TLP-NEGCN: Temporal Link Prediction via Network Embedding and Graph Convolutional Networks

Akshi Kumar, Abhishek Mallik, and Sanjay Kumar

***Abstract*—Temporal link prediction (TLP) is a prominent problem in network analysis that focuses on predicting the existence of future connections or relationships between entities in a dynamic network over time. The predictive capabilities of existing models of TLP are often constrained due to their difficulty in adapting to the changes in dynamic network structures over time. In this paper, an improved temporal link prediction model, denoted as TLP-NEGCN, is introduced by leveraging network embedding, graph convolutional networks (GCN), and bidirectional long short-term memory (BiLSTM). This integration provides a robust model of TLP that leverages historical network structures and captures temporal dynamics leading to improved performances. We employ graph embedding with self-clustering (GEMSEC) to create lower-dimensional vector representations for all nodes of the network at the initial timestamps. The node embeddings are fed into an iterative training process using GCNs across timestamps in the dataset. This process enhances the node embeddings by capturing the network's temporal dynamics and integrating neighborhood information. We obtain edge embeddings by concatenating the node embeddings of the end nodes of each edge, encapsulating the information about the relationships between nodes in the network. Subsequently, these edge embeddings are processed through a BiLSTM architecture to forecast upcoming links in the network. The performance of the proposed model is compared against several baselines and contemporary temporal link prediction models on various real-life temporal datasets. The obtained results based on various evaluation metrics demonstrate the superiority of the proposed work.**

***Index Terms*—Complex networks, Graph Convolutional Networks (GCN), Graph Embedding with Self Clustering (GEMSEC), Network embedding, Temporal link prediction**

I. Introduction

Temporal network analysis is a branch of network science that focuses on the study of dynamic or time-varying networks. Unlike traditional static networks, where the connections between entities are fixed and do not change over time, temporal networks foster links or relationships between nodes that appear over time. Various real-life networks like communication networks, social networks, collaboration networks, and biological networks, are complex and dynamic networks. A temporal network provides enormous scope for data analysis, especially the prediction of probable future links amongst various elements of a system. The task of predicting probable links amongst the nodes based on the topological and structural details of the network is known as link prediction [1]–[3]. In other words, it can be stated as, the problem of predicting upcoming links amongst the nodes of a network using the existing structural information of the network. As one of the instances of social network analysis, link prediction has many real-life applications like network forecasts, recommendation systems, disease outbreaks, etc., wherein existing network information can be used to reveal interactions and associations amongst the users of the network.

Temporal link prediction (TLP) is a prominent task in network analysis which involves forecasting potential future connections between nodes in a network based on their past topological patterns. In a temporal link prediction problem, typically a historical record of interactions or connections between entities at different time points is provided. These entities could be users in a social network, products in an e-commerce platform, genes in a biological network, or any other relevant nodes in a network. The goal is to predict which new connections or relationships are likely to form in the future. By anticipating the evolution of the network, TLP can be instrumental in various applications such as estimating future interactions, proactive resource allocation, improved recommendations, and others. Mathematically, temporal link prediction can be formulated as follows. Given a temporal network represented as a set of time-stamped edges $E = (u, v, t)$ where u and v are nodes in the network, t is the timestamp of the edge, and a historical record of such edges observed up to time k, $E = (u, v, t)|t \leq t_k$. The task of temporal link prediction is to predict the existence of edges at future time points $t > k$, typically represented as

$$E' = (u, v, t)|t > t_k \tag{1}$$

The problem of temporal link prediction is primarily tackled using probabilistic-based techniques, maximum likelihood-based techniques, and similarity-based techniques. Recently, deep learning-based techniques have been used extensively in various domains and have given exceptional results [4], [5]. In the recent literature, several deep learning-based techniques have been explored for link prediction with reasonably good performances.

The real-life temporal networks are usually vast in size, which incurs a substantial computational cost for any useful analysis and mining work [6]. The high cost proves

Manuscript received-; revised- .

Akshi Kumar is with the Department of Computing, Goldsmiths, University of London, 8 Lewisham Way, London SE14 6NW United Kingdom. Email:- Akshi.Kumar@gold.ac.uk

Abhishek Mallik and Sanjay Kumar are with the Department of Computer Science and Engineering, Delhi Technological University, New Delhi-110042, India, Email:- abhishekmallik265@gmail.com, sanjay.kumar@dtu.ac.in

Corresponding author: Sanjay Kumar (sanjay.kumar@dtu.ac.in)

0000–0000/00$00.00 © 2021 IEEE

as an impediment to performing any efficient analysis of such networks. This is where network representation learning comes into play [7], [8]. It is a representational paradigm that preserves the network topology and structural details and represents the nodes in a lower dimensional vector space [9]. However, this approach has a major drawback - it requires a large amount of memory to store the dense adjacency matrix, making it difficult to scale up to larger networks. Such representation of the network into smaller vector space is known as a graph or network embedding, which generates an embedding for all nodes of the network. The graph embedding technique uses the nodes' proximity substructure to create a lower-dimensional vector representation, thereby optimally capturing the neighborhood information of the node under consideration. These techniques consider the first and higher-order proximities in a network. Hence, they can capture the inherent dynamics either explicitly or implicitly, thus enabling their use in link prediction techniques [10]. However, traditional graph embedding techniques suffer from another limitation - they cannot handle dynamic networks, where the links change over time [11], [12].

In this paper, we introduce an improved temporal link prediction model, named TLP-NEGCN, by integrating graph embedding with self-clustering (GEMSEC) and graph convolutional networks (GCN). The proposed model starts by generating the node embeddings for all the nodes at the initial timestamp using the GEMSEC algorithm. The node embeddings are then fed to GCN iteratively on the timestamps of the datasets to obtain improved embeddings by capturing the changing opinions and the information contributed by the neighborhood of the users in the network. The node embeddings are then employed to produce the edge embeddings between two nodes by concatenating the node embeddings. The dimensionality of the node embeddings is varied from 4 to 256 to estimate the performance of the node embedding with increasing dimensionality. The generated embeddings are then passed through a bidirectional long short-term memory (BiLSTM) to make the link prediction on future timestamps. The BiLSTM helps in capturing the short-term periodicity in opinion changes for link prediction. The experiments are performed on various real-life and synthetic temporal networks. The acquired results based on various evaluation metrics demonstrate the efficacy of the proposed model of temporal link prediction. The innovation lies in integrating Graph Embedding with Self Clustering (GEMSEC) and Graph Convolutional Networks (GCN) to capture evolving opinions and network dynamics while optimally preserving topological features. This module assembly provides a robust model of TLP that leverages historical network structures and captures temporal dynamics leading to improved performances. We exploit the topological feature preservation nature of the node embedding, the closeness of similar node embedding in the feature space, and the neighborhood aggregation capability of the GCN. The contribution of this work is a specialized model for temporal link prediction, demonstrated through experiments on diverse temporal networks. Overall, the proposed model showcases its superior performance in dynamic settings along with being time efficient. The main contributions of this article can be summarized as follows.

(i) The proposed model efficiently captures the structural information of the network and transforms the high-dimensional node features into lower-dimensional vector representations through GEMSEC.
(ii) Through iterative training with GCN, the model effectively learns the changes in link connectivity patterns as the network evolves over time. By continuously updating the node embeddings based on neighborhood interactions and opinion changes, the TLP-NEGCN model ensures that it adapts to the dynamic nature of temporal networks.
(iii) The incorporation of a Bidirectional Long Short-Term Memory (BiLSTM) architecture further reinforces the model's capability to explore and leverage temporal dependencies within the data. BiLSTM enables the model to consider both past and future contexts while processing the edge embeddings.
(iv) The research encompasses real-life datasets from various domains, as well as synthetic datasets with varying sizes and applications. The proposed model's efficacy is rigorously assessed through extensive experimentation on diverse temporal network datasets.
(v) By evaluating the confidence intervals over AUC values for all datasets, the study provides a comprehensive assessment of the model's performance. This statistical analysis adds an extra layer of confidence to the findings, strengthening the credibility of the proposed TLP-NEGCN model and its superiority over other methods.

