Predictive Temporal Path Queries

A Thesis

submitted to the designated
by the Assembly
of the Department of Computer Science and Engineering
Examination Committee

by

Christos Gkartzios

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN DATA AND COMPUTER SYSTEMS ENGINEERING

WITH SPECIALIZATION
IN DATA SCIENCE AND ENGINEERING

University of Ioannina
School of Engineering
Ioannina 2022

Examining Committee:

- **Evaggelia Pitoura**, Professor, Department of Computer Science Engineering, University of Ioannina (Supervisor)
- **Panayiotis Tsaparas**, Associate Professor, Department of Computer Science Engineering, University of Ioannina
- **Nikos Mamoulis**, Professor, Department of Computer Science Engineering, University of Ioannina

Acknowledgements

With the completion of this thesis I would like to extend my sincere thanks to my supervisor, professor Evagelia Pitoura, whose expertise, collaboration and friendship have constituted an invaluable help.

In addition, I would like to thank my parents for their constant support all these years. Lastly, I would like to thank my dear friends who have always been there to share my concerns and some times become a much needed distraction from them.

12/10/2022 Christos Gkartzios

Table of Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Christos Gkartzios, M.Sc. in Data and Computer Systems Engineering, Department of Computer Science and Engineering, School of Engineering, University of Ioannina, Greece, 2022.
Predictive Temporal Path Queries.
Advisor: Evaggelia Pitoura, Professor.

Previous research has focused on graph queries applied either on the current state or the past states of a time-varying graph. In this thesis, we introduce the problem of querying the future state of a graph. Specifically, given a time-varying graph, our goal is to predict the length of the shortest temporal path between two nodes u and v at a future time point. The shortest temporal path is a path that minimizes both the weight of the edges that constitute the path and in addition it preserves an always increasing temporal sequence between the edges of the path.

We propose an approach that combines an efficient algorithm for computing the shortest temporal paths at the present with predictions based on node embeddings. For the computation of the shortest temporal path, we use a one pass algorithm that takes advantage of a sorted stream representation of the edges of the graph. For the prediction step, we exploit three different methods, namely link prediction, next interaction prediction and distance prediction.

For the link prediction task we combine the embeddings of nodes with a classifier that outputs the probability of the existence of an edge. The classifier is tested both on temporal and non-temporal embeddings.

The next interaction prediction task uses temporal embeddings and a recursive neural network to predict the next interaction of a source node u. Given a future time instance and a source node, a Recurrent Neural Network predicts the node v that the source node will interact with at the specified time.

The next distance prediction method uses a a multi-layer perceptron for the prediction of the distance between two nodes. The non-temporal embedding representations of the source node u and and the destination node v, of a shortest temporal path, are used as input of the perceptron for the prediction of the temporal distance. We evaluated our approach on several real datasets and embedding methods. We found that temporal embeddings work best in terms of training time efficiency and prediction accuracy. Furthermore we found that the temporal reachability task works best on classifying the existence of a temporal path.

Εκτεταμενη Περιληψη

Χρίστος Γκάρτζιος, Δ.Μ.Σ. στη Μηχανική Δεδομένων και Υπολογιστικών Συστημάτων, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πολυτεχνική Σχολή, Πανεπιστήμιο Ιωαννίνων, 2022.
Ερωτήματα πρόβλεψης Χρονικών Μονοπατιών.
Επιβλέπων: Πιτουρά Ευαγγελία, Καθηγήτρια.

Το πρόβλημα της συντομότερης διαδρομής, μπορεί να οριστεί ως το πρόβλημα της εύρεσης μιας διαδρομής μεταξύ δύο κόμβων η οποία ελαχιστοποιεί τα βάρη των άκρων που αποτελούν την εν λόγω διαδρομή. Ένα μονοπάτι μπορεί να οριστεί ως μια ακολουθία ακμών η οποία συνδέει μια ακολουθία κορυφών. Επομένως το πρόβλημα της εύρεσης της συντομότερης διαδρομής μεταξύ δύο τοποθεσιών σε έναν χάρτη μπορεί να μοντελοποιηθεί ως ένα πρόβλημα συντομότερης διαδρομής.

