Chapter 3

Features Fusion based Link Prediction in Dynamic Networks

This chapter [1] studies the local, global and quasi-local feature of social networks. The fusion of local, global and quasi-local similarity indices is then applied to find missing links in networks. One of the well-known methods for link prediction is the similarity-based method, which uses similarity-based score. The three widely used similarity-based indices are Local (L), Global (G), and Quasi-local (Q) for calculating similarity scores. In our proposed LGQ model, these wide categories of indices are used in different combinations (L, G, Q, LG, LQ, GQ, LGQ) for feature set generation that can be used with various machine learning techniques for link prediction. In local similarity indices, we have considered Common Neighbors (CN), Adamic/Adar Index (AA), Jaccard Coefficient (JC), and Preferential Attachment (PA). In global similarity indices, we have considered cos+, Average Commute Time (ACT), Shortest Path (SP), MFI (Matrix Forest Index), and in quasi-local indices, local path Index (LP), Path of Length 3 (L3).

[1]Published in Journal of Computational Science. Feature Fusion based Link Prediction in Dynamic Networks

3.1 Introduction

Social networks are a kind of network where individuals are represented by nodes (entity), and the relationship between individuals is represented as the edge between nodes. These relationships can be of different types, such as friendship, common interests, citations, places, work, etc. The links among entities are continuously changing. In real world scenarios, social networks are dynamic in nature. Dynamic networks [129, 130] is characterized by the phenomenon of nodes and edges appearing and disappearing over time. Numerous techniques for link prediction have been implemented. These methods can be categorized into several categories, like similarity-based, probabilistic models, learning-based models etc [1]. The similarity-based approach involves three categories - Local similarity indices (L), Global similarity indices (G), and Quasi-local indices (Q).

The local similarity indices are normally calculated using the information of the local neighborhood of nodes. The example of local similarity indices includes Common Neighbors (CN), Adamic/Adar Index (AA), Jaccard Coefficient (JC), Preferential Attachment (PA).

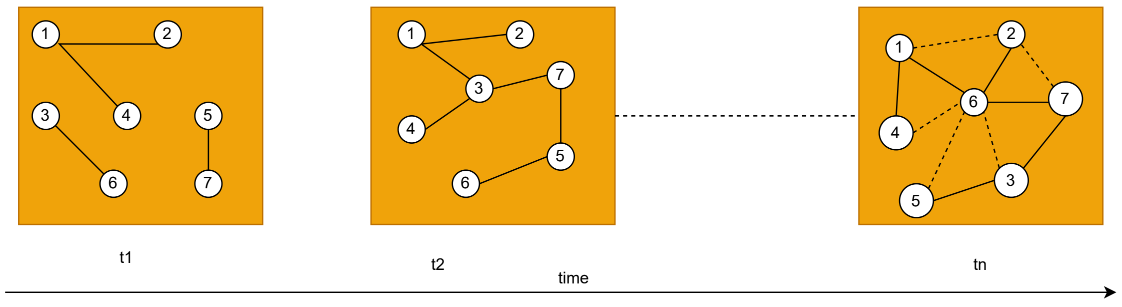
The global similarity indices are calculated with topological information of a network. The global similarity indices are computationally more complex than local similarity-based methods but they provide a better outline of the whole structure of the graph. Some global similarity indices are shortest path (SP), cos+ (Cosine based on L^+ (Cos^+)), Matrix Forest Index (MFI), and Average Commute Time (ACT).

The Quasi-local similarity indices try to create a balance between the properties of local and global similarity measures. Some quasi-local indices are local path Index (LP) and Path of Length 3 (L3) methods.

The detailed explanation and formulas of all the three similarity indices (local, global and quasi-local) are explained in Chapter 2 which deals with the preliminaries required for our proposed framework of link prediction. Link prediction in dynamic networks has a different requirement than that of static networks and hence requires some targeted techniques. Different techniques available for dynamic link prediction include:

- Matrix Factorization
- Probabilistic
- Spectral Clustering
- Time Series
- Deep Learning: Deep learning involves the following category: Embedding, RBM (Restricted Boltzmann machine) and Other deep learning.

