Link Prediction in Foursquare Social Network

Francisco Izaguirre
fizaguir@stanford.edu
Thawsitt Naing
thawsitt@stanford.edu

Stanford University

1 Introduction

Link prediction is a classic problem in networks of great practical relevance – it can suggest to users who they should follow in a social network, which song they might like based on their listening history, etc. Our proposed project is to perform link prediction on Foursquare Social Network, which is a bipartite network between users and venues they checked in at.

The diversity of the data set makes this a very interesting network to analyze. Given the current state of the Foursquare social network, how accurately can we predict future edges between users and venues? Real work networks can be vast and dynamic. Waiting for computation times to compute over days may simply not be an option. With that in mind, how well can we predict missing links when we sample from our large graph? What is the accuracy, precision, and recall based on the network partitions sampled? These are the questions we aim to answer at the end of our research project.

2 Relevant Prior Work

Liben-Nowell et al. formalized link prediction problem in their paper and performed link prediction on academic co-authorship network [1]. They used a number of similarity metrics such as graph distance, common neighbors, Jaccards Coefficient, Adamic/Adar (Frequency-Weighted Common Neighbors), Preferential Attachment, Katz (Exponentially Damped Path Counts), and rooted PageRank. These methods constitute the foundation of link prediction algorithms and provide a solid starting point for our project.

Boyao Zhu et al. explored the potential of the link prediction in weighted networks [2], which is directly relevant to our project. They proposed Weighted Mutual Information (WMI) model which combines both structural features and link weights to offer improved feature calculations for Weighted Common Neighbor (WCN), Weighted Adamic-Adar (WAA), and Weighted Resource Allocation (WRA). These methods are essentially the same as those used by Liben-Nowell et al., but they consider edge weights as an additional input in the calculation of similarity scores.

3 Methodology

3.1 Data Collection Process

Our dataset is made up of five files.

- `users.dat`: consists of a set of users such that each user has a unique id and a geospatial location (latitude and longitude) that represents the user home town location.
- `venues.dat`: consists of a set of venues (e.g., restaurants) such that each venue has a unique id and a geospatial location (latitude and longitude).
- `checkins.dat`: marks the checkins (visits) of users at venues. Each checkin has a unique id as well as the user id and the venue id.

- `socialgraph.dat`: contains the social graph edges (connections) that exist between users. Each social connection consists of two users (friends) represented by two unique ids ($first\_user\_id$ and $second\_user\_id$).
- `ratings.dat`: consists of implicit ratings that quantifies how much a user likes a specific venue.

The core of our link prediction algorithms focus on `checkins.dat`, which constructs a bipartite graph between users and venues. Other files will be added later for edge weights and additional heterogeneous data. We used a Python script to process the data source, yielding white-space delimited mappings between user and venue nodes that can be imported as an undirected graph with SNAP.

We noticed that the graph is extremely sparse and the majority of the nodes skew towards low degrees. In addition, most of the users nodes have a degree of 1 (i.e. only checked into one venue), and there are a lot of islands disconnected from the rest of the graph. This is problematic because we cannot calculate graph distance on these nodes. In addition, this will cause our link prediction algorithms to have an extremely low accuracy since most of the links predicted will not exist in the first place (since 60% of users only checked into one venue). In order to solve this problem, we sampled our data set using the following reduction methods:

1. Retrieve the largest weakly connected component, which includes approximately 85% of nodes.
2. Remove user and venue nodes that have a degree less than d, where d is the minimum degree for each node.
3. Repeat step 2 until all nodes in the network have at least degree d.

Based on the minimum degree d, this reduced our network size down to the connected sample networks listed in table 2.

Minimum Degree	# of nodes	# of edges	# of users	# of venues
2	203590	548331	177521	26069
3	91654	334981	79670	11984
4	42779	193949	36650	6129
5	19576	104805	16477	3099
6	8094	49892	6678	1416
7	3156	21574	2537	619
8	613	4909	525	88
9	259	2154	205	54

Table 1: Network sample size per Minimum Degree

3.2 Problem Definition

Given a social network $G = \langle V, E \rangle$ in which each edge $e = \langle u, v \rangle \in E$ represents an interaction between a user node u and a venue node v, we want to accurately predict future edges based on the current state of the network. First, the network is divided into training and testing sub-graphs.

