Chapter 4

Level-2 Node Clustering Coefficient-based Link Prediction

This chapter [1] studies the clustering feature of social networks and extends its concept to the next level. The extended clustering coefficient is then applied to find missing links in networks.

4.1 Introduction

With the success of network science theory, the complex network has turned into a standard and powerful tool to model communication of a group or community of persons in the real-world. For example, recommender systems [48, 237] for friends, products, movies, etc. on different online platforms, protein-protein interaction (PPI) in biological networks [271], websites hyperlink prediction in the Internet [19, 20], hidden link detection among terrorist organizations, and so on. These networks can be represented as graphical models in which a vertex (or node) maps to a person or social entity, and an edge (or link) maps to an interaction between two persons or social entities. These

[1] Published in Applied Intelligence

networks are quite complex as nodes and edges are continuously being introduced and removed, which results in complicated relationships in the system.

Informally, Link Prediction can be defined as finding the missing links between two vertices in an observed network (called static link prediction) or predicting the likelihood of future link by assuming that there is no link between them at the current state of the network (dynamic link prediction). Figure 4.1 illustrates the link prediction problem.

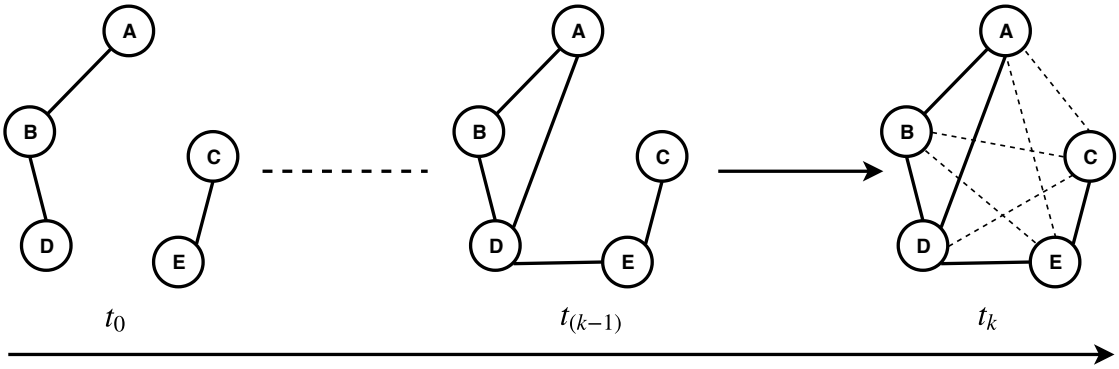


FIGURE 4.1: Initially at time t_0, three links are present in the disconnected graph. As the time progress, more links are formed as shown at the time instant $t_{(k-1)}$. Now, at the time instant t_k, which of the non-existing links (i.e. AC, AE, BC, BE, CD) will be formed ? Finding the potential links that will appear at t_k is called the link prediction problem.

Huang [5] presented a paper on graph topology based link prediction where generalized clustering coefficient is used as a predictive parameter. The author introduces a cycle formation model which shows the relationship between link occurrence probability and its ability to form different length cycles. Further, Liu et al. [171] proposed degree related clustering coefficient to quantify the clustering ability of nodes. They applied the same to paths of shorter lengths and introduced a new index called degree related clustering ability Path (DCP). They performed the degree of robustness (DR) test for their index and showed that missing links have a small effect on the index. Recently Wu et al. [35] extracted triangle structure information in the form of node clustering coefficient of common neighbors. Their experiments on several real datasets show comparable results to CAR index in [68]. The same concept of clustering coefficient also introduced in the work presented by Wu et al. [71]. Authors introduce both node and link

clustering information in their work [71]. Their experiments on small, middle and large network datasets showed better performance results against existing methods, especially on middle and large network datasets.

Clearly, node and link clustering information play an essential role in the evolution of complex networks. The above paragraph shows some research works on link prediction using this property and still more efforts need to be applied. Our work is also an effort in this direction.

4.2 Proposed work

Evidences [11, 169] suggest that many real networks demonstrate consistent topological features across different domains viz., small-world [28, 29], clustering, and scale-free [9]. Their corresponding basic measures are path length, clustering coefficient, and degree distribution. most empirically observed networks show their behavior resembles small-worlds in which any two nodes can find each other in a few steps even if the network is large enough, i.e., the diameter increases logarithmically with the number of nodes in such networks. Small-world networks are highly clustered and characterized by the clustering coefficient. Our work focuses on clustering coefficient measure which is extended up to next level. This work exploits more local information as level-2 common neighbors and clustering properties of such nodes in the network.

This work relaxes the notion of CAR index where only common neighbors and link information among them (i.e. local communities) [68] are considered and extends the notion of clustering information of the CCLP index. Our work considers level-1, level-2, and level-3 link information (also higher level links in some cases) to extract level-2 triangles (clustering information). It selects level-2 and level-3 link information to a greater extent in the triangle formation as compared to level-1 links. The proposed

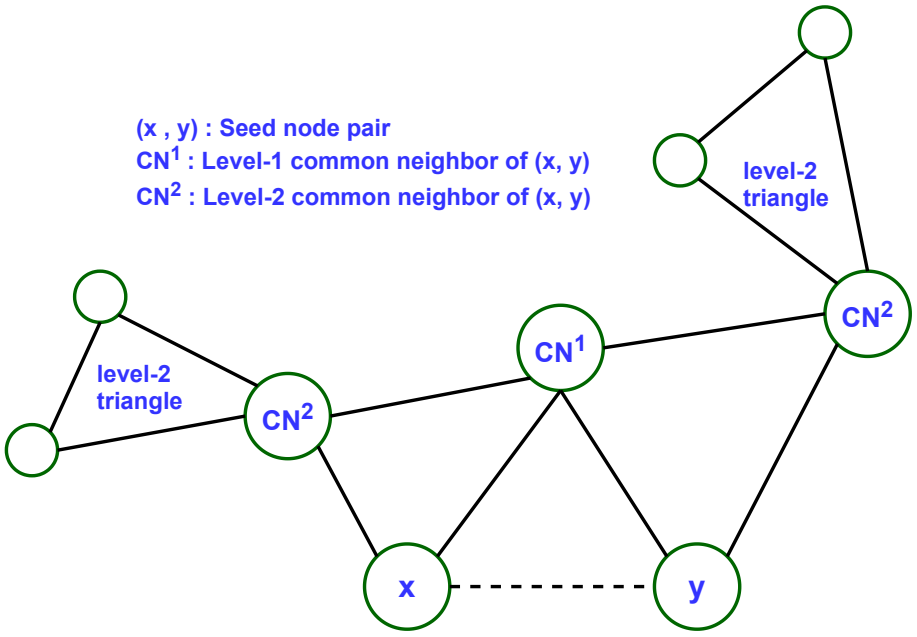


FIGURE 4.2: Notion of level-2 clustering coefficient

method explores a large portion (global to some extent) of the underlying network (Figs. 4.2 and 4.3).

Extracting more local information While selecting level-2 common neighbors (CN^2), there exist some possibilities that level-2 common neighbors are also treated as level-1 common neighbors as shown in figure 4.3. In such cases, level-2 common neighbors are extended to next level in the network and continue until further identification of level-2 common neighbors as level-1 common neighbors.