FIGURE 3.1: Dynamic Networks with different snapshots. In the snapshot-based paradigm for link prediction on dynamic graphs, feature sets are used to track the behavioral changes in edges. These features are then used to make predictions for non-existing edges.



Our proposal for link prediction is mainly based on similarity indices between nodes. Different similarity indices extract the different types of information from the graph, which is used for link prediction [1]. Different similarity methods have their own advantages and disadvantages. Therefore, all three types of methods, Local similarity indices, Global similarity indices, and Quasi-local similarity indices, can be combined to create a feature set which we believe would be more useful for link prediction than its components individually. In this paper, we have proposed a novel model for building feature vectors for machine learning model-based link prediction that combines the properties of three types of similarity indices. The primary motivations for formulating this method are as follows -

- In the other algorithms [131–133] for link prediction in a dynamic network, the amount of information present is limited. From our approach LGQ model, more information can be extracted from the graph because our model takes into account both the immediate neighborhood of nodes as well the overall topological structure of the graph.
- Earlier solutions have been more focused on the limited number of local and global indices. The quasi-local indices have not been used in combinations for link prediction in dynamic networks.
- Similarity indices are designed for usage in static networks. They can not be directly applied to the dynamic network. In most of the literature, this issue has been overcome by creating a set of static snapshots of a dynamic network and then applying static techniques for prediction.

We have attempted to provide a feature set which tries to represent all the information required for link prediction in a dynamic network. To transform these feature sets into prediction probabilities, we have used machine learning-based classification. Primarily logistic regression, neural network, and boosting algorithms such as XGBoost have been used in literature.

The main contributions of this paper are as follows:

- We have assembled four local, four global, and two quasi-local similarity indices into three groups and studied the result of feature sets generated by them for the problem of link prediction in dynamic graphs.
- Apart from the individual groups, in order to maximize the information represented by feature sets, we have created and tested different combinations of these groups

with each other. This is done in order to create a feature set which represents all the relevant properties of the graph, which may be useful for link prediction and to test the performance of these variations with different machine learning models.

- The combination of groups is inspired by the methodology of unique quasi-local indices, which aim to highlight both local and global features of the graph. As far as we know, this combination approach has not been attempted for feature sets generation using dynamic graphs.
- After creating these feature sets, for the task of link prediction, we have also compared two different learning models for predicting actual edge probabilities. The used models are neural networks and XGBoost.
- Testing these models with our combination feature sets, we have found that for the largest model LGQ, XGBoost is much more suitable than neural network for training and testing datasets. We have also compared our best combination results with state-of-the-art using results in four evaluation matrices, i.e., AUPR, AUC, Balanced accuracy score (BAC), and F1-score.
- Also, we have compared the best performing models in both the neural network as well as the XGBoost category with state-of-art link prediction algorithms using the matrices mentioned above. We have observed a significant increase in performance on three of the four matrices used.

3.2 Proposed work

We present our proposed feature generation model, LGQ, for link prediction problems in dynamic networks. Features used in the proposed LGQ model provide more comprehensive information for link prediction. Features based on common neighborhood, node degree as well as path-based information are used in this model. We have also considered quasi-local indices for the proposed model. These extract more relevant information about local neighborhoods than local similarity indices. Also, the less relevant information used in global similarity indices would be neglected. Both these attributes contribute to the increase in prediction performance [134]. The quasi-local indices perform remarkably better than the neighborhood-based index [135].

In our proposed LGQ model, local similarity indices (L), global similarity indices (G), and quasi-local similarity indices (Q) are combined to generate edge feature set vector for link prediction in dynamic network. Here, we have assembled ten link prediction similarity indices. In L, four local similarity indices (CN, AA, JC and PA) are used. Also in G, four global similarity indices (COS+, ACT, SP and MFI) are used. Finally in Q, two quasi-local similarity indices (LP and L3) are used. To get a rich feature set vector, we have fused these L,G and Q similarity categories in different combinations (L, G, Q, LG, LQ, GQ, LGQ) in dynamic networks. These wide categories of indices are used for feature set vector generation that can be used with different machine learning techniques for link prediction in dynamic network. We have also studied the results obtained from using the LGQ model and its variations in different machine learning algorithms(Neural network and XGBoost).