All link prediction methods in this paper are score-based: all pairs of node $\langle u, v \rangle$ are assigned a score(x, y) based on input graph G. Then, the scores are sorted in descending order such that pairs with higher scores are more likely to be connected than those with lower scores.

We subtract the edges found in the training sub-graph from the ranked list of pairs. This results in predicted new links, in decreasing order of confidence. We take the top n pairs and evaluate the performance based on how many of those pairs appear as edges in the test sub-graph.

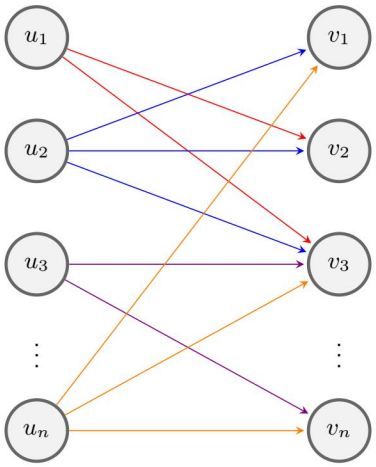


Fig. 1: Graph Representation of Checkins Network

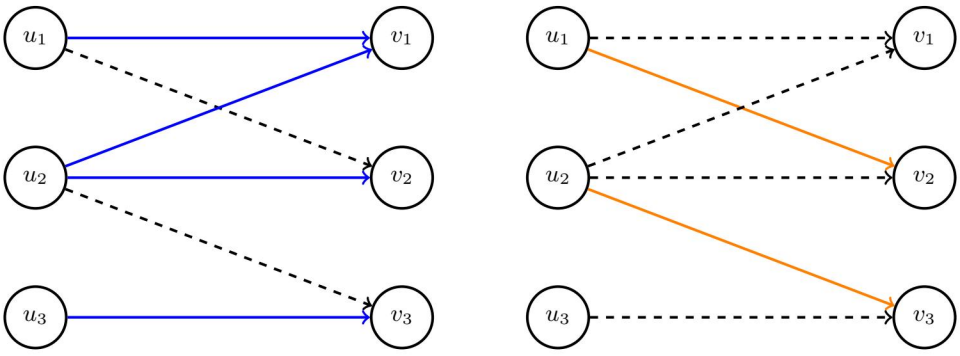


Fig. 2: Network edges split into training set (left, blue edges) and test set (right, orange edges)

3.3 Algorithm

After sampling nodes from the data-set, we need to split the data-set into train and test sets. However, we can't split it by nodes because train and test sets need to contain the same nodes. Therefore, we split it by edges. Given a sampled data-set, we remove random edges from the graph until we have removed 20% of the edges. We store the removed edges as a test set. Therefore, we now have a training set with 80% of the original edges and a test set with 20% of the edges.

Then, we compute the similarity score of all possible (user, venue) pairs. Next, we choose the top 20% with the highest scores and see how many of them appear in the test set. The metrics we used are listed in detail below.

3.3.1 Distance: In this metric, we define score(x, y) to be negative of the shortest path distance from x to y because we want the shorter paths to have higher scores. In order to compute the shortest path efficiently, we exploit the small-world property of the social network and apply expanded ring search.

Specifically, we initialize $S = \{x\}$ and $D = \{y\}$. In each step, we either expand set S to include its members neighbors (i.e., $S = S \cup \{v | \langle u, v\rangle \in E \cap u \in S\}$) or expand set D to include its members neighbors (i.e., $D = D \cup \{v | \langle u, v\rangle \in E \cap u \in D\}$). We stop whenever $S \cap D \ != \emptyset$. The number of steps taken so far gives the shortest path distance. For efficiency, we always expand the smaller set between S and D in each step.

3.3.2 Common Neighbors: The common-neighbors predictor captures the notion that two strangers who have a common friend may be introduced by that friend. In a unipartite network, it is usually computed as

$$\text{Score}(x, y) = |\Gamma(x) \cap \Gamma(y)|$$

However, since our network is bipartite, we adapt this to

$$\text{Score}(x, y)^{\text{user}} = \max(|\Gamma(x) \cap \Gamma(x_i)|) \qquad \text{for all } x_i \in \Gamma(y)$$

$$\text{Score}(x, y)^{\text{venue}} = \max(|\Gamma(y) \cap \Gamma(y_i)|) \qquad \text{for all } y_i \in \Gamma(x)$$

where $\Gamma(x)$ = the neighbors of user x and $\Gamma(y)$ = the neighbors of venue y. Therefore, we end up with two metrics: common neighbors of users and common neighbors of venues.