The remainder of this paper is organized as follows. Section II discusses some of the existing works previously done in the field of temporal link prediction. Section III discusses the proposed model for temporal link prediction in detail. The datasets used and the performance evaluation metrics chosen are presented in Section IV. Section V presents the experimental analysis of the proposed model. Finally, section VI concludes the paper.

II. Related Work

In network science, temporal link prediction has drawn huge attraction from all over the research community. Various temporal link prediction techniques have been presented over the years that can be categorized into some broad classes like matrix factorization-based, maximum likelihood-based methods, time series, and deep learning-based techniques [13]. Matrix factorization-based temporal link prediction represents the network features by adjacency matrices. These techniques factorize matrices to create the characteristics needed to carry out the link prediction task. Ma et al. [14] introduced a technique by utilizing non-negative matrix factorization (NMF) and graph communicability. The method considered the temporal changes in network structure and leveraged the communicability measure to capture indirect paths between nodes. Ahmed et al. [15] presented a technique, named DeepEye, for predicting links in dynamic networks through the use of non-negative matrix factorization (NMF). They employed NMF to derive low-dimensional representations of nodes, which are then leveraged to forecast future linkages. Yet, matrix

factorization-based techniques depend on the information in the input matrix that may be incomplete or noisy that may be incomplete or noisy. This can limit the accuracy of the predictions. Also, these methods can become computationally expensive for large networks, making them less scalable. Maximum likelihood-based algorithms work by ascertaining the network structure's organizational details by maximizing the probability of the observed structure. These organizational details are based on some set of principles and rules, which can be helpful in determining the probability of non-existent links. However, the problem of high computational complexity and difficulty in modeling dynamic changes are the main concerns of maximum likelihood-based techniques.

Recently various machine learning and deep learning techniques emerged toward link predictions in temporal networks. Ozcan et al. [4] introduced an approach for predicting links in heterogeneous networks that evolve over time. The proposed method utilized the Nonlinear AutoRegressive beside inputs (NARX) neural network, which is designed to handle the heterogeneity and dynamic nature of the network. An ensemble-learning model integrating logistic regression and the Xgboost method for link prediction was introduced by Li et al. [16]. In order to extract the link structure information from the subgraph using a residual attention network-based model, Wang et al. [17] constructed a deep convolutional neural network-based technique. Kumar et al. [18] introduced features fusion-based link prediction in temporal networks, named LGQ. The method involved three stages: feature selection, feature fusion, and link prediction. In the feature selection stage, the authors extracted various local (L), global (G), and Quasi-local (Q) node centralities. Then, in the feature fusion stage, the chosen features are merged using a weighted combination method that considers the relevance of each feature toward the link prediction task. Authors in [19] explored a method for temporal link prediction on the WikiLinkGraphs dataset. They first analyzed baseline methods that utilize the structural features of the network. Then formulated a link prediction task as a supervised learning activity and applied different Graph Neural Network techniques for link prediction. Finally, they obtained the best performance with GraphSAGE and CTDNE. Qiu et al. [20] introduced a motifs-based link prediction model for temporal link prediction. A temporal network is divided into several snapshots. They then offered a triad transition matrix prediction tool to determine how the distribution of triads has changed between the various snapshots. The dynamic evolution of the network can be captured by the learned changes in the distribution of triads. A temporal link prediction on weighted dynamic graphs was proposed by Qin et al. [5] utilizing the inductive dynamic embedding aggregation (IDEA) technique. They argued that because IDEA employs an inductive dynamic embedding approach with a careful node alignment unit and adaptive embedding aggregation module, it can handle the temporal link prediction on weighted networks.

The existing methods for temporal link prediction have their advantages and disadvantages, but there is still room for improvement. Specifically, most of the methods do not capture temporal information and global structural information. Moreover, recently introduced neural network-based models produce good results but are computationally intensive and difficult to scale for large networks. Also, balancing accuracy with computational efficiency is a challenge. Therefore, there is a need for a hybrid model that combines the strengths of multiple techniques to capture both temporal and structural information along with being computationally efficient. The proposed TLP-NEGCN framework is a hybrid temporal link prediction model obtained by combining the strengths of GEMSEC, GCN, and BiLSTM to capture both temporal and structural information in temporal networks. The key advantage of our method is that it can capture structures in the network and improve node representations, which in turn improves the performance of temporal link prediction.

III. Proposed work

This section discusses the proposed model of temporal link prediction, named TLP-NEGCN, in detail. We present the description of the various phases of the model such as dataset creation, initial feature generation, iterative feature processing using GCN, and link prediction using BiLSTM. The proposed work introduces a refined temporal link prediction model by leveraging the power of graph Embedding with self-clustering (GEMSEC) and a graph convolutional network (GCN). The motivation behind this work stems from the need to make accurate predictions of future links in temporal networks. The existing models often struggle to adapt to the evolving network structures over time, which limits their predictive capabilities. Also, existing models are computationally intensive and time-consuming, especially for large datasets. The innovation in our approach lies in the systematic integration of GEMSEC and GCN to iteratively capture network dynamics, thereby overcoming the limitations of existing models. GCNs complement GEMSEC's node embeddings by enabling iterative learning, capturing opinion changes over time, and incorporating neighborhood information of users, essential in temporal link prediction. The use of edge embeddings, derived from the node embeddings, enriches link predictions, as it facilitates the representation of edge relationships and captures temporal dynamics in evolving networks. Additionally, employing a BiLSTM architecture for link prediction with edge embeddings strengthens our temporal modeling capabilities, effectively capturing temporal dependencies in the data for more accurate predictions. Overall, the integration of GEMSEC and GCN in the model provides a robust framework, offering an improved approach to leverage historical network structures and capture temporal dynamics to enhance link prediction accuracy in real-world applications.

A. Dataset Creation

For our study, we utilize temporal networks which have changing network structures as time advances. Let a temporal network be denoted as $G(V, E, T)$ with V as the node set, E as the edge set, and T as the number of timestamps. More formally, $G(V, E, T)$ can be represented by Eq. 2 where g_i represents the topology of the network at i^{th} timestamp.

$$G(V, E, T) = \{g_0, g_1, g_2, ..., g_{T-1}\} \tag{2}$$

We split the entire dataset into two parts, one for training our model and the remaining for testing. This is done by keeping the initial k timestamps for training while the rest for testing. The training dataset constitutes the initial 80% of the total timestamps in the dataset leaving the rest 20% timestamps for testing. This is done in accordance with realistic scenarios in which we only have structural information about the network up to a certain timestamp which is used to make predictions for future timestamps. Thus, the training timestamps, T^{Train} can be represented as $\{0, 1, 2, ..., k\}$ while the testing timestamps, T^{Test} is represented by $\{k+1, k+2, ..., T-1\}$. We also split the edge list E into E^{Train} and E^{Test} to represent the edges for the training and testing timestamps. Hence we obtain $G^{Train}(V, E^{Train}, T^{Train})$ as the training network while $G^{Test}(V, E^{Test}, T^{Test})$ as the testing network. To have a balanced dataset, we consider the existing edges as positive samples and perform negative sampling by considering the non-existing edges as negative samples which act as labels for the dataset. Following along these lines, we modify the edge lists E^{Train} and E^{Test} to label lists L^{Train} and L^{Test}, where each pair of nodes (u, v) is associated by a label l having values 1 and 0 on the basis of the existence and non-existence of edges between a pair of nodes, respectively. Thus, the training and testing graphs are modified according to Eq. 3 and Eq. 4.

$$G^{Train}(V, L^{Train}, T^{Train}) = generateLabels(G^{Train}(V, E^{Train}, T^{Train})) \quad (3)$$

$$G^{Test}(V, L^{Test}, T^{Test}) = generateLabels(G^{Test}(V, E^{Test}, T^{Test})) \quad (4)$$

Fig. 1 shows the process of dataset creation employed by us to have well-balanced training and testing datasets. The connected circles refer to the state of the network across the timestamps. We can see the process of reserving the initial k timestamps for training while using the remaining timestamps for testing. The training and testing timestamps constitute 80% and 20% of the total timestamps, respectively. Here, the connected circles refer to the state of the network across each timestamp. The circles represent the nodes while the lines represent the edges. The features for the datasets are generated in the next step.

