Chapter 2

Literature Review

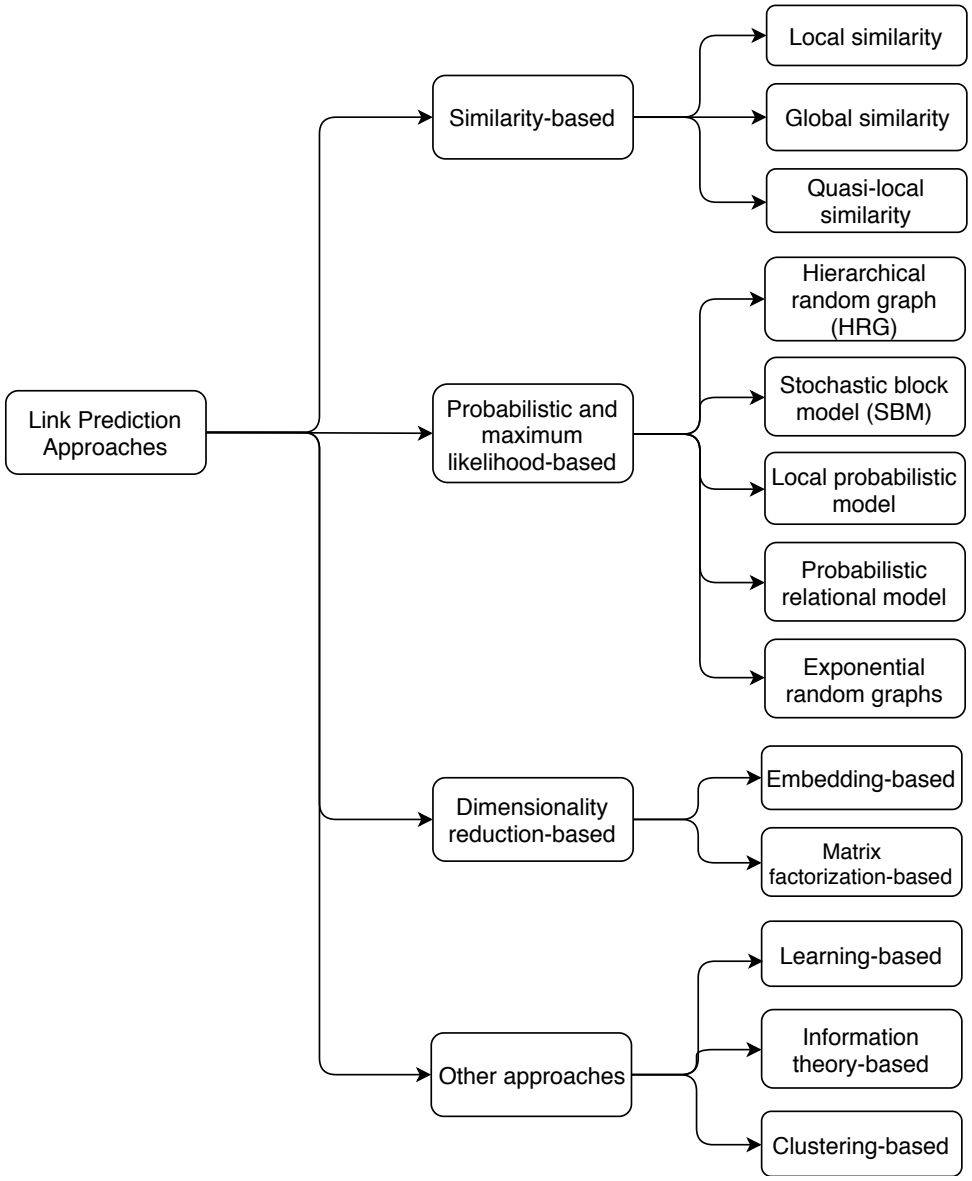
In this chapter, we present a review of earlier techniques emphasizing link prediction with the point of convergence mostly on social network graphs. We classify these approaches into a number of categories. One of these categories computes a similarity score between pairs of vertices, where higher scored pairs are considered to have links between them.

Recently, numerous methodologies of link prediction have been implemented. These methods can be grouped into several categories, like similarity-based, probabilistic models, learning-based models, etc.

2.1 Similarity-based methods

Similarity-based metrics are the simplest one in link prediction, in which for each pair x and y, a similarity score $S(x,y)$ is calculated. The score $S(x,y)$ is based on the structural or node's properties of the considered pair. The non-observed links (i.e., $U - E^T$) are assigned scores according to their similarities. The pair of nodes having a higher score represents the predicted link between them. The similarity measures between every pair can be calculated using several properties of the network, one of which is structural

FIGURE 2.1: Taxonomy of Link Prediction Approaches [1]



property. Scores based on this property can be grouped in several categories like local and global, node-dependent and path-dependent, parameter-dependent and parameter-free, and so on.

2.1.1 Local similarity based indices

Local indices are generally calculated using information about common neighbors and node degree. These indices consider immediate neighbors of a node. Examples of such indices includes.

- **Common neighbors (CN)** . The Common neighbors [51] for a given pair of nodes x and y in a specific network or graph, the size of the intersection of the two nodes' neighborhoods is used to calculate the size of the shared neighborhoods.

$$S(x,y) = |\Gamma(x) \cap \Gamma(y)|, \tag{2.1}$$

  where $\Gamma(x)$ and $\Gamma(y)$ are neighbors of the node x and y respectively. With more common neighbors between them, the likelihood that there is a link between x and y increases.

