

Final Report

Eric L. Lee, Shu-Hao Chang, Yu-Cheng Weng, Can Jiang

November 13, 2019

1 Abstract

In our last proposal report, we planned to perform link prediction on network structured data. By now, we have collected data and established our evaluation framework. We implemented 6 baseline models in 8 different datasets, performed data analysis and visualized our datasets. In the following two weeks, we will be focusing on implementing factorization models and state-of-the-art methods as well as further improving our current results.

2 How to Reproduce Our Result

We believe that reproducibility is very important when it comes to research. Thus, we carefully recorded all of our experiment results on Github. All the results presented in the following sections can be easily reproduced using this github repository:

https://github.com/miamiasheep/Purdue_ML_Course_Project

To reproduce all of our result (including data analysis part), simply type the following commands:

```
python script.py
```

To reproduce the result of certain data sets, type the following command:

```
python main.py --input [file1 ,file2 ,...] --goal [auc/f1]
```

To draw the graph of a network, type the following command:

```
python main.py --input [file] --draw [N(Integer)]
```

When N is larger, the sample network will be larger.

3 Data Analysis

3.1 Data Set

We collect 8 different datasets which are all unique and have different characteristics. Table 1 shows basic statistics of each data set.

3.1.1 Facebook

This is a dataset downloaded from SNAP(Staford Network Analysis Project)[4]. The data is an egonet in facebook. Each node represents an account in Facebook and each link represents friendship in Facebook.

3.1.2 Power

Power is an electrical grid of western US[9].

3.1.3 NS

NS is a collaboration network of resarchers who publish paper on network sciences [6].

3.1.4 PB

PB is a network of US political blogs.[1]

3.1.5 Router

Router is a router-level Internet.[7]

3.1.6 USAir

USAir is a network of US air lines. [2]

3.1.7 Yeast

Yeast is a protein-protein interaction network in yeast.[8]

3.1.8 Celegan

Celegan is a neural network of C. elegans.[9]

3.2 Visualizing Our Data Set

We implemented a tool to visualize our datasets. And all of our visualization graphs can be seen in the following link:

https://github.com/miamiasheep/Purdue_ML_Course_Project/tree/master/graph

data set	nodes	edges	average degree	max degree
Facebook	4039	88234	21.85	1045
Power	4941	6595	1.33	19
NS	1461	2742	1.87	34
PB	1222	16714	13.68	351
Router	5022	6258	1.26	106
USAir	332	2126	6.40	139
Yeast	2375	11683	4.91	118
Arxiv	18772	198110	10.55	504
Celegan	297	2148	7.23	134

Table 1: Basic Information of different data set

Since the network is too big and it is hard to find any clue regarding the network structure if we draw all the edges and nodes, we only sampled a small portion of the network on every dataset.

By visualizing our data sets, we can see that the characteristic of each data is actually very different. Let’s take Facebook (figure1) and Router (figure2) as examples. Facebook dataset has multiple triangles in the middle of the network. In contrast, Router has no triangle in the data set.

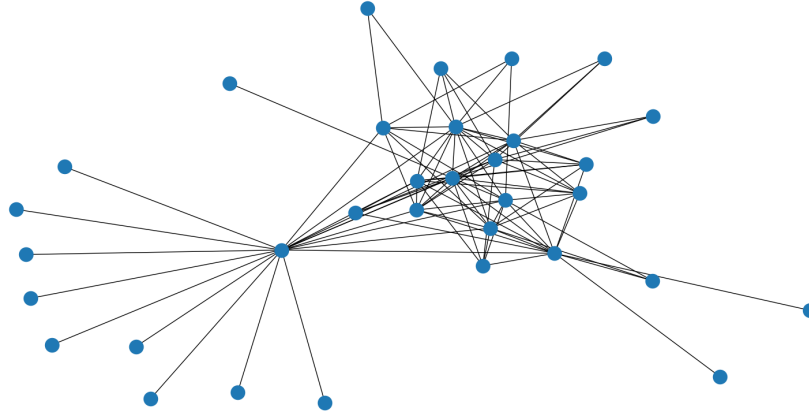


Figure 1: facebook

4 Evaluation and Result

4.1 Baselines

For the following sections, we denote x and y as two nodes, between which we want to predict if there is a link. And we denote the set of neighbors of x as $N(x)$. We implemented six very

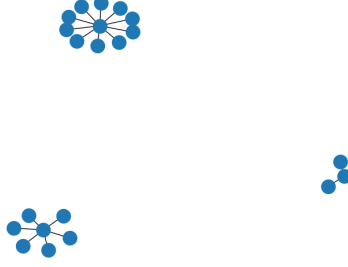


Figure 2: Router

common heuristics. And in table 2, we organized the formulas of the six heuristics.

4.1.1 Common Neighbors(CN)

It is a very simple heuristic. The intuition is that the more common neighbors the two nodes have, the more likely that there is a link between them. The score can be calculated as $|N(x) \cap N(y)|$.