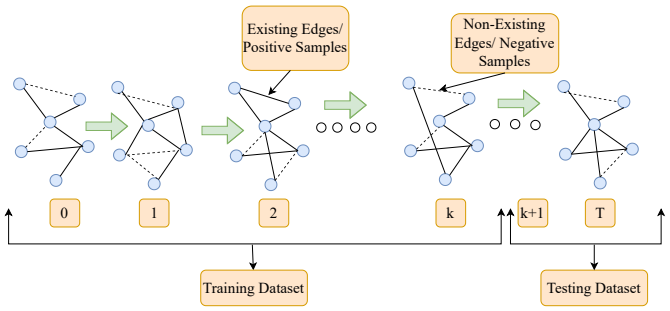

Fig. 1. Splitting of the entire temporal dataset by keeping 80% dataset for training dataset and the remaining for 20% for testing dataset.

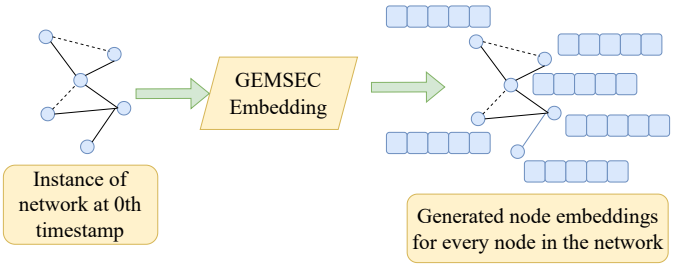

Fig. 2. Generating the initial features for the network's node using GEMSEC embedding. The network structure at the first timestamp is used to generate the embeddings. Here, the connected circles refer to the state of the network at the first timestamp. The circles represent the nodes while the lines represent the edges. The array of squares shows the node embeddings for each node in the network.

B. Initial Feature Generation

The dataset obtained in the previous step has labels, but it lacks features. In this phase, we generate features for the input network. There are various methods to generate features for the nodes in the network, like the adjacency matrix, and modularity matrix. However, these methods generate feature sets of massive size, making it computationally infeasible to process these matrices. To generate features for all nodes, we adopt the recently proposed GEMSEC network embedding [21]. It generates lower-dimensional vector representations for every node of the network in a feature space of dimensions d. This improves the computation cost of our algorithm while preserving the topological features of the network. Moreover, since GEMSEC is a self-clustering-based embedding, it helps generate embeddings with nodes clustered in tight communities. The generated embeddings also ensure that similar nodes are placed closer in the feature space. The dimensionality of feature space (d) is chosen to be 256, which helps ensure feasible computational costs while capturing sufficient network details to be used for link prediction. Let $g_0(V, L_0^{Train}, 0)$ be the structure of the network at the 0^{th} timestamp. So the initial feature vectors are generated using Eq. 5.

$$F_0^{n \times d} = GEMSEC(g_0(V, L_0^{Train}, 0), d) \quad (5)$$

Here, $GEMSEC()$ represents the embedding generation using the GEMSEC technique. Also, $F_0^{n \times d}$ represents the 2-dimensional feature matrix generated having dimensions $n \times d$, where n corresponds to the number of nodes, and d refers to the dimensionality of the feature space. Here, $F(v)$ represents the feature vector for the node v. The generation of the initial features for the network's nodes using the GEMSEC embedding in our model is depicted in Fig. 2. From the figure, we can see how the feature embeddings are generated for every node in the network at the 0^{th} timestamp. The circles represent the nodes while the lines represent the edges. The array of squares shows the node embeddings for each node in the network.

C. Iterative Feature Processing using GCN

In the previous phase, we generated initial features for the dataset. However, these features capture only the initial state

of the network. To further enhance the features of the nodes, we iteratively improve them using a Graph Convolutional Network over the advancing timestamps of the dataset. For iterative training, the embedding generated at timestamp t_i is used as the features of the nodes for GCN at timestamp t_{i+1}. These embeddings then serve as the foundation for iterative encoding and decoding. At each subsequent timestamp, the embeddings generated are utilized as input features for a Graph Convolutional Network (GCN). The GCN learns to model the changing network topologies by aggregating information from neighboring nodes. This iterative encoding process effectively encodes the evolving network structure and dynamics. The resulting embeddings reflect the network's state over time and are used for temporal link prediction. This strategy ensures the model adapts to temporal changes and learns temporal patterns and evolving topology. The encoding at any timestamp can be represented as per Eq. 6.

$$Em(t_i) = \text{GCN}(F(t_i)) \tag{6}$$

where $Em(t_i)$ represents the embeddings generated at timestamp t_i. $F(t_i)$ denotes the features of the nodes at timestamp t_i. GCN signifies the Graph Convolutional Network that encodes the features. The key innovation here is in how the GCN parameters adapt over time to model changing network topologies, which can be mathematically expressed as the evolution of GCN parameters. This is shown in Eq. 7.

$$\Theta(t_{i+1}) = \text{Update}(\Theta(t_i), E(t_i), F(t_{i+1})) \tag{7}$$

where $\Theta(t_{i+1})$ signifies the updated parameters of the GCN at timestamp t_{i+1}. $\Theta(t_i)$ represents the parameters of the GCN at timestamp t_i. $E(t_i)$ represents the embeddings generated at timestamp t_i. $F(t_{i+1})$ represents the features of the nodes at timestamp t_{i+1}. The "Update" function reflects how the GCN parameters are adjusted to model the evolving network structure effectively. This iterative process, along with the neighborhood aggregation feature of GCN, optimally incorporates contributions from neighboring nodes.

This trains the parameters of the GCN to model the changing topology of the networks. The neighborhood aggregation feature of the GCN enables to appropriate incorporation of the contributions of the nodes in the neighborhood of a node under consideration. Thus, we have a feature set that is robust to the varying network structure instead of the predisposed features generated in the previous step. Hence, now we have a proper well-labeled dataset with a proper feature set optimally capturing the topological changes, and neighborhood information of the network structure. We iteratively process the feature vectors of every node in the network as per Eq. 8 till the k^{th} timestamp.

$$F_i^{n\times d} = GCN(F_{i-1}^{n\times d}, g_i(V, L_i^{Train}, i), d), i \in [1, k] \tag{8}$$

Fig. 3 shows how the features or embeddings of the nodes at timestamp t_i are processed through a GCN which in turn are used as the embedding for the nodes at timestamp t_{i+1}. Here, the array of squares shows the node embeddings for each node in the network. The initial embeddings are generated for the 0^{th} timestamp using the GEMSEC embedding technique. The generated embeddings are then processed iteratively while taking into account the embeddings generated in the previous timestamp and the network structure in the current timestamp. This helps our model to iteratively capture the changing interconnectivity structure of the graph as it evolves over time. Here, the connected circles refer to the state of the network across the timestamps. The circles represent the nodes while the lines represent the edges. The array of squares shows the node embeddings for each node in the network.