3.2.1 Proposed Feature generation model Algorithm

The detailed pseudo code of the feature set generation algorithm is shown in Algorithm 1. In line 1-4 is the initialization phase. We have initialized the $true\_es$ (true edges), $false\_es$ (false edges) and all_es (all edges) as empty sets. Also, we have taken an empty dictionary as a means of combining features of edges between snapshots. In our analysis, we have taken five snapshots. The LGQ feature set for edges all_es is generated in lines 8-25. The for loop in line number 8 is for iteration among snapshots. In line 9, the for loop is used for iteration on all edges of set all_es. The local similarity measures are computed in lines 11-14, global similarity measures are calculated, and quasi-local similarity measures are calculated in lines 19-20. Lines 21-24 are for retrieving features of the same edge which was generated for the previous snapshot. Finally, in line 25, features of the current snapshot for the current edge are combined with features for the same edge in the previous snapshot. Line 27-31 are used for defining labels for all edges in set all_es (existence of edge in the final graph). If the edge is present in the last snapshot, the label is defined as 1; otherwise, 0. It is shown in lines 29 and 31. Finally, the generated feature labels for all edges in set all_es are returned in line 32.

We consider the link prediction problem as a binary classification problem. The presence or absence of links denotes the class label. When the link is present between two nodes, the label is denoted by 1; otherwise, the label is taken as 0. The features generated by LGQ model are fed into the Neural Network and XGBoost machine learning algorithms as input.

3.3 Result Analysis

We compare the performance the proposed LGQ model and its variations in terms of AUPR, F1 score, Balanced accuracy score (BAC), and AUC. We also demonstrate and contrast the preference of Neural Network and Xgboost based prediction models for different variations of proposed feature sets. Lastly, we have compared our proposed LQ-NN and LGQ-XGB with three state-of-the-art algorithms [131–133].

3.3.1 Performance of LGQ model and its variations with Neural Network (NN)

In this section, we are going to analyze the experimental details and performance of LGQ model with its variations when training and prediction are made using neural network. For the creation of our machine learning model, Tensorflow has been used to build the model. It is a very popular machine learning framework developed and maintained by Google. Tensorflow provides the inbuilt tools that are very helpful in building the model. While

Algorithm 1: Feature generation Algorithm for proposed LGQ method

```
Input: $G(V,E)$: Dynamic social graph, $v$: no. of nodes, $e$: no. of edges, $t$: Number
       of snapshot, $all\_es$ true and false edge count, $true\_es$ true edges of final
       snapshot, $false\_es$ possible random edges
Output: $P[v][v]$: LGQ feature set for edges
1  $all\_es \leftarrow \emptyset$                                        ▷ Initialization Phase
2  $true\_es \leftarrow \emptyset$
3  $false\_es \leftarrow \emptyset$
4  $edge\_fs \leftarrow dict$        ▷ Edge dictionary for combining feature set between snapshots
5  $true\_es \leftarrow true\_edges(s_n)$
6  $false\_es \leftarrow random\_fasle\_edges(s_n) \ni |true\_es| = |false\_es|$
7  $all\_es \leftarrow true\_es + false\_es$
8  for each snapshot in $s_0, s_1, s_2.....s_{n-1}$ do
9      for each edge_curr in all_es do
10         $node1, node2 \leftarrow edge\_curr$
11         $cn \leftarrow CommonNeighbour(node1, node2, snapshot)$     ▷ For Local similarities
             indices
12         $pa \leftarrow PreferentialAttachment(node1, node2, snapshot)$
13         $aa \leftarrow AdamicAdar(node1, node2, snapshot)$
14         $jc \leftarrow JaccardCoefficient(node1, node2, snapshot)$
15         $mfi \leftarrow Matrixforestindex(node1, node2, snapshot)$  ▷ For Global similarities
             indices
16         $act \leftarrow Averagecommutetime(node1, node2, snapshot)$
17         $cosp \leftarrow Cosinebased(node1, node2, snapshot)$
18         $sp \leftarrow Shortestpath(node1, node2, snapshot)$
19         $lp \leftarrow Localpath(node1, node2, snapshot)$         ▷ For Quasi-local similarities
             indices
20         $l3 \leftarrow Pathoflength3(node1, node2, snapshot)$
21         if edge_fs(edge_curr) not empty then
22             $temp = edge\_fs[edge\_curr]$
23         else
24             $temp = []$
25         $edge\_fs[edge\_curr] \leftarrow temp + [cn, pa, aa, jc, mfi, act, cosp, sp, l3, lp]$
26 $s_n \leftarrow lastsnapshot$                                     ▷ For last snapshot
27 for edge_curr in all_es do
28     if edge_curr in s_n then
29         $edge\_fs[edge\_curr] = edge\_fs[edge\_curr] + ['1']$
30     else
31         $edge\_fs[edge\_curr] = edge\_fs[edge\_curr] + ['0']$
32 return edge_fs
```

