Supervised Learning for Link Prediction Using Similarity Indices

Sergey Korolev[1] and Leonid Zhukov

Higher School of Economics, Moscow, Russia,
sokorolev@edu.hse.ru

Abstract. The problem of link prediction gathered a lot of attention in the last few years, arising in different applications ranging from recommendation systems to social networks. In this paper, we will describe most popular similarity indices, compare their performance in their ability to show links with the highest probability of being removed from initial network and describe the approach that allows to use them to predict missing links using supervised machine learning. We will show the accuracy of prediction of this method on examples of real networks.

Keywords: network analysis, graph theory, link prediction

1 Introduction

Over the last few years the topic of link prediction has become very popular due to the rise of recommendation systems and social networks. The ...

The aim of this paper is to try to devise an approach to predict the missing links in the network using only structural information of said network. And also test for underlying properties of the index space of the links of the network.

2 Similarity Indices

In this section, we will describe most popular (as presented in [1]) local and global similarity indices and measure their accuracy in ranking links according to the probability that said links were removed from the network.

We'll start with describing the method for measuring said accuracy. We start with the set of edges E of existing graph G. Then we shuffle it and split it into n parts $E_i',\ i \in \{1, \dots, n\}$. Then for each step we subtract E_i' from E and look at the complement of the remaining set of edges $\bar{E}_i$ to the set of edges of the complete graph on these nodes, so that $E^* = E \cup \bar{E} = E_i \cup E_i' \cup \bar{E} = E_i \cup \bar{E}_i$, where E^* is the set of edges of the complete graph and $\bar{E}$ is complement of graph G.

The main tool for measuring the accuracy of the ranking is AUC [2] score, which can be described as the probability that for a pair of edges (e_a, e_b) $e_a \in E_i',\ e_b \in \bar{E}$, the ranking is such that $score(e_a) > score(e_b)$ plus 0.5 times the probability that $score(e_a) = score(e_b)$. Or as a formula – $AUC = \frac{n' + 0.5n''}{n}$, where n is the number of pairs (e_a, e_b) $e_a \in E'_i$, $e_b \in \bar{E}$, n' is the number of pairs such that $score(e_a) > score(e_b)$ and n'' is the number of pairs such that $score(e_a) = score(e_b)$.

We'll first describe all indices used for the prediction of links and then compare their AUC scores on different networks.

2.1 Local Similarity Indices

Common Neighbours (CN) This is the most obvious approach to the idea of capturing the similarity of two nodes in the network. As such, the number of indices described below will use this measure with different weights. So if we denote $\Gamma(x)$ the number of neighbours of node x in the graph, the formula looks like

$$s_{xy}^{CN} = |\Gamma(x) \cap \Gamma(y)|, \tag{1}$$

where s_{xy} is the index. For symmetric graphs it is also obvious that $s_{xy} = (A^2)_{xy}$, where A is the adjacency matrix of said graph.

Salton Index (SaI) [3] This index is also called cosine similarity and is defined as

$$s_{xy}^{Salton} = \frac{|\Gamma(x) \cap \Gamma(y)|}{\sqrt{k_x \times k_x}}, \tag{2}$$

where k_x is the degree of node x.

Jaccard Index (JI) [4] This index was proposed by Jaccard and measures the number of common neighbours weighted by the size of the union of the sets of neighbours:

$$s_{xy}^{Jaccard} = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}. \tag{3}$$

Sørensen Index (SoI) [5]

$$s_{xy}^{Sørensen} = \frac{2|\Gamma(x) \cap \Gamma(y)|}{k_x + k_y}. \tag{4}$$

Hub Promoted Index (HPI) [6] This index gives higher scores to edges adjacent to hubs since it is weighted by the minimum of node degrees:

$$s_{xy}^{HPI} = \frac{|\Gamma(x) \cap \Gamma(y)|}{min\{k_x, k_y\}}. \tag{5}$$

Hub Depressed Index (HDI) This index penalises scores for the edges adjacent to hubs:

$$s_{xy}^{HDI} = \frac{|\Gamma(x) \cap \Gamma(y)|}{max\{k_x, k_y\}}. \tag{6}$$

LeichtHolmeNewman Index (LHN1) [7] This index promotes edges adjacent to the nodes with lower degrees:

$$s_{xy}^{LHN1} = \frac{|\Gamma(x) \cap \Gamma(y)|}{k_x \times k_y}. \tag{7}$$

Preferential Attachment Index (PAI) This index is derived from the preferential attachment model of network growth [8]. It promotes edges adjacent to the nodes of higher degrees:

$$s_{xy}^{PA} = k_x \times k_y. \tag{8}$$

AdamicAdar Index (AAI) [9] This index is a modified version of counting the amount of common neghbours as it values neighbours with lower degree higher:

$$s_{xy}^{AA} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{logk_z}. \tag{9}$$

