A New Link Prediction Algorithm Based on Local Links

Juan Yang, Lixin Yang(✉), and Pengye Zhang

Key Lab of Intelligent Telecommunication Software and Multimedia,
Beijing University of Posts and Telecommunications,
100876 Beijing, China
yangjuan@bupt.edu.cn, ylxsyf@163.com

Abstract. Link prediction refers to estimating the possibility of the existence of non-existent links between the nodes. The link prediction algorithms based on local information merely consider nodes' attributes or a small amount of topology information about common neighbors. In this paper, we proposed a new measure motivated by the cohesion between common neighbors and the predicted nodes——LNL (Local Neighbors Link). Experiments show that, compared with four classical algorithms on seven real networks, LNL has the higher accuracy and robustness. Furthermore, we apply the link prediction algorithms into large-scale networks. We implement the LNL method in both MapReduce and Spark, the experiments show that the implementation by Spark has higher efficiency than using MapReduce.

Keywords: Link prediction · Complex network · Adjacent nodes · Parallel algorithm · Spark

1 Introduction

Link Prediction is a key direction in complex network research, refers to estimating the possibility of the existence of non-existent links between node pairs according to the known network structure information [1]. It has been applied to the prediction of unknown information on protein network [2], relationship recommendation of microblogging users [3], evaluation of network evolving mechanisms [4], recommend collaborations [5], classification in partially labelled networks [6] and other fields.

The majority similarity-based link prediction algorithms are classified into several aspects: based on local, global and quasi-local information. Methods based on local information have lower time complexity, yet lower prediction accuracy, and may perform poorly on networks with different clustering coefficients. Global information considers longer paths of network instead of only nearest neighbors for sufficient information. Random walk link prediction algorithms calculate the probabilities of the

This work is supported in part by the National Key Basic Research and Department (973) Program of China (No. 2013CB329606), and the National Natural Science Foundation of China (No. 71231002, 61375058).

X. Xiao and Z. Zhang (Eds.): WAIM 2015, LNCS 9391, pp. 16–28, 2015.
DOI: 10.1007/978-3-319-23531-8_2

source node to the target node as the similarity between this node pair. The supervised learning methods transfer the link prediction problem into a classification problem. Both random walk and supervised learning methods are difficult to be parallelized.

The above works can handle specific networks; the approaches based on local information have the least time complexity. Considering the rapid growth of network, we focus on local information which is easy to be implemented and has low time cost. Meanwhile, link prediction has a broad applicability, the local index is simple but can be used in many ways, that is to say, it is worthy of improving its performance. Furthermore, we apply the link prediction algorithms based on common neighbors with adjacency information on large-scale networks. Thus, our researches focus on following problem:

1. Present a new approach LNL based on local links with better performances in accuracy and robustness.
2. Design the parallel LNL link prediction method, implement the parallel algorithm in both MapReduce and Spark, and compare their efficiency.

2 Related Work

Previous researches have investigated some classical algorithms: LP algorithm was introduced by Lü L et al. [7] with time complexity $O(Nk^3)$ (where N is the number of vertexes, k is time complexity to traverse the neighborhood of a node), this method based on CN method yet considering more path information within length 3. Katz [8] considered all the paths information which the shorter path has a higher weight. The usage of path information can make a good prediction for sparse networks of both Katz and LP methods, yet the computational complexity is high. Prediction with random walk has several indices including LWR [9], SWR [9] et al., and the integral accuracy is high. Mohammad A H et al. [10] transformed link prediction problem into classification problem, the method extracted a set of features of network as input for supervised learning for link prediction. Fire M et al. [11] took topological features for supervised learning, and ranked the importance of each feature.

Comparing with the above methods, the approaches based on local information have the least time complexity and broad applicability. This paper focuses on the algorithms based on local information including: Common Neighbors (CN), Jaccard [12], Adamic-Adar (AA) [13], PA [14], RA [15]. CN takes the number of common neighbors as the existing possibility of nodes. On the basis of CN, AA and RA improve the accuracy by assigning the less connected neighbors more weight. The similarity index AA is defined as: $S_{xy} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} 1/logk(z)$, $k(z)$ represents the degree of node z, $\Gamma(x)$ is the neighbor of node x. AA refines the nodes with lower degree have more weight. RA has a similar form with AA while it decays faster, RA is motivated by the resource allocation process taking place on networks with definition: $S_{xy} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} 1/k(z)$. Compared with AA, RA performs better [4] in most cases. Connecting possibility of x and y is

proportional to their degree in PA, namely $S_{xy} = k(x) \cdot k(y)$. Noted that PA requires less information which has the least computational complexity, yet the performance is very poor in the highly clustered network.