- **Jaccard coefficient (JC)**. Jaccard coefficient [52] is similar to the common neighbor. It normalizes common neighbor score. It is calculated as.

$$S(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}, \tag{2.2}$$

  where $\Gamma(x)$ and $\Gamma(y)$ are neighbors of the node x and y respectively.

- **Adamic/Adar index (AA)**. Adamic and Adar [53] presented a metric to calculate a similarity score between two web pages based on shared features, which are further used in link prediction after some modification by Liben-Nowell et al. [20].

$$S(x,y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log k_z}, \tag{2.3}$$

  where k_z is the degree of the node z. It is clear from the equation that more weights are assigned to the common neighbors having smaller degrees. This is also intuitive in the real-world scenario, for example, a person with more number of

friends spend less time/resource with an individual friend as compared to the less number of friends.

- **Preferential attachment (PA)**. Using the concept of preferential attachment [54], a growing scale-free network is created. The phrase "expanding" refers to the network's nodes' gradual emergence over time. The degree of the node, kx, determines the possibility of an incrementing new connection linked with node x. The preferential attachment score between two nodes x and y is computed as.

$$S(x,y) = k_x.k_y. \tag{2.4}$$

  In a supervised learning framework, Hasan et al. [55] showed that aggregate functions (e.g., sum, multiplication, etc.) over feature values of vertices could be applied to compute link feature value.

- **Resource allocation Index (RA)** . The original dynamics of this similarity index is originated from Ou et al. [56] work published in "Physical Review E" on resource allocation [57] dynamics on complex networks. Consider two non-adjacent vertices x and y. If node x transmits certain resources to node y through the shared nodes of both x and y, then the similarity between the two vertices is computed in terms of the resources delivered from x to y. Mathematically, it is expressed as.

$$S(x,y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{k_z}. \tag{2.5}$$

  This similarity measure and the Adamic/Adar are very similar to each other, as shown by the equations 2.5 and 2.3, respectively. The RA index, in contrast to the AA index, severely penalizes to higher degree nodes.

2.1.2 Global similarity indices

Global indices are computed using entire topological information of a network. The computational complexities of such methods are higher and seem to be infeasible for large networks. Global similarity-based features are usually calculated using information from the whole graph structure. The following are examples of link prediction methods of this category used in this paper.

- **Shortest Path(SP)** . There are several different algorithms that can be used to determine the shortest path [20] between two vertex pairs in a graph depending on circumstances [58–60]. Shortest Path is calculated as.

$$SP(x,y) = -|d(x,y)|, \tag{2.6}$$

  where the shortest path $d(x,y)$ between the node pair (x,y) is calculated using the Dijkstra algorithm [58].

- **Cos+ (COSP)**. Any inner product metric, such as the cosine similarity [61], can be used to determine how similar two nodes x and y are to one another. The cosine similarity time metric is based on $L^{\dagger}$ by calculating similarity of two vectors. It is calculated by the following formula.

$$COSP(x,y) = \frac{L^{\dagger}_{x,y}}{\sqrt{L^{\dagger}_{x,x}L^{\dagger}_{y,y}}}. \tag{2.7}$$

- **Matrix Forest Index (MFI)**.

  MFI [62] employs the spanning tree principle. It contains fewer links than the original graph.

$$MFI = (I+L)^{-1}, \tag{2.8}$$

where $(I+L)_{(x,y)}$ is the count of spanning rooted forests (x as root) that include both the nodes x and y. This value is also identical to the co-factor of $(I+L)_{(x,y)}$.

- **Average Commute Time(ACT).**

  It is based on the random walk concept. A random walk is a Markov chain [63, 64] which describes the movements of a walker. To calculate the average commute time [65], the random walk method is employed.

$$n(x,y) = m(x,y) + m(y,x). \tag{2.9}$$

  This above equation can be made simpler using the pseudo-inverse of the Laplacian matrix L^+

$$n(x,y) = |E|(l^+_{xx} + l^+_{yy} - 2l^+_{xy}), \tag{2.10}$$

  where l^+_{xy} denotes the (x,y) entry of the matrix L^+. Pseudo-inverse of the Laplacian, L^+ can be computed as

$$L^+ = (L - \frac{ee^T}{n})^{-1} + \frac{ee^T}{n}, \tag{2.11}$$

  where e denotes a column vector consisting of 1's.

2.1.3 Quasi-local indices

Quasi-local indices have been developed as a combination of local and global perspectives, or between performance and complexity, as shown in Table 2.1. As with local indices, these metrics can be computed quickly. Some of these indices analyze the network's whole topological data and extract it. When compared to global techniques, these indices' time complexity is still lower. Examples of such indices include local path index, local random walk index [66], local directed path (LDP) [67], etc.

- **Local path index (LP)**. With the intent to furnish a good trade-off between accuracy and computational complexity, the local path-based metric [68] is

considered. Mathematically, it is computed as.

$$S^{LP} = A^2 + \varepsilon A^3, \tag{2.12}$$

where ε represents a free parameter. Clearly, the measurement converges to common neighbor when $\varepsilon = 0$. If there is no direct connection between x and y, $(A^3)_{xy}$ is equated to the total different paths of length 3 between x and y. The index can also be expanded to generalized form

$$S^{LP} = A^2 + \varepsilon A^3 + \varepsilon^2 A^4 + ... + \varepsilon^{(n-2)} A^n, \tag{2.13}$$

where n is the maximal order. Computing this index becomes more complicated with the increasing value of n.

