a list in decreasing order. Then we select the top M high scored edges and see how many of them are in G_2. Furthermore, we will show how the accuracy changes for different sizes of M. The more the accuracy increases, the more valuable is the feature score in our link prediction.

There are mainly three groups of similarity measurement scores, where each gives different results based on the problem. Those are:

- **Local Measurement Scores** - where we calculate the score mostly based on the information depending on nodes u and v.
- **Global Measurement Scores** - where we use all paths in the graph to calculate the score. These measurement scores usually give good results, but depending on the graph it may take very long time to compute.
- **Quasi-Local Measurement Scores** - which are in-between measurement scores, that balance the trade-off between accuracy and computing complexity.

We must emphasize that in this work we focus only on local measurement scores and estimate their importance for link prediction.

2) Link Prediction as a Supervised Learning Problem: From the original graph G we choose node pairs (u, v) that do not exist in the graph. We choose equal number of non-existing edges and existing edges in order to solve the problem with class imbalance while training. Whenever there is a present edge in the graph, we label it as positive, whereas the non-present edges are labeled as negative. This is crucial pre-processing step before we split the original graph in two subgraphs G_1 and G_2.

For each sample, we use a variety of attributes, such as: topological (global), neighbourhood-based (local) etc. in order to form a dataset for training and test, which is then fed into the machine learning and deep learning models. In the next subsection, we will give an insight of the types of attributes used by our models.

B. Attributes - Feature Scores

As we mentioned before, in each of our methods we will use attributes (feature scores) that will help us build models and evaluate them later on. All of these attributes represent some score for a given pair of nodes u and v.

1) Common Neighbors Score: The idea of common neighbours comes from social networks where it states that if two strangers who have a friend in common a more likely to be introduced to each other than those who do not have any friends in common. For our problem, we use Laishram's [5] variation of Common Neighbours in directed weighted graphs:

$$CN_{i,weighted}(u,v) = \sum_{z\in(\Gamma_i(u)\cap\Gamma_i(v))} \frac{w(z,u)+w(z,v)}{2},$$

$$CN_{o,weighted}(u,v) = \sum_{z\in(\Gamma_o(u)\cap\Gamma_o(v))} \frac{w(u,z)+w(v,z)}{2},$$

where u and v are the nodes, Γ_i are the predecessors of a given node and Γ_o are the successors of a given node and $w(x,y)$ is the link's weight.

2) Preferential Attachment Score: Preferential attachment is measure to compute the closeness of two nodes based on their neighbours. If both of the nodes u and v have more neighbours, the chance is bigger for them to connect. We will use Laishram's [5] weighted variation:

$$PA_{i,weighted}(u,v) = \left(\frac{\sum\limits_{z\in\Gamma_i(u)} w(z,u)}{|\Gamma_i(u)|}\right) * \left(\frac{\sum\limits_{z\in\Gamma_i(v)} w(z,v)}{|\Gamma_i(v)|}\right),$$

$$PA_{o,weighted}(u,v) = \left(\frac{\sum\limits_{z\in\Gamma_o(u)} w(z,u)}{|\Gamma_o(u)|}\right) * \left(\frac{\sum\limits_{z\in\Gamma_o(v)} w(z,v)}{|\Gamma_o(v)|}\right).$$

3) Adamic Adar Score: Adamic Adar score [8] is based on a concept that if a person has a lot of friends, and he is a common friend or acquaintance of other two people, then it is less likely to introduce the two people to each other, other than when he would have very few friends [9]. We will use Laishram's [5] weighted variation of Adamic Adar:

$$AA_{i,weighted}(u,v) = \frac{1}{2}\left(\sum_{z\in(\Gamma_i(u)\cap\Gamma_i(v))} \frac{w(z,u)+w(z,v)}{\log(\sum\limits_{x\in\Gamma_i(z)} w(x,z))}\right),$$

$$AA_{o,weighted}(u,v) = \frac{1}{2}\left(\sum_{z\in(\Gamma_o(u)\cap\Gamma_o(v))} \frac{w(u,z)+w(v,z)}{\log(\sum\limits_{x\in\Gamma_o(z)} w(z,x))}\right).$$