Definition 4.2.1. **(Clustering Coefficient)** It is a measure of the degree to which nodes of a graph tends to be clustered. In graph theory, the clustering coefficient of a node represents its neighbor's tendency to become a clique or complete graph. Mathematically this measure [11] is expressed as

$$C(i) = \frac{2|\{e_{jk:v_j,v_k \in \Gamma(i), e_{jk} \in E}\}|}{k_i(k_i - 1)} \tag{4.1}$$

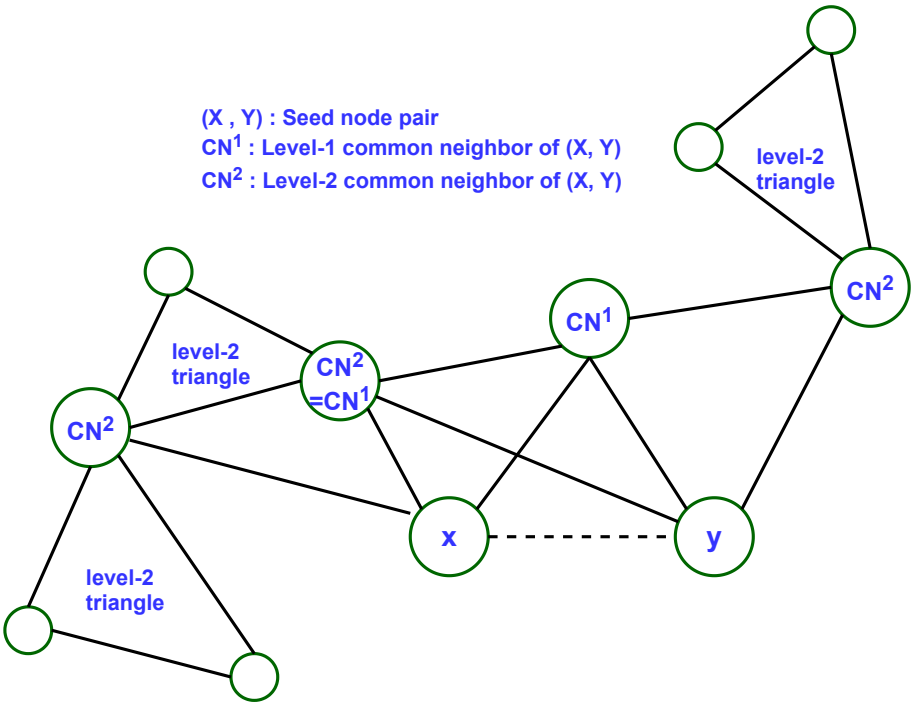


FIGURE 4.3: Exploring local to global structure

where k_i is the degree of the node, i and $\Gamma(i)$ is immediate neighbors of i. We refer this measure as Level-1 clustering coefficient, based on which Wu et al. [35] presented a paper on link prediction.

Definition 4.2.2. **(Level-2 node clustering coefficient)** We extend the definition of node clustering coefficient [35] to next level named level-2 node clustering coefficient which exploits level-2 common neighbors and their clustering information for every pair of nonexistent nodes in the network.

Figure 4.4 represents the best explanation of level-2 node clustering coefficient. For the seed node pair (x,y), level-1 common neighbors or simply common neighbors (CNs) are p, q, and r shown in the right upper part of the figure. Further, all those pairs are selected in which the first node is either x or y (one of the seed node pair) and the second node is one of the level-1 common neighbors. For all such pair, level-2 common neighbors are computed based on common nodes of their respective pair. Based on this definition nodes s and t are level-2 common neighbors shown in right bottom part of the figure. Now the

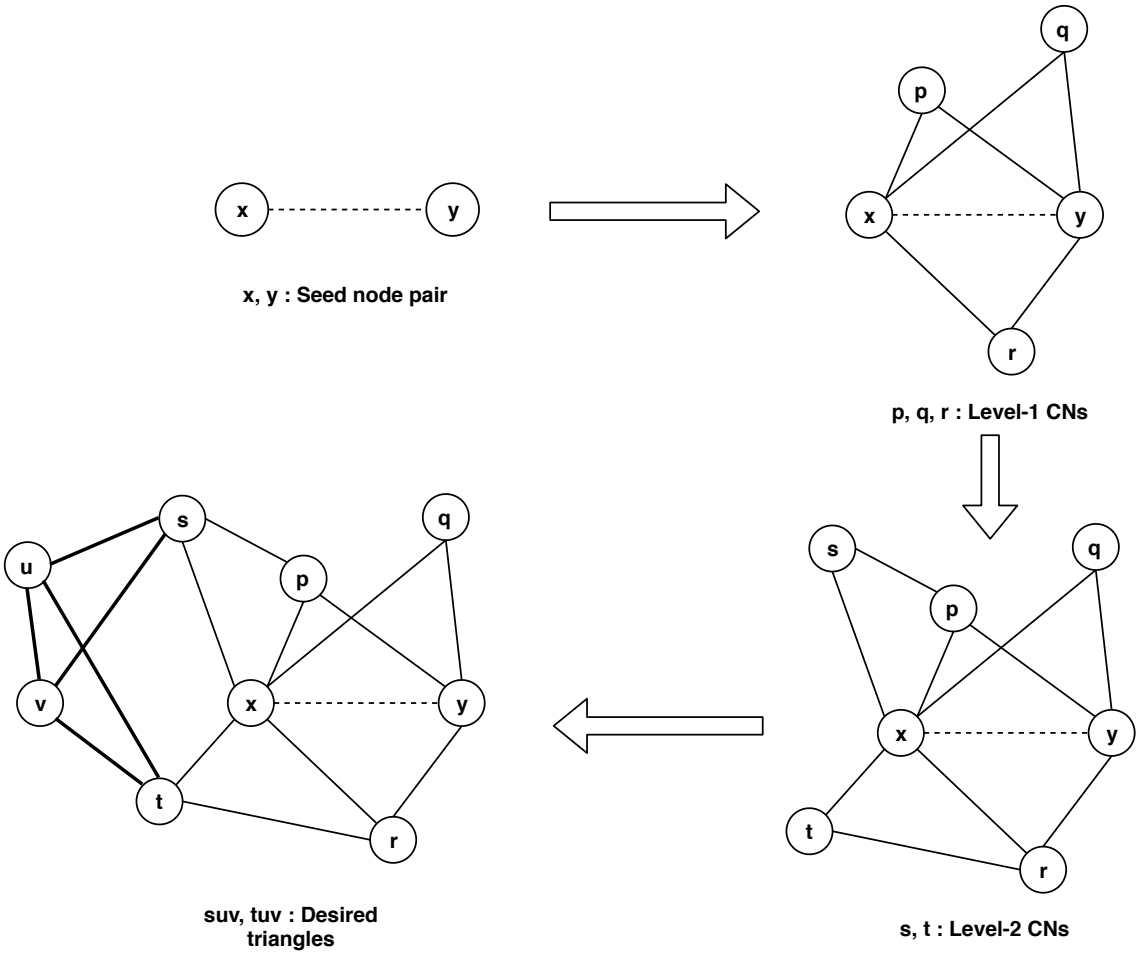


FIGURE 4.4: Computing level-2 node clustering coefficient

total number of triangles passing through each level-2 common neighbor is computed and summed over all such neighbors to find level-2 clustering coefficient of the seed node pair.

Link prediction based on Level-2 node clustering coefficient. Our work focuses on level-2 clustering coefficient that explores clustering information of level-2 common neighbors which is more informative than the clustering coefficient used in [35]. We extend the notion of clustering coefficient of a node the next level (level-2) in the network. We compute the level-2 node clustering coefficient according to the equation 4.2

$$
\begin{aligned}
S_{(x,y)}^{CCLP2} &= \sum_{CN_x^2 \in \Gamma(x) \cap \Gamma(CN^1)} CC(CN_x^2) \\
&\quad + \sum_{CN_y^2 \in \Gamma(CN^1) \cap \Gamma(y)} CC(CN_y^2) \\
&= \sum_{CN^2 \in (\Gamma(x) \cap \Gamma(CN^1)) \cup (\Gamma(CN^1) \cap \Gamma(y))} CC(CN^2)
\end{aligned}
\tag{4.2}
$$

where $CC(CN^2)$ is having the usual definition of node (i.e. CN^2) clustering coefficient value and is computed using the equation 4.1. CN_x^2 is the level-2 common neighbor corresponding to node x and the common node of the pair (x,y). CN^1 is the level-1 common neighbor defined in the literature and is computed as

$$CN^1 = \Gamma(x) \cap \Gamma(y).$$

The pseudo code of the proposed algorithm is presented in the Algorithm 1.