- **Path of length 3 (L3)**.

  Georg Simmel, a German sociologist, first coined the concept "triadic closure" and made popular by Mark Granovetter in his work [69] "The Strength of Weak Ties". The authors [70] proposed a similarity index in protein-protein interaction (PPI) network, called path of length 3 (or $L3$) [71] published in the Nature Communication. They experimentally show that the triadic closure principle (TCP) does not work well with PPI networks. They showed the paradoxical behavior of the TCP (i.e., the path of length 2), which does not follow the structural and evolutionary mechanism that governs protein interaction. The TCP predicts well to the interaction of self-interaction proteins (SIPs), which are very small (4%) in PPI networks and fails in prediction between SIP and non SIP that amounts to 96%. They showed that the $L3$ index performs well in such conditions and give mathematical expression to compute this index [70] as

$$L3(x,y) = \sum_{u,v} \frac{a_{x,u}.a_{u,v}.a_{v,y}}{\sqrt{k_u.k_v}}. \tag{2.14}$$

Recently, Pech et al. [71] in Physica A, proposed a work that models the link prediction as a linear optimization problem. They introduced a theoretical explanation of how direct count of paths of length 3 significantly improves the prediction accuracy.

- **Similarity based on local random walk and superposed random walk (LRW and SRW)** .

  Liu and Lü [66] proposed new similarity measures by exploiting the random walk concept on graphs with limited walk steps [66]. They defined node similarity based on random walks of lower computational complexity compared to the other random walk based methods [72]. The probability of getting from a random walker starting at node x to node y in t steps is given by

$$\vec{\pi}_x(t) = P^T \vec{\pi}_x(t-1), \tag{2.15}$$

  where $\vec{\pi}_x(0)$ is a column vector with x^{th} element as 1 while others are 0's and P^T is the transpose of the transition probability matrix P. P_{xy} entry of this matrix defines the probability of a random walker at node x will move to the next node y. It is expressed as $P_{xy} = \frac{a_{xy}}{k_x}$, where a_{xy} is 1 when there is a link between x and y and 0, otherwise. The authors computed the similarity score (LRW) between two nodes based on the above concept as

$$S^{LRW}(x,y) = \frac{k_x}{2|E|}\pi_{xy}(t) + \frac{k_y}{2|E|}\pi_{yx}(t). \tag{2.16}$$

  This similarity measure focus on only few steps covered by the random walker (hence quasi-local) and not the stationary state compared to other approaches [66, 72].

  Random walk based methods suffer from the situation where a random walker moves far away with a certain probability from the target node whether the target node is closer or not. This is an obvious problem in social networks that show a high clustering index i.e., clustering property of the social networks. This degrades

the similarity score between the two nodes and results in low prediction accuracy. One way to counter this problem is that continuously release the walkers at the starting point, which results in a higher similarity between the target node and the nearby nodes. By superposing the contribution of each walker (walkers move independently), SRW is expressed as

$$S^{SRW}(x,y)(t) = \sum_{l=1}^{t} S^{LRW}(l), \tag{2.17}$$

2.1.4 Clustering based Features

The clustering coefficient is a measurement of how closely nodes in a graph cluster together. In a social network, the clustering coefficient measures the likelihood that a user's friends are also friends with one another [73].

- **CAR-based Common Neighbor Index (CARCN)** . The concept behind CAR-based indices [38] is that a relationship between two nodes is more likely if their shared neighbours are members of a local community.

$$\begin{aligned} CAR(x,y) &= CN(x,y) \times LCL(x,y) \\ &= CN(x,y) \times \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{|\gamma(z)|}{2}, \end{aligned} \tag{2.18}$$

  where $CN(x,y) = |\Gamma(x) \cap \Gamma(y)|$ represents the number of common neighbors. $LCL(x,y)$ denotes the number of local community links, which are defined as connections between seed nodes x and y's shared neighbors. The subset of neighbors of node z that are also common neighbors of x and y is called $\gamma(z)$

- **Node clustering coefficient (CCLP).**

  This index [36] is also based on the clustering coefficient property of the network in which the clustering coefficients of all the common neighbors of a seed node pair are

computed and summed to find the final similarity score of the pair. Mathematically, this index can be expressed as follows.

$$S(x,y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} C(z), \tag{2.19}$$

where

$$C(z) = \frac{t(z)}{k_z(k_z-1)},$$

is clustering coefficient of the node z and $t(z)$ is the total triangles passing through the node z.

- **Node and link clustering coefficient (NLC)**.