4) Shortest Path Score: Shortest path score is a global measurement score that calculates reciprocal of the length of all shortest paths from u to v. We use Laishram's [5] weighted variation of Shortest Path:

$$SP_{weighted}(u,v) = \frac{1}{|\rho(u,v|l_{min})|}\left(\sum_{p\in\rho(u,v|l_{min})} \frac{\sum\limits_{(x,y)\in p} w(x,y)}{l_{min}}\right),$$

where $|\rho(u,v|l_{min})|$ are number of paths between u and v with length l_{min}, and l_{min} is the length of the shortest path.

5) Fairness and Goodness: The Fairness and Goodness algorithm [6] is an extension of HITS [10] for directed signed graphs. The fairness attribute of a node is a measure of how fair is the node in giving ratings to other nodes (users). The goodness attribute of a node measures how much the other nodes "like" him and what is his true quality. Fairness and Goodness are calculated as follows:

$$\text{Let } f^0(u) = 1 \text{ and } g^0(u) = 1, \forall u \in V,$$

$$g^{t+1}(u) = \frac{1}{|\Gamma_i(u)|} \sum_{z\in\Gamma_i(u)} f^t(z) \times w(z,u),$$

$$f^{t+1}(u) = 1 - \frac{1}{|\Gamma_o(u)|} \sum_{z\in\Gamma_o(u)} \frac{|w(u,z) - g^{t+1}(z)|}{R},$$

where R is the maximum difference between an edge weight and goodness score [6]. The full algorithm is described in Kumar's paper [6].

6) Jaccard Similarity: Jaccard Similarity is a measure that shows us how similar are two nodes in a graph by calculating how many common neighbours they have from all their neighbours. The Jaccard Similarity for directed graphs will be calculated as follows:

$$JC_i(u,v) = \frac{|\Gamma_i(u)\cap\Gamma_i(v)|}{|\Gamma_i(u)\cup\Gamma_i(v)|},$$

$$JC_o(u,v) = \frac{|\Gamma_o(u)\cap\Gamma_o(v)|}{|\Gamma_o(u)\cup\Gamma_o(v)|}.$$

7) Cosine Similarity: Cosine Similarity in a directed graph is calculated as follows:

$$CS_i(u,v) = \frac{|\Gamma_i(u)\cap\Gamma_i(v)|}{|\Gamma_i(u)| * |\Gamma_i(v)|},$$

$$CS_o(u,v) = \frac{|\Gamma_o(u)\cap\Gamma_o(v)|}{|\Gamma_o(u)| * |\Gamma_o(v)|}.$$

8) Katz Score: Katz score is similar to Shortest path score, however instead of taking only the shortest paths, it takes all paths in the graph. The Katz score in directed graph is calculated like this:

$$KS(u,v) = \sum_{l=1}^{\infty} \beta^l * |\rho(u,v|l)|,$$

where $|\rho(u,v|l)|$ are the number of paths between u and v with length l and β is damping factor between 0 and 1. We used the unweighted variation of Katz score since the weighted one was computationally expensive.

9) PageRank Score: The PageRank Score is a node based feature and is calculated as follows:

$$\text{Let } PR(u) = \frac{1}{|V|}, \forall u \in V,$$

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)},$$

where B_u are the links that u is connected to and $L(v)$ are the number of links that node v points to.

10) HITS: HITS [10] or *Hubs and Authorities* was initially created for rating Web pages. Authorities in our case will be the users who participated in most of the transactions, while hubs are those users that can point us to the authoritative users. HITS is calculated as follows:

$$\text{Let } u = (1, ..., 1),$$

$$v = A^T u, \quad u = Av,$$

where A is the adjacency matrix, u is the hub weight vector and v is the authority weight vector. Since we use Fairness and Goodness as weighted variation of HITS, we also wanted to use the unweighted variant for comparison and determine which one is more valuable in the prediction.