3.3.3 Adamic/Adar (Frequency-Weighted Common Neighbors): Adamic/Adar refines the simple counting of common features by weighting rarer features more heavily. For example, two users who have both visited a less-known venue are probably more similar than those who have both visited a popular tourist attraction. It is usually computed as

$$\text{Score}(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log|\Gamma(z)|} \qquad \text{where } |\Gamma(z)| \text{ is the degree of } z.$$

However, since our network is bipartite, we adapt this to

$$\text{Score}(x, y)^{\text{user}} = \sum_{z \in A} \frac{1}{\log|\Gamma(z)|} \qquad \text{where } A = \max(|\Gamma(x) \cap \Gamma(x_i)|) \quad \text{for all } x_i \in \Gamma(y)$$

$$\text{Score}(x, y)^{\text{venue}} = \sum_{z \in B} \frac{1}{\log|\Gamma(z)|} \qquad \text{where } B = \max(|\Gamma(y) \cap \Gamma(y_i)|) \quad \text{for all } y_i \in \Gamma(x)$$

3.3.4 Preferential Attachment: Preferential Attachment implies that users with many friends tend to create more connections in the future. In our context, we predict that a more outgoing user (i.e. a user node with a high degree) is more likely to visit a venue which is more popular (i.e. a venue node with a high degree). In this metric, we calculate the score as the product of the degrees of both nodes.

$$\text{Score}(x, y) = |\Gamma(x)| \cdot |\Gamma(y)|$$

3.3.5 Katz (Exponentially Damped Path Counts): The Katz-measure is a variant of the shortest-path measure. The idea behind the Katz-measure is that the more paths there are between two vertices and the shorter these paths are, the stronger the connection. It is defined as

$$\text{Score}(x, y) = \sum_{l=1}^{5} \beta^l \cdot |\text{Path}^l_{x,y}|$$

where β = a constant that is exponentially damped by length
$|\text{Path}^l_{x,y}|$ = the number of possible paths of length l between x and y.

We choose $\beta = 0.05$ and compute $|\text{Path}^l_{x,y}|$ by doing a breath first search from node x. We stop at $l = 5$ because paths of length 3 or more contribute very little to the overall score. It's worth noting that the path length from a user to a venue is always an odd number since the graph is bipartite.

4 Results

We run our algorithm on three different sizes of sampled data set. For each sample, we calculate the precision and recall of each of the predictors. As discussed in the paper by Liben-Nowell et. al. [1], many edges fail to form for reasons outside the scope of the network. Therefore, the raw performance of our predictors is relatively low. To more meaningfully represent predictor quality, we compare the results against a random predictor baseline which simply randomly selects (user, venue) pairs that do not occur in the training network. The results are shown in Table 2 below. A **bold** entry represents the highest performance.

Similarity Metric	$d > 6$ (3156 nodes)			$d > 7$ (613 nodes)			$d > 8$ (259 nodes)		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
Random Predictor	1.00	0.27%	19.84%	1.00	2.19%	20.59%	1.00	3.52%	18.14%
Distance	1.27	0.35%	25.27%	0.72	1.44%	14.78%	0.27	0.95%	4.88%
Common Neighbors (user)	2.90	0.79%	57.44%	0.98	2.08%	20.18%	0.36	1.26%	6.51%
Common Neighbors (venue)	3.24	0.80%	64.30%	2.02	4.57%	41.59%	1.15	4.07%	20.93%
Adamic/Adar (user)	3.23	0.88%	64.14%	0.99	2.15%	20.29%	0.41	1.45%	7.44%
Adamic/Adar (venue)	3.27	0.89%	64.84%	2.04	4.47%	**42.10%**	1.12	3.93%	20.23%
Preferential Attachment	2.89	0.80%	57.30%	1.91	4.19%	39.25%	1.24	4.38%	**22.56%**
Katz (β=0.005)	3.34	0.91%	**66.25%**	1.73	3.58%	35.68%	0.55	1.94%	10.00%

Table 2: Results: Relative accuracy, precision, and recall for each dataset sample of minimum node degree d

As seen in Table 2, in the smallest sample of 259 nodes, the relative accuracy of some predictors is even lower than the baseline. This shows that there is not enough information in the network to predict the missing links. As the sample size increase, the relative performance of the predictors begin to improve. In the 3156-node network, all predictor metrics outperform the random baseline by a factor of 2.8 on average. We expect this trend to continue in larger sampled networks.