  This similarity index is based on the basic topological feature of a network called "Clustering Coefficient" [14, 15]. The clustering coefficients [39] of both nodes and links are incorporated to compute the similarity score.

$$S(x,y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{|\Gamma(x) \cap \Gamma(z)|}{k_z - 1} \times C(z) + \frac{|\Gamma(y) \cap \Gamma(z)|}{k_z - 1} \times C(z). \tag{2.20}$$

- **Level-2 node clustering coefficient (CCLP2)**. The level-2 [37] node gets more clustering information from the seed node pair's level-2 common neighbors and uses it to compute the similarity score. In contrast to the level-1 common neighbors and their corresponding clustering coefficients, it explores significantly more information about networks. The lavel-2 node clustering coefficient is calculated using the following equation:

$$\begin{aligned} CCLP^2_{(A,B)} = & \sum_{CN^2_A \in \Gamma(A) \cap \Gamma(CN^1)} CC(CN^2_A) + \\ & \sum_{CN^2_B \in \Gamma(CN^1) \cap \Gamma(B)} CC(CN^2_B) \end{aligned} \tag{2.21}$$

TABLE 2.1: Comparison of similarity-based approaches [1]

Properties	Local Indices	Global Indices	Quasi-local Indices
Nature	Simple	Complex	Moderate
Features employed	Local neighborhood	Entire network	More local neighborhood
Computational complexity	Low	high	Moderate
Parallelization	Easy	More complex	Moderate
Implementation	Feasible for large networks	Feasible for small networks	Feasible for large networks

In order to calculate the similarity score, similarity-based techniques mainly concentrate on the structural characteristics of the networks. Local approaches typically take into account local information (immediate neighbors or neighbors of neighbors), which requires less computing time. This characteristic makes the local techniques feasible for large real-world network datasets. Since global approaches consider the complete network's structural data, it takes more time to gather the data than it does for local and quasi-local approaches. Additionally, particularly in a decentralized context, complete topological information might not always be accessible at the time of computation. In contrast to local and quasi-local approaches, parallel processing over global approaches may not be viable or be exceedingly difficult. When compared to local and global techniques, quasi-local approaches extract more structural information.

2.2 Dynamic Networks

Compared to a static network, a dynamic network allows for a more thorough depiction of complex relationships. Dynamic network analysis is deployed in several fields. In the literature, link prediction in dynamic networks has been explored in various methods. Recently link prediction in dynamic networks has become an emerging topic of research and various authors such as, Ma et al.[23], Ahmed et al.[24], Yasami et al.[25], Wu et al.[22] have presented a solution to this problem.

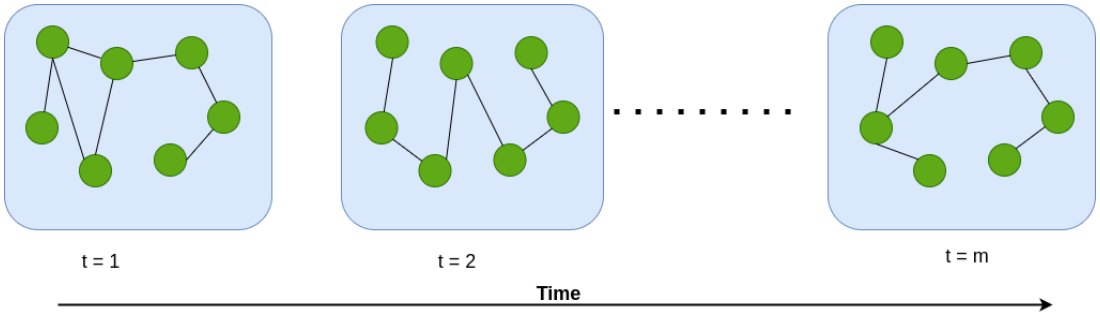
In dynamic networks, link prediction refers to the technique of identifying new or absent linkages in the graph at various timestamps $(time = 1, time = 2, ..., time = m)$. Formally, we can define link prediction in dynamic graph as "Let $G = (V, E)$ be a dynamic network,

where V is the set of vertices and each edge (u, v) $\in$ E represents the relation or link between u and v. Let G_1,G_2,G_3.....G_n are the networks at different snapshots t_1, t_2, t_3, ..., t_n, we need to predict G_{t+1} at $t+1$ time." In a dynamic network, prospective linkages can become genuine links as the network evolves. In these kind of networks, improving link prediction accuracy is always an objective. Fig. 2.2 demonstrates the schema of a dynamic graph containing different snapshots.

Several efforts have been employed by the researchers in this direction in the last decade. Purnamrita et al. [74] introduced a nonparametric method for temporal network link prediction where the time dimension is partitioned into subsequences of snapshots of the graph. This approach predicts links based on topological features and local neighbors. Dunlavy et al. [75] employ matrix and tensor techniques in a framework where matrix part collapses sequence of snapshots of networks into a single matrix and computes link scores using truncated svd and extended Katz methods. The tensor part computes the scores using heuristics and temporal forecasting. The tensor part captures the temporal patterns effectively in the network, but it costs heavily also. Moreover, Gao et al. [76] proposed a model based on latent matrix factorization that employs content values with the structural information to capture the temporal patterns of links in the networks. Table 2.2 shows some more works in this direction.

Machine learning methods may be used to detect a node pair's missing or future link. There are a variety of machine learning-based link prediction algorithms, and the basic link prediction method is based on similarity. Link prediction was considered a supervised learning task by Hasan et al. [77], Fire et al. [78] created a set of node similarity attributes based on network topology, then forecast the link using a standard machine learning classification technique, which produced better results. The machine learning algorithm treats the prediction process as a classification problem. Other recent research has focused on the development of network analysis and machine learning techniques for modeling dynamic networks [79]. Zhu et al. [80] have attempted to incorporate a temporal regularise into the matrix factorization framework in order to

FIGURE 2.2: Structure of a dynamic network with different snapshots



generate a temporal latent space using snapshot networks. The machine learning model considers the prediction process as a classification task [81, 82]. The classifier estimates the class of each edge, whether it exists or not.