V. Research Study

In this section, we study the problem of link prediction in Weighted Signed Network. There are many studies on link prediction focused on the big online social networks, which prompted us to experiment on smaller networks with the previously mentioned approaches and attributes.

Before we explain how we used the methods, we need to point out that certain weight mapping was used in order to calculate the feature scores. For all attributes except Shortest Path Score, the mapping used is $[-10,10] \to [1,20]$, while in Shortest Path a mapping of $[-10,10] \to [20,1]$ is used. The mapping was not used for the Fairness and Goodness attribute, since the algorithm itself can handle negative weights.

In the first method (i.e. Attribute estimation based on Similarity Scores link prediction), we find the maximal strongly connected component and we split it in two subgraphs G_1 and G_2 where we take 15% of the edges in the maximal component to the subgraph G_2 and the rest to G_1. We mentioned above that we will use different sizes for M which is the number of top high scored edges that do not exist in graph G_1. M takes values $l_{test}2^i$, $\forall i \in [0,1,...,k]$, where k is a number which satisfies the condition $l_{test}2^k$ equals 15% of the number of all possible edges in G_1 and l_{test} is the number of edges in graph G_2. We also mentioned that for this method we will use only local feature scores. Those are **Common Neighbours**, **Adamic Adar** and **Preferential Attachment**. With each of these measures, we calculate the score for all edges that do not exist in graph G_1 and we select the top M high scoring edges and see how much of them belong in G_2 as we change M.

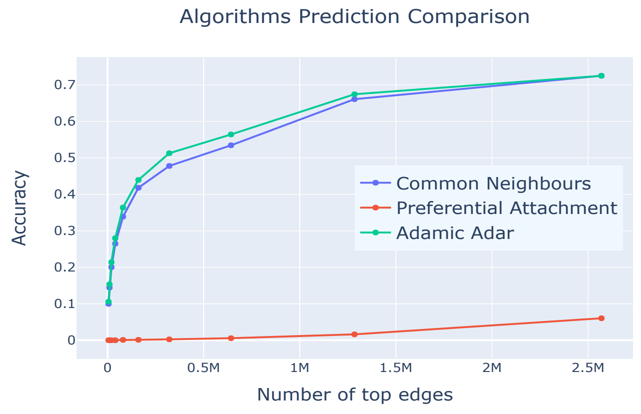

Fig. 3: Algorithms accuracy based on top M edges selected

TABLE I: Accuracy for each algorithm

Value of M	Algorithms		
	Common Neigh.	Adamic Adar	Preferential Attach.
5019	0.099	**0.105**	0.0
10038	0.144	**0.153**	0.0
20076	0.2	**0.213**	0.0
40152	0.264	**0.28**	0.0
80304	0.339	**0.363**	0.001
160608	0.418	**0.439**	0.001
321216	0.477	**0.512**	0.002
642432	0.534	**0.564**	0.006
1284864	0.66	**0.674**	0.016
2569728	0.724	0.724	0.06

From the results above, we can see that Adamic Adar gives the best results. Common Neighbours has similar accuracy results compared to Adamic Adar in the beginning and same accuracy in the end. However, Preferential Attachment gave very poor result. These results gives us a rough estimate about which local features are more significant for link prediction when using the second method, i.e. Link Prediction as a Supervised Learning Problem.

In the second method, we select edges whose links are not present in the original graph G. Since the graph is very sparse, the number of selected non-present edges will be very high which will produce very high class imbalance. For the edges that exist in the graph, we label them as positive, while the other as negative. Next, we split our main graph G with the generated edges into G_1, which is used for training and has equal amount of existing and non-present edges in order to solve the problem of class imbalance while training, and G_2, which is used for test and uses 15% of the existing edges and 15% of the non-present edges, which makes it highly unbalanced. For each of these graphs, we generate a dataset where the attributes are all the feature scores we previously mentioned along with their node pair whose feature scores are calculated. Afterwards, we remove the node pairs from the dataset in order to prevent the machine learning algorithms to be biased based on the edges. These datasets were used for training and evaluating several machine learning and deep learning models, which are shown in the following figures.