Recent research found that clustering coefficient of networks has a large impact on link prediction algorithms' applicability [4], some algorithms focus on topological characteristic have been developed. In 2011, Dong Y. X et al. [16] designed a new algorithm IA exploiting the interactions between common neighbors defined as:$S_{xy} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} e_z / k(z)$, where e_z refers to the links between z and other common neighbors of x and y. It proved well performance while maintain low time complexity. Another method LCP accounts for the singular topology was proposed [17], the connecting possibility of node pairs is strictly limited, it has a great accuracy in precision, but not satisfactory in overall accuracy evaluation.

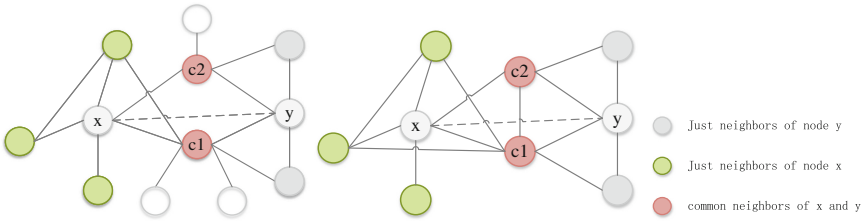
Classified similarity indexes cannot distinguish the importance of common neighbors when the node pair shares no more than two common neighbors while it is common in many networks. This paper puts forward a new index based on local neighbors' link degree——LNL (Local Neighbors Link) index, considering both attribute and topological features between the target points' adjacent nodes and their common neighbors. Extensive experiments on disparate real world networks demonstrate that LNL outperformed existing algorithms. In the next section, we will present the LNL index in detail. In Sect. 4, the parallel implementation is designed for link prediction algorithms. In Sect. 5, the experimental result of LNL will be compared with previous methods; compare the parallel performance between MapReduce and Spark.

3 Local Neighbors Link Method

Given an undirected network G (V, E), where V presents the set of nodes, E is the set of edges, e_{wv} refers to the link between w and v. There is no loopback in the network. Link Prediction refers to calculate the possibility of the existence of non-existent links between node pairs.

In Fig. 1, for the classical algorithms CN, AA and RA, c1 and c2 are treated respectively the same weight in Fig. 1 (a) and (b), therefore S_{xy} is same in Fig. 1 (a) and (b). However, we can observe from Fig. 1 that (take c1 as examples):

(1) In Fig. 1 (a), c1 has total four links with neighbors of x and y (include x and y), yet in Fig. 1 (b) c1's friends are all neighbors of x and y (include x and y). That is to say, c1 in Fig. 1 (b) may have more contact with x and y than in Fig. 1 (a).
(2) Walk from x to y, there exist three paths passed c1 that length less than three in Fig. 1 (a), in contrast, there exist five paths passed c1 in Fig. 1 (b), namely, x and y have more possibilities to connect each other by passing c1 in Fig. 1 (b).



(a) Network with less links (b) Network with more links

Fig. 1. Example network: x and y are the predicted nodes, c1 and c2 are the common neighbors of x and y, S_{xy} is the link probability of x and y.

LNL is motivated by the cohesion between common neighbors and the predicted nodes. The predicted node pair is x and y, c is one of their common neighbors; $k(c)$ is the degree of c. The basic idea is that if c has more links with the predicted nodes occupied $k(c)$, more weight will be assigned to c.

It should cover the above observation: (1) if c has more common friends with the predicted nodes occupied its total friends, meaning that it may have more contact with the predicted node pairs. (2) If c has more common friends, refers to c will generate more possibilities for the connections between x and y, for it provides more paths within 3 steps. Thus the weight of c is defined as:

$$w(c) = \frac{\sum_{v\in\Gamma(x)\cup x}\delta(c,v) + \sum_{w\in\Gamma(y)\cup y}\delta(c,w)}{k(c)} \tag{1}$$

Where $\Gamma(x)$ is the set of neighbor of node x, $\Gamma(y)$ is the set of neighbors of node y, the $\delta(w,v)$ represents whether there exist a link between w and v which is defined as:

$$\delta(w,v) = \begin{cases} 0 & e_{wv} \notin E \\ 1 & e_{wv} \in E \end{cases} \tag{2}$$

If there is no link between w and v, $\delta(w,v) = 0$, conversely $\delta(w,v) = 1$. Compared Eq. 1 with existing CN, AA and RA, a main difference in assigning weight for c is that Eq. 1 considering second-order neighbors information instead of merely nearest common neighbors.