TABLE 2.2: Link Prediction in Dynamic Networks

Models	Network Types	Characteristics	References
Learning-based models	Coauthorship networks	High computational cost	Vu et al.[83], Pujari et al.[84], Zeng et al.[85], He et al.[86], Bao et al.[87], Madadhain et al.[88], Bringmann et al. [89]
Heuristics-based models	Twitter, Collaboration and Coauothorship networks	Fast convergence and high precision	Catherine et al.[90], Sherkat et al. [91]
Probabilistic model	Nodes-attributed graphs	Characterize the stochastic and dynamic relations. Need prior link distribution so impractical for real networks	Hu et al.[92], Barbieri et al.[93], Gao et al.[76], Ji Liu et al.[94], Hanneke et al. [95]

2.2.1 Dynamic Datasets

Seven well-known temporal networks with various graph sizes, densities, and time periods were used to evaluate our results. The datasets are publicly available in Stanford Large Network Dataset Collection [96]. Also, the various network data statistics are shown in Table [1] 2.3. The information about datasets and created snapshots used in experiment is

[1] https://networkrepository.com/

shown in Table 2.3. It comprises information on the datasets we've collected as well as the snapshots into which the datasets are being divided. We adopted an equal time intervals strategy to create the snapshots. The approximate time between each snapshot's starting and completing edges remains constant. Due to the temporal nature of the datasets, each edge has a timestamp associated with it. The initial edge's time stamp appears in the Timestamp Start column, while the last edge's time stamp appears in the Timestamp end column.

The various datasets used are.

- **mit**[2] [97]. It contains 96 nodes and 1,086,404 edges. This is an undirected network incorporates human contact data from 100 students at the Massachusetts Institute of Technology (MIT), acquired as part of the Reality Commons project's Reality Mining experiment in 2004. The information was gathered over a nine-month period utilizing 100 mobile phones. A individual is represented as a node, and an edge indicates that the nodes in question were in physical contact.

- **radoslaw-email**[3] [98]. It has 82876 edges and 167 nodes. It's a mid-sized industrial company's internal email communication network for employees. Employees are represented by nodes, while individual emails between two users are represented by edges.

- **Eu-core**[4]. The number of nodes and edges are 986 and 332334. An edge represents an email sent from one institution member to another at a certain moment in this dataset, which contains over 300,000 emails sent among teachers at a European university. The dataset stands out for its dense graph structure and well-defined community structure.

- **fb-forum** [5]. It has 899 nodes and 33686 edges. The Facebook-like Forum Network arose from the same online community as the online social network. However, the

[2]http://konect.cc/networks/mit/
[3]http://networkrepository.com/ia-radoslaw-email.php
[4]http://snap.stanford.edu/data/email-Eu-core.html
[5]http://networkrepository.com/fb-forum.php

focus in this network is on user participation in the forum rather than on private messages sent among users.

- **CollegeMsg**[6]. It contains 1899 nodes and 59835 edges. Over the course of a half-year, the CollegeMsg dataset documents student interactions in a social network at the University of California, Irvine. An edge here indicates a private message sent at a given moment from one student to another.

- **Lkml-reply**.[7] There are 26885 vertices and 1,028,233 connections in this graph. This is the Linux kernel mailing list's communication network. Every directed edge is a response from one member to the other, whereas nodes are people (recognized by respective email addresses).

- **Mathoverflow**.[8] There are 24818 vertices and 506550 connections in this graph. On the stack exchange platform Math Overflow, this is a temporal network of exchanges. The Stack Exchange Data Dump was used to create these graphics. The 'OwnerUserId' label in the data dump corresponds to the node ID numbers.

2.2.2 Link Prediction Framework in dynamic networks

The overall process of link prediction model is depicted in Figure 2.3. It's broken down into the steps below.

- The edge list of the overall graph is used as input in the first stage. It is made up of the network's complete edge list.

- We partition the whole edge list into snapshots like G_0, G_1, ..., G_n based on equal time intervals because this is a dynamic graph. We partition the dataset's whole time range into equal-sized chunks such that the time range difference between

[6]https://snap.stanford.edu/data/CollegeMsg.html
[7] https://networkrepository.com/lkml-reply.php
[8] https://snap.stanford.edu/data/sx-mathoverflow.html

TABLE 2.3: Dataset Information along with corresponding snapshot information and time intervals (AVG DEG - Average Degree, AVG SP - Average Shortest Path Length, CLUSTER - Average Clustering Coefficient, HETERO - Heterogeneity, ASSOC - Associativity)