Στην παρούσα εργασία εξετάζουμε μια ειδική περίπτωση του προβλήματος της συντομότερης διαδρομής σε χρονικούς γράφους, το πρόβλημα της συντομότερης χρονικής διαδρομής. Ως συντομότερη χρονική διαδρομή μπορεί να οριστεί μια διαδρομή η οποία καταφέρνει ταυτόχρονα να ελαχιστοποιεί το βάρος των άκρων που αποτελούν τη διαδρομή, αλλά και να διατηρεί μια διαρκώς αυξανόμενη χρονική ακολουθία μεταξύ των άκρων της. Η προηγούμενη έρευνα έχει επικεντρωθεί σε ερωτήματα σχετικά είτε με την τρέχουσα είτε με παρελθοντικές χρονικές καταστάσεις ενός εξελισσόμενου γραφήματος. Στην παρούσα εργασία εισάγουμε το ζήτημα των ερωτημάτων πανω στην μελλοντική κατάσταση ενός γραφήματος. Δοθέντω ενός γράφηματος το οποίο μεταβάλλεται στον χρόνο, ο στόχος μας είναι να βρούμε το μήκος της συντομότερης χρονικής διαδρομής μεταξύ δύο κόμβων u και v του γραφήματος σε μελλοντική χρονική στιγμή.

Προτείνουμε μια μέθοδο που συνδυάζει τον υπολογισμό των συντομότερων χρονικών διαδρομών για το παρόν, τις χωρικές πληροφορίες των embeddings των κόμβων και έναν classifier ο οποίος προβλέπει την πιθανότητα εμφάνισης σύνδεσης με-

ταξύ δύο κόμβων. Δοκιμάζουμε τη μέθοδό μας με δύο τύπους embeddings, οι οποίοι διαφοροποιούνται με βάση τη μέθοδο που χρησιμοποιήθηκε για τη δημιουργία τους.

Ο πρώτος τύπος εκμεταλλεύεται μόνο τις μορφολογικές πληροφορίες του γράφου, ενώ ο δεύτερος εκμεταλλεύεται τις αλλαγές που επίκεινται στην κατάσταση του δικτύου με την πάροδο του χρόνου. Για τον υπολογισμό της συντομότερης χρονικής διαδρομής, αξιοποιούμε έναν αλγόριθμο one pass ο οποίος εκμεταλλεύεται την αναπαράσταση των ακμών του γραφήματος ως ταξινομημένη ροή, ως προς τον χρόνο.

Εξετάζουμε επίσης δύο παραλλαγές της προαναφερθείσας στρατηγικής. Στην πρώτη μέθοδο, δεδομένης της μελλοντικής χρονικής στιγμής που θέλουμε να προβλέψουμε και ενός κόμβου πηγής, αξιοποιούμε ένα Recurrent Neural Network για την πρόβλεψη της επόμενης αλληλεπίδρασης κόμβου. Στη δεύτερη στρατηγική χρησιμοποιείται ένα multi-layer perceptron για την πρόβλεψη της απόστασης μεταξύ δύο κόμβων. Οι προτεινόμενες μέθοδοι είναι αξιολογήθηκαν πειραματικά για την ποιότητα των προβλέψεών τους.

Chapter 1

Introduction

1.1 Motivation

A plethora of networks in our world, such as communication and disease spread networks, change over time. New objects and connections may appear and and old ones may disappear as the time passes. Temporal changes are observed on the nodes and the edges of the corresponding networks.

Temporal graphs, also known as dynamic or time-varying graphs, help us model these networks, understand them and derive useful conclusions about the patterns that have defined the evolution of these network and give us insight about their future development.

A graph is a structure that amounts to a set of objects, pairs of which display a number of relations between them. These objects are most commonly called vertices or nodes and their relations are called edges or links. There are two main differentiating factors that define a graph based on its edges, whether the edges correspond to ordered pairs of vertices and if multiple edges that have the same end nodes are permitted to exist.

One of the main problems to be solved in the analysis of graphs is the computation of the shortest path between two nodes. The shortest path problem, can be defined

as the problem of finding the path between two nodes that minimizes the weights of the edges that constitute said path. A path can be defined as a sequence of edges that connects a sequence of vertices. Little attention has been given on temporal networks, in which the temporal features of the edges determines the dissemination of information.