Algorithm 1: CCLP2

Input: Graph $G(V,E)$
Output: Top-L predicted links

1 **foreach** *Node pair* $(x,y) \notin E$ **do**
2 find all (level-1) common neighbors, $CN_{(x,y)}^1$
3 **foreach** (p_i,q_i), $p_i \in \{x,y\}$ *and* $q_i \in CN_{(x,y)}^1$ **do**
4 find all (level-2) common neighbors, $CN_{(x,y)}^2$
5 Calculate (level-2) clustering coefficient (CC) of all nodes in $CN_{(x,y)}^2$ and add them to get final similarity score as in step 6.
6 Similarity score; $S_{(x,y)}^{CCLP2} = \sum_{i \in CN_{(x,y)}^2} \frac{t_i}{k_i(k_i-1)/2}$, where t_i is number of triangles passing through node i and k_i is the degree of i.
7 Sort all node pairs in descending order based on computed similarity score.
8 **return** Top-L node pairs as predicted links.

Algorithm description. For a given simple undirected graph, the algorithm finds top-L missing links. The main crux of the algorithm is to find level-2 common neighbors from which level-2 clustering coefficient can be calculated.

For each pair of nodes (seed node pair (x,y)) having no edge between them, the algorithm finds all common neighbors (level-1 CNs) [Line: $1-2$]. Level-2 common neighbors are then computed for all those node pairs (p_i,q_i) in which first node p_i belongs to $\{x,y\}$ while second node in level-1 common neighbors of (x, y) [Line: $3-4$]. Now for all nodes in level-2 common neighbors, clustering coefficient values are computed and added to get final similarity score for the seed node pair (x,y) [Line: $5-6$]. Once scores of all non-existent node pair have been computed, the next step [Line: 7] arranges them in descending order, and finally, top-L node pairs returned as predicted links [Line: 8].

4.3 Experimental study

4.3.1 Evaluation metrics

The link prediction problem is treated as a binary classification task [63] so most of the evaluation metrics of any binary classification task can be used in link prediction evaluation. The evaluation of a binary classification task having two classes can be represented as a confusion matrix [180].

In the confusion matrix,

- True Positive (TP): positive data item predicted as positive
- True Negative (TN): negative data item predicted as negative
- False Positive (FP): negative data item predicted as positive
- False Negative (FN): positive data item predicted as negative

Based on the confusion matrix, several metrics can be derived as follows [180].

True Positive Rate (TPR)/Recall/Sensitivity

$$TPR = \frac{\#TP}{\#TP + \#FN} \tag{4.3}$$

False Positive Rate (FPR)

$$FPR = \frac{\#FP}{\#FP + \#TN} \tag{4.4}$$

True Negative Rate (TNR)/Specificity

$$TNR = \frac{\#TN}{\#TN + \#FP} \tag{4.5}$$

$$Precision = \frac{\#TP}{\#TP + \#FP} \tag{4.6}$$

Our approach is evaluated on four metrics viz., Area under the ROC curve (AUROC) [52, 182], Area under the precision-recall curve (AUPR) [181], Average precision (AP) [180] and *Recall* [180].

Area under the Receiver Operating Characteristics Curve (AUROC) An ROC curve is a plot between the true positive rate (sensitivity) on the Y-axis and the false positive rate (1-specificity) on the X-axis. The true positive rate and false positive rate can be evaluated using equation 4.3 and 4.4 respectively. The area under the ROC curve [182] is a single point summary statistics between 0 and 1 that can be computed using the trapezoidal rule which sums all the trapezoids under the curve. The value of the AUROC of a predictor should be greater than 0.5 which is the value of a random predictor, i.e., higher the value of AUROC better the performance of the predictor.

Area under the precision-recall curve (AUPR) The precision-recall curve is more useful and informative when applied to binary classification on imbalanced datasets [272]. So

we have also considered the area under the precision-recall curve (AUPR). This value is computed based on the precision-recall curve which is a plot between the precision values on the Y-axis and the recall values on the X-axis. The precision and recall values can be computed using equation 4.6 and equation 4.3 respectively.

Average Precision (AP) This metric is also a single point summary value computed based on varying threshold[2] values of recall. The average precision value is equal to the precision averaged over all values of recall between 0 and 1, i.e.,

$$AP = \int_{r=0}^{1} p(r)dr, \tag{4.7}$$

where p is the precision at different threshold value of recall r.

Practically, integral is approximated to sum over the precision at each threshold value, multiplied by the change in the recall, i.e.,

$$AP = \sum_{k=1}^{R} p(k)\triangle r(k), \tag{4.8}$$

where R is the set of different threshold values.

Recall This metric[3] intuitively finds all positive samples (existence of links in this case) in the data and equates to the metric given in equation 4.3.

4.3.2 Datasets description

This work used 11 network datasets from various fields to study the performance of our approach. Macaque[4] [176]: is a biological network of cerebral cortex of Rhesus Macaque. Football[5] [177]: American football games network played between Division

[2]https://sanchom.wordpress.com/tag/average-precision/

[3]`https://ils.unc.edu/courses/2013_spring/inls509_001/lectures/10-EvaluationMetrics.pdf`

[4]https://neurodata.io/project/connectomes/

[5]http://www-personal.umich.edu/ mejn/netdata/

IA colleges during regular season Fall 2000. Celegansneural[4] [11]: A neural network of C. Elegans compiled by D. Watts and S. Strogatz in which each node refers a neuron and, an edge joins two neurons if they are connected by either a synapse or a gap junction. USAir97[6] is an airline network of US where a node represents an airport and an edge shows the connectivity between two airports. Political bolgs[4] [273] is a directed network of hyperlinks in political blogs leaning towards the conservatives and the democrats preceding US election 2004. Yeast[7] [274] is also a biological network of proteins in a cell where a node represents a protein and edge denotes the interaction between two proteins. Amazon web graph[6] [275] is an informational network of web pages of Amazon.com and its sister companies. Power grid[4] [11] is an undirected and unweighted network of power grid located in western states of the United States. Ca-GrQc[8], Ca-HepTh[7], and Ca-HepPh[7] are collaboration networks of arXiv General Relativity, High Energy Physics Theory, and High Energy Physics respectively.

TABLE 4.1: Topological information of real-world network datasets

Datasets	$\|V\|$	$\|E\|$	$\langle D \rangle$	$\langle K \rangle$	$\langle C \rangle$
Macaque	91	1401	1.658	30.791	0.742
Football	115	613	2.486	10.661	0.403
Celegansneural	297	2148	2.447	14.456	0.308
USAir97	332	2126	2.738	12.807	0.749
Political blogs	1490	16718	2.738	22.440	0.361
Yeast	2361	7182	4.376	6.084	0.271
Amazon web graph	2880	3904	3.433	2.711	0.818
Power grid	4941	6594	18.989	2.669	0.107
Ca-GrQc	5242	14496	6.049	5.531	0.687
Ca-HepTh	8361	15751	7.025	3.768	0.636
Ca-HepPh	12008	118521	4.673	19.74	0.699

Table 4.1 shows some basic topological properties of the considered networks datasets. $|V|$, $|E|$, $\langle D \rangle$, and $\langle K \rangle$ are number of vertices, number of edges, average distance, and average degree of the network respectively. $\langle C \rangle$ is the average clustering coefficient of the network.