For evaluation metrics, we used ROC Curve and Precision-Recall Curve for each of the models. We used *5-fold Cross Validation with Grid Search* in order to find the optimal hyper-parameters for our machine learning models.

For our deep learning model, we used classical Feed Forward Network with the idea of Residual Networks [11] by using only the *skip connections*. For simplicity, we called it SkipConnNet. We used 3 hidden Linear layers, each with 100 neurons. The optimizer used was Ranger [12] with learning rate of 0.0001. We used validation split of 10% and batch size of 128. The loss function we used was Mean False Error (MFE) [13]. The model was created using Pytorch[14] and Poutyne [15].

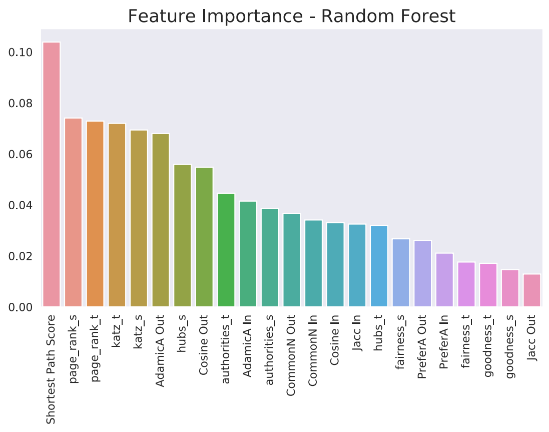

Fig. 4: Feature Importance using Random Forest

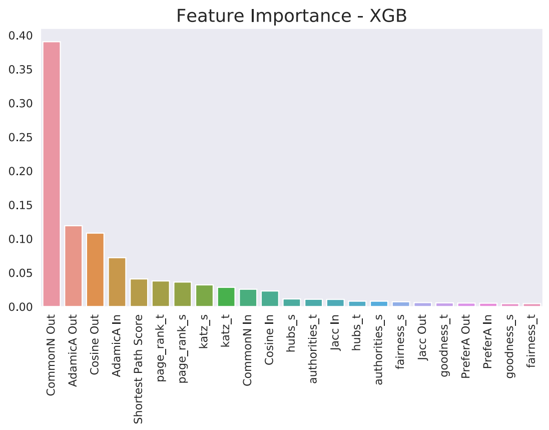

Fig. 5: Feature Importance using XGB

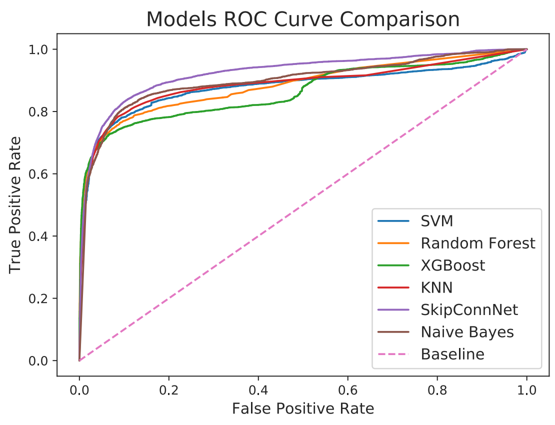

Fig. 6: Models ROC Curves

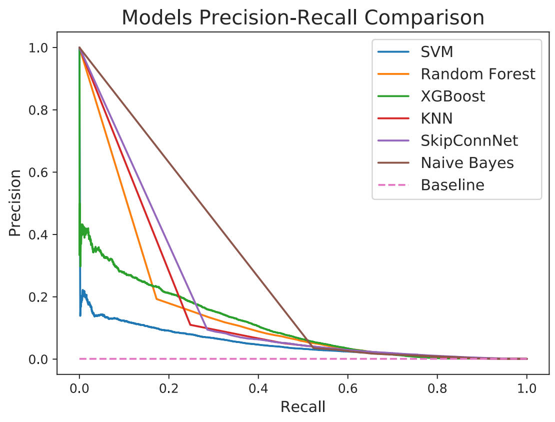

Fig. 7: Models Precision-Recall Curves

As we mentioned above, we have a highly imbalanced test set for evaluation, where the dominant class is the negative one (ratio of around 1:1000). From Fig. 6 we can see that the **SkipConnNet** has the best ROC curve than the other models. However, in Fig. 7 is hard to distinguish which model performs better. All of them vary significantly for certain precision and recall, however for recall above 0.5, all of them converge to a certain precision value.