S_{xy} is the accumulation of each common neighbor c, the definition of LNL index as:

$$S_{xy} = \sum_{c\in\Gamma(x)\cap\Gamma(y)} w(c) \tag{3}$$

In Eq. 3 suggests that x and y have more common friends, S_{xy} will get higher scores by accumulating each weight of common neighbors. Take Fig. 1 as an example, S_{xy} is calculated by LNL index as:

Figure 1 (a): $S_{xy} = w(\mathrm{c1}) + w(\mathrm{c2}) = (2+2)/6 + (1+2)/4 = 17/12$;
Figure 1 (b): $S_{xy} = w(\mathrm{c1}) + w(\mathrm{c2}) = (4+3)/6 + (2+3)/4 = 29/12$;

The result from (b) is greater than (a), the LNL is proved to distinguish the possibility of (a) and (b), while other indices such as CN, AA and RA are generally not available in this situation.

4 Parallel Implementation

To apply link prediction into large complex network, this paper design and implements a parallel method based on existing parallel method [19]. Existing parallel link prediction algorithm based on local information using MapReduce [18, 19] is proposed in 2012, this method can process large scale network of millions nodes, it proved to be $O(N/U)$ time complexity, where N is the number of nodes, U is the number of processing units. However, it doesn't give clear solutions to dealing the link prediction with additional common neighbors' information, thus we design a pre-process of carrying additional attributes for common neighbors. Besides, MapReduce is not good at multiple job execution for it will cost more starting time. A parallel programming model named Spark [20] will be used to implement the parallel link prediction and compared its efficiency with the implementation by MapReduce.

4.1 Spark

In MapReduce, each job need to reload the data from the disk which produced by Map task, it is time-consuming. While Spark outperforms MapReduce in iterative jobs for its memory computing which can save read/write time and starting time.

Furthermore, the parallelization is transparent to the developers and easy to implement. Spark provides two abstractions for the programming, RDD and parallel operations on datasets. The developer only needs to implement the high-level control flow of the application and launches various operations in parallel.

4.2 Link Prediction Parallelization

We design the implementation of parallel link perdition that enhance the applicability, apart from the classical link prediction such as CN,AA, RA et al. (except PA), it can handle the many similarity-based algorithms such as IA, LNL et al. which require common neighbors' additional attributes like neighbors or degree information.

The adjacency information can be saved in a file, and be read into the relevant Map task, however, some large-scale networks are too large to be read into memory. For link prediction algorithm in this paper which frequently query the attributes about common

neighbors which needs O (1) query time complexity. We design a parallel pre-process to keep them in memory. The whole pseudo code of link prediction is divided into three parts as follows:

(1) **Pre-process Parallelization**

```
Input: <X, Y> :<X, Y>  E from G (V, E)
Output: <X, Q(info)[1~n]> : Q(info)[1~n] is the
neighbors of node X with additional information
(Output)
Calculate the adjacent nodes Q[1~n] for each node X
begin
   var i := 0;
   repeat
      EMIT(Q[i], X(Q[1~n]))
      i:= i + 1
   until i:= n
end.
Calculate the adjacent neighbors carrying additional
information Q(info)[1~n] for each node X
```