training the neural network model, we have used two hidden layers with 1024 neurons each. The learning rate used here is 0.001. The batch size of 32 is taken for training purposes with epoch 5. An Adam optimizer has been used for minimizing cross-entropy. We have used the sigmoid function and ReLu function as an activation function.

The comparison of the performance of LGQ model with its different variations with neural networks is shown in Table 3.1. We have analyzed the best two models as shown in the before mentioned table. The corresponding accuracy values of different models with different metrics are shown in the table. After doing the analysis of all metrics on all datasets, we have come across many interesting results. Although the LGQ feature set has more information, when LQ feature set is used, the performance is better in a neural network in all datasets. The reason behind the LQ-NN's performance is that it has more precise information relevant for link prediction. It a combination of local and quasi-local measures while not taking into account the global ones. This feature set seems to represent the most relevant information about local neighborhoods when using a neural network based prediction model. The LGQ model performs well on redoslaw-email, mit dataset in AUPR metric. We can see from the table, Q-NN is also performing well in most of the cases. Also, L-NN gives better results in some datasets.

For the AUPR metric, for mit dataset, LQ-NN is best, and the second best is Q-NN. In the radoslaw-email dataset, LGQ-NN is best closely followed by LQ-NN and LG-NN. The radoslaw-email seems to be a dataset where global similarity indices give better performance when compared to the relative pattern for other datasets. Q-NN is the best on fb-forum and CollegeMsg datasets, with LQ-NN being a close second. In mathoverflow and lkml-reply datasets, there is a close competition between L-NN, Q-NN, and LQ-NN feature sets.

For the F1 metric, LQ-NN is the best performing for all datasets except lkml-reply, where it is slightly worse than Q-NN. For the Balanced Accuracy Score (BAC), LQ-NN has the best performance in all datasets. But Q-NN has very close performance with LQ-NN in fb-forum, CollegeMsg, mathoverflow and lkml-reply datasets. For mit and radowslaw-email dataset, LGQ-NN gets the second best performance. For AUC, LQ-NN has the best performance in all datasets except lkml-reply, where it is slightly outperformed by Q-NN. Q-NN has the second best performance in fb-forum, CollegeMsg, and mathoverflow datasets. For mit and radowslaw-email datasets, L-NN and LGQ-NN have a decent performance for the AUC metric.

From all the experiments based on Neural Network based prediction we can see that for maximum performance, the prediction model is partial towards feature sets which contain quasi local based features. But in LGQ which is largest feature sets by feature count its performance degrades as compared to LQ model, hence we can conclude that this approach is very susceptible to the number of features offered as input to the model and their overall relevance to prediction.

TABLE 3.1: Performance of LGQ model and its variations with Neural Network (NN)