We also used Random Forest and XGB (Fig. 4 and Fig. 5) to see which features impacted their prediction the most. On both plots, on the x-axis, we see two formats of labeling our features **xxx_s/xxx_t**, which means that this feature was calculated individually for both Source and Target nodes respectively, and **xxx Out/xxx In**, which means that this feature was calculated based on the successors and predecessors of both Source and Target nodes respectively. It can be seen that Common Neighbours and Adamic Adar have bigger impact than Preferential Attachment, which is what we roughly estimated with the first method.

VI. Conclusion

In our research we have shown:

- how we use the problem of link prediction to predict a possible transaction in Bitcoin OTC Network,
- that certain weight mapping from negative to positive can improve the score from Common Neighbours, Adamic Adar etc. and with that, the overall model performance,
- that link prediction using similarity scores can also be used to give a rough estimation of how much each feature is important in the prediction,
- that link prediction as supervised learning problem with machine learning models can give decent result on a sparse network with very big class imbalance like the one we used.

In the future we will extend this work by using state-of-the-art networks like Node2Vec to generate additional features and improve the performance of our models even more.

References

[1] Peng Wang et al. "Link Prediction in Social Networks: the State-of-the-Art". In: *CoRR* abs/1411.5118 (2014). arXiv: 1411.5118. URL: http://arxiv.org/abs/1411.5118.

[2] Mohammad Al Hasan et al. "Link prediction using supervised learning". In: 2006.

[3] Catherine A. Bliss et al. *An Evolutionary Algorithm Approach to Link Prediction in Dynamic Social Networks*. 2013. arXiv: 1304.6257 [physics.soc-ph].

[4] Haris Mandal et al. "Multilayer Link Prediction in Online Social Networks". In: *2018 26th Telecommunications Forum (TELFOR)*. IEEE. 2018, pp. 1–4.

[5] Laishram Ricky. "Link Prediction in Dynamic Weighted and Directed Social Network using Supervised Learning". In: (2015). Dissertations - ALL. 355. URL: https://surface.syr.edu/etd/355/.

[6] Srijan Kumar et al. "Edge weight prediction in weighted signed networks". In: *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE. 2016, pp. 221–230.

[7] Srijan Kumar et al. "Rev2: Fraudulent user prediction in rating platforms". In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM. 2018, pp. 333–341.

[8] Lada A. Adamic and Eytan Adar. "Friends and neighbors on the Web". In: *Social Networks* 25 (2001), pp. 211–230.

[9] Fei Gao et al. "Link Prediction Methods and Their Accuracy for Different Social Networks and Network Metrics". In: *Scientific Programming* (2015), p. 172879. ISSN: 1058-9244. DOI: 10.1155/2015/172879. URL: https://doi.org/10.1155/2015/172879.

[10] Jon M. Kleinberg. "Authoritative Sources in a Hyperlinked Environment". In: *J. ACM* 46.5 (Sept. 1999), pp. 604–632. ISSN: 0004-5411. DOI: 10.1145/324133.324140. URL: https://doi.org/10.1145/324133.324140.

[11] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[12] Michael R. Zhang et al. *Lookahead Optimizer: k steps forward, 1 step back*. 2019. arXiv: 1907.08610 [cs.LG].

[13] Shoujin Wang et al. "Training deep neural networks on imbalanced data sets". In: July 2016, pp. 4368–4374. DOI: 10.1109/IJCNN.2016.7727770.

[14] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[15] Frédérik Paradis. *Poutyne: A Keras-like framework for PyTorch*. https://poutyne.org. 2018.