4.1.2 Jaccard(JC)

It is very similar to common neighbors except that Jaccard considers the fact that two higher degree nodes unsurprisingly tend to have more common nodes. Thus, if two lower degree nodes have common neighbors, maybe they really have a strong relationship. The score can be calculated as $\frac{|N(x) \cap N(y)|}{|N(x) \cup N(y)|}$

4.1.3 Adamic Adar(AA)

It is also very similar to common neighbors. It assumes that a low degree common neighbor have more contribution to the probability of a link between x and y. Thus, the score divides the weight of each common neighbor by the logarithm of its degree. It is actually very similar to TFIDF. And we can calculate the Adamic Adar by $\sum_{z \in N(x) \cap N(y)} \frac{1}{\log(|N(z)|)}$

4.1.4 Total Neighbors(TN)

It is the simplest baseline, which is just summing up the total size of neighbors of x and y. It can be calculated as $|N(x)| + |N(y)|$

4.1.5 Preferential Attachment(PA)

It is assumed that a high degree node have more chances to have a link with other nodes. Preferential Attachment calculate the product of degree of x and y. It can be calculated as $|N(x)| * |N(y)|$

method	description
common neighbor(CN)	$ N(x) \cap N(y) $
Jaccard(JC)	$\frac{ N(x) \cap N(y) }{ N(x) \cup N(y) }$
Adamic and Adar(AA)	$\sum_{z \in N(x) \cap N(y)} \frac{1}{\log(N(z))}$
Total Neighbor(TN)	$ N(x) + N(y) $
Preferential Attachment (PA)	$ N(x) * N(y) $
Page Rank(PR)	$\log r_x + \log r_y$

Table 2: link prediction methods

4.1.6 Page Rank(PG)

It is very similar to PA. But instead of using the product of nodes' degrees, it uses the product of two nodes' scores derived from performing page rank on our network. Since the product can be very small, we use the sum of logarithm instead. Let r_x be the score of x after performing page rank and r_y be the score of y after page rank. It can be calculated as $\log r_x + \log r_y$

4.2 Evaluation

Evaluation is not trivial when it comes to link prediction problems. If we want to evaluate all pairs of nodes, it will need $O(N^2)$ predictions, where N is the size of the nodes. And even for a small graph with 1000 nodes in testing set, the evaluation time can be really long because we have to make $1000 * 999 / 2$ predictions. In fact, finding a fair and effective evaluation is also a research topic. Ryan and Nitech published a paper [5] about how to evaluate a link prediction problem. We choose the most common way to evaluate our model: Down-sampling negative samples. (By negative samples, we mean a pair of nodes with no link between them. In link prediction problems, negative samples are usually way more than positive ones.)

First, we randomly divided our edges into training, validation and testing. The ratio is 0.8:0.1:0.1. Training set is for training while validation set is only for grid search for the best parameters. And testing set is for evaluation of our model.

Second, we randomly sampled negative samples for validation and testing set. The sample size is equal to the set size.

Third, we train the models using training set, and tuned our model using validation set, and evaluate the models to our testing set. The higher score a pair gets, the higher the ranking it is and the more possible that there would be a link between them. For evaluation, we used the ranking metrics. We mainly used AUC-ROC score (area under Receiver Operating Characteristics curve, abbreviated to AUC here). AUC has a very good characteristic that the sampling AUC is actually an approximation of actual AUC even if we do down-sampling.

Data Set	CN	JAC	AA	PA	TN	PG
Celegans	0.8314	0.7669	0.8443	0.7554	0.734	0.7587
facebook	0.9882	0.9863	0.9893	0.8324	0.7345	0.8085
NS	0.9742	0.9747	0.9741	0.6803	0.5204	0.5206
PB	0.9152	0.8669	0.9176	0.908	0.8776	0.9094
Power	0.5965	0.5964	0.5965	0.528	0.5198	0.5536
Router	0.6109	0.6102	0.6111	0.9298	0.9198	0.9418
USAir	0.9509	0.9164	0.9626	0.9021	0.868	0.8971
Yeast	0.902	0.901	0.9029	0.8561	0.7926	0.8488

Table 3: AUC of the six baselines

Data Set	CN	JAC	AA	PA	TN	PG
Celegans	0.7407	0.5926	0.713	0.5694	0.5787	0.5833
facebook	0.9546	0.9312	0.9544	0.6742	0.5618	0.6455
NS	0.9524	0.9524	0.9524	0.5281	0.3939	0.3723
PB	0.8052	0.6927	0.801	0.7762	0.7223	0.7792
Power	0.1947	0.1947	0.1947	0.396	0.3518	0.3252
Router	0.2279	0.2279	0.2279	0.761	0.7574	0.7721
USAir	0.8278	0.7608	0.8612	0.799	0.7799	0.7608
Yeast	0.815	0.815	0.815	0.7024	0.6184	0.6944

Table 4: F1 score of the six baselines

We also implemented $f1@k$ score as an evaluation metric. For the parameter k , we tentatively set the k equal to the size of positive samples. For this k , recall is actually equal to precision. And $f1@k$ thus has a very intuitive physical meaning: the accuracy of finding all of the missing links given the size of missing links. In the future, we’ll try to set different k ’s and see the influence of them.