DATASET	NODES	EDGES	Start Time	End Time	AVG DEG	DENSITY	AVG SP	CLUSTER	HETERO	ASSOC
MIT	96	1086404	1095183096	1115253696	52.9	0.5568	1.43	0.75	1.13	-0.06
MIT-SNAP1	96	1792	1095183096	1099197216	37.33	0.393	1.61	0.65	1.23	-0.03
MIT-SNAP2	96	1778	1099197216	1103211336	37.04	0.3899	1.56	0.65	1.25	-0.01
MIT-SNAP3	96	817	1103211336	1107225456	17.02	0.1792	1.92	0.49	1.77	-0.08
MIT-SNAP4	96	622	1107225456	1111239576	12.96	0.1364	1.96	0.5	2.05	-0.29
MIT-SNAP5	96	362	1111239576	1115253696	7.54	0.0794	1.93	0.37	2.95	-0.44
Radoslaw-Email	167	82876	1262454010	1285884492	38.92	0.2345	1.96	0.59	1.66	-0.3
Radoslaw-Email-SNAP1	167	2018	1262454010	1267140106	24.17	0.1456	1.94	0.53	1.98	-0.28
Radoslaw-Email-SNAP2	167	1851	1267140106	1271826202	22.17	0.1335	2	0.47	1.99	-0.25
Radoslaw-Email-SNAP3	167	1294	1271826202	1276512298	15.5	0.0934	2.24	0.31	1.86	0
Radoslaw-Email-SNAP4	167	1238	1276512298	1281198394	14.83	0.0893	2.15	0.32	1.88	-0.05
Radoslaw-Email-SNAP5	167	1428	1281198394	1285884490	17.1	0.103	2.1	0.37	1.91	-0.11
EU-Core	986	332334	0	69459254	32.58	0.0331	2.58	0.41	2.29	-0.03
EU-Core-SNAP1	986	8012	0	13891850	16.25	0.0165	2.84	0.3	2.59	0.02
EU-Core-SNAP2	986	8259	13891850	27783700	16.75	0.017	2.79	0.31	2.73	-0.02
EU-Core-SNAP3	986	9247	27783700	41675550	18.76	0.019	2.79	0.32	2.48	0
EU-Core-SNAP4	986	5526	41675550	55567400	11.21	0.0114	3.03	0.28	2.73	-0.01
EU-Core-SNAP5	986	1093	55567400	69459250	2.22	0.0023	4.24	0.11	5.07	-0.2
FB-Forum	899	33686	1084585996	1098798101	15.65	0.0174	2.83	0.06	2.16	-0.11
FB-Forum-SNAP1	899	5950	1084585996	1087428417	13.24	0.0147	2.85	0.06	2.28	-0.1
FB-Forum-SNAP2	899	1663	1087428417	1090270838	3.7	0.0041	3.67	0.01	3.1	-0.05
FB-Forum-SNAP3	899	1255	1090270838	1093113259	2.79	0.0031	3.6	0.01	4.82	-0.15
FB-Forum-SNAP4	899	858	1093113259	1095955680	1.91	0.0021	4.04	0.01	5.37	-0.08
FB-Forum-SNAP5	899	728	1095955680	1098798101	1.62	0.0018	4.48	0	5.41	-0.14
CollegeMsg	1899	59835	1082040961	1098777142	14.57	0.0077	3.05	0.11	3.82	-0.19
CollegeMsg-SNAP1	1899	8289	1082040961	1085388197	8.73	0.0046	3.05	0.08	5.15	-0.21
CollegeMsg-SNAP2	1899	4802	1085388197	1088735433	5.06	0.0027	3.45	0.04	4.53	-0.15
CollegeMsg-SNAP3	1899	1116	1088735433	1092082669	1.18	0.0006	3.7	0.02	15.14	-0.2
CollegeMsg-SNAP4	1899	790	1092082669	1095429905	0.83	0.0004	4	0.01	15.86	-0.21
CollegeMsg-SNAP5	1899	497	1095429905	1098777141	0.52	0.0003	4.38	0	16.43	-0.26
Mathoverflow	24759	390441	1254192988	1457262355	15.19	0.0006	3.23	0.31	19.33	-0.22
Mathoverflow-SNAP1	24818	102359	1254192988	1294806861	4.1	0.0002	3.01	0.07	38.86	-0.2
Mathoverflow-SNAP2	24818	83030	1294806861	1335420734	3.71	0.0001	3.25	0.07	32.02	-0.17
Mathoverflow-SNAP3	24818	76310	1335420734	1376034607	3.39	0.0001	3.39	0.07	30.01	-0.18
Mathoverflow-SNAP4	24818	65789	1376034607	1416648480	2.87	0.0001	3.58	0.05	27.81	-0.18
Mathoverflow-SNAP5	24818	62953	1416648480	1457262353	2.74	0.0001	3.63	0.05	29.26	-0.17
Lkml-Reply	26885	1028233	1136080607	1388528616	11.9	0.0004	5.37	0.31	28.32	-0.18
Lkml-Reply-SNAP1	27927	194692	1136080607	1186570208	3.26	0.0001	6.25	0.1	53.26	-0.17
Lkml-Reply-SNAP2	27927	213186	1186570208	1237059809	2.95	0.0001	5.26	0.09	56.44	-0.18
Lkml-Reply-SNAP3	27927	198691	1237059809	1287549410	2.64	0.0001	5.38	0.08	52.9	-0.17
Lkml-Reply-SNAP4	27927	194763	1287549410	1338039011	2.62	0.0001	5.32	0.08	48.53	-0.15
Lkml-Reply-SNAP5	27927	226900	1338039011	1388528612	2.64	0.0001	4.72	0.08	47.46	-0.15