Resource Allocation Index (RAI) [10] This index is similar to the (9) and is derived from the model of resource allocation in the networks, assuming that node's ability to transfer something is inversely proportional to its degree:

$$s_{xy}^{RA} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{k_z}. \tag{10}$$

2.2 Global Similarity Indices

Katz Index (KI) [11] This index measures the number of paths of length l weighted by β^l, where β is the dampening factor. Since the number of paths in a symmetric graph can be expressed as the power of the adjacency matrix, it can be expressed in the matrix form as:

$$S^{Katz} = (I - \beta A)^{-1} - I, \tag{11}$$

where β should be less than the reciprocal of the largest eigenvalue of A in order for the equation to converge.

LeichtHolmeNewman Index (LHN2) [7] This index is the modification of Katz index that takes into account that two nodes tend to be similar if their neighbours are similar. It can be expressed in the matrix form [1] as:

$$S^{LHN2} = 2m\lambda_1 D^{-1}(I - \frac{\phi A}{\lambda_1})^{-1} D^{-1}, \tag{12}$$

where m is the number of edges in the network, λ_1 is the largest eigenvalue of adjacency matrix A, D is a diagonal degree matrix of form $diag\{k_1, \dots, k_n\}$, $\phi:\ 0 < \phi < 1$ is a free parameter that assigns higher weights to shorter paths if it's closer to 0 and to longer paths if it's closer to 1.

Average Commute Time (ACT) This index assigns higher scores to the pair of nodes for which the average length of the random walk from one node to another is shorter. It can be expressed as:

$$s_{xy}^{ACT} = \frac{1}{l_{xx}^{+} + l_{yy}^{+} - 2l_{xy}^{+}}, \tag{13}$$

where l_{ij}^{+} is the (i, j) entry of pseudoinverse of the Laplacian matrix $L = D - A$.

Cosine based on L^+ (CBL) This index is defined as the cosine of two node vectors:

$$s_{xy}^{cos^+} = cos(x, y)^+ = \frac{v_x^T v_y}{|v_x| \cdot |v_y|} = \frac{l_{xy}^{+}}{\sqrt{l_{xx}^{+} \cdot l_{yy}^{+}}}. \tag{14}$$

Random Walk with Restart (RWR) This index is derived from the PageRank algorithm and can be expressed as:

$$s_{xy}^{RWR} = q_{xy} + q_{yx}, \tag{15}$$

where q_{xy} is the yth element of vector $\overrightarrow{q_x}$. $\overrightarrow{q_x} = (1 - c)(I - cP^T)^{-1}\overrightarrow{e_x}$, where c is probability in the PageRank algorithm, P is the transition matrix with $P_{xy} = \frac{1}{k_x}$ if x and y are connected and 0 otherwise and $\overrightarrow{e_x}$ is the vector of form (δ_{ix}), where δ_{ix} is the Kronecker function.

Matrix Forest Index (MFI) [12] This index can be described as the ratio of the number of spanning rooted forests such that y belongs to the tree rooted in x to the number of all spanning rooted forests of the network:

$$S = (I + L)^{-1}. \tag{16}$$

We can now implement and compare AUC scores of all indices described above on different networks. (See [27] for the code and results using Python's NetworkX [13], Pandas [14] and NumPy [15] in IPython Notebook [16])

3 Method

We begin by defining the test dataset to check the performance of the algorithm. We shuffle the set of existing edges and split it into n parts $E'_i,\ i \in \{1, \ldots, n\}$. It is said that $n = 10$ achieves the best complexity-precision tradeoff. Now we can use the set $\bar{E}_i = E'_i \cup \bar{E}$ as test dataset, where if $e \in E'_i$ then $class_e = 1$ and if $e \in \bar{E}$ then $class_e = 0$.