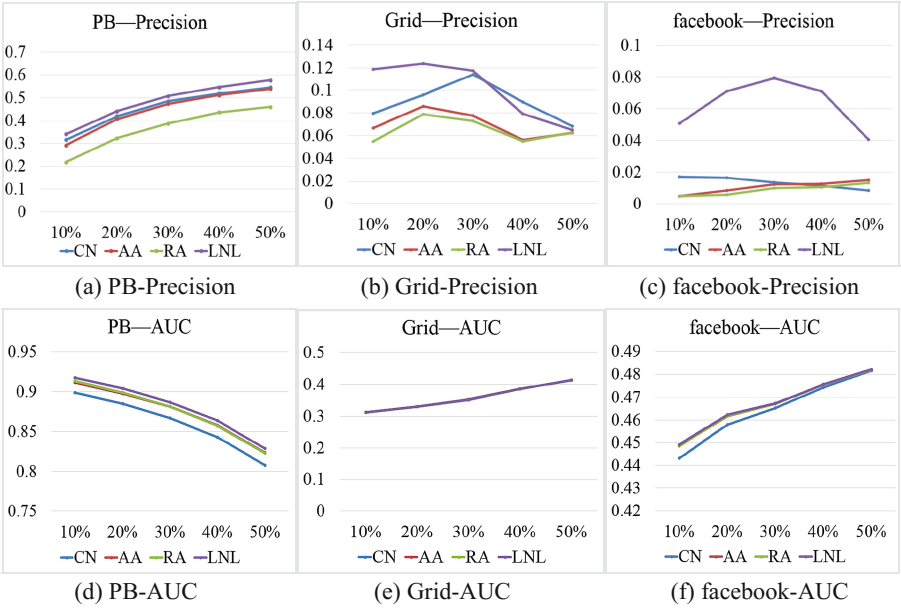
(2) **Parallel the predicted node pairs using literature [19], the main pseudo code as follows**

```
Input: <X,Q(info)[1~n]> : defined as above mentioned
Output: <(X,Z),S_ (X, Z)> : (x, Z) is one node pair,
S_(X, Z) is weight of one common neighbors
begin
   var i := 0; j:=0; score=0;
   newpair: String Input
   repeat
      repeat
         j:=i+1
         newpair:= sorted Order(Q[i],Q[j])
         score : =Similarity(X,Q(info)[i],Q(info)[j])
         EMIT(newpair, score)
         j:= j+1
      until j:= n
   until i:= n-1
end.
```

(3) **Parallel the accumulation of score fragments**

```
Input: <(X,Z),S[1~k]> : (x, Z) is one node pair, S[i]
is the weight of i-th common neighbors, total k common
neighbors
Output: <(X, Z), S(X, Z)> : S(X, Z) prediction score
of (X, Z)
begin
   var i := 0; sum :=0;
   repeat
      sum := sum + S[i]
      i:= i + 1
   until i:= k
   EMIT((X, Z), sum)
end.
```

Time Complexity: The first part is to compute the adjacent neighbors of nodes, the time complexity is $O(Nk)$, where N is the number of nodes and k is the average degree of nodes. According to literature [19], there are k $(k-1)/2$ node pairs for every common neighbor. Algorithms for LNL, it needs to compute the link between common neighbors and the predicted node pairs, suppose common neighbors respectively have average m and n links, the computational complexity is $O((l^2+m^2)k^2)$ for one



(a) PB-Precision (b) Grid-Precision (c) facebook-Precision

(d) PB-AUC (e) Grid-AUC (f) facebook-AUC

Fig. 2. Comparison about robustness of three real network with different ratio of E_t and E_p.

common neighbors, furthermore, $O(N(l^2 + m^2)k^2)$ for all the nodes. Last step is to accumulate the score of each node pair, supposed d is average number of common neighbors, the accumulated time is $O(Nd)$, the total time complexity is $O(N((l^2 + m^2)k^2 + d + k))$. Suppose the splits parameter of the data is K, then the time complexity is $O(N((l^2 + m^2)k^2 + d + k)/K)$, the higher of K, the shorter running time.

5 Experiments

5.1 Data

In this paper, we compare LNL with CN, AA, RA, and IA four classical algorithms, take standard evaluation method Precision and AUC. Also consider seven representative networks including: C.elegans [21], USAir [22], Grid [21], PB [23], Yeast [24], ego-facebook [25] and H.friendships [26]. Table 1 summarizes the basic topological features of these networks. N and M are the total numbers of nodes and links, respectively. C is the clustering coefficient, <d> is the average path length, <k> refer to the average degree of the network. Literature [4] concludes that clustering coefficient has high representative of link prediction algorithms. Consequently, we choose seven networks with comprehensive coverage of different types and clustering coefficient.

In order to test the algorithms' accuracy, the exist links E is divided into two parts: the training set E_t and the probe set E_p, where $E = E_t + E_p$. In the experiments, we use the classical evaluation methods Precision [27] and AUC [28].