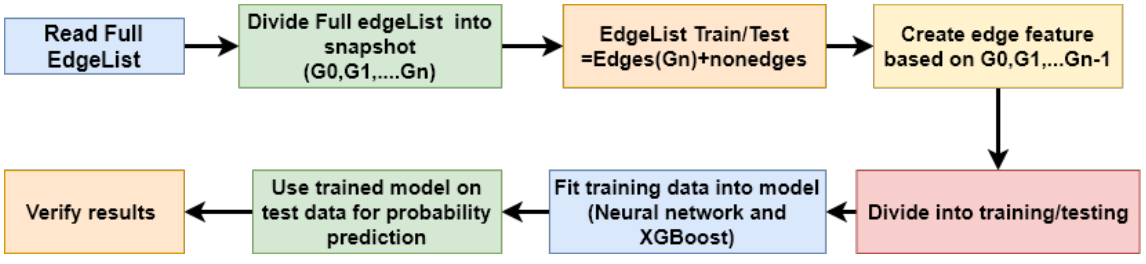
each snapshot is roughly equal. As indicated in Table 2.3, each snapshot has only edges that belong to this time range.

- For our analysis we have taken five snapshots. To create our training and testing edge lists, we combine the most recent snapshot with the randomized non edges. The training and testing edge sets are created by randomly dividing the combined set of true and non edges into training and testing edge sets, with a ratio of training edges to all edges of 0.7.

- In the next step we create edge features based on $G_0, G_1, \ldots, G_{n-1}$ for all edges of training and testing edge sets. The feature set is based on topological features and clustering features.
- Based on the presence of an edge in either the training or testing subset, we divide the data into training and testing sets. We took five snapshots in all, four of which were used to generate features and the fifth was utilized to identify the class label.
- Machine learning models such as Neural Network (*NN*), Logistic Regression (*LR*), XGBoost (*XGB*), Random Forest Classifier (*RFC*), linear Discriminant Analysis (*LDA*) and Gaussian Naive Bayes (*GNB*) are used to process the training data. The probabilities of existing edges on test data are then predicted using these models.
- Finally, we establish performance benchmark metrics based on these predictions, which are compared to findings from state-of-the-art approaches.

The link prediction problem is considered as a binary classification problem. The class label is determined by the existence or absence of links. When a link exists between two nodes, the label is set to 1; otherwise, it is set to 0. If the edge was present in the previous snapshot, the label is set to 1; otherwise, it is set to 0. Finally, all edges' feature labels are returned. The overall link prediction methods in dynamic network

FIGURE 2.3: Flow of link prediction framework in dynamic networks



2.3 Machine learning classifiers in link prediction

Link prediction problem is considered as a supervised learning problem [77, 90]. The employment of a machine learning classifier in a link prediction problem has a number of benefits.

- Automation of Everything (More reliable)
- Wide Range of Applications
- Scope of Improvement
- Efficient Handling of Data
- Best for Education and Online Shopping

A link prediction problem that uses a machine learning classifier has a number of shortcomings as well.

- Algorithm Selection
- Data Acquisition (Machine Learning requires massive datasets to train on, and these should be inclusive/unbiased, and of good quality.)
- Time and Resources
- High error-susceptibility.

The purpose of this research is not only to present and analyze several link methods that have been proposed in the literature but also to look into some novel approaches to enhance the accuracy.

The link prediction problem is considered as a binary classification problem [1, 21, 99]. There are a number of machine learning-based link prediction algorithms in addition to

the conventional similarity-based techniques [77, 78, 100]. The machine learning prediction-based paradigm addresses the process of prediction as a classification issue. Other recent research has focused on the development of network analysis and machine learning techniques for modeling dynamic networks [79, 80, 101].

Due to the fact that a dynamic network evolves with time, time-series modeling and prediction is an appropriate technique for researching link prediction in dynamic networks. The purpose of time series forecasting is to forecast possible values of a variable based on historical data of the same variable. Numerous situations have made effective use of time series forecasting. In addition, the literature gives a thorough summary of complex network methodologies for nonlinear time series analysis [102, 103].

2.3.1 Machine learning classifier

Using a similarity or probabilistic function, similarity and probabilistic approaches produce a score for each non-observed connection. The link prediction problem, on the other hand, can be approached as a learning-based model that utilizes topological graph features and attribute data. In a binary classification problem where several classifiers like different machine learning models are used, which can be used to predict the label of unknown data points (corresponding to missing links in the network). The selection of a suitable feature set is one of the primary issues of this model [1, 35, 50]. The many machine learning algorithms that were employed are highlighted in this section.

- **Neural Network (NN)** . Neural networks [104, 105] are a sort of machine learning technology that describes intricate patterns in datasets by employing multiple hidden layers and non-linear activation functions. A single layer neural network has a straightforward training method. The error or loss function can be calculated using the weight. As a result, gradient is simple to compute. Because of the levels

of hidden layers in a multi-layer network, calculating the loss function is difficult. Back-propagation can be used to calculate it. The direct application of dynamic programming is the back-propagation approach.

- **Logistic Regression (LR)** .

  Logistic regression [106, 107] is a probability classifier that maps feature variables to class probabilities by employing modeling assumptions. For link prediction, Logistic Regression is deployed as a classifier. It is calculated by conditional probability $P(Y = 1|X_1, ..., X_n)$ through

$$P(Y = 1|X_1, ..., X_n) = \frac{exp(\beta_0 + \beta_1 X_1 + ... + \beta_n X_n)}{1 + exp(\beta_0 + \beta_1 X_1 + ... + \beta_n X_n)} \tag{2.22}$$

  where $Y = 1$ means positive class, $X_1 ... X_n$ are feature variables, and $\beta_1 ... \beta_n$ are regression coefficients, which are estimated by maximum-likelihood from the considered data set.