[6] http://vlado.fmf.uni-lj.si/pub/networks/data/
[7] https://icon.colorado.edu/#!/networks
[8] https://snap.stanford.edu/data/

4.3.3 Results analysis

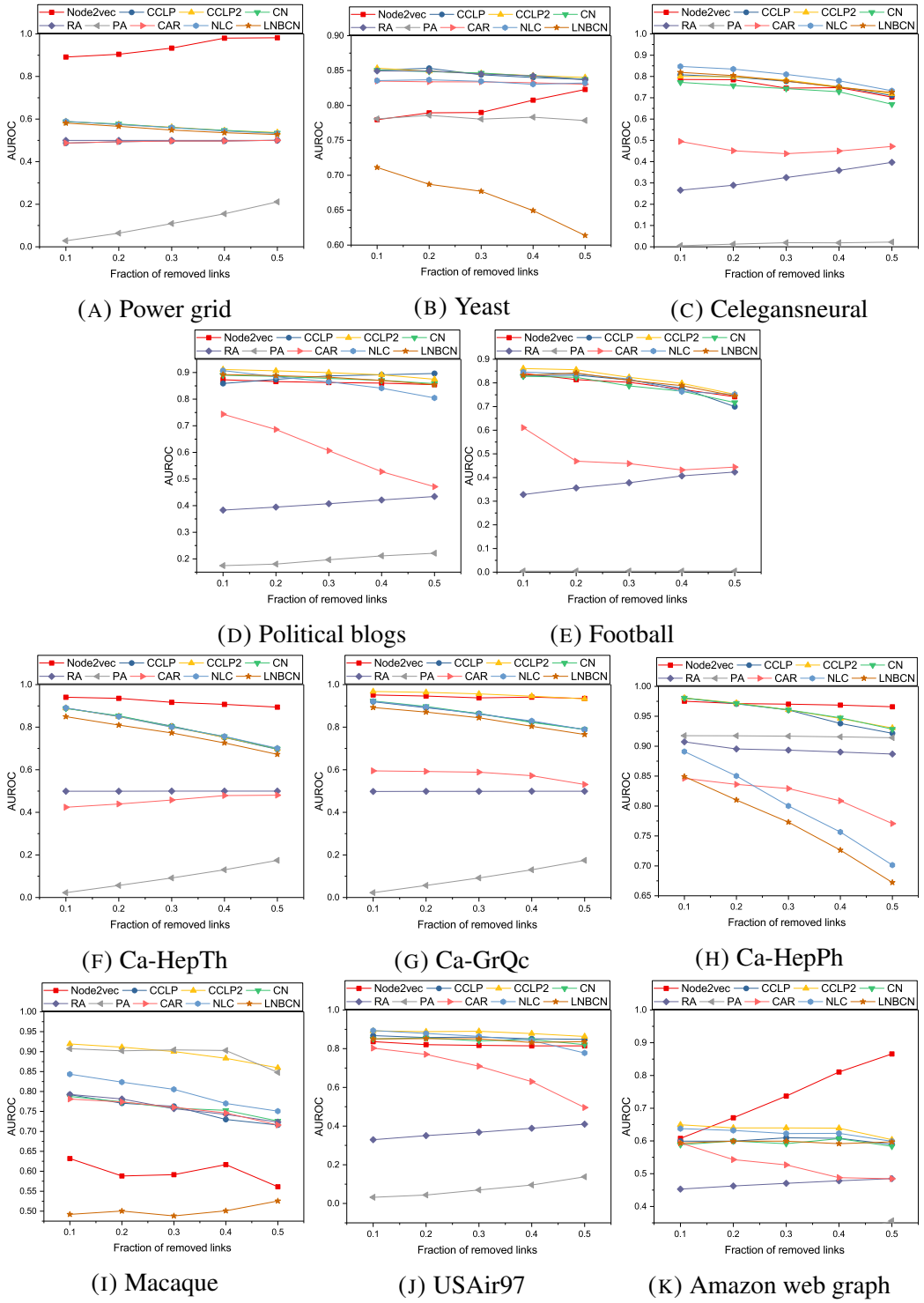
This section investigates the effectiveness of our proposed work on different network datasets against the baseline methods. Our method is tested on four well-known accuracy measures of link prediction namely area under the ROC curve (AUROC), area under the precision-recall curve (AUPR), average precision (AP), and recall as explained in the section 4.3.1. Five sets of probe links (i.e., percentage of removed links = 10, 20, 30, 40, 50) (sparsification levels) are used to evaluate each performance metric. Increasing the percentage of removed links beyond 50% may disconnect the graph, so we consider the sparsification level up to 50% only. The fraction of removed links and the individual metric are displayed on X-axis and Y-axis respectively. We demonstrate our results (Figures 4.5, 4.6, 4.7, and 4.8) based on the clustering values of the networks. First, three of each figure with low clustering, next two figures having medium clustering values, and last six figures are shown with high clustering values. The proposed method entitled "Level-2 node clustering-coefficient" is abbreviated as "CCLP2" and other baseline methods are also abbreviated in accordance with the abbreviation given in the literature review section.

AUROC Figure 4.5 shows the AUROC results of different methods (proposed+baseline) on 11 real-world network datasets. The X-axis represents the five sets of probe links (or fraction of removed links), and AUROC is shown on the Y-axis. With low clustered ($\langle C \rangle \leq 0.3$) networks, the proposed method (CCLP2) shows comparable results with CN, and NLC on Power grid (4.5a), CCLP, CN, and RA on Yeast (4.5b), and CCLP, and LNBCN on Celegansneural data (4.5c). Our method performs better than remaining methods accordingly. The CCLP2 performs overall best on Political blogs (4.5d) and Football (4.5e) networks (networks with medium clustering values ($0.3 < \langle C \rangle \leq 0.5$)). The Node2vec and the NLC perform best on the Power grid and Celegansneural data respectively. The NLC method also shows good results on these networks but slightly lags CCLP2. For large clustered networks ($\langle C \rangle > 0.5$), the CCLP2

performs overall best on collaboration network Ca-GrQc (4.5g), Macaque (4.5i), and USAir97 (4.5j) datasets. Our method outputs comparable results with CCLP and CN on Ca-HepTh (4.5f)and Ca-HepPh (4.5h). Moreover, it is significantly better than the remaining methods. The Node2vec shows best results on Ca-HepTh (4.5f) and Amazon web graph (4.5k) where our method is the second-best performer on Amazon web graph.

AUPR Most of the real-world networks are sparse, i.e., the number of existing links are very less as compared to the number of non-existing links. In other words, these networks are highly imbalanced, and the literature suggested that the precision-recall curve (AUPR) [272] is more informative than the ROC curve for the evaluation of such networks. Hence AUPR is also considered as one of the evaluation approaches of the link prediction.

Figure 4.6 shows the result of the area under the precision-recall curve (AUPR). From the figures, we observe that the Node2vec is the best performing method against all datasets except the Macaque where CCLP2 and PA are best when the fraction of removed links are 40% and 50%. After Node2vec, the proposed method (CCLP2) performs best on Power grid (at sparsification levels 10%, 20%, 30%) and Yeast networks (except at 20%). The CCLP2 also performs best on medium clustered networks (Political blogs 4.6d and Football 4.6e) and high clustered networks (Ca-GrQc 4.6g, Macaque 4.6i, and Amazon web graph 4.6k). Moreover, It beats all methods except CN on Ca-HepTh and CAR on Ca-HepPh as depicted in 4.6f and 4.6h respectively. On Celegansneural and USAir97 datasets, our method shows average performance compared to others. Being high clustering value of the USAir97, our method performs average because of the lower number of common neighbors between the pairs (local airports (LAs), local centers (LCs)), (local airports (LAs), hubs), and between two local airports [69]. The lower performance of common neighborhood-based methods also due to the same reason. Note that we have used sparsification levels and the fraction of removed links interchangeably.