Also, something to note here is that some nodes may only exist in validation or testing set but not in training set. We discarded the pairs of nodes where at least one node is not in the training set when evaluating, i.e., We only considered pairs of which both nodes are in the training set.

5 Result and Discussion

Table 3 shows AUC’s of the six different baselines and Table 4 shows the F1 scores.

According to the result above, we can see that no heuristic method can outperform all other heuristics, and that Adamic Adar(AA), Jaccard, and common neighbors perform pretty well in the Facebook and NS dataset. However, for Power and Router dataset, the results of common neighbor based methods are pretty bad. This is not surprising because in Power

and Router dataset, there are so few triangles that the methods aren't able to achieve good performances.

To achieve a better result, we want to resort to machine learning approach in the following two weeks. Instead of designing a heuristic before actually seeing the data, we want to train our model using the existing link in the training data. The main problem of machine learning approach is to find the feature vector x for each node. We have surveyed two kinds of method in our proposal.

The first approach is using factorization model(a.k.a Matrix Factorization) [3]. We give each node a latent feature vector and use the inner product be the probability whether the node will have the link. This method has a great performance in recommendation problem (link prediction in bipartite graph). We want to test wheter it will have the same great performance when it comes to a much more general link prediction problem.

The second approach is to sample a set of subgraph and enumerate all the nodes in each subgraph. As a result, the adjacency matrix in each subgraph of these subgraph can be the feature vector for machine learning approach. In fact, there are two recent works proposed in KDD 2017 [10] and NIPS 2018 [11] using this method.

6 Matrix Factorization

In this section, we try to apply Matrix Factorization to link prediction.

When we do the experiments, we find a surprising fact: "Lots of people using matrix factorization in the wrong way". And the mistake is still existing in a very recent NIPS paper. To be more surprised, the paper was accepted in KDD 2017 and then submitted in NIPS 2018 uses this incorrect result and being accepted without notice the bug. We then contact the author and surprisingly find that we are the first one to find this bug. Because the paper is already cited by 50 papers, we are really surprised.

Through the conversation with the author, I found that the author is really smart and very willing to correct the result in the later publication. So, I believe it is still a very good paper with carefully experimentation and the mistake may be not so easy to find. So, we want to point out this mistake in the following section and our final project presentation.

6.1 Myth of Matrix Factorization

Matrix Factorization is extremely popular in solving recommendation problem which can be one of link prediction problem. The only difference is that recommendation problem only focus on the bipartite graph.

6.2 Zero Prediction Problem

6.3 Result

7 Ensemble Heuristics

8 Sampling Subgraph

References

- [1] ACKLAND, R. Mapping the us political blogosphere: Are conservative bloggers more prominent?
- [2] BATAGELJ, V., AND MRVAR, A. Usair data set. <http://vlado.fmf.uni-lj.si/pub/networks/data/>, 2006.
- [3] BELL, R., KOREN, Y., AND VOLINSKY, C. Matrix factorization techniques for recommender systems. *Computer* 42, 08 (aug 2009), 30–37.
- [4] LESKOVEC, J., AND KREVL, A. SNAP Datasets: Stanford large network dataset collection. <http://snap.stanford.edu/data>, June 2014.
- [5] LICHTNWALTER, R., AND CHAWLA, N. V. Link prediction: Fair and effective evaluation. In *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (Aug 2012), pp. 376–383.
- [6] NEWMAN, M. E. J. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E* 74, 3 (Sep 2006).
- [7] SPRING, N., MAHAJAN, R., WETHERALL, D., AND ANDERSON, T. Measuring isp topologies with rocketfuel. *IEEE/ACM Trans. Netw.* 12, 1 (Feb. 2004), 2–16.
- [8] VON MERING, C., KRAUSE, R., SNEL, B., CORNELL, M., OLIVER, S., FIELDS, S., AND BORK, P. Comparative assessment of large-scale data sets of protein-protein interactions. *Nature* 417 (06 2002), 399–403.
- [9] WATTS, D. J. Collective dynamics of small-world networks. *Nature* 393.
- [10] ZHANG, M., AND CHEN, Y. Weisfeiler-lehman neural machine for link prediction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2017), KDD ’17, ACM, pp. 575–583.
- [11] ZHANG, M., AND CHEN, Y. Link prediction based on graph neural networks. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems* (USA, 2018), NIPS’18, Curran Associates Inc., pp. 5171–5181.