- **XGBoost (XGB)**. Based on the enhancement and expansion of the GBDT, XGBoost [108, 109] is a scalable end-to-end tree boosting system. Regularization models (L1 and L2) are included in XGBoost to control over-fitting by smoothing the final learned weights and therefore improve performance. In addition, XGBoost features a better tree learning algorithm for sparse data, and a good weighted quantize sketch procedure for approximation tree learning can handle instance weights. As a result of this advantage, XGBoost uses parallel and distributed computing to speed up learning.

- **Random Forest Classifier (RFC)** . Random forest [110, 111] is an "ensemble learning" technique that uses averaging to aggregate the predictions of multiple randomized decision trees. Random forest reduces variance when compared to a single decision tree.

- **Linear Discriminant Analysis (LDA)**. It is a widely used dimensionality reduction technique [112, 113]. As the name implies, dimensionality reduction procedures

reduce the number of dimensions (i.e. variables) in a dataset while preserving as much data as possible.

2.3.2 Ensemble learning

Ensemble learning is based on the deliberate generation and optimal combination of many learning models for addressing the issues like classification. Ensemble learning is widely employed in a variety of disciplines, including sentiment analysis [114], customer credit risk [115], aviation incident risk [116] and many more. Usually, it is used in supervised machine learning tasks. Multiple classifier systems are another term for ensemble learning systems. If there are large variations or diversity among the models, ensemble methods get superior results [57, 117]. Ensemble learning can be divided into three types- Stacking, boosting, and bagging.

- **Stacking**. By altering the model types fit on the training data and utilizing a model to combine predictions, stacking [118] is an ensemble method for finding a diverse group of members. Stacking has its own terminology, with level-0 models referring to ensemble members and level-1 models referring to the model that is used to integrate the forecasts. Although more levels of models can be utilized, the most frequent strategy is a two-level hierarchy of models. Instead of a single level-1 model, we might have three or five level-1 models and a single level-2 model that integrates level-1 model predictions to generate a forecast.

- **Boosting**. Boosting [119] is a type of ensemble learning in which the weighting of the samples varies over time, allowing the system to optimize its choice by taking into account the results of the samples in proportion to their (positive) impact on overall system accuracy. The samples are initially uniformly weighted during boosting. The samples that are adequately assigned are weighted lower after each iteration of the algorithm than the ones that are inappropriately assigned.

- **Bagging**. The ensemble learning method of tagging, also known as bootstrap aggregation, is often used to reduce variance within a noisy dataset. Bagging [120] is the process of selecting a random sample of data from a training set with replacement, that is, the individual data points might be chosen many times. These weak models are then trained individually after multiple data samples are collected, and depending on the type of task, regression or classification, the average or majority of those predictions yield a more accurate estimate.

2.3.3 Performance evaluation metrics

We used a number of measures in this thesis, primarily threshold-curve-focused methods that are effective with unbalanced datasets. We have evaluated our approach on four evaluation matrices - Area under the precision–recall curve (AUPR) [121, 122], F1 score [123], Balance-accuracy [124], Area under the ROC curve (AUROC) [125, 126], Average precision [127], Precision [128] and Recall [127]. In this work, link prediction problem is considered as a binary classification task.

- **Area under the precision–recall curve (AUPR).** AUPR is the average of precision across all recall values. When applied to binary classification, it is more useful and informative. The precision-recall curve, which is a plot between the precision values on the y-axis and the recall values on the x-axis, is used to calculate it.

- **F1 score.** The F1-score is a measure of a test's accuracy in binary classification statistical analysis. It can be calculated using the following formula:

$$F1\_score = 2 * \frac{Precision * Recall}{Precision + Recall}. \tag{2.23}$$

- **Balance-accuracy.** To deal with imbalanced data sets, the balanced accuracy in binary and multi-class classification problems is used. It's the average of the recall

scores for each class. When adjusted=False, the best value is 1 and the worst value is 0.

- **Area under the ROC curve (*AUROC*).**

  It is used to determine the accuracy of predictive algorithms. The AUC determines the likelihood that a non-observed but existing link will have a higher score than other truly nonexistent relationships. Mathematically, *AUROC* is expressed as.

$$AUC = \frac{n_1 + 0.5n_2}{n}, \tag{2.24}$$

  where n=independent comparison , n_1 times = the true link has higher score than the false link and n_2 times = equal score. For any network, the set of existing edges, E, is randomly divided into two sets, the set of training edges, E^T and the set of test edges E^P. where $E^T \cap E^P = \phi$ and $E^T \cup E^P = E$.

- **Average precision.**

  The average precision value [9] is calculated by averaging the precision across all recall values between 0 and 1. It is calculated as:

$$Average\ precision = \int_{r=0}^{1} p(r)dr,$$

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

- **Precision**. It's calculated by dividing the number of correctly anticipated positive cases by the total number of positive examples predicted. The precision is calculated as.

$$Precision = \frac{true\_positives}{(true\_positives + false\_positives)}. \tag{2.25}$$

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