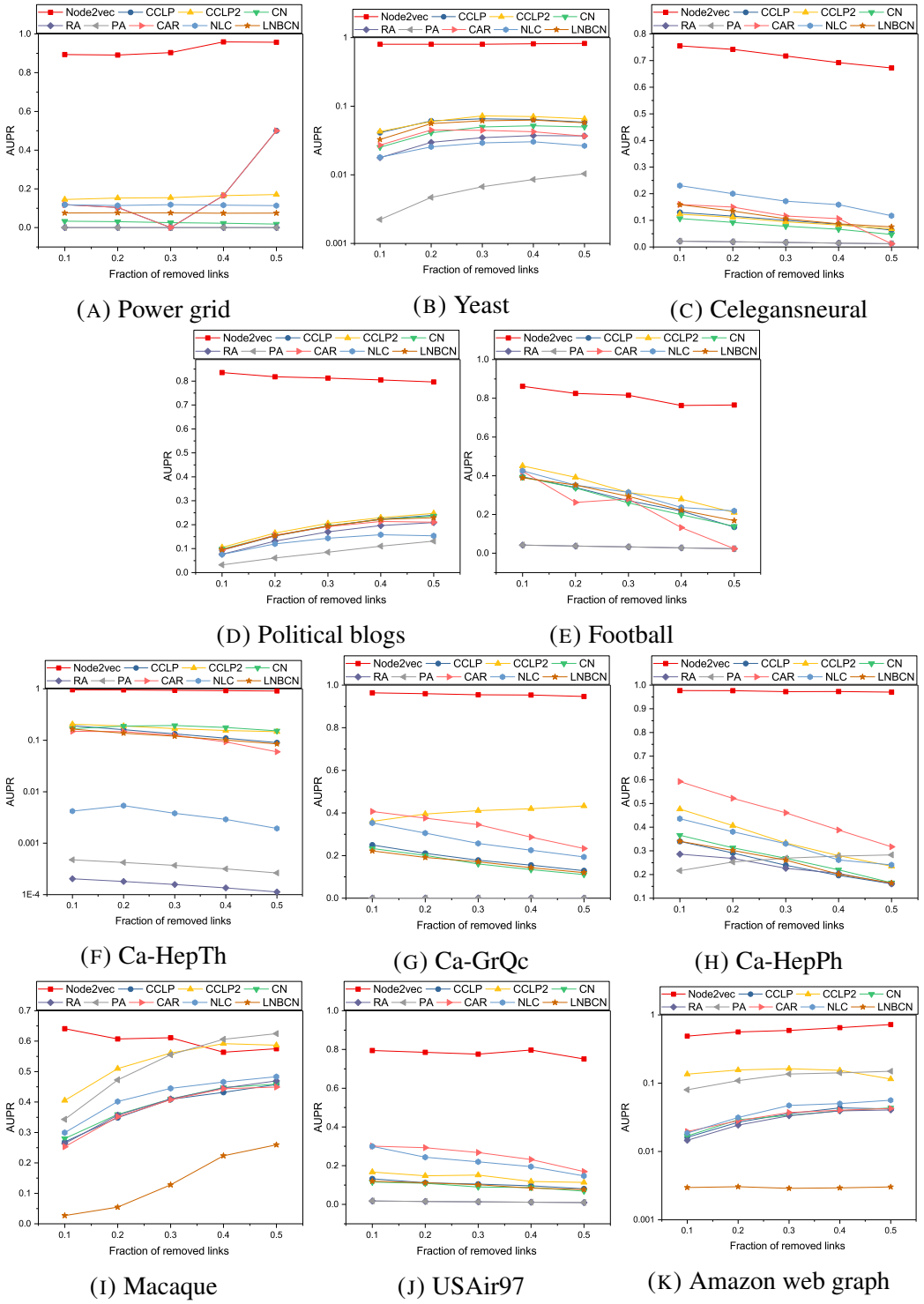
(A) Power grid (B) Yeast (C) Celegansneural

(D) Political blogs (E) Football

(F) Ca-HepTh (G) Ca-GrQc (H) Ca-HepPh

(I) Macaque (J) USAir97 (K) Amazon web graph

FIGURE 4.5: AUROC Results



(A) Power grid (B) Yeast (C) Celegansneural

(D) Political blogs (E) Football

(F) Ca-HepTh (G) Ca-GrQc (H) Ca-HepPh

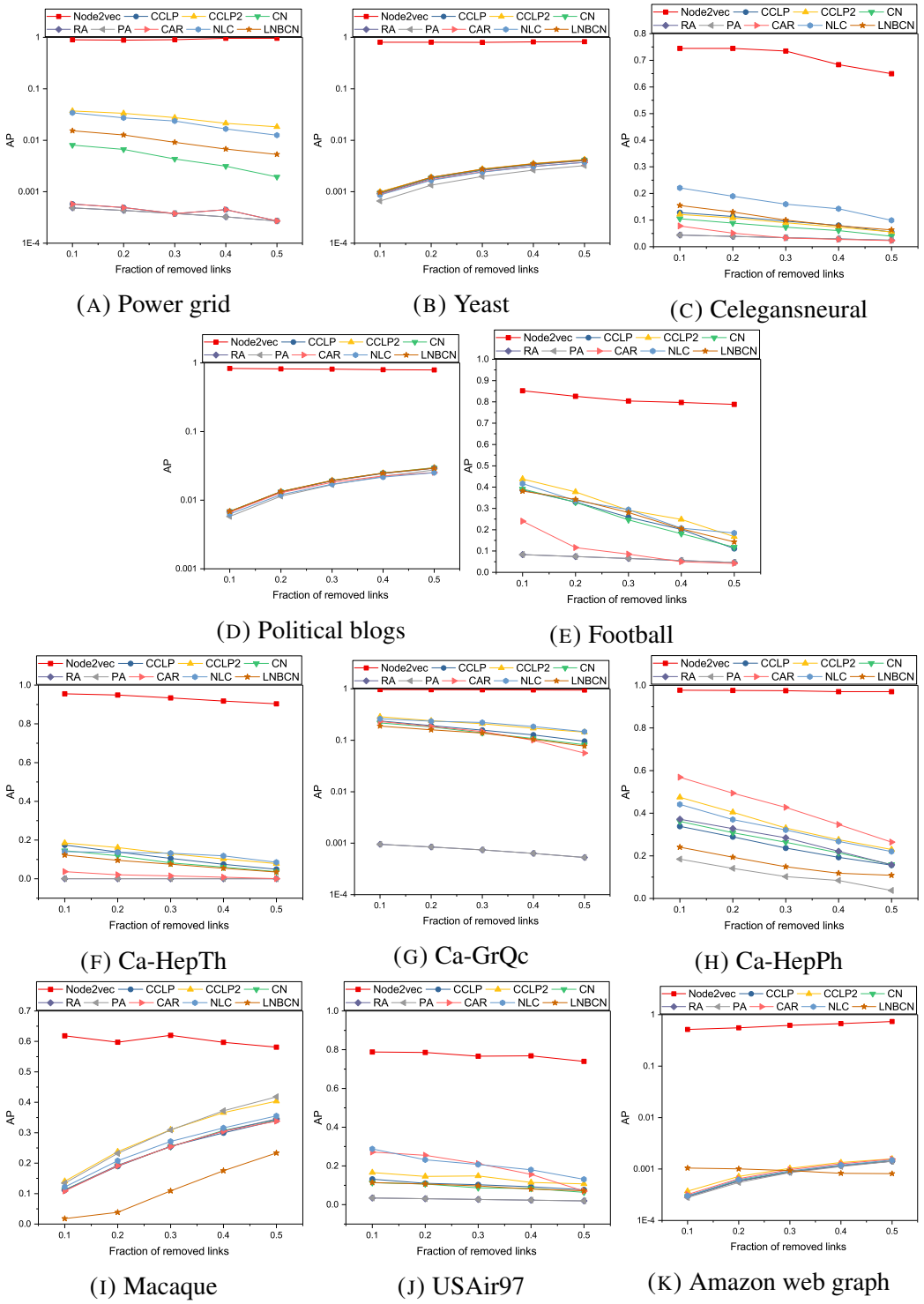
(I) Macaque (J) USAir97 (K) Amazon web graph

FIGURE 4.6: AUPR Results

AP Figure 4.7 shows the average precision results on 11 real-world network datasets. Similar to the AUPR result, Node2vec shows outstanding performance on all datasets. The considered methods except for Node2vec show very low average precision results on all datasets. Our methods performs best after Node2vec on Power grid (4.7a) and Football (4.7e) networks. The CCLP2 and other methods show comparable results on Yeast 4.7b), Political blogs (4.7d), and Amazon web graph 4.7k). On collaboration networks (i.e. Ca-HepTh (4.7f) and Ca-Grqc (4.7g)), our method show equivalent results as that of the NLC with some fluctuation. The same results are obtained on the Macaque network (4.7i), but, the two equivalent methods are CCLP2 and PA. On Ca-HepPh, The CCLP2 lags behind the CAR method. Finally, Our method show average performance on Celegansneural (4.7c) and USAir97 (4.7j) datasets. The average performance of the CCLP2 and other common neighbor based methods are due to the same reason explained for AUPR in the previous paragraph.