The concept of a path can be altered in order to take into account the temporal information of a temporal graph. This new type of path is defined as a temporal or time-respecting path. A path is temporal only if the timestamps, of the edges used in the edge sequence, are non-decreasing. This means that the links appearing in a temporal path correspond to interactions that happened closer to the present time, the nearer the links appear to the destination node. Therefore the shortest path problem can be redefined in the context of temporal networks. The shortest temporal path problem can be defined as the task of finding the path that both minimizes the weight of the edges that constitute the path and also preserves an always increasing temporal sequence between the edges of the path, from the furthest in the past to the nearest to the present time.

Previous research has focused on graph queries either on the current or the past states of a time evolving graph. Specifically previous research considers present-time shortest path queries, like what is the current shortest path between two nodes, or historical shortest path queries, trying to find what was the shortest path in some time instance in the past. In this thesis, we introduce the problem of querying the future state of a graph. Given the state of a temporal graph at the present we try answer queries about future changes that will happen on said graph.

We propose an approach that combines an efficient algorithm for the computation of the shortest temporal paths at the present with predictions based on node embeddings. For the calculation of the shortest temporal paths, from a specific source node of a graph, to all others nodes we use a one pass algorithm that takes advantage of a sorted stream representation of the edges of the graph.

For the prediction step, we exploit three different methods, namely link prediction, next interaction prediction and distance prediction. For the link prediction task we combine the embeddings of nodes with a classifier that outputs the probability of the existence of an edge. The classifier is tested both on temporal and non-temporal embeddings.

The next interaction prediction task uses temporal embeddings and a recurent

neural network to predict the next interaction of a source node u. Given a future time instance and a source node, a recurrent neural network predicts the node v that the source node will interact with at the specified time.

Finaly the next distance prediction method uses a a multi-layer perceptron for the prediction of the distance between two nodes. The non-temporal embedding representations of the source node u and and the destination node v, of a shortest temporal path, are used as input of the perceptron for the prediction of the temporal distance. We evaluated our approach on several real datasets and embedding methods. We found that temporal embeddings are the most efficient methods concerning the time needed for the training of the model and accomplish accurate predictions. Furthermore we found that the temporal reachability task works best on classifying the existence of a temporal path.

1.2 Structure of the Thesis

The rest of this thesis is structured as followed. In Chapter 2 we present the concepts and algorithms that we will use in this thesis. In Chapter 3 we define the problems we will tackle. In Chapter 4 we formalize the strategies and define the algorithms that we will use for the prediction of the future shortest paths and the problem of temporal reachability. In Chapter 5 we define the datasets, we will test our strategies on, and the metrics we will evaluate these strategies with. In Chapter 6 we present the conclusions of our evaluations and propose future expansions on the present work.

Chapter 2

Related Work

Graphs have been used to represent many complex sociological and natural phenomena. The temporal information that can be observed on networks with the passing of time presents the need to find new ways to model and analyze these networks. Temporal graphs help us model and complete queries on these networks. In this section we present previous research on the modeling and analysis of temporal graphs, the creation of embeddings and the computation of the temporal path problem.

2.1 Temporal Graphs

Temporal Graphs are a representation method that retain the temporal information of a network. [1] presents the concept of the temporal graph as a type of graph that retains all the analytical benefits of a static graph while also retaining all the temporal data.

In [2] the concept of temporal exploration on temporal graphs is presented. Specifically they focus on the problem of computing a foremost exploration schedule for

a temporal graph. Their work focuses on undirected temporal graphs that are connected at each time step. Finally they find that the approximation of the temporal exploration problem is NP-hard.

An integral problem in the context of temporal networks is the problem of modeling, storing, and querying temporal property graphs. This is addressed in [3]. In this paper a temporal graph data model, where nodes and relationships contain attributes (properties) timestamped with a validity interval along with a high-level graph query language *T-GQL* are presented.

Another approach to tackle the problem of modeling and querying temporal networks is based on modifying the concept of graph neural networks (GNNs). The framework of Temporal Graph Networks (TGNs) is proposed in [4] for deep learning on dynamic graphs that are represented as a sequence of timed events. Furthermore an efficient training method is proposed, so that the model can learn accurately from the sequentiality of the data. Lastly, in [5] the concept of Temporal Graph Neural Networks (TGNNs) is presented. Given a time series of a graph to be explained the TGNN framework identifies dominant explanations in the form of a probabilistic model in a time period.