Metric	Dataset	L-NN	G-NN	Q-NN	LG-NN	LQ-NN	GQ-NN	LGQ-NN
AUPR	mit	0.74001	0.79012	**0.83622**	0.79243	**0.86005**	0.82943	0.82676
	radoslaw-email	0.77975	0.66601	0.7807	0.8311	**0.83293**	0.79934	**0.83612**
	fb-forum	0.61241	0.65964	**0.93703**	0.62234	**0.93382**	0.65259	0.68112
	CollegeMsg	0.56813	0.57976	**0.7309**	0.57465	**0.7279**	0.52799	0.56521
	mathoverflow	**0.82286**	0.72548	0.82144	0.74644	**0.82761**	0.76928	0.76855
	lkml-reply	**0.87844**	0.72789	0.87633	0.83033	**0.87732**	0.83099	0.83512
F1 - score	mit	0.64895	0.64105	0.64426	0.68518	**0.77668**	**0.73841**	0.67895
	radoslaw-email	0.73979	0.44799	0.69288	0.71613	**0.76959**	0.66662	**0.74428**
	fb-forum	0.47277	0.45509	**0.86898**	0.36094	**0.87294**	0.4587	0.45445
	CollegeMsg	0.40961	0.45544	**0.56207**	0.4715	**0.56596**	0.44462	0.50068
	mathoverflow	0.58846	0.64125	0.63682	0.63883	**0.64246**	**0.64165**	0.64035
	lkml-reply	0.74367	0.74386	**0.7558**	0.7443	**0.75549**	0.74181	0.74531
BAL ACC SCORE	mit	0.67307	0.66572	0.70225	0.6876	**0.77701**	**0.73832**	0.71409
	radoslaw-email	0.73344	0.55966	0.7166	0.73147	**0.76814**	0.69382	**0.74388**
	fb-forum	0.56611	0.55055	**0.87014**	0.5108	**0.87732**	0.55372	0.56287
	CollegeMsg	0.5287	0.48704	**0.62453**	0.51092	**0.64381**	0.46428	0.50287
	mathoverflow	0.69673	0.68334	**0.71741**	0.68119	**0.72145**	0.68293	0.68276
	lkml-reply	0.78395	0.7605	**0.7893**	0.76102	**0.78949**	0.75959	0.76191
AUC	mit	0.70261	0.69264	0.72294	0.72524	**0.82084**	**0.7486**	0.74573
	radoslaw-email	**0.79088**	0.58126	0.76202	0.76297	**0.82246**	0.71961	0.77798
	fb-forum	0.5933	0.55647	**0.91529**	0.51503	**0.91398**	0.5576	0.56493
	CollegeMsg	0.54474	0.49404	**0.67693**	0.51346	**0.6861**	0.45908	0.50898
	mathoverflow	0.70339	0.68201	**0.73245**	0.68584	**0.73703**	0.69329	0.69067
	lkml-reply	0.79766	0.76146	**0.81125**	0.7685	**0.81123**	0.77112	0.77507

3.3.2 Performance of LGQ and its variations with XGBoost

In this section, we are analyzing the performance of LGQ model and its variations using XGBoost as a training and testing model on feature sets. In the XGBoost model, we have used the learning rate as 0.01 with n_estimators equal to 50. Here n_estimators is the number of boosting rounds. The maximum depth of a single tree is considered as six. XGBoost can handle different types of sparsity patterns in the input data more efficiently. It uses max_depth parameter and starts tree pruning from the backward direction. This improves the computational performance and speed of the XGBoost framework.

The performance of LGQ model and its variations with XGBoost is shown in Table 3.2. After doing the analysis of the accuracy of different metric, we observe that LGQ feature set performs very well over its all variations on all datasets. The second variation which performs well is GQ model. We can form a conclusion that while using the Neural Network model, the best features used were of the quasi-local type. Instead, on using the XGBoost model, both global and quasi-local features become important to the prediction performance of the model. As from the result table, we can The LG model also performs well on some datasets.

For the AUPR metric, in mit and radowslaw datasets, G-XGB, LG-XGB, and LGQ-XGB show comparable performance. For mathoverflow and lkml-reply, LQ-XGB, GQ-XGB, and LGQ-XGB show comparable performance. For fb-forum dataset, LG-XGB, GQ-XGB and LGQ-XGB perform very competitively. For CollegeMsg dataset, there is close performance gap between LG-XGB, LQ-XGB and LGQ-XGB methods. For F1 score, there is a minimal performance gap between LG-XGB, GQ-XGB, and LGQ-XGB for all datasets except CollegeMsg and lkml-reply. For these datasets, LQ-XGB shows comparable performance to the other algorithms. For the Balanced Accuracy Score, GQ-XGB and LGQ-XGB show comparable performance. G-XGB has a decent performance in all datasets except CollegeMsg. For the AUC metric, LGQ-XGB and GQ-XGB show comparable performance across all datasets and G-XGB shows decent performance for all datasets except CollegeMsg.