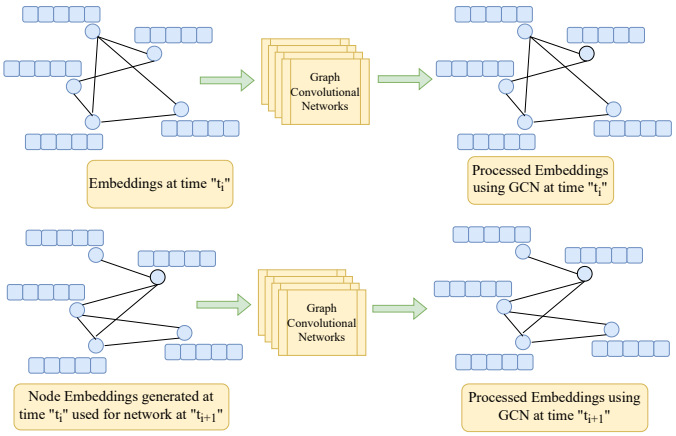

Fig. 3. Processing the features of the nodes iteratively using GCN. The features of the nodes from the previous timestamp are used as the inputs for the current network structure and the embeddings are processed using the GCN architecture. The output of the first step forms the input of the second step.

D. Link Prediction using BiLSTM

In this step, we train a deep Bidirectional Long Short-Term Memory (BiLSTM) model on the training dataset and make link predictions on the testing dataset. BiLSTM is a sequential network model that processes the input both forward and backward across timeframes. It is a collection of multiple LSTM cells. Now, we create a well-labeled dataset using the feature vectors generated in the previous step and the labels generated in section III-A. Eq. 9 and Eq. 10 show the process of generating the training dataset, D^{Train} and testing dataset, D^{Test}.

$$D^{Train} = \{F(u) \oplus F(v), L^{Train}(u, v)\}, u, v \in V \tag{9}$$

$$D^{Test} = \{F(u) \oplus F(v), L^{Test}(u, v)\}, u, v \in V \tag{10}$$

The obtained well-labeled dataset optimally captures the network details iteratively across the timestamps in the training set. The training dataset is employed to train a BiLSTM model sequentially across the timestamps in the training dataset. The sequential processing of BiLSTM allows the model to effectively encode the temporal relationships and the evolving network structure. BiLSTMs consist of two LSTMs, one processes the input sequence in the forward direction, and the

other processes in the backward direction. The BiLSTM's bidirectional processing captures short-term periodicity and dynamic changes in node associations, enhancing link prediction accuracy. The forward LSTM computes hidden states $h\_f[t]$ as per Eq. 11 and the backward LSTM computes the backward hidden states $h\_b[t]$ as per Eq. 12. Here, $LSTM\_forward$ and $LSTM\_backward$ are the LSTM cell operations, and $h\_f[1]$ and $h\_b[T]$ are initialized to zeros. The hidden states from the forward and backward LSTMs are concatenated for each time step to obtain a comprehensive representation of the temporal context as per Eq. 13.

$$h_f[t] = LSTM\_forward(D^{Train}(t), h_f[t-1]) \tag{11}$$

$$h_b[t] = LSTM\_backward(D^{Train}(t), h_b[t+1]) \tag{12}$$

$$h[t] = [h\_f[t], h\_b[t]] \tag{13}$$

For temporal link prediction, we use the concatenated hidden states $h[t]$ as the basis for our predictions. We feed these representations into a fully connected layer and apply a sigmoid activation function to predict the probability of a link existing between two events at different time steps. The sigmoid function outputs a value between 0 and 1, where values closer to 1 indicate a higher likelihood of a link. Let $P(y = 1|t_i, t_j)$ represent the probability of a link between events at time t_i and t_j. We can calculate it as per Eq. 14. Here, W is the weight matrix of the fully connected layer, $[h[t_i], h[t_j]]$ is the concatenated hidden state representation of events at time t_i and t_j, and b is the bias term.

$$P(y = 1|t_i, t_j) = sigmoid(W * [h[t_i], h[t_j]] + b) \tag{14}$$

Our model can thereby predict future links without prior knowledge of the network's future structure. This innovation, as illustrated in Figure 4, enables our method to excel in temporal link prediction, making it a significant advancement in leveraging temporal information for robust link prediction in dynamic networks. Once the BiLSTM is trained on the training dataset, it is used to make predictions on the testing dataset to make link predictions on future timestamps without knowing the network structure of future timestamps.

Fig. 4 shows the process of using a well-labeled and well-balanced dataset for link prediction using the BiLSTM model. The node embeddings generated act as the features of the nodes while the existence or non-existence of edges act as labels. We perform proper hyperparameter tuning for the BiLSTM architecture to achieve optimal results. The training of the BiLSTM model is done on the initial k timestamps as mentioned in Section III-A. The trained BiLSTM model is utilized to perform the final temporal link prediction on the remaining k to T timestamps. Here, the connected circles refer to the state of the network with circles representing the nodes and the lines representing the edges. The array of squares shows the node embeddings for each node in the network. The box with the BiLSTM title shows the BiLSTM architecture. We then evaluate the various link prediction-based performance metrics for our algorithm.

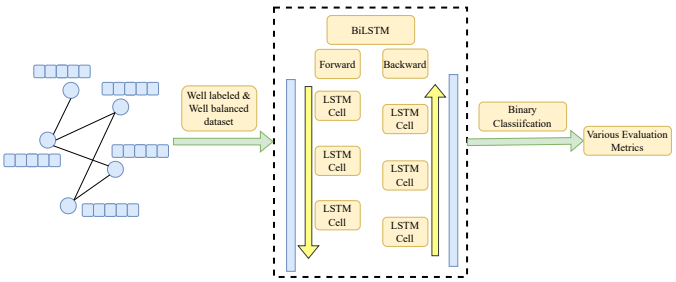

Fig. 4. Using a well-labeled and well-balanced dataset with embeddings as features and existence and non-existence of edges as labels dataset, for link prediction using the BiLSTM model.

IV. Datasets and Evaluation Criteria

This section presents the different datasets used for the evaluation of the proposed model and the different performance metrics that are calculated. For this study, we use six real-life and two synthetic temporal datasets belonging to social networks, biological networks, and information networks.

A. Datasets

For our study, we have used six different real-life and two synthetic temporal networks having varied sizes, topologies, and dimensions. Brief information regarding the real-life datasets used is as follows:

1) MathOverflow [22]: It is a temporal network representing the interaction amongst the users of Math Overflow. It contains 24,818 nodes, and 506,550 collected over 2350 days.
2) Email [22]: This is a dataset of email messages gathered for the European research organization. It contains 986 vertices and 332,334 edges which are collected over a period of 803 days.
3) CollegeMsg [23]: It is a temporal network formed via the private messaging service of the University of California, Irvine. It has 1899 vertices followed by 59835 edges collected over 193 days.
4) Bitcoin-OTC [24]: This is a who-trusts-whom temporal network formed amongst the users of the Bitcoin OTC platform (http://www.bitcoin-otc.com). This dataset contains 5,881 nodes and 35,592 edges collected from 2010 to 2016.
5) Bitcoin-Alpha [24]: This is also a who-trusts-whom temporal network formed between the traders on the platform Bitcoin Alpha (http://www.btcalpha.com/). It has 3,783 nodes and 24,186 edges collected over a period from 2010 to 2016.
6) Reddit [25]: It is a temporal network representing the interaction amongst the users on Reddit. It has 55863 nodes and 858490 edges collected from January 2014 to April 2017.

We also use two synthetic networks in our study to obtain a more profound study. For generating the synthetic networks,

we use the Stochastic Block Model (SBM) proposed by Goyal et al. [26]. It is a recent and popular technique for randomized graph modeling to simulate community structures and evolutions. The snapshots for the real-life networks are represented as the time period across which they're captured. The details about the generated synthetic datasets are given below:

1) SBM1000: It contains 1000 nodes and around 56000 edges. The number of snapshots taken as 8.
2) SBM5000: It contains 5000 nodes and around 157000 edges. The total number of snapshots is kept as 8.