2.2 Embeddings

Network embeddings is a collective term that describes a plethora of techniques that aim to map the nodes of a graph to vectors in a multidimensional space. In order to be useful these embeddings should, in their most basic form, at least preserve the structure of the graph they derive from. These embeddings are used as inputs in a lot of graph analysis tasks. The methods of node embedding creation are divided into two categories based on the strategy followed for their creation. Graph Neural Networks (*GNN*s) and random-walk approaches. Graph Neural Networks are neural networks that can be used on graphs achieve prediction tasks on the whole graph or specifically on the node or edge levels of the graph. A random-walk on a graph is a special case of a Markov chain, with a special property called *reversibility*. This property dictates that the probability of a given path to be traversed in in one direction or the other have a very simple connection between them. Due to their efficiency and their inductive learning capabilities GNNs are the most popular approach.

Other than the strategy followed for their creation, embeddings can be categorized based on whether they utilize the temporal information of the graph. Temporal embeddings take into account the information of the time instance that an edge was created on, for the creation of the node embeddings. We experimentally evaluate our system, for the task of link prediction, on both temporal embeddings like tNodeEmbed [6] and Multilens [7] and non-temporal like Node2vec[8].

Node2vec [8] is an algorithmic framework that helps in the learning of continuous feature representations for the nodes of a network. This framework maps the nodes to a low-dimensional space of features. By doing this the likelihood of preserving the neighborhoods of nodes in a network, is maximized. In the context of word embeddings, given the linear nature of text, neighborhoods are defined by using a sliding window over consecutive words. Contrary to text, networks are not linear, therefore a different sampling strategy needs to be followed for the creation of node embeddings.

To resolve this issue Node2vec utilizes a randomized sampling procedure that samples from different neighborhoods given a source node. There are two extreme sampling strategies Breath-first (*BFS*), that samples the immediate neighbors of a node and Depth-first (**DFS**), that samples nodes sequentialy at an increasing distance. These sampling strategies correspond with the two main types of similarity that exist, structural equivalence and homophily. When a network is evaluated based on homophily it is expected that highly interconnected nodes would have similar embeddings. Likewise when the network is evaluated based on structural equivalence, nodes with similar structural roles are expected to have similar embeddings. Therefore when neighborhoods are sampled by BFS the embeddings correspond to the strustural equivalence of the network. The opposite is true when neighborhoods are sampled using DFS, therefore embedding construction is influenced highly by homophily.

Node2vec utilizes a flexible sampling strategy, based on biased random walks, that interpolates smoothly between BFS and DFS. To achieve this Node2vec uses two parameters, p and q, that control the speed at which a walk explores and leaves a neighborhood. In more detail, p is the parameter that controls the probability of a walk to immediately revisit a node. This means that it is less likely to sample again an already visited node, if the value of p is high, and more likely to backtrack, if p is low. Parameter q determines whether the random walk exploration will be biased

toward "inwards" or "outwards" nodes. If $q > 1$ the random walk has behavior similar to BFS, meaning that it is biased toward nodes close to the starting node. On the contrary if $q < 1$ the random walk is biased toward nodes that are away from the starting node, which means that the exploration has similar behavior to DFS.

Table 2.1: Binary operations for learning of edge features

Operator	Symbol	Definition
Average	$\oplus$	$[\mathrm{f(u)}\oplus\mathrm{f(v)}]_i = \frac{f_i(u)+f_i(v)}{2}$
Hadamard	*	$[\mathrm{f(u)}\oplus\mathrm{f(v)}]_i = f_i(u) * f_i(v)$
Weighted-L1	$\|\cdot\|_{\bar{1}}$	$\|\ \mathrm{f}_i(u)\cdot\ \mathrm{f}_i(v)\|_{\bar{1}i} = \|f_i(u) - f_i(v)\|$
Weighted-L2	$\|\cdot\|$	$\|\ \mathrm{f}_i(u)\cdot\mathrm{f}_i(v)\|_{\bar{2}i} = \|f_i(u) - f_i(v)\|^2$

We are interested in link prediction, a task that involves pairs of nodes in a network, in order to predict whether a link exists between them. We use an extension of the Node2vec embedding representation, proposed in [8], that defines the method to create the embedding of pair of nodes (u,v). With the help of binary operators on the corresponding feature vectors f(u) and f(v) the embedding for the corresponding pair of nodes is created. The binary operators are shown in Table 1.