To test the performance of parallel link prediction, we choose four large scale datasets from different type of network provide by Stanford Large Network Dataset Collection [29]. The detailed information about the datasets are described in Table 2.

Table 1. Topological features of representative networks.

Network	N	M	C	<d>	<k>
C.elegans	297	2,359	0.308	2.455	14.465
USAir	332	2,126	0.749	2.738	12.807
Grid	4,941	6,595	0.107	18.756	2.67
PB	1,490	19,090	0.36	2.738	27.312
Yeast	2,361	7,182	0.2	4.376	5.63
ego-facebook	2,888	2,981	0.803	3.867	2.064
H.friendships	1,858	12,534	0.167	3.453	13.491

Table 2. Topological features of large-scale networks

ID	Network	N	M	C	Type
D1	ca-HepTh	9877	25998	0.4714	Collaboration networks
D2	email-Enron	36692	183831	0.497	Communication networks
D3	DBLP	317080	1049866	0.6324	Networks with ground-truth communities
D4	roadNet-PA	1088092	1541898	0.0465	Road networks

5.2 Accuracy

In the experiments, we randomly select the training set E_t (contains 90 % links) and the probe set E_p (contains 10 % links), repeated ten times the experiment to calculate the average accuracy, averages is taken to four decimal places. In Precision evaluation method, L takes 100. In AUC method, we randomly select the compare pairs N = 20,000,000 times.

As evidenced in Table 3, apart from USAir, LNL achieves more accurate predictions than other existing algorithms, especially significant improvement for ego-facebook and H.friends. From the experimental results in Table 3, LNL has a better performance in sparse network, since LNL consider a wider range of connections between nodes instead of just common neighbors, it differs the contribution of each common neighbor. Table 4 shows the result evaluated by AUC, LNL maintain the optimum of accuracy except USAir. The accuracy of USAir is not the optimal but the difference in an acceptable range, that is to say, LNL proved to be the better methods and perform better.

Table 3. Accuracies of five similarity indices, measured by Precision method

Precision	C.elegans	USAir	Grid	PB	Yeast	ego-facebook	H.friends
CN	0.123	0.607	0.0890	0.4090	0.2020	0.017	0.0230
AA	0.132	0.623	0.0630	0.3610	0.1880	0.008	0.0170
RA	0.126	**0.636**	0.0530	0.2230	0.1530	0.007	0.0110
IA	0.138	0.624	0.1070	0.4510	0.2740	0.007	0.0240
LNL	**0.149**	0.63	**0.1200**	**0.4540**	**0.2920**	**0.067**	**0.1090**

Table 4. Accuracies of five similarity indices, measured by AUC method

AUC	C.elegans	USAir	Grid	PB	Yeast	ego-facebook	H.friends
CN	0.7877	0.9204	0.3144	0.9003	0.5146	0.4427	0.6732
AA	0.8345	0.9412	0.3145	0.9124	0.5168	0.4487	0.6831
RA	0.8380	**0.9473**	0.3145	0.9134	0.5167	0.4488	0.6837
IA	0.8392	0.9444	0.3145	0.9139	0.5168	0.4487	0.6842
LNL	**0.8457**	0.9405	**0.3145**	**0.9186**	**0.5174**	**0.4493**	**0.6888**

5.3 Robustness

To test the robustness of LNL, we choose three real networks: PB, Grid, ego-facebook and compared with CN, AA, RA indices. We randomly select the training set E_t and the probe set E_p, E_t from 90 % to 50 % decrease by 10 %, accordingly, E_p from 10 % $\sim$ 50 % increase in 10 %, repeated ten times the experiment to calculate the average accuracy. The parameter N is set as the previous section described. L takes 20 % probes set links (If L is lower than 100, then L takes 100) for considering widely for the larger test set.

Figure 2 (a) demonstrates that the prediction accuracy increased with the proportional change of E_t and E_p, and LNL perform best overall. In Fig. 2 (b), besides CN, other algorithms achieve the peak in 20 % ratio for Grid, through 10 %–50 %, LNL get the highest precision. For ego-facebook network, LNL reach the peak at 30 %, and the accuracy is significantly higher than other methods. In terms of AUC, except PB decrease along with the increase of E_p, the others increase oppositely, LNL maintain the optimum of accuracy. Therefore, with different proportion of E_t and E_p, LNL performs best comparing with the others, thus it is quite robustness.