The precision values of our predictors are very low. We tried to make it better by taking a smaller percentage of the top scores as our predictions. However, in that case, both precision and recall drops closer to

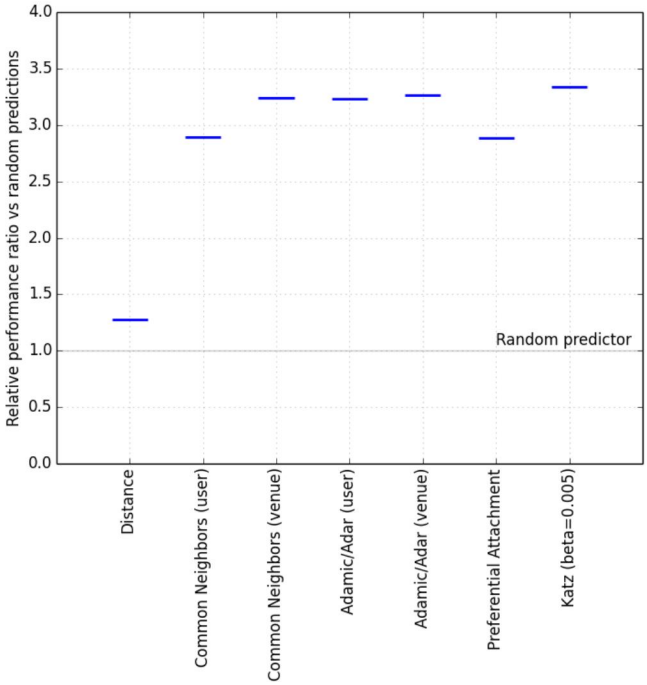


Fig. 3: Relative performance of each metric in 3156-node network ($d > 6$)

0%. This is the result of a sparse input network. Nonetheless, we expect better recall values as the sample size increases. For example, in 3156-node graph, we are able to predict 66.35% of missing edges with the Katz matrix.

In addition, we can see how different similarity metrics perform relative to each other even in a sparse network. We found Common Neighbors (venues), Adamic/Adar (venue) and Katz to have consistently higher performance. Another interesting thing we found is that Adamic/Adar performance mirrors that of the common neighbor metric.

5 Conclusion

In this paper, we outline a variety of methods to extract proximity-based features and path-base features in order to calculate node scores and predict "missing" edges between our training set and testing set. We investigate the accuracy relative to a coin-flip, precision, and recall for varying network samples where each node has at least degree d. The recorded metrics were considerably low in precision, but drastically improved in recall as the d decreased and the network consisted of more nodes.

Real work networks can be vastly large. Feature extraction has proven to have very high runtimes, to the point that by the time a similarity metric is calculated for each node, the dynamic nature of real work networks could make the scores irrelevant. We demonstrate the relative accuracy, precision, and recall of predicting missing links where we sample a large network and calculate similarity scores under 2 hours.

6 Future Steps

Computation time was a growing concern for us. Though drastically reduced the size of our checkins dataset after filtering out nodes that were not connected to the largest weakly connected component and had less than the minimum degree threshold d, we encountered exceedingly long runtimes for computing similarity

scores on dataset partitions where $d < 6$, consisting of more than 8000 nodes. These computations took more than 48 hours to execute and sometimes would hit memory heap limitations, causing our laptops to force restart and discard the running process.

It would be interesting to run these metrics for network samples where $d < 6$. To do so, consider the following techniques to remedy the computational bottleneck:

- Multi-threading. A divide-and-conquer approach to allow a single virtual process dedicate itself to calculate similarity scores on a fixed range of user IDs.

- Utilizing dedicated computer clusters. Offloading the work to one of Stanfords machine clusters would remarkably boost our computational resources and help execute processes faster. Alternatively, could also dedicate GPU cores for running calculations, but like we mentioned above, the heap limitation is also of concern.

- Implementing approximation methods. After researching some potential approximation techniques for our metrics of concern, we have been considering running with more of these ideas. However, there are several contingencies required for a network to approximate these metrics accordingly. These restraints include network size, clustering coefficient, degree distribution, etc.