In addition to the normal embedding methods such as Node2Vec, we examine our methods with temporal embeddings as well. Specifically we use tNodeEmbed and a modified case of Multilens that takes into account the temporal information of the edges for the creation of the embeddings.

Specifically, tNodeEmbed [6] is a semi-supervised algorithm that exploits a network's temporal behavior, in order to create node embeddings that incorporate the dynamics of the network over time. The two main goals are to preserve the static network neighborhoods of the nodes in a feature space of d-dimensions and to preserve the dynamics of the temporal network. These goals are achieved by leveraging static embedding methods, such as Node2vec, in order to create the d-dimensional representations of the nodes in every time instance of the network. The historical embeddings of every node are utilized for the creation of a final embedding that represents each node.

Depending on the desired prediction task a different loss function is preferred. Specifically for node classification categorical cross-entropy loss is chosen, while for link prediction binary classification loss is preferred. As a result we can formulate the problem of the creation of a node's temporal embedding as an optimization problem that minimizes the corresponding loss function. With the help of static graph snapshots, the embeddings of every node for different time points are computed. Due to the fact that Node2vec does not guarantee the consistency of the embeddings across different trainings an alignment strategy is followed between the static embeddings of consecutive time steps.

Therefore tNodeEmbed, is defined as an architecture that is initialized with the embeddings of static nodes, followed by the alignment of the embeddings of said nodes for consecutive time points and optimized for a given task.

On the other hand, Multilens[7] is an inductive multi-level latent network summarization approach that captures the structure of egonets and higher-order subgraphs. To achieve this Multilens leverages a set of relational operators and relational functions. The structure is stored in low-rank, size-independent structural feature matrices, which along with the relational functions comprise the latent network summary.

Table 2.2: Embedding Methods

	Temporal	Random Walk	Neural Network	Temporal Snapshots
Node2Vec		✓		
tNodeEmbed	✓	✓		✓
Multilens	✓			✓
P-GNN			✓	
Jodie embeddings	✓		✓	
DANE[9]	✓			✓
DynGEM[10]	✓			✓
CTDNE[11]	✓	✓		✓

Previous work on embeddings has focused on representing a timestamped edges stream by using a time-series of graphs which is based on a specific time scale τ. A different approach, that is used with Multilens, is to use the ϵ-graph that uses a fixed number of edges for each graph. By fixing the number of edges in each graph, the models used are allowed to capture the actual changes that happen to the structure

of the graphs over time.

Different strategies have been followed for the creation of embeddings that take advantage of the temporal information of the corresponding Network. The Domain Adaptive Network Embedding framework [9] (DANE) applies graph convolutional network to learn the embeddings, the embeddings are further aligned by adversarial learning regularization. DynGEM[10] takes advantage of deep autoencoders to handle growing dynamic graphs and create embeddings that are stable over time. *Continuous-time dynamic networks* (CTDNEs)[11] constitute an embedding creation framework that incorporate temporal dependencies into existing node embeddings.

The second category, based on the strategy followed for the creation of the embeddings, is Graph Neural Networks. The main weakness that is observed in the existing GNN architectures, is that they fail to capture the position of the node in relation to the general context of the structure of the graph. *Position-aware Graph Neural Networks (P-GNNs)*[12] constitute a possible solution to this problem.

Position-aware Graph Neural Networks (P-GNNs) incorporate the positional information of a node compared to all the other nodes of the network, during the task of the embedding creation. P-GNNs use k random subsets of nodes, called *anchor-sets* for the computation of the embedding. In each forward pass multiple anchor sets are sampled. A non-linear aggregation scheme combines the node features information of each anchor-set and weights it by the distance between the node and and the anchor-set. In order to enhance the expressiveness of the model this aggregation is combined into multiple layers.

Traditional approaches tackle the problem of predicting the problem of the computation of the shortest path by traversing the network. This approach is computation and cost heavy. To address this limitation [13] proposes strategies for the learning of embeddings the preserve the distances from every vertex to all the other vertices of a network. These embeddings are used to train a *multi-layer percetron* (MLP) to predict the distances between two vertices, given the embedding of each vertex.