The various statistical details of all the real-life and synthetic temporal datasets have been tabulated in Tab. I. Here, $\mid V \mid$ and $\mid E \mid$ refer to the number of nodes and number of edges, respectively. The time period represents the time period across which the dataset was accumulated.

B. Evaluation metrics

For evaluating the performance of our model, we use several popular evaluation metrics for temporal link prediction, namely, AUC (area under the ROC curve), Precision, Recall, and F1 Score.

m

V. Experimental results and analysis

This section discusses the experimental analysis of the proposed model for link prediction in temporal networks. In this comprehensive experimental analysis, we conducted an extensive evaluation of the proposed TLP-NEGCN model on a diverse set of temporal networks. These networks, consisting of six real-life datasets and two synthetic datasets, vary in size, complexity, and application domains, as detailed in Section IV-A. Our evaluation encompasses a thorough examination of performance metrics, as outlined in Section IV-B, to provide a holistic assessment of the model's capabilities. Furthermore, we performed an in-depth exploration of parameter tuning to identify the optimal hyperparameters for the TLP-NEGCN model. This step ensures that our results accurately reflect the model's full potential. The performance metrics for all experiments are rigorously computed to provide a robust basis for comparison.

To gauge the effectiveness of the TLP-NEGCN model, we compared its performance against a range of baseline temporal link prediction methods, including GCN [27], GraphSage [28], TemporalWalk [29], as well as recent techniques like Ensemble-model-based link prediction (EMLP) [16], deep convolutional neural network (DCNN) [17], Local-Global-Quasi (LGQ) [18], GraphSAGE with CTDNE [19], and TLPSS [30]. Notably, we optimized the parameters for both baseline and contemporary techniques to ensure a fair comparison and to provide insights into the TLP-NEGCN model's superiority.

In addition to these comparisons, as part of our ablation study, we assessed the performance of the TLP-NEGCN model against four state-of-the-art node embedding techniques: Laplacian eigenmaps [31]-based node embedding, HOPE [32], node2vec [33], and SDNE [34]. This analysis offers valuable insights into the utility and advantages of our proposed model in the context of temporal link prediction. The evaluation metrics outlined in Section IV-B were consistently applied across all selected networks to maintain consistency and ensure a comprehensive evaluation. To gain a more holistic understanding of the TLP-NEGCN model's performance, we conducted additional analyses. Firstly, we computed confidence intervals for Area Under the Curve (AUC) values, providing statistical validation for the robustness of our results. Secondly, we performed runtime complexity analysis to assess the model's efficiency and practicality in real-world applications.

In summary, our experimental results and analysis provide a thorough and rigorous assessment of the TLP-NEGCN model's performance. We present insights into its effectiveness compared to baseline methods, contemporary techniques, and state-of-the-art node embedding approaches. This comprehensive evaluation, including parameter tuning, statistical validation, and runtime analysis, offers a holistic view of the model's capabilities in the domain of temporal link prediction. The simulation of the proposed model and other comparing methods are performed in a Python programming environment including several Python libraries like NumPy, TensorFlow, sklearn, etc.

A. Parametric Setting

This section discusses the initial parametric setting for the proposed model. We performed careful hyperparameter tuning to select the best parameters for our model. We used the Random Search heuristic to determine suitable parameters to attain optimal results. It is a hyperparameter tuning heuristic, in which several random combinations of various hyperparameters are tried from a list of pre-determined hyperparameters to achieve the best results. The obtained hyperparameters are tabulated in Tab. II. From Tab. II, it can be seen that we use one BiLSTM layer and two dense layers. The dimensionality of the feature space d to be chosen as 256. The recurrent dropout and dropout values are chosen to be 0.2 and 0.25, respectively. We use ReLU and Softmax as the activation function. Stochastic gradient descent is employed as an optimizer with a learning rate (lr) equal to 1e-4 and the momentum being 0.9. The number of epochs is chosen to be 50. Since temporal link prediction is referred to as a binary classification, hence, we used "Binary-crossentropy" as the loss function. The "ReduceLRonPlateau" is used as the callback function to reduce the value of the learning rate as the results tend to plateau.

Further, we performed the dimensionality analysis, and Splitting Ratio vs. AUC as a part of hyperparameter tuning.

1) Dimensionality Analysis: We analyzed the simulations performed to determine the appropriate dimensionality of the feature space. Fig. 5 shows the accuracy percentages obtained for various datasets as the dimensionality is varied. From the figure, we see that the accuracy for all the datasets increases as the dimensionality is increased. This is due to the additional knowledge associated with the higher dimensionality. It is

TABLE I
THE VARIOUS STATISTICAL DETAILS OF THE SEVERAL TEMPORAL NETWORKS USED BY US.

Datasets	| V |	| E |	Time Period	Description
MathOverflow	24818	506550	2350 days	Math stack exchange interaction network
Email	986	332334	803 days	Email interaction network
CollegeMsg	1899	59835	193 days	College messaging network for UC, Irvine
Bitcoin-OTC	5881	35592	6 years	Bitcoin trading platform trust network
Bitcoin-Alpha	3783	24186	6 years	Bitcoin trading platform trust network
Reddit	55863	858490	3 years	Reddit users interaction network
SBM1000	1000	56000	8 Snapshots	Stochastic Block Model
SBM5000	5000	157000	8 Snapshots	Stochastic Block Model

TABLE II
CHOSEN HYPERPARAMETERS FOR THE PROPOSED TLP-NEGCN APPROACH.

Hyperparameter	Description or Value
Number of BiLSTM layers	1
Dimension of the network embedding	256
Number of Dense layers	2
Recurrent dropout rate	0.20
Dropout rate	0.25
Activation function	ReLU and Softmax
Optimizer	SGD(lr=1e-4, momentum=0.9)
Number of epochs	50
Loss function	Binary-crossentropy
Callbacks	ReduceLROnPlateau

evident that the results obtained an almost saturation point for dimensionality equal to 256, beyond which the improvement in accuracy is very minute. But as the dimensionality increases the computation cost increases drastically. Hence, for our study, we considered the dimensionality of the feature space to be 256, to achieve an optimal balance between accuracy and computational cost.

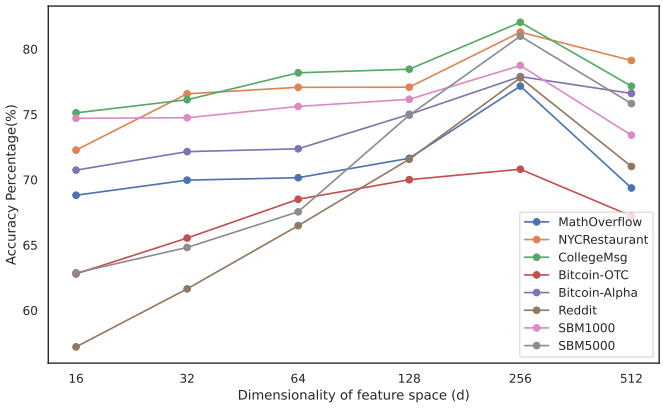

Fig. 5. Dimensionality of feature space (d) vs. Accuracy for various datasets.