TABLE 3.2: performance of LGQ and its variations with XGBoost (XGB)

Metric	Dataset	L-XGB	G-XGB	Q-XGB	LG-XGB	LQ-XGB	GQ-XGB	LGQ-XGB
AUPR	mit	0.85154	**0.89876**	0.86876	0.89015	0.86557	0.88866	**0.90123**
	radoslaw-email	0.84737	0.88926	0.84392	**0.89586**	0.85345	0.88786	**0.89265**
	fb-forum	0.64346	0.93226	0.92509	**0.93431**	0.9121	**0.93784**	0.93386
	CollegeMsg	0.58647	0.75384	0.7789	0.77921	**0.78289**	0.77529	**0.7836**
	mathoverflow	0.81949	0.81776	0.82107	0.81649	0.82097	**0.82872**	**0.82647**
	lkml-reply	0.86731	0.867	0.86598	0.86663	0.86749	**0.86823**	**0.86851**
F1 - score	mit	0.80817	0.85138	0.83453	0.85444	0.82934	**0.85708**	**0.86197**
	radoslaw-email	0.80275	0.84669	0.7972	**0.85595**	0.81072	0.84793	**0.85327**
	fb-forum	0.47107	0.88944	0.88124	0.88936	0.86514	**0.90248**	**0.89446**
	CollegeMsg	0.32536	0.52724	0.57962	0.57458	0.58562	**0.60842**	**0.59052**
	mathoverflow	0.58996	0.64115	**0.6437**	0.64181	0.63856	**0.6451**	0.64253
	lkml-reply	0.7395	0.75693	0.75672	0.75587	0.75589	**0.7584**	**0.75764**
BALL ACC SCORE	mit	0.80297	0.84915	0.83262	0.84792	0.81903	**0.8501**	**0.85686**
	radoslaw-email	0.79412	0.84546	0.78285	**0.85318**	0.80239	0.84478	**0.84945**
	fb-forum	0.56229	0.89712	0.8891	0.89508	0.87252	**0.90684**	**0.90082**
	CollegeMsg	0.51764	0.6472	**0.68023**	0.66402	**0.67916**	0.66575	0.66816
	mathoverflow	0.69812	0.71786	0.72119	0.7178	0.7193	**0.72431**	**0.72252**
	lkml-reply	0.78342	0.7921	0.79152	0.79105	0.79145	**0.79349**	**0.79312**
AUC	mit	0.80297	0.84915	0.83262	0.84792	0.81903	**0.8501**	**0.85686**
	radoslaw-email	0.79412	0.84546	0.78285	**0.85318**	0.80239	0.84478	**0.84945**
	fb-forum	0.56229	0.89712	0.8891	0.89508	0.87252	**0.90684**	**0.90082**
	CollegeMsg	0.51764	0.6472	**0.68023**	0.66402	**0.67916**	0.66575	0.66816
	mathoverflow	0.69812	0.71786	0.72119	0.7178	0.7193	**0.72431**	**0.72252**
	lkml-reply	0.78342	0.7921	0.79152	0.79105	0.79145	**0.79349**	**0.79312**