We believe that running these calculation on a larger portion of the network would yield higher accuracy and a clearer trend on which scores work best for for missing edge detection.

References

1. Liben-Nowell, David, and Jon Kleinberg. 'The Link Prediction Problem for Social Networks.' Proceedings of the Twelfth International Conference on Information and Knowledge Management - CIKM 03, 2003. doi:10.1145/956958.956972.

2. Zhu, Boyao, and Yongxiang Xia. 'Link Prediction in Weighted Networks: A Weighted Mutual Information Model.' Plos One 11, no. 2 (May 2016). doi:10.1371/journal.pone.0148265.

3. WSong, Han Hee, et al. 'Scalable Proximity Estimation and Link Prediction in Online Social Networks.' Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference - IMC 09, 2009, doi:10.1145/1644893.1644932.

4. H. Chen, X. Li and Z. Huang, "Link prediction approach to collaborative filtering," Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '05), Denver, CO, 2005, pp. 141-142. doi: 10.1145/1065385.1065415

5. N. Benchettara, R. Kanawati and C. Rouveirol, "Supervised Machine Learning Applied to Link Prediction in Bipartite Social Networks," 2010 International Conference on Advances in Social Networks Analysis and Mining, Odense, 2010, pp. 326-330. doi: 10.1109/ASONAM.2010.87

6. Holme, P., Liljeros, F., Edling, C.R., Kim, B.J.: On network bipartivity. Phys. Rev. E 68, 66536673 (2003)

7. O. Allali, C. Magnien and M. Latapy, "Link prediction in bipartite graphs using internal links and weighted projection," 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Shanghai, 2011, pp. 936-941. doi: 10.1109/INFCOMW.2011.5928947

8. Wang, Liang, et al. Robustness of Link-Prediction Algorithm Based on Similarity and Application to Biological Networks. Current Bioinformatics, vol. 9, no. 3, 2014, pp. 246252., doi:10.2174/1574893609666140516005740.

9. Newman, M. E. J. Clustering and Preferential Attachment in Growing Networks. Physical Review E, vol. 64, no. 2, 2001, doi:10.1103/physreve.64.025102.

10. Dong, Yuxiao, et al. "Link Prediction and Recommendation across Heterogeneous Social Networks." 2012 IEEE 12th International Conference on Data Mining, 2012, doi:10.1109/icdm.2012.140.

Appendix

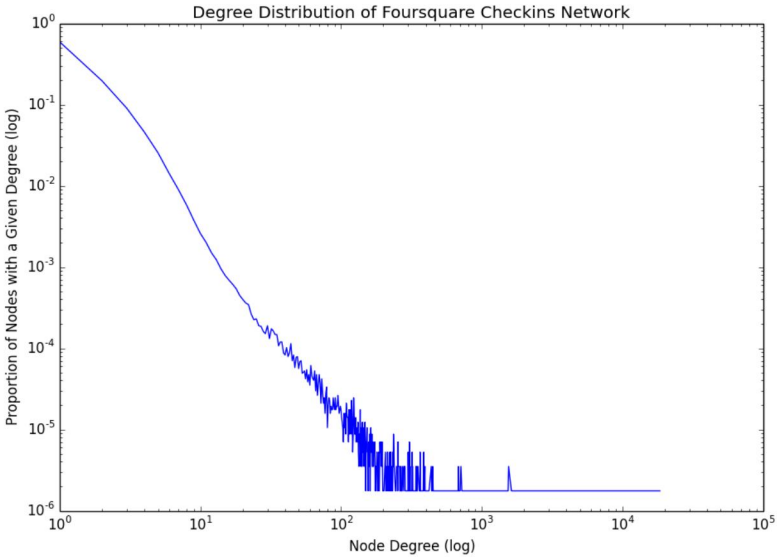
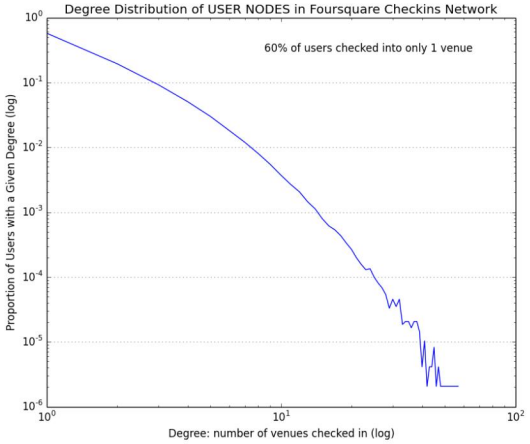
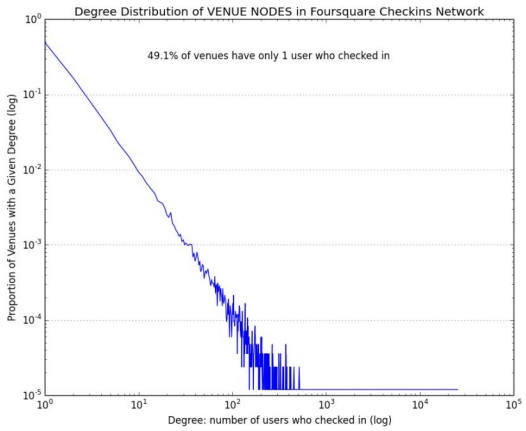