Recall Figure 4.8 shows recall results for all methods (proposed+baseline). With the low clustered networks, the CCLP2 shows its best on Yeast (4.8b) network after Node2vec and on Celegansneural (4.8c) after CAR method and comparable results with CN and NLC on Power grid data (4.8a). It also best performs on Political blogs (4.8d) but average performance on Football (4.8e). With high clustered networks, our method overall best on USAir97 network (4.8j) and Amazon web graph (4.8k), while second-best performing method on Macaque and Ca-HepPh networks after Node2vec and CN respectively. On Amazon web graph and Macaque networks, PA shows good results over the CCLP2 when sparsification level is increased to 40% and 50%. On arXiv networks (i.e., Ca-HepTh and Ca-GrQc), the CCLP2 result is comparable to CN, CCLP, and NLC. Our results in Figure 4.8 show that the Node2vec is the best performing method on Power grid, Yeast, Ca-HepTh, and Macaque networks, and CAR is overall best on the Ca-GrQc dataset.

Concluding remarks. By analyzing the AUROC results, we observe that the proposed method (CCLP2) shows comparable results on low clustered networks while



(A) Power grid (B) Yeast (C) Celegansneural

(D) Political blogs (E) Football

(F) Ca-HepTh (G) Ca-GrQc (H) Ca-HepPh

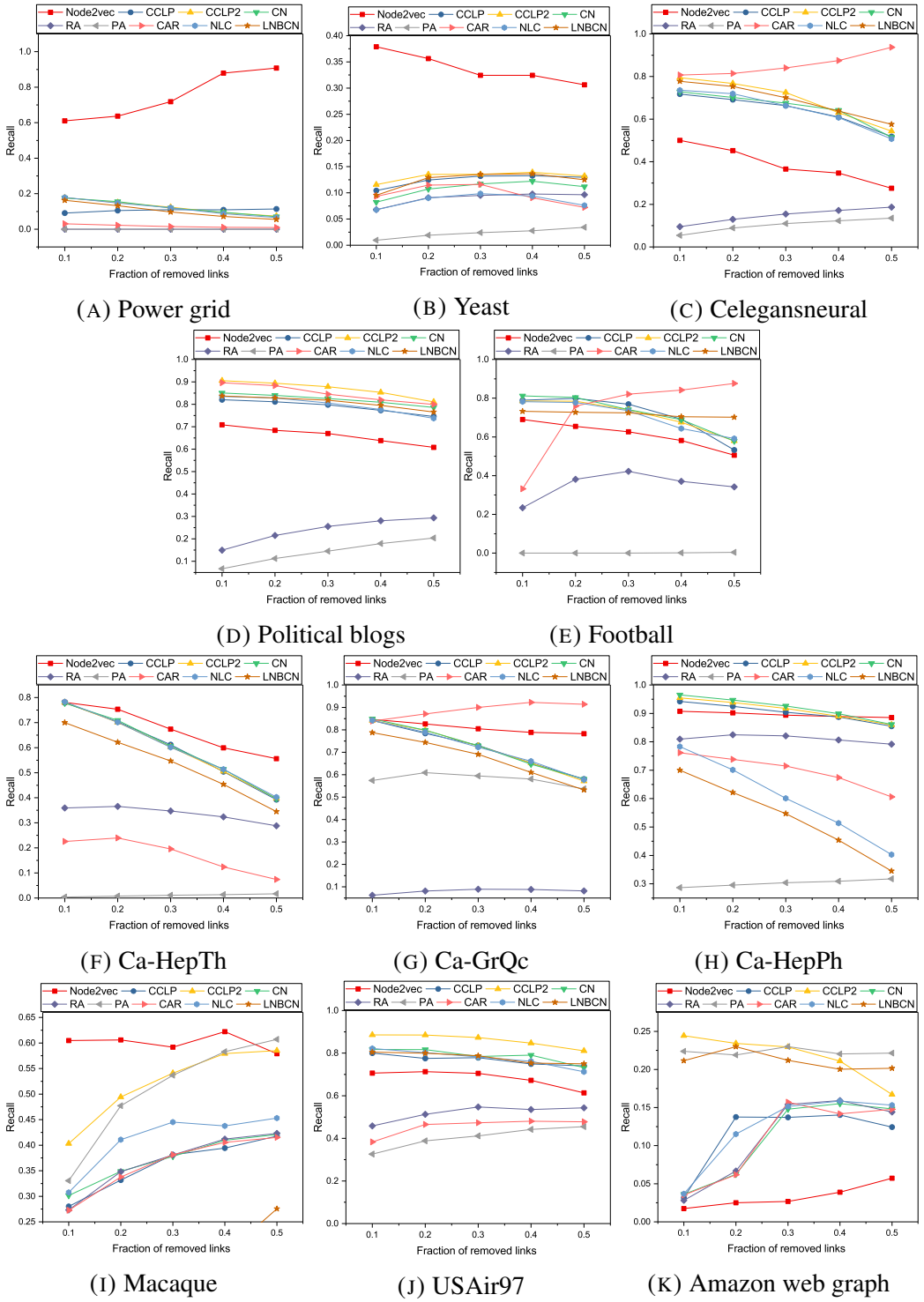
(I) Macaque (J) USAir97 (K) Amazon web graph

FIGURE 4.7: AP Results

best prediction results on medium clustered networks. With high clustered networks, it is best on three datasets (Ca-GrQc, Macaque, and USAir97) and second-best performer on the remaining three datasets (Ca-HepTh, Ca-HepPh, and Amazon web graph). When observing the AUPR results, we find that the Node2vec gives best results on all datasets. The proposed method is second best performing on all networks except the Celegansneural, Ca-HepPh and USAir97. On Ca-HepTh, the CN beats our method only on 30% and 40%, lags on 10%, and comparable on 20% and 50% of removed links. Further, Node2vec also performs overall best on average precision metric against 11 network datasets. Our method is the second-best performer on Power grid and Football, and comparable on high clustered networks except for USAir97 where CCLP2 show average performance. The recall result shows that Node2vec is best on 4 datasets (Power grid, Yeast, Ca-HepTh, and Macaque). The CCLP2 is overall best on Political blogs and USAir97 networks while second-best performer on Yeast, Celegansneural, Ca-HepPh, Macaque, and Amazon web graph. It shows comparable results on Power grid, Football, Ca-HepTh, and Ca-GrQc datasets.

Finally, we observe that Node2vec is the best performing method on all datasets with some exception. The CCLP2 shows better performance after the Node2vec method on average and high clustered networks except USAir97. Though USAir97 is having high clustering coefficient value, our method performs average (after CAR, and NLC) on AUPR and average precision metrics. The possible reason may be the existence of least number of common neighbors between two local airports (LAs), between local airports (LAs) and local centers (LCs), and between local airports (LAs) and hubs [69]. This results the least probability of such links to present in the top of the list and hence the precision gets reduced of our method and the common neighborhood-based methods. As a result low performance in AUPR and average precision.