2) Splitting Ratio vs. AUC: Here, we discuss the AUC results obtained by the proposed TLP-NEGCN model for temporal link prediction as the splitting ratio is varied for each dataset. The splitting ratio for dividing the dataset into training and testing datasets is varied from 50 to 95% in steps of 5%. Fig. 6 depicts the obtained results. It is noted that as the splitting ratio increases, the value of AUC also increases. This is due to the availability of sufficient data available for training the model. However, the poor performance of the

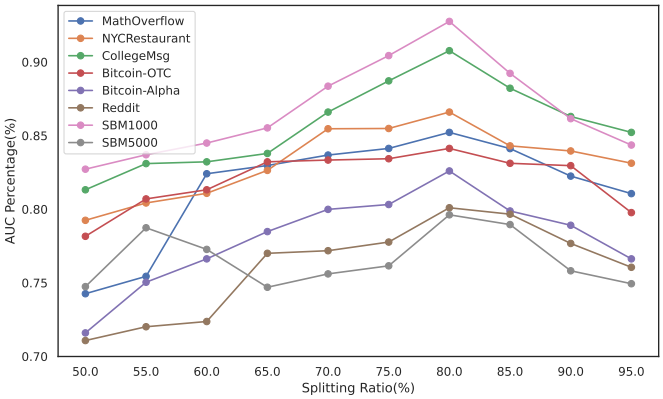

Fig. 6. AUC values obtained by our algorithm as the value of the splitting ratio for the dataset is varied on all the datasets.

introduced model at a lower splitting ratio is due to under-fitting. Moreover, for all the datasets it achieves maxima at a splitting ratio value of around 80%. As the splitting ratio increases further, the performance of our framework deteriorates. This is due to the over-fitting of our model to the dataset. The above analysis shows that the splitting ratio value of 80% is an optimal choice to achieve better results.

B. Comparison with existing link prediction techniques

we examined the performance of the proposed model against some baseline and contemporary temporal link prediction methods. Tab. III shows the results obtained for various comparing methods as compared to the proposed model, TLP-NEGCN. The results were evaluated on all the temporal networks and for all the evaluation metrics. From Tab. III we see that our proposed TLP-NEGCN approach performs the best across all the datasets in terms of AUC. In terms of Precision also, our TLP-NEGCN approach performs optimally well for all the datasets except for Bitcoin-Alpha by a very minute margin. In terms of recall, our TLP-NEGCN algorithm lags behind some algorithms only for Bitcoin-OTC, Reddit, and Ubuntu datasets by a very small percentage. For all the other datasets, the proposed TLP-NEGCN model performs better as compared to other models. This can be inferred as a result of the inverse proportionality of Precision and Recall. Further, the proposed model performed well in terms of F1 Score owing to the better results obtained for Precision and Recall. The improved results of the proposed model can be attributed to the appropriate sequential capturing of the opinion

TABLE III
THE RESULTS OBTAINED FOR VARIOUS CLASSICAL AND CONTEMPORARY LINK PREDICTION ALGORITHMS AS COMPARED TO OUR PROPOSED TLP-NEGCN METHODOLOGY ACROSS ALL THE DATASETS AND ALL THE EVALUATION METRICS. THE BEST RESULT IN EACH CASE IS MARKED IN BOLD.

Methods	MathOverflow				Email				CollegeMsg				Bitcoin-OTC			
	AUC	Prec.	Rec.	F1	AUC	Prec.	Rec.	F1	AUC	Prec.	Rec.	F1	AUC	Prec.	Rec.	F1
TemporalWalk	0.7772	0.7697	0.5565	0.6459	0.7776	0.7959	0.7391	0.7665	0.7201	0.6719	0.8101	0.7345	0.6879	0.6767	0.3233	0.4804
GraphSage	0.8232	0.8229	0.2475	0.3917	0.8157	0.7957	0.8692	0.8308	0.8331	0.8278	0.8278	0.8278	0.6891	0.6851	0.4029	0.5633
GCN	0.8274	0.8153	0.4606	0.5962	0.8370	**0.8799**	0.7757	0.8245	0.8480	0.8300	0.8627	0.8460	0.7772	0.7697	0.5565	0.6459
EMLP	0.8051	0.8331	0.1604	0.2751	0.8121	0.7518	0.8879	0.8142	0.8204	0.8095	0.8360	0.8225	0.7596	0.6979	0.3683	0.5006
DCNN	0.8018	0.8287	0.7705	0.7985	0.7961	0.7799	0.8160	0.7975	0.8098	0.8368	0.7692	0.8016	0.7662	0.7190	0.8493	0.7787
LGQ	0.7999	0.8054	0.3888	0.5558	0.7875	0.7857	0.8684	0.8250	0.8225	0.8051	0.8796	0.8407	0.7407	0.7160	0.4051	0.5569
CTDNE	0.7272	0.6643	0.8788	0.75668	0.8491	0.8296	0.8865	0.85714	0.8680	0.8786	0.8658	**0.8722**	0.7866	0.7469	**0.9002**	**0.8164**
TLPSS	0.8113	0.7480	0.8980	0.8162	0.8589	0.8466	0.8750	0.8605	0.7042	0.6164	0.9492	0.7474	0.7581	0.7722	0.7223	0.7464
TLP-NEGCN	**0.8522**	**0.8677**	**0.9077**	**0.8708**	**0.866**	0.8484	**0.8944**	**0.8708**	**0.9077**	0.8669	**0.9572**	**0.9098**	**0.8413**	**0.8308**	0.4440	0.6056
	Bitcoin-Alpha				Reddit				SBM1000				SBM5000			
	AUC	Prec.	Rec.	F1	AUC	Prec.	Rec.	F1	AUC	Prec.	Rec.	F1	AUC	Prec.	Rec.	F1
TemporalWalk	0.7107	0.6765	0.7819	0.7254	0.6891	0.6851	0.4029	0.5633	0.7008	0.6663	0.8731	0.7558	0.6046	0.6212	0.5632	0.5908
GraphSage	0.7718	**0.8175**	0.7254	0.7687	0.6879	0.6767	0.3233	0.4804	0.7418	0.6735	0.8647	0.7572	0.5730	0.5562	0.7787	0.6489
GCN	0.7700	0.7952	0.7213	0.7564	0.6057	0.5511	0.8860	0.6796	0.7977	0.7546	0.8741	0.8100	0.6391	0.6081	0.7422	0.6685
EMLP	0.7237	0.7043	0.7546	0.7286	0.7012	0.6929	0.7409	0.7161	0.8296	0.7819	0.9162	0.8437	0.6627	0.6114	0.6858	0.6464
DCNN	0.7201	0.6719	0.8101	0.7345	0.7056	0.7095	0.6739	0.6913	0.8454	0.8020	0.9157	0.8551	0.7748	0.7422	**0.8622**	0.7977
LGQ	0.7776	0.7959	0.7391	0.7665	0.7675	0.7630	0.7678	0.7654	0.8536	0.8331	0.8927	0.8619	0.7254	0.7075	0.8041	0.7527
CTDNE	0.8149	0.7743	0.8687	0.8188	0.7584	0.7573	0.7731	0.7651	0.8726	0.8694	0.8968	0.8829	0.7898	0.7636	0.8229	0.7922
TLPSS	0.6971	0.6535	**0.9210**	0.7645	0.7042	0.6164	**0.9492**	0.7474	0.8113	0.7480	0.8980	0.8162	0.7277	0.6835	0.8549	0.75973
TLP-NEGCN	**0.8260**	0.8083	0.8437	**0.8256**	**0.8010**	**0.8035**	0.8083	**0.8059**	**0.9276**	**0.8970**	**0.9403**	**0.9181**	**0.7962**	**0.7867**	0.8222	**0.8041**

and relationship changes amongst the nodes in the network across the various timestamps. Moreover, the use of GCN processes the features of the nodes to better incorporate the neighborhood information for every node of the network to produce the feature vectors that represent the network topology in a better way.