Now we need to construct a training dataset to train our classifier on it and then check its performance on the test dataset. We take the remaining set of edges E_i, shuffle it and split it into n parts $(E'_i)'_j$. To achieve balanced training dataset with the same amount of edges with class 0 and 1 we split the complement $\bar{E}_i$ into subsets of the same size as $(E'_i)'_j$. Now for each subset $(E'_i)'_j$ we take this

Table 1. AUC scores of local indices. AN - Word adjacencies network[18], CN - Neural network[19], Dp - Dolphin social network network[20], FB - American College football network[21], LM - Les Miserables network[22], PB - Books about US politics network[23], KC - Zachary's karate club network[24], UA - US air transportation system network[25]. Best results for given network are in bold.

Net	CN	SaI	JI	SoI	HPI	HDI	LHN1	PAI	AAI	RAI
AN	0.662	0.606	0.603	0.603	0.618	0.603	0.566	**0.744**	0.662	0.659
CN	0.844	0.797	0.790	0.790	0.805	0.779	0.725	0.750	0.861	**0.866**
Dp	0.779	0.774	0.779	0.779	0.763	0.780	0.762	0.619	**0.781**	**0.781**
FB	0.846	0.856	0.858	0.858	0.856	0.857	**0.859**	0.270	0.846	0.846
LM	0.910	0.882	0.880	0.880	0.847	0.878	0.820	0.776	0.918	**0.919**
PB	0.887	0.884	0.875	0.875	0.894	0.863	0.848	0.653	**0.897**	0.890
KC	0.700	0.636	0.607	0.607	0.712	0.593	0.600	0.712	0.726	**0.733**
UA	0.934	0.908	0.897	0.897	0.869	0.891	0.767	0.885	0.945	**0.951**

Table 2. AUC scores of global indices.

Net	KI	LHN2	ACT	CBL	RWR	MFI
AN	0.714	0.549	**0.742**	0.586	0.738	0.667
CN	0.852	0.717	0.738	0.848	0.725	**0.865**
Dp	0.799	**0.825**	0.760	0.791	0.649	0.804
FB	0.857	0.879	0.588	**0.885**	0.273	0.878
LM	**0.884**	0.813	0.863	0.825	0.794	0.867
PB	0.891	0.858	0.729	0.891	0.618	**0.899**
KC	**0.755**	0.610	0.666	0.739	0.618	0.749
UA	**0.920**	NaN	0.892	0.913	0.862	0.913

subset out of E_i, then calculate indices on the complement of the remaining set and mark classes as $class_e$, where if $e \in (E_i')_j'$ then $class_e = 1$ and if $e \in \bar{E}_i$ then $class_e = 0$. And to get balanced dataset, on each step we take edges from only one subset of $\bar{E}_i$ as examples of class 0.

After n steps of the above process, we get the training dataset of size $2|E_i|$. Now we can train our classifier on this dataset and then test its performance on the test dataset. After testing SVM with linear kernel, decision trees, k nearest neighbours and random forests it was observed that the best performance is achieved with classification using random forests (as suggested in [17] and also tested in [28]).

4 Results

We test the performance our method on proposed test datasets and measure its F1 score, precision, recall, AUC score and accuracy on the class 1 of edges removed from the network. We then compile all these results and take the mean and standard deviation and compare these scores for different networks. (See [28] for the code and results using Python's Scikit-learn [26])

Table 3. Scores of the algorithm performance on the class 1 of marked links (mean ± std).

Net	F1	Precision	Recall	ROCAUC	Accuracy
AN	0.027 ± 0.003	0.014 ± 0.002	0.579 ± 0.072	0.638 ± 0.032	0.696 ± 0.023
CN	0.041 ± 0.001	0.021 ± 0.000	0.811 ± 0.018	0.809 ± 0.008	0.808 ± 0.004
Dp	0.049 ± 0.008	0.025 ± 0.005	0.635 ± 0.082	0.704 ± 0.040	0.771 ± 0.035
FB	0.199 ± 0.025	0.116 ± 0.017	0.721 ± 0.044	0.831 ± 0.019	0.939 ± 0.011
LM	0.163 ± 0.021	0.090 ± 0.013	0.831 ± 0.068	0.875 ± 0.032	0.918 ± 0.013
PB	0.080 ± 0.009	0.042 ± 0.005	0.780 ± 0.080	0.812 ± 0.040	0.842 ± 0.012
KC	0.092 ± 0.026	0.049 ± 0.014	0.641 ± 0.145	0.717 ± 0.082	0.790 ± 0.043
UA	0.083 ± 0.008	0.044 ± 0.004	0.901 ± 0.024	0.910 ± 0.011	0.919 ± 0.009

5 Conclusion

The proposed approach of supervised learning for link prediction showed interesting results on the selected networks. It appears that the algorithm classifies most of the removed links correctly, but also classifies a lot of links that were not present in the network in the first place as the ones that were removed.