Three models are proposed for the prediction of the distances between the vertices, Vdist2vec, Vdist2vec-L and Vdist2vec-S. The Vdist2vec model takes as input two one-hot vectors that represent two vertices. The embeddings of these vectors are used as input in an MLP in order to predict the distance between them. The loss function that is used is the mean squared error between the predicted distance and the actual distance between the vertices. It is found that a very small portion of vertex pairs,

with a much larger prediction error than the rest, dominate the training error and force the model to focus on them. Vdist2vec-L optimizes the model by changing the loss function, presented in 4.2.

$$L_n = E_\phi[f_\delta(d(v_i, v_j) - \widehat{d}_{i,j})] \tag{2.1}$$

The f_δ function is based on the *Huber* loss function and δ is the top 1% with the largest prediction error.

Existing dynamic embedding methods generate embeddings only when the nodes of a graph take an action without modelling the future trajectory of the corresponding embeddings. Jodie [14] is an attempt to address this issue.

Jodie is a coupled recurrent neural network model whose goal is to learn the future trajectory of embeddings. In the context of this work the trajectory of an embedding is defined as the sequence of dynamic embeddings that a node has over a time interval.

In the context of the model proposed in [14], it is assumed that every node has both stationary and dynamic properties, as dynamic are defined the properties of a node that change over time. This means that every node has both static and dynamic embeddings. The static embeddings represent the long-term stationary properties of a node, while the dynamic embeddings represent its time varying properties. Both embeddings are used for the predictions.

Jodie classifies the nodes of a graph in two categories as users, who are the source of an interaction, and items, which are the destination of an interaction. The two operations that Jodie consists of, the update and projection operations, utilize this classification.

The update operation consists of two Recurrent Neural Networks (RNNs) for the generation of the embeddings. The two RNNs are coupled to inocorporate the interdependency between the users (sources) and the items (destinations). At the conclusion of an interaction between a user and an item, the user RNN utilizes the embedding of the interacting item in order to update the user embedding.

The second operation constitutes also the major innovation of Jodie. The projection operation predicts the future trajectory of the embedding of the users. The model utilizes a temporal attention layer in order to project the embedding of a user after some time Δ. This predicted user embedding is used for the prediction of the item embedding the user is most likely to interact with.

For the measurement of the performance of the algorithm for the prediction of

a future interaction, the metrics used are the mean reciprocal rank (MRR) and recall@10. MRR is the average of the reciprocal ranks and recall is a fraction of the interactions in which the ground truth is ranked in the top 10. The embeddings of the items items are ranked based on the L2 distance from the predicted item embedding.

2.3 Shortest Temporal Paths

Past research has focused on the problems of shortest temporal paths and reachability on past or present present time instances of temporal Networks. Specifically, in [15], the problem of conducting queries concerning the historical reachability of temporal graphs in a time interval in the past. Furthermore a reachability index appropriate for queries on the past of a temporal graph, termed *TimeReach* index is presented.

Further work on historical graphs exploits the representation of the state of the graph at different time instants, as a sequence of graph snapshots. The focus of [16] is to find *durable graph pattern queries*, the matches that exist for the longest period of time. In [17] the *Best Friends Forever* (BFF) is defined. Given a set of graph snapshots, which correspond to the state of a temporal graph at different time instances, the goal is to identify the set of nodes that are the most densely connected in all snapshots.

A number of algorithms for the computation of minimum temporal paths are proposed in [18]. In this paper algorithms for the computation of the shortest and the fastest paths of a temporal network are defined. The proposed algorithms can be separated in two categories those that take advantage of the stream representation of the edges of a graph, and those that demand the transformation of a graph.

In [19] is presented a method of link prediction in temporal uncertain social networks. Uncertainty is a feature that is present in real world networks. In social networks, the likelihood of social interactions and relations may be affected by trust and influence issues. In this paper the method proposed transforms the link prediction problem in uncertain networks to a random walk in a deterministic network. Furthermore this method is extended to temporal uncertain networks by integrating the time and global topological information of the network.

Chapter 3

Problem Definition

Our work focuses on queries on temporal graphs. Specifically we concentrate our research on the problem of the prediction of the future shortest temporal between two nodes. Furthermore we tackle the problem of future temporal reachability between two nodes.

Informally we can define a temporal graph as a graph that changes over time. When the time instances are discrete, temporal graphs can be represented as a sequence of static graphs G_1, G_2,...,G_t over the same set of nodes V.