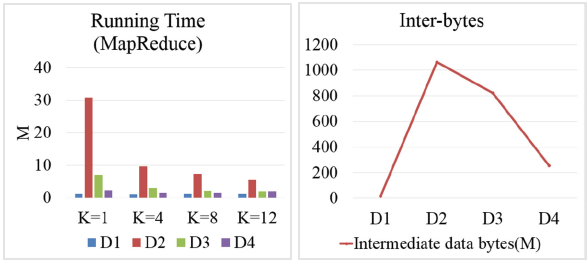
Time Complexity: The computational complexity of CN is $O(Nk^2)$(where N is the total number of nodes, k is the average number of neighbors), AA and RA is consistent with CN in complexity. IA has a time complexity of $O(Nk^2 + Nkn^2)$(n is the number of links between one common neighbors and other common neighbors). LNL respectively computes the number of links between common neighbor c and predicted x and y, it cost $O(Nkl^2 + Nkm^2)$ (where l is the total links of c and x, m is the total links of c and y), thus the computational complexity is $O(Nk^2 + Nkl^2 + Nkm^2)$.

5.4 Efficiency of Parallelization

Cluster Setup: We ran the large scale experiments on 5-node Hadoop cluster. Hadoop version is 2.20. Every node has the same hardware configuration: Dual CPU, 24-core, 64G of memory, configuration about each core: Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10 GHz. In the experiments: K refers to the data split number, for Spark, K is the parallels parameter; for MapReduce, K is the Reduce number.

Performance: We first implement the parallel link prediction by Map Reduce, and analyze its performance. In map Reduce, we totally generate three Jobs of this algorithm. The first Job is to generate the adjacent information of each node; the second Job need to produce all node pairs and compute the score of each common neighbor. The last Job is to accumulate the final prediction score of each node pair.

As shown in Fig. 3 (a), the X-axis is the parallel parameter K, Y-axis is the running time (Minute). Running time decreases sharply with the increasing of K by 4 steps. We also record the inter-bytes (M) of the algorithms as Fig. 3 (b) shown. Compare Fig. 3 (a) with Fig. 3 (b), we can notice that the running time is proportional to the inter-bytes.



(a) MapReduce with different K (b) Inter-bytes

Fig. 3. Running time of MapReduce with different K, K represents the number of data splits.

And the space complexity of LNL is $S(Nk^2)$, its inter-bytes are proportional to the N and k (where k is the average degree of the network).

MapReduce vs Spark: We implement the parallel link prediction in both MapReduce and Spark. As evidenced in Fig. 4, we observe that the running time of Spark is shorter than MapReduce, the running time is less than 50 % for most datasets with the same parameter K >=4. The reason for the difference can be explained as follows: (1) Spark is based on memory computation, while MapReduce need to write its inter-bytes into the local disk in Map task and reload in the Reduce task. It will cost I/O read/write time especially performs badly for large inter-bytes. (2) Spark only needs to start one time the Job of the link prediction algorithm, while the MapReduce needs to start three times. This case cannot represent the typical examples of iterative operation, but it still reduces the running time for using Spark.

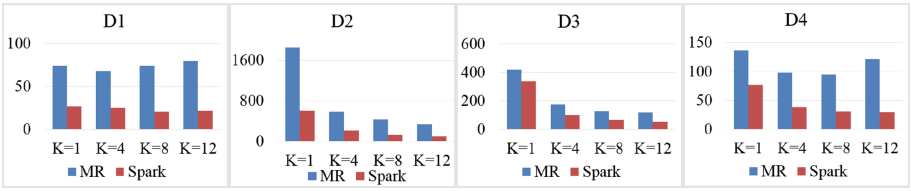


Fig. 4. MapReduce vs Spark on running time: the left Y-axis is the running time (seconds), and the X-axis is the parallel parameter K. D1, D2, D3, D4 are the four networks.

6 Conclusion and Future Work

This paper proposed a new index based on local neighbors links——LNL (Local Neighbors Link), in which attributes and topological connections between nodes are considered. We empirically compared some prediction algorithms. Numerical results of seven real networks shows that LNL outperformed in accuracy. Besides, we design the parallel pre-process of link prediction to enhance applicability of existing method [19]. We improve its efficiency by implementing LNL in a parallel programming model named Spark, and the experiments show that the performance is better by using Spark than MapReduce for large-scale networks.