3.3.3 Comparison of LGQ model with state-of-the-art methods

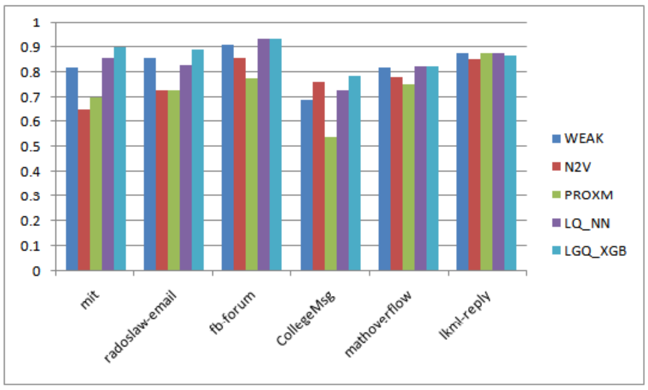
From Table 3.1, we observe that LQ model with neural network performs better than other variations of the LGQ model. This is because neural network are more sensitive with respect to the number of features as compared to XGBoost. Also, when we compare the results of Table 3.2, we find our LGQ model with XGBoost gives a better result compared to all other variations of the LGQ model. Finally, we compare our best models from both the neural network and XGBoost based prediction with three state-of-arts methods [131–133]. In Table 3.3, we have compared the performance of LGQ model with the XGBoost and LQ model with neural network in terms of AUPR, F1 score, Balance Score, and AUC values with the three state-of-arts methods [131–133]. Our LGQ model outperforms in terms of AUPR, F1 score, Balance Score. In terms of AUC measure, our model LGQ outperforms very closely to one of the state-of-the-art methods. Fig.3.2 shows the comparison of our model with state-of-arts methods.

TABLE 3.3: Comparison of LGQ model and its variation with state-of-the-art methods

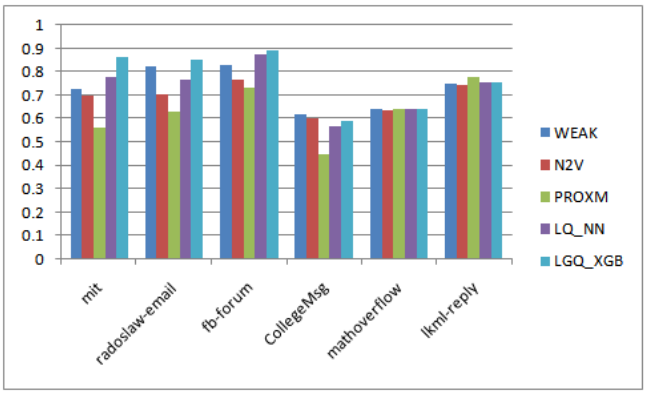
	Dataset	WEAK	N2V	PROXM	LQ-NN	LGQ-XGB
AUPR	mit	0.81955	0.65357	0.69731	0.86005	**0.90123**
	radoslaw-email	0.85881	0.72948	0.72606	0.83293	**0.89265**
	fb-forum	0.91248	0.86042	0.77348	0.93382	**0.93386**
	CollegeMsg	0.69009	0.76162	0.53835	0.7279	**0.7836**
	mathoverflow	0.81535	0.77766	0.7513	**0.82761**	0.82647
	lkml-reply	0.87729	0.85572	**0.87916**	0.87732	0.86851
F1 - score	mit	0.72793	0.70159	0.56461	0.77668	**0.86197**
	radoslaw-email	0.82365	0.70648	0.62973	0.76959	**0.85327**
	fb-forum	0.83066	0.76669	0.73051	0.87294	**0.89446**
	CollegeMsg	**0.61901**	0.59964	0.4492	0.56596	0.59052
	mathoverflow	0.64051	0.63353	0.63933	0.64246	**0.64253**
	lkml-reply	0.74845	0.74559	0.77663	0.75549	**0.75764**
BAL ACC SCORE	mit	0.72368	0.63528	0.80921	0.77701	**0.85686**
	radoslaw-email	0.8141	0.66511	0.84302	0.76814	**0.84945**
	fb-forum	0.8384	0.78121	0.8232	0.87732	**0.90082**
	CollegeMsg	0.60863	0.6583	0.65769	0.64381	**0.66816**
	mathoverflow	0.70965	0.69433	**0.74458**	0.72145	0.72252
	lkml-reply	0.78599	0.77391	**0.83135**	0.78949	0.79312
AUC	mit	0.80174	0.66724	**0.90279**	0.82084	**0.85686**
	radoslaw-email	0.88331	0.73781	**0.91677**	0.82246	0.84945
	fb-forum	0.90646	0.84315	0.79896	**0.91398**	**0.90082**
	CollegeMsg	0.67109	**0.71939**	0.6565	0.6861	0.66816
	mathoverflow	0.73305	0.71538	**0.78729**	0.73703	0.72252
	lkml-reply	0.81362	0.80291	**0.92396**	0.81123	0.79312