These results also show that there is difference in index space between the edges removed and edges from complement that differentiates removed edges within the complement.

References

1. Lü, L., Zhou, T.: Link prediction in complex networks: A survey. Physica A 390, 1150-1170 (2011)
2. Hanely, J. A., McNeil, B. J.: The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology 143, 29 (1982)
3. Salton, G., McGill, M. J.: Introduction to Modern Information Retrieval. McGraw-Hill, Auckland (1983)
4. Jaccard, P.: Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bull. Soc. Vaud. Sci. Nat. 37, 547 (1901)
5. Sørensen, T.: A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons. Biol. Skr. 5, 1 (1948)
6. Ravasz, E., Somera, A. L., Mongru, D. A., Oltvai, Z. N., Barabási, A.-L.: Hierarchical organization of modularity in metabolic networks. Science 297, 1551 (2002)
7. Leicht, E. A., Holme, P., Newman, M. E. J.: Vertex similarity in networks. Phys. Rev. E 73, 026120 (2006)
8. Barabási, A.-L., Albert, R.: Emergence of scaling in random networks. Science 286, 509 (1999)
9. Adamic, L. A., Adar, E.: Friends and neighbors on the web. Soc. Networks 25, 211 (2003)
10. Zhou, T., Lü, L., Zhang, Y.-C.: Predicting missing links via local information. Eur. Phys. J. B 71, 623 (2009)
11. Katz, L.: A new status index derived from sociometric analysis. Psychmetrika 18, 39 (1953)
12. Chebotarev, P., Shamis, E. V.: The matrix-forest theorem and measuring relations in small social groups. Autom. Remote Control 58, 1505 (1997)
13. Hagberg, A. A., Schult, D. A., Swart, P. J.: Exploring network structure, dynamics, and function using NetworkX. Proceedings of the 7th Python in Science Conference 11–15 (2008)
14. McKinney, W.: Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference, 51–56 (2010)
15. van der Walt, S., Colbert, S. C., Varoquaux, G: The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science & Engineering, 13, 22–30 (2011)
16. Pérez, F., Granger, B. E.: IPython: A System for Interactive Scientific Computing. Computing in Science & Engineering, 9, 21–29 (2007)
17. Cukierski, W., Hamner, B., Yang, B.: Graph-based Features for Supervised Link Prediction. Proceedings of International Joint Conference on Neural Networks, San Jose, California, USA (2011)
18. Newman, M. E. J. Phys. Rev. E 74, 036104 (2006) Available at: `http://www-personal.umich.edu/~mejn/netdata/`.

19. Watts, D. J., Strogatz, S. H. Nature 393, 440–442 (1998) Available at: http://www-personal.umich.edu/~mejn/netdata/.
20. Lusseau, D., Schneider, K., Boisseau, O. J., Haase, P., Slooten, E., Dawson, S. M. Behavioral Ecology and Sociobiology 54, 396–405 (2003) Available at: http://www-personal.umich.edu/~mejn/netdata/.
21. Girvan, M., Newman, M. E. J. Proc. Natl. Acad. Sci. USA 99, 7821–7826 (2002) Available at: http://www-personal.umich.edu/~mejn/netdata/.
22. Knuth, D. E. The Stanford GraphBase: A Platform for Combinatorial Computing, Addison-Wesley, Reading, MA (1993) Available at: http://www-personal.umich.edu/~mejn/netdata/.
23. Krebs, V. Datasets. Available at: http://www-personal.umich.edu/~mejn/netdata/.
24. Zachary, W. W.: An information flow model for conflict and fission in small groups. Journal of Anthropological Research 33, 452–473 (1977) Available at: http://www-personal.umich.edu/~mejn/netdata/.
25. Batageli, V., Mrvar, A. Pajek datasets. Available at: http://vlado.fmf.uni-lj.si/pub/networks/data/default.htm.
26. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830 (2011)
27. http://github.com/libfun/...
28. http://github.com/libfun/...