The research about LNL may be used as topological features for supervised learning of better link prediction accuracy.

References

1. Getoor, L., Diehl, C.P.: Link mining: a survey. ACM SIGKDD Explor. Newslett. **7**(2), 3–12 (2005)
2. Li, J., Shang, X.Q., Guo, Y., Li, X.Y.: New approach of link prediction in PPI network. Appl. Res. Comput. **11**, 016 (2012)

3. Wu, M.: Research on Relationship recommender systems based on link prediction, Beijing University of Posts and Telecommunications (2012)
4. Lü, L., Zhou, T.: Link prediction in complex networks: a survey. Physica A **390**(6), 1150–1170 (2011)
5. Guns, R., Rousseau, R.: Recommending research collaborations using link prediction and random forest classifiers. Scientometrics **101**(2), 1461–1473 (2014)
6. Zhang, Q., Shang, M., Lü, L.: Similarity-based classification in partially labeled networks. Int. J. Mod. Phys. C **21**(06), 813–824 (2010)
7. Lü, L., Jin, C.H., Zhou, T.: Similarity index based on local paths for link prediction of complex networks. Phys. Rev. E **80**(4), 046122 (2009)
8. Katz, L.: A new status index derived from sociometric analysis. Psychometrika **18**(1), 39–43 (1953)
9. Liu, W., Lü, L.: Link prediction based on local random walk. EPL (Europhy. Lett.) **89**(5), 58007 (2010)
10. Al Hasan, M., Chaoji, V., Salem, S., Zaki, M.: Link prediction using supervised learning. In: SDM 2006: Workshop on Link Analysis, Counter-terrorism and Security (2006)
11. Fire, M., Tenenboim, L., Lesser, O., Puzis, R., Rokach, L., Elovici, Y.: Link prediction in social networks using computationally efficient topological features. In: 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), pp. 73–80. IEEE (2011)
12. Jaccard, P.: Etude comparative de la distribution florale dans une portion des Alpes et du Jura. Impr. Corbaz. (1901)
13. Adamic, L.A., Adar, E.: Friends and neighbors on the Web. Soc. Netw. **25**(3), 211–230 (2003)
14. Barabási, A., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
15. Zhou, T., Lü, L., Zhang, Y.C.: Predicting missing links via local information. Eur. Phy. J. B. **71**(4), 623–630 (2009)
16. Dong, Y.X., Ke, Q., Wu, B.: Link prediction based on node similarity. Comput. Sci. **38**(7), 162 (2011)
17. Cannistraci, C.V., Alanis-Lobato, G., Ravasi, T.: From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks. Scientific reports, 3 (2013)
18. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM. **51**(1), 107–113 (2008)
19. Rao, J., Wu, B., Dong, Y.X.: Parallel link prediction in complex network using MapReduce. Ruanjian Xuebao/J. Softw. **23**(12), 3175–3186 (2012)
20. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, pp. 10–10 (2010)
21. Watts, D.J., Strogatz, S.H.: Nature **393**, 440–442 (1998)
22. Batagelj, V., Mrvar, A.: Pajek datasets (2006). http://vlado.fmf.uni-lj.si/pub/networks/data/mix/USAir97.net
23. Adamic, L.A., Glance, N.: The political blogosphere and the 2004 US election: divided they blog. In: Proceedings of the 3rd International Workshop on Link Discovery, pp. 36–43. ACM (2005)
24. Vladimir Batagelj and Andrej Mrvar (2006): Pajek datasets. http://vlado.fmf.uni-lj.si/pub/networks/data/bio/Yeast/Yeast.htm
25. Facebook (NIPS) network dataset – {KONECT} (2015)
26. Hamsterster friendships network dataset – {KONECT} (2015)

27. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. (TOIS) **22**(1), 5–53 (2004)
28. Hanley, J.A., McNeil, B.J.: The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology **143**(1), 29–36 (1982)
29. Leskovec J. Stanford large network dataset collection. http://snap.stanford.edu/data/

http://www.springer.com/978-3-319-23530-1

Web-Age Information Management
WAIM 2015 International Workshops: HENA, HRSUNE,
Qingdao, China, June 8-10, 2015, Revised Selected Papers
Xiao, X.; Zhang, Z. (Eds.)
2015, IX, 113 p. 31 illus., Softcover
ISBN: 978-3-319-23530-1