Complexity analysis. Here, we estimate the efficiency of the proposed method as well as the baseline predictors (algorithms). Only off-line parts of all algorithms are considered for the time estimation where off-line refers to the similarity matrix computation for all



(A) Power grid (B) Yeast (C) Celegansneural

(D) Political blogs (E) Football

(F) Ca-HepTh (G) Ca-GrQc (H) Ca-HepPh

(I) Macaque (J) USAir97 (K) Amazon web graph

FIGURE 4.8: Recall Results

pair of nodes. The time needed to compute level-2 common neighbors is the same as the normal common neighbor of a pair of nodes, i.e., $O(n^2)$ when the data structure is the adjacency matrix and $O(n)$ in case of the adjacency list. The clustering coefficient of a node takes $O(n^3)$ in the worst case and $O(n\langle K\rangle^2)$ after applying some optimization, although an approximate algorithm of $O(n)$ time by Schank and Wagner [276] also exists.

The main crux of our algorithm is the computation of the level-2 clustering coefficient in step $3-6$ where for loop of step 3 iterates to $2\langle K\rangle$ times. Line 4 costs $O(n)$ to compute CNs for a given pair and $O(n\langle K\rangle)$ for computing clustering coefficients, resulting in total $2\langle K\rangle \times (O(n)+O(n\langle K\rangle))$ time.. The outer for loop iterates to $O(n^2)$ in the worst case, so the total time complexity of the proposed algorithm is $O(n^3\langle K\rangle^2)$ which is comparable to the $O(n^3\langle K\rangle)$ of existing CCLP. The computational complexity of the NLC and the LNBCN are $O(n^4\langle K\rangle)$ and $O(n.O(f(z)+n\langle K\rangle^3))$ where $f(z)$ is the influence function. The CAR costs $O(n\langle K\rangle^4)$ which is more complex as it computes time-consuming local community links (LCL). Other methods like CN, AA, RA estimates $O(n\langle K\rangle^3)$ while PA costs $O(n\langle K\rangle^2)$ where $\langle K\rangle$ is the average degree of the network. The Node2vec [125] method is based on a random walk sampling which is efficient compared to pure BFS/DFS. The effective time complexity of the Node2vec is $O(\frac{l}{k'(l-k')})$ per sample where l is walk length and k' is neighborhood size.

Statistical test In this paragraph, we conduct statistical test [277] to show the significant difference of the proposed method with the baseline methods. We perform the Friedman test [278, 279] to analyze whether there is a significant difference among multiple methods. It is a non-parametric counterpart of the repeated measures ANOVA. If the test result shows a significant difference, we further applied post hoc analysis to check the degree of rejection of each hypothesis. for the post hoc analysis, several methods are available in the literature and we applied post hoc counterpart of the Friedman test known as Posthoc Friedman Conover method. The proposed method CCLP2 is considered as the control algorithm in the posthoc analysis. We selects the level of confidence $\alpha_c = 0.05$ and the degree of freedom $D_f = 8$.

The Friedman test result for the metric area under the ROC curve is tabulated in the Table 4.2. The table shows the computed Friedman test values F_f on different percentage (10, 20, 30, 40, and 50) of removed links (or sparsification levels). The Friedman test rejects the null hypothesis H_0 if the test value F_f is greater than $\chi^2(\alpha_c, D_f)$, i.e. $F_f > 15.51$. We have performed the same test for remaining three metrics viz., AUPR, Recall and AP where we found that the null hypothesis is rejected for each of the metrics. We have not shown the remaining here.

Since this test rejects the null hypothesis on each percentage of removed links so we go for the post hoc analysis. The results of the post hoc analysis are shown in table 4.3 for all four metrics used in this paper. With the confidence level $\alpha_c = 0.05$, we observe that the proposed method (CCLP2) is significantly different from all the baseline methods except NLC method on AUROC and AP. our method is insignificant on AUROC against 10% of removed links and insignificant on AP against 30%, 40%, and 50% of removed links. Moreover, the CCLP2 shows its significance on recall against CCLP (except 50%), RA, PA, CAR, NLC (except 10% and 40%), and LNBCN methods. It is insignificant from the CN and the Node2vec (except 10% and 20%). With AUPR, our method is significantly different from the baseline method except for NLC, where it is insignificant only for 50% of the removed links.

4.4 Conclusion and future works

Motivated by the intuition that more local information of topology of a network may improve the accuracy of link prediction, we extracted common neighbors and clustering information up to next level. The proposed method computes clustering coefficients of level-2 common neighbors of the seed node pair. The similarity score sums over all such common neighbors for the seed node pair. The experiments have been conducted 11 real-world networks and results are organized as low, medium, and high clustered networks. The comprehensive results show that the proposed method performs better

TABLE 4.2: The Friedman test on Area under the ROC Curve (AUROC)