C. Ablation study: Comparison with various node embedding methods:

The Ablation study aims to evaluate the effectiveness of various node embedding techniques in the TLP-NEGCN model for temporal link prediction. To perform a suitable comparative study, we replace the GEMSEC node embedding in the initial feature generation phase with well-known node embedding techniques like Laplacian eigenmaps (Lap) [31] based node embedding, HOPE [32], Node2Vec [33], SDNE [34]. Tab. IV presents the results obtained for the various evaluation metrics obtained on all the datasets for all the chosen node embeddings augmented in our framework. The results indicate that the proposed methodology surpasses other node embeddings in AUC, Precision, and F1 Score. However, Our model falls slightly behind in Recall for only the Bitcoin-OTC and Ubuntu datasets by a narrow margin. The results obtained exhibit the efficacy of the proposed model of temporal link prediction. The better performance of the proposed work also exhibits the utility of the model in choosing the GEMSEC node embedding for initial feature generation over other node embedding methods.

The GEMSEC embedding used by us helps in yielding compact and informative embeddings that retain crucial network structure while reducing computational complexity. Moreover, the iterative nature of our algorithm also helps in learning the evolving temporal patterns for temporal link prediction. The Laplacian eigenmaps on the other hand are sensitive to the connectivity and density of the graph, which can lead to suboptimal representations, especially in networks with varying density or disconnected components. HOPE's reliance on higher-order proximity information may pose challenges in obtaining such information in real-world temporal networks, potentially limiting its effectiveness. Node2Vec, while proficient in capturing neighborhood structures, is highly parameter-dependent and may produce biased representations if suboptimal parameters are chosen for random walk simulations. SDNE, with its deep learning approach, demands careful hyperparameter tuning and considerable computational resources, potentially leading to longer training times, which might be less practical for real-time temporal link prediction applications. The Ablation study's comparative analysis enables us to identify the most suitable node embedding method for the TLP-NEGCN model in temporal link prediction tasks.

D. Confidence Interval on AUC

Here, we discuss the statistical significance of the results obtained by the introduced model by evaluating the confidence interval over AUC values for all the datasets. Tab. V presents the confidence intervals obtained by us. From Tab. V, it is evident that the confidence interval for our proposed model lies very near across the AUC results obtained by the model. Even the lower limit of the interval is significant in terms of performance. The obtained results further justify the exemplary performance of the proposed model of temporal link prediction.

E. Run time Complexity Analysis

Now, we compare the absolute runtime taken by various baseline and contemporary methods along with the proposed model, TLP-NEGCN, to perform temporal link predictions on different datasets. Here, the absolute runtime refers to the

TABLE IV
THE RESULTS OBTAINED FOR VARIOUS WELL-KNOWN NODE EMBEDDING METHODS AS COMPARED TO THE PROPOSED TLP-NEGCN METHODOLOGY ACROSS ALL THE DATASETS. THE BEST RESULT IN EACH CASE IS MARKED IN BOLD.

Methods	MathOverflow				Email				CollegeMsg				Bitcoin-OTC			
	AUC	Prec.	Rec.	F1	AUC	Prec.	Rec.	F1	AUC	Prec.	Rec.	F1	AUC	Prec.	Rec.	F1
SDNE	0.7258	0.7475	0.6877	0.7164	0.7479	0.7237	0.8203	0.7690	0.7413	0.8196	0.6176	0.7044	0.7545	0.7225	**0.8615**	0.7859
Node2Vec	0.7249	0.7423	0.6919	0.7249	0.7526	0.7453	0.7832	0.7638	0.7611	0.8295	0.6636	0.7373	0.8015	0.8046	0.8046	**0.8046**
HOPE	0.8007	0.8504	0.7368	0.7895	0.7642	0.7545	0.7850	0.7695	0.7971	0.8569	0.7133	0.7785	0.7150	0.7398	0.6946	0.7165
Lap	0.8436	0.8506	0.8276	0.8389	0.7762	0.7612	0.8081	0.7840	0.8370	**0.8799**	0.7757	0.8245	0.7458	**0.8398**	0.7398	0.7398
TLP-NEGCN	**0.8522**	**0.8677**	**0.9077**	**0.8708**	**0.866**	**0.8484**	**0.8944**	**0.8708**	**0.9077**	0.8669	**0.9572**	**0.9098**	**0.8413**	0.8308	0.4440	0.6056
	Bitcoin-Alpha				Reddit				SBM1000				SBM5000			
	AUC	Prec.	Rec.	F1	AUC	Prec.	Rec.	F1	AUC	Prec.	Rec.	F1	AUC	Prec.	Rec.	F1
SDNE	0.7054	0.7007	0.7441	0.7218	0.5628	0.6415	0.2741	0.3841	0.7191	0.7118	0.7347	0.7230	0.5935	0.5690	0.8110	0.6688
Node2Vec	0.6451	0.6814	0.5877	0.6311	0.6089	0.6213	0.5203	0.5663	0.7533	0.7183	0.8079	0.7605	0.5202	0.4755	**0.9145**	0.6257
HOPE	0.7647	0.7739	0.7295	0.7510	0.6536	**0.8474**	0.3816	0.5263	0.7674	0.7602	0.7846	0.7722	0.6708	0.6508	0.8333	0.7308
Lap	0.7216	**0.8226**	0.6991	0.7107	0.6895	0.7422	0.5760	0.6486	0.7791	0.7624	0.8159	0.7882	0.6918	0.6187	0.8389	0.7122
TLP-NEGCN	**0.8260**	0.8083	**0.8437**	**0.8256**	**0.8010**	0.8035	**0.8083**	**0.8059**	**0.9276**	**0.8970**	**0.9403**	**0.9181**	**0.7962**	**0.7867**	0.8222	**0.8041**

TABLE V
STATISTICAL SIGNIFICANCE OF THE PERFORMANCE OF THE PROPOSED MODEL ON THE BASIS OF CONFIDENCE INTERVAL OVER AUC VALUES

Dataset	90%	95%	98%	99%
MathOverflow	[85.2208, 85.2192]	[85.2210, 85.2190]	[85.2212, 85.2188]	[85.2213, 85.2187]
Email	[86.6010, 86.5990]	[86.6012, 86.5988]	[86.6014, 86.5986]	[86.6015, 86.5985]
CollegeMsg	[90.7719, 90.7681]	[90.7723, 90.7677]	[90.7728, 90.7672]	[90.7731, 90.7669]
Bitcoin-OTC	[84.1332, 84.1268]	[84.1338, 84.1262]	[84.1345, 84.1255]	[84.1350, 84.1250]
Bitcoin-Alpha	[82.6040, 82.5960]	[82.6048, 82.5952]	[82.6057, 82.5943]	[82.6063, 82.5937]
Reddit	[80.1007, 80.0993]	[80.1008, 80.0992]	[80.1010, 80.0990]	[80.1011, 80.0989]
SBM1000	[92.7618, 92.7582]	[92.7621, 92.7579]	[92.7626, 92.7574]	[92.7628, 92.7572]
SBM5000	[79.6217, 79.6183]	[79.6220, 79.6180]	[79.6224, 79.6176]	[79.6226, 79.6174]

TABLE VI
ABSOLUTE RUN TIME (IN MS.) COMPARISON OF VARIOUS BASELINE AND CONTEMPORARY MODELS TO PERFORM TEMPORAL LINK PREDICTION ON DIFFERENT DATASETS.

Methods	MathOverflow	Email	CollegeMsg	Bitcoin-OTC	Bitcoin-Alpha	Reddit	SBM1000	SBM5000
TemporalWalk	759	165	345	604	454	1533	258	509
GraphSage	717	145	378	633	487	1504	272	589
GCN	762	142	360	659	464	1526	281	585
EMLP	985	308	558	756	686	1761	437	764
DCNN	1008	375	536	787	645	1713	451	777
LGQ	1058	350	598	769	665	1754	419	727
TLP-NEGCN	963	246	489	697	563	1618	339	681

amount of time required by the trained model to perform the temporal link prediction amongst the nodes of the network. The absolute run time is calculated in milliseconds (ms). The obtained results are listed in Tab. VI that illustrate the baseline methods take the least time to make link predictions over all the datasets. However, compared to the contemporary methods, the proposed TLP-NEGCN model takes the least amount of time. The difference in the absolute runtimes of the contemporary and the baseline algorithms is due to the deep learning architecture used by all the chosen contemporary algorithms. The baseline algorithms, on the other hand, use very trivial architectures which improves their computation time. The consistent performance of the proposed model shows its stability across the varying dimensionalities and intricacies of the datasets. The grouping of the dataset based on timestamps to establish appropriate granularity while removing the irrelevant information helps our model to achieve good computational time. Moreover, performing choosing the suitable parameters also helps in speeding up the task of link prediction and generating efficient results.