For the AUPR metric, LGQ-XGB has the best performance across all datasets except mathoverflow and lkml-reply, where it is the second and third best algorithm, respectively. PROXM algorithm has the best result for lkml-reply, and LQ-NN has the best performance in the mathoverflow dataset. For the F1 score, LGQ-XGB has the best performance across all datasets except CollegeMsg, in which the WEAK algorithm shows the best results. In the Balanced Accuracy score metric, LGQ-XGB has the best performance for radowslaw-email, mit, fb-forum, and CollegeMsg datasets. It is the second best algorithm for mathoverflow and lkml-reply datasets, only behind the PROXM algorithm. For the AUC matrix, the PROXM algorithm has the best results in mit, radowslaw-email, mathoverflow and lkml-reply datasets. Our algorithms LQ-NN and LGQ-XGB show the best performance only in the fb-forum dataset.

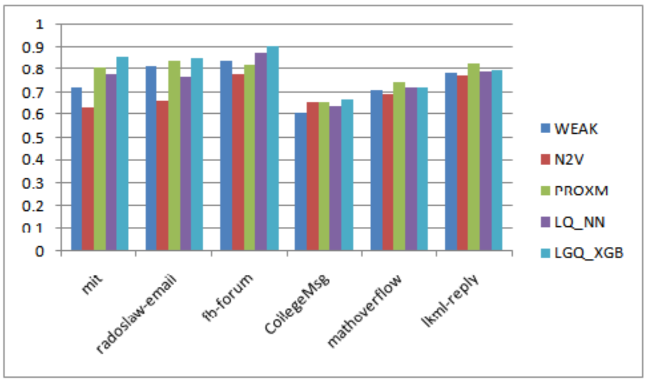
FIGURE 3.2: Comparison of LGQ approach with state-of-the-art



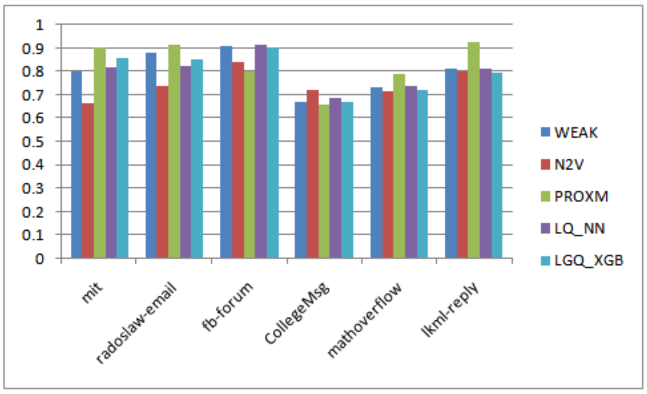
(A) AUPR



(B) F1 Score



(C) Balance score



(D) AUC

We can also observe from all our results that there are fundamental differences between Neural Network and XGBoost when performance over the same kind of feature sets are observed. The LQ variation performs best with Neural network, but they are unable to show the same kind of performance when used with XGBoost, which favors variations with global similarity matrices. Hence, we can infer that at the time of assembling a feature set, if Neural Network have to be used, more emphasis should be given to quasi-local similarity indices. But if XGBoost has to be used, global similarity indices should be preferred.

3.4 Conclusion

In this work, we proposed the LGQ model and its variations to generate feature sets for link prediction in a dynamic network. Our model provides a rich feature set for link prediction by combining different similarity scores in dynamic networks. The addition of global and quasi-local similarity scores shows significant changes to the performance of link prediction. Combined, these features represent different properties of the graph ranging from local neighborhoods to the full structure of the graph. The patterns in these properties help us in improving the performance of link prediction in dynamic graphs by a decent margin. We have also tested the performance of these features with different machine learning models (NN and XGB) to further enhance the understanding of the pairing of different types of feature sets with different models. We conclude from the results that Neural Network based prediction model shows best performance with feature sets having low number of features and has a preference for quasi local similarity indices while XGBoost doesn't have any such preference o specific datasets and works best with LGQ based feature set. Experimental results show that our method achieves significant improvement to state-of-art methods for six datasets. In cases where our algorithm is not the best, it shows comparable results with other state-of-the-art algorithms.