Removed links (%)	Dataset	IS-value									Test value	State Result
		Node2vec	CCLP	CCLP2	CN	RA	PA	CAR	NLC	LNBCN	F_f	Is $F_f > \chi^2$?
10	Macaque	0.63199	0.79209	0.91921	0.7875	0.79259	0.91732	0.78093	0.84349	0.49197	47.466	Null Hypothesis Rejected
	Football	0.84021	0.83049	0.86067	0.8281	0.32826	0.00474	0.61076	0.84617	0.83241		
	Celegansneural	0.78599	0.80817	0.80028	0.77269	0.26587	0.00527	0.49472	0.84701	0.81955		
	USAir97	0.83602	0.86762	0.89145	0.85007	0.32958	0.03225	0.80266	0.89348	0.8508		
	Political blogs	0.87281	0.85881	0.91166	0.88962	0.38327	0.17453	0.7436	0.90752	0.89255		
	Yeast	0.77971	0.85077	0.85354	0.85065	0.84938	0.78088	0.83494	0.83562	0.71119		
	Amazon web graph	0.60775	0.5995	0.64935	0.58821	0.45291	0.08295	0.59378	0.63773	0.59335		
	Power grid	0.89143	0.48755	0.58843	0.5872	0.49946	0.02824	0.48755	0.58948	0.58129		
	Ca-GrQc	0.95144	0.91962	0.96748	0.92301	0.49802	0.02237	0.59494	0.92142	0.89244		
	Ca-HepTh	0.94038	0.88902	0.88972	0.88813	0.49917	0.02237	0.42399	0.89102	0.84927		
	Ca-HepPh	0.97528	0.9801	0.98051	0.98051	0.90737	0.91754	0.84623	0.89102	0.84927		
20	Macaque	0.58817	0.77075	0.91119	0.7742	0.78123	0.91188	0.77379	0.82347	0.50015	41.541	Null Hypothesis Rejected
	Football	0.81396	0.83259	0.85578	0.82336	0.35586	0.0047	0.46912	0.83809	0.84206		
	Celegansneural	0.78504	0.79719	0.79833	0.75735	0.28911	0.01219	0.45128	0.83489	0.80416		
	USAir97	0.82055	0.85718	0.88811	0.85534	0.35042	0.04408	0.77034	0.87852	0.85176		
	Political blogs	0.86617	0.87417	0.90649	0.88584	0.39449	0.18029	0.68585	0.88432	0.88852		
	Yeast	0.78944	0.85339	0.84905	0.84859	0.84942	0.78592	0.83401	0.83678	0.68717		
	Amazon web graph	0.67059	0.59935	0.63993	0.59964	0.46255	0.16024	0.54331	0.63223	0.59926		
	Power grid	0.9043	0.49376	0.57517	0.57715	0.49957	0.06411	0.49376	0.57362	0.56606		
	Ca-GrQc	0.94598	0.89122	0.96368	0.89782	0.49833	0.05681	0.59172	0.8945	0.87084		
	Ca-HepTh	0.93529	0.85119	0.85012	0.85396	0.4993	0.05681	0.43903	0.85024	0.81001		
	Ca-HepPh	0.97136	0.97089	0.97213	0.9714	0.8955	0.91742	0.83615	0.85024	0.81001		
30	Macaque	0.5915	0.76278	0.90004	0.75815	0.75693	0.90442	0.76048	0.80546	0.4882	44.904	Null Hypothesis Rejected
	Football	0.80292	0.8125	0.82345	0.78719	0.37782	0.00465	0.45868	0.81644	0.81101		
	Celegansneural	0.74513	0.77752	0.78369	0.74319	0.32566	0.019	0.43761	0.80986	0.77897		
	USAir97	0.81677	0.85996	0.88998	0.83981	0.36812	0.07017	0.70941	0.8636	0.85032		
	Political blogs	0.86271	0.88731	0.89981	0.87817	0.4072	0.19667	0.60622	0.86517	0.88266		
	Yeast	0.79004	0.84402	0.84603	0.84651	0.84519	0.78042	0.83386	0.83471	0.67703		
	Amazon web graph	0.73707	0.60955	0.64005	0.59175	0.47066	0.23873	0.5268	0.62252	0.59947		
	Power grid	0.93327	0.49698	0.56191	0.55913	0.49966	0.10937	0.49698	0.56015	0.5482		
	Ca-GrQc	0.93761	0.8644	0.95637	0.86322	0.49862	0.09151	0.58845	0.8608	0.84441		
	Ca-HepTh	0.91689	0.80546	0.80138	0.80254	0.49943	0.09151	0.45759	0.80022	0.77299		
	Ca-HepPh	0.97038	0.96051	0.96084	0.96084	0.89339	0.9169	0.82919	0.80022	0.77299		
40	Macaque	0.61685	0.73003	0.8832	0.75282	0.74258	0.9028	0.74629	0.76991	0.50097	42.62	Null Hypothesis Rejected
	Football	0.77291	0.77777	0.79831	0.76483	0.40681	0.0046	0.43195	0.76349	0.78828		
	Celegansneural	0.74764	0.75169	0.75133	0.72846	0.35868	0.0189	0.44974	0.77997	0.74867		
	USAir97	0.81444	0.85122	0.87774	0.84676	0.38865	0.09531	0.62982	0.84167	0.83187		
	Political blogs	0.86056	0.89142	0.89165	0.87088	0.42104	0.21101	0.5276	0.84148	0.87017		
	Yeast	0.80769	0.84003	0.84286	0.84217	0.84236	0.78319	0.83267	0.83036	0.64939		
	Amazon web graph	0.81067	0.60829	0.63898	0.60711	0.47841	0.30388	0.48793	0.62335	0.59182		
	Power grid	0.97994	0.49685	0.54597	0.54742	0.49974	0.15502	0.49685	0.54368	0.53582		
	Ca-GrQc	0.94134	0.82526	0.94552	0.8224	0.4989	0.13022	0.57193	0.82966	0.80411		
	Ca-HepTh	0.90694	0.75157	0.75261	0.75634	0.49957	0.13022	0.47881	0.75659	0.72636		
	Ca-HepPh	0.96875	0.93794	0.94639	0.94741	0.89036	0.91561	0.80858	0.75659	0.72636		
50	Macaque	0.56126	0.71594	0.85942	0.72602	0.72339	0.88796	0.7159	0.75059	0.52559	44.876	Null Hypothesis Rejected
	Football	0.74172	0.69971	0.75182	0.71695	0.42341	0.00456	0.44407	0.75167	0.74535		
	Celegansneural	0.7037	0.7108	0.71897	0.66977	0.39684	0.02219	0.47157	0.73282	0.7257		
	USAir97	0.81418	0.84813	0.86302	0.82049	0.41003	0.13808	0.49573	0.77795	0.8365		
	Political blogs	0.85452	0.8967	0.87475	0.8587	0.43442	0.22092	0.47083	0.8047	0.85456		
	Yeast	0.82299	0.83725	0.84017	0.83749	0.83697	0.77849	0.83032	0.83232	0.61378		
	Amazon web graph	0.86576	0.58945	0.60367	0.58361	0.4849	0.35637	0.48466	0.59915	0.59714		
	Power grid	0.98211	0.5	0.53698	0.53492	0.49982	0.21136	0.5	0.53272	0.52725		
	Ca-GrQc	0.93511	0.7897	0.93304	0.78912	0.49914	0.17413	0.53079	0.78822	0.76549		
	Ca-HepTh	0.894	0.69597	0.6987	0.69777	0.49968	0.17413	0.4804	0.7012	0.67221		
	Ca-HepPh	0.96589	0.92157	0.93042	0.92844	0.88683	0.9142	0.77064	0.7012	0.67221		

than the baseline methods except for the Node2vec with medium and large average clustering coefficients. Recently, some sophisticated methods like Node2vec [125] and SPM [183] have been proposed which show outstanding performance. Although the prediction performance of these methods are significantly better, however, in the case of large networks the proposed method (CCLP2) should be considered with these methods at least.

In this work, we have considered simple undirected and unweighted networks (datasets) i.e., only one type of relationship between two nodes have been selected. If we consider

TABLE 4.3: The Posthoc Friedman Conover Test (Control method = CCLP2)

Metric	Removed links (%)	p-value							
		Node2vec	CCLP	CN	RA	PA	CAR	NLC	LNBCN
AUROC	10	0.00219	0.00126	0.00039	3.60E-09	1.00E-11	9.70E-10	0.17335	2.40E-06
	20	0.01193	0.00551	0.06079	3.00E-07	5.70E-10	3.60E-08	0.06079	5.70E-05
	30	0.01079	0.02287	0.00242	5.00E-09	4.90E-11	1.10E-08	0.01468	4.40E-06
	40	0.0346	0.01283	0.0346	5.50E-08	2.40E-10	7.00E-08	0.00811	5.60E-06
	50	0.04009	0.00571	0.00927	6.60E-09	6.50E-11	5.10E-09	0.00673	2.10E-05
Recall	10	0.04495	0.02078	1.37E-01	4.30E-08	1.70E-08	7.40E-05	0.05717	8.93E-03
	20	0.0344	0.03045	0.18656	7.20E-07	5.60E-08	1.07E-02	0.00526	1.54E-03
	30	0.05862	0.02915	0.07288	6.60E-07	1.90E-07	2.58E-02	0.00284	3.30E-03
	40	0.25412	0.04778	0.44606	1.10E-04	9.20E-06	3.06E-02	0.05914	3.84E-02
	50	0.1836	0.0724	0.1292	1.90E-05	9.00E-06	1.89E-02	0.0304	3.82E-02
AUPR	10	0.00923	0.00092	2.30E-05	1.60E-12	7.20E-11	3.25E-02	0.00276	1.50E-06
	20	0.01215	0.00024	7.20E-05	2.30E-11	2.60E-10	2.05E-02	0.00578	9.60E-06
	30	0.01221	0.00019	0.00024	2.40E-10	3.70E-08	7.23E-03	0.02358	9.70E-05
	40	0.0245	0.00812	0.00034	4.60E-09	7.50E-07	2.84E-02	0.01323	9.80E-05
	50	0.01669	0.01435	0.00332	3.40E-08	1.00E-05	1.05E-02	0.05125	7.80E-04
AP	10	0.01492	0.00088	1.40E-04	6.60E-11	1.60E-12	8.20E-06	0.01244	8.30E-05
	20	0.01115	0.00131	6.90E-05	1.70E-09	1.50E-11	8.80E-05	0.02223	1.31E-03
	30	0.01046	0.00126	2.70E-05	6.70E-09	2.10E-10	5.50E-05	0.11963	6.70E-04
	40	0.00881	0.00515	0.00089	4.00E-08	2.50E-09	2.00E-05	0.16345	1.00E-04
	50	0.01434	0.00051	0.00041	1.00E-08	1.80E-09	1.30E-07	0.13366	4.00E-05

multiple relationships into account, the prediction performance can be enhanced [280]. In the future, we will try to explore such an idea in a supervised setting. Moreover, we will also validate our method with the networks having negative links (Signed network) like Epinions[7] and Slashdot[7] networks.