The above discussion reveals the usefulness of the proposed work, named TLP-NEGCN, for temporal link prediction as compared to various baseline and contemporary temporal link prediction methods. The experimental analysis also demonstrates the use of GEMSEC for generating the initial features for all nodes of the network. The better performance of our model can be inferred due to the appropriate representation of the topological properties of the relationships amongst the nodes via the GEMSEC embedding which are further processed by the GCN architecture. This helps our model to aggregate the neighborhood contribution of nodes in generating the feature set for every node of the network. Moreover, the use of a BiLSTM architecture to make the final link helps in capturing the short-term periodicity and changes in the association amongst the nodes optimally.

VI. CONCLUSION

Temporal link prediction is among the most-studied topic in network science. In this paper, an improved temporal link prediction model has been introduced using GEMSEC graph

embedding and graph convolutional networks (GCN). We adopted the GEMSEC node embedding to generate the initial feature vectors for each node of the network. The necessary changes are made in the GCN architecture to iteratively process these feature vectors for the dataset's timestamps, which enables better capture of the opinion changes over time and neighborhood information of the users in the network. Then, using the node features vectors and the existence or absence of the edges between nodes as labels, we produced a well-labeled and well-balanced dataset. Following that, the dataset is divided into training and testing parts. The BiLSTM model is trained using the training dataset, and the trained model is then used to predict links based on future timestamps in the testing datasets. The hyperparameter studies are carried out to acquire appropriate model parameters. We conducted intensive experiments on six real-life and two synthetic temporal datasets and computed various evaluation metrics. Our model's performance was compared to various baseline and contemporary temporal link prediction methods. Further, as a part of the ablation study, the result of the proposed model is also compared with some prominent node embedding techniques. The results obtained illustrate the proposed model's effectiveness in predicting links on temporal networks. Further, we can apply a similar idea for link prediction on weighted temporal links in the near future.

REFERENCES

[1] C. Muro, B. Li, and K. He, "Link prediction and unlink prediction on dynamic networks," *IEEE Transactions on Computational Social Systems*, 2022.

[2] L. Wang, J. Ren, B. Xu, J. Li, W. Luo, and F. Xia, "Model: Motif-based deep feature learning for link prediction," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 2, pp. 503–516, 2020.

[3] C. Muro, B. Li, and K. He, "Link prediction and unlink prediction on dynamic networks," *IEEE Transactions on Computational Social Systems*, 2022.

[4] A. Ozcan and S. G. Oguducu, "Link prediction in evolving heterogeneous networks using the narx neural networks," *Knowledge and Information Systems*, vol. 55, no. 2, pp. 333–360, 2018.

[5] M. Qin, C. Zhang, B. Bai, G. Zhang, and D.-Y. Yeung, "High-quality temporal link prediction for weighted dynamic graphs via inductive embedding aggregation," *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[6] H. Gao, J. Huang, Y. Tao, W. Hussain, and Y. Huang, "The joint method of triple attention and novel loss function for entity relation extraction in small data-driven computational social systems," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 6, pp. 1725–1735, 2022.

[7] Z. Cao, L. Xu, D. Z. Chen, H. Gao, and J. Wu, "A robust shape-aware rib fracture detection and segmentation framework with contrastive learning," *IEEE Transactions on Multimedia*, vol. 25, pp. 1584–1591, 2023.

[8] H. Gao, J. Xiao, Y. Yin, T. Liu, and J. Shi, "A mutually supervised graph attention network for few-shot segmentation: the perspective of fully utilizing limited samples," *IEEE Transactions on neural networks and learning systems*, 2022.

[9] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *IEEE transactions on Big Data*, vol. 6, no. 1, pp. 3–28, 2018.

[10] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[11] M. Xing, W. Ding, T. Zhang, and H. Li, "Stcgcn: a spatio-temporal complete graph convolutional network for remaining useful life prediction of power transformer," *International Journal of Web Information Systems*, vol. 19, no. 2, pp. 102–117, 2023.

[12] M. Chen, X. Luo, H. Shen, Z. Huang, Q. Peng, and Y. Yuan, "A chinese nested named entity recognition approach using sequence labeling," *International Journal of Web Information Systems*, vol. 19, no. 1, pp. 42–60, 2023.

[13] A. Divakaran and A. Mohan, "Temporal link prediction: A survey," *New Generation Computing*, vol. 38, pp. 213–258, 2020.

[14] X. Ma, P. Sun, and G. Qin, "Nonnegative matrix factorization algorithms for link prediction in temporal networks using graph communicability," *Pattern Recognition*, vol. 71, pp. 361–374, 2017.

[15] N. M. Ahmed, L. Chen, Y. Wang, B. Li, Y. Li, and W. Liu, "Deepeye: link prediction in dynamic networks based on non-negative matrix factorization," *Big Data Mining and Analytics*, vol. 1, no. 1, pp. 19–33, 2018.

[16] K. Li, L. Tu, and L. Chai, "Ensemble-model-based link prediction of complex networks," *Computer Networks*, vol. 166, p. 106978, 2020.

[17] W. Wang, L. Wu, Y. Huang, H. Wang, and R. Zhu, "Link prediction based on deep convolutional neural network," *Information*, vol. 10, no. 5, p. 172, 2019.

[18] M. Kumar, S. Mishra, and B. Biswas, "Features fusion based link prediction in dynamic neworks," *Journal of Computational Science*, vol. 57, p. 101493, 2022.

[19] N. Bowman, R. Jones, and S. Shafi, "Temporal link prediction on the wikilinkgraphs dataset."

[20] Z. Qiu, J. Wu, W. Hu, B. Du, G. Yuan, and P. Yu, "Temporal link prediction with motifs for social networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35(3), 2023.

[21] F. Spezzano, W. Chen, and X. Xiao, *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ACM, 2019.

[22] A. Paranjape, A. Benson, and J. Leskovec, "Motifs in temporal networks in: Proceedings of the international conference on web search and data mining, 601–610," 2017.

[23] P. Panzarasa, T. Opsahl, and K. M. Carley, "Patterns and dynamics of users' behavior and interaction: Network analysis of an online community," *Journal of the American Society for Information Science and Technology*, vol. 60, no. 5, pp. 911–932, 2009.

[24] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. Subrahmanian, "Rev2: Fraudulent user prediction in rating platforms," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 2018, pp. 333–341.

[25] S. Kumar, W. L. Hamilton, J. Leskovec, and D. Jurafsky, "Community interaction and conflict on the web," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 933–943.

[26] P. Goyal, N. Kamra, X. He, and Y. Liu, "Dyngem: Deep embedding method for dynamic graphs," *arXiv preprint arXiv:1805.11273*, 2018.

[27] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[28] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[29] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Companion proceedings of the the web conference 2018*, 2018, pp. 969–976.

[30] R. Zhang, Q. Wang, Q. Yang, and W. Wei, "Temporal link prediction via adjusted sigmoid function and 2-simplex structure," *Scientific Reports*, vol. 12, no. 1, p. 16585, 2022.

[31] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," *Advances in neural information processing systems*, vol. 14, 2001.

[32] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 1105–1114.

[33] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[34] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 1225–1234.