Fig. 4: Degree distribution of Foursquare Checkins Network



(a) Degree distribution of user nodes



(b) Degree distribution of venue nodes

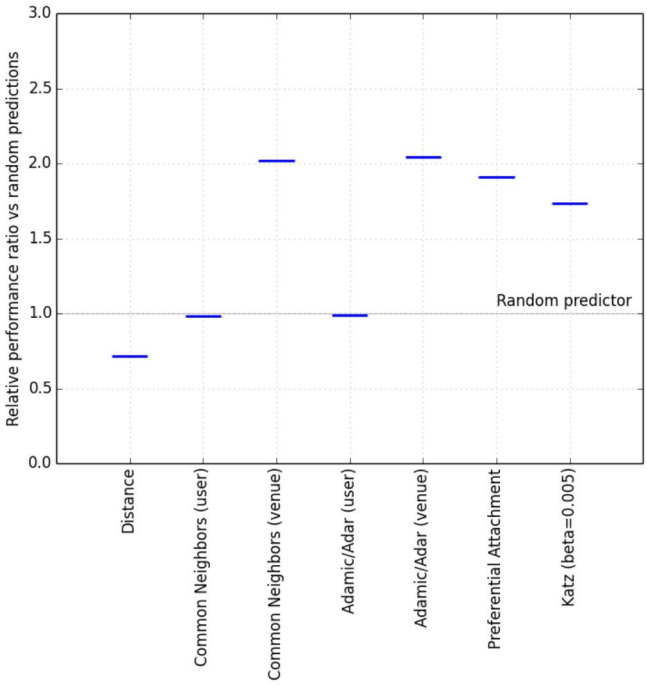


Fig. 5: Relative performance of each metric in 613-node network ($d > 7$)

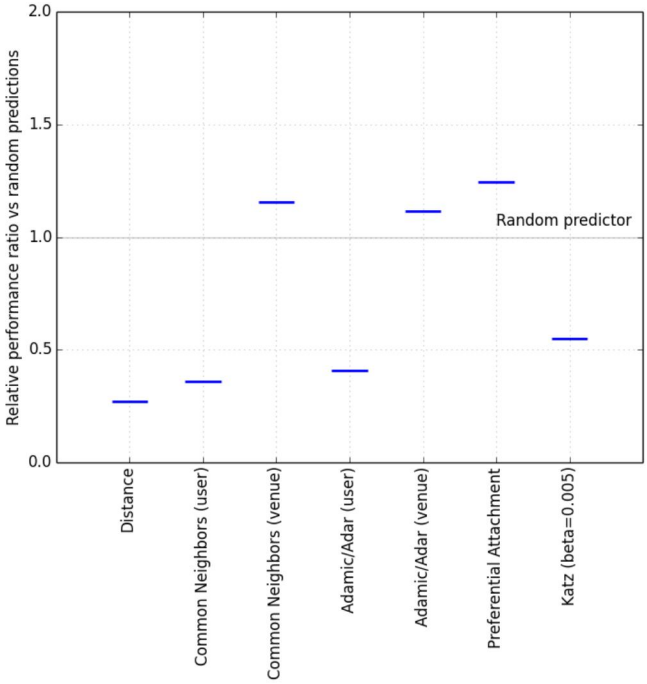


Fig. 6: Relative performance of each metric in 259-node network ($d > 8$)