Definition 3.1. A *temporal network* is a network whose links carry temporal information of the time they were active.

Time evolving networks can be modeled and studied with the help of temporal graphs. We assume temporal graphs where each edge is annotated with a timestamp indicating the time that the edge appeared.

Definition 3.2. A temporal graph is a graph $G = (V, E)$ where V is a set of nodes and E is a set of edges where each edge is a triple (u, v, t) where u, $v \in V$ and t is the time instance when the edge appeared.

We will focus on graph queries and in particular on shortest path queries. This problem can be distinguished in three variations. Single-source shortest path problem, in which we are tasked to find the shortest path, that starts from a single source v to all the other nodes of the graph.The single-destination shortest path problem, which constitutes a variation of the single-source shortest path problem, where given

a destination node u the shortest path from all vertices to that end node needs to be found. The final variation is the all-pair shortest path problem, where we have to find all the shortest paths between every possible pair of source and destination vertices of the graph. We focus our work on finding the shortest paths between pairs of vertices, where the source and destination nodes are clearly defined.

Previous research considers present-time shortest path queries (what is the current shortest path between two nodes), or historical shortest path queries (what was the shortest path in some time instance in the past). We will study future-time shortest path queries. We will focus on temporal paths.

Definition 3.3. A temporal path is a path (u_1, u_2, ... u_{k+1}), where $(u_i, u_{i+1}, t_i) \in E$ and $t_i < t_{i+1}$, for 1 $\leq$ i $<$ k.

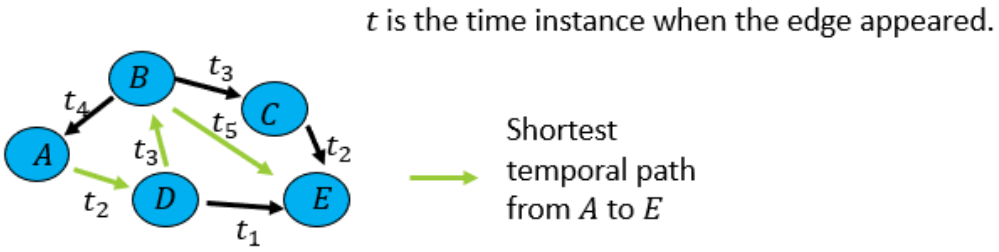

Figure 3.1: Shortest temporal path example

Figure 3.1 presents the shortest temporal path between two nodes at A and E at a specific time instance. The green arrows show the temporal path between the specified nodes.

Temporal paths also known as time respecting paths, are defined as sequences of links that traverse a temporal network. Their common property with the paths in directed graphs is that the source a and destination b nodes of the edges of the path are clearly defined. This means that it is not given that if a path from a to b exists the reverse path from b to a will exist. In addition temporal paths are time-varying which means that their existence is valid only in the course of a specific time interval.

Definition 3.4. Given the graph $G = (V, E)$ at time instance t, where V is the set of all the nodes and E is the set of all the edges that have appeared by time T. Assuming that u and v are two nodes that belong to V, our goal is to predict the shortest path distance between these nodes at a ***future*** time instance $T + 1$.

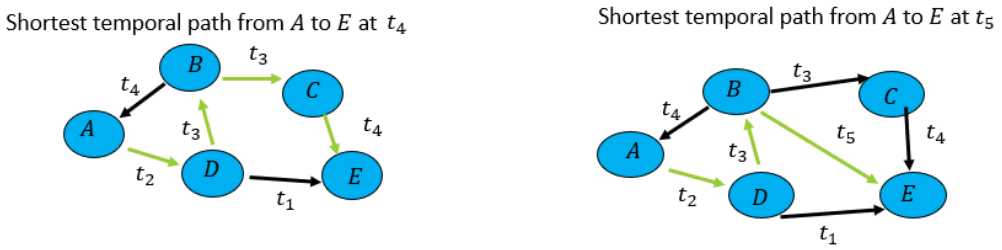

Figure 3.2: Shortest temporal path change in time

Figure 3.2 presents the change of the shortest temporal path between nodes A and E in time. At the time instance t_4 the shortest temporal path between A and E is of length 4, while at time instance t_5 it is of length 3.

Additionally we address the problem of temporal reachability on time varying networks. In graph theory reachability can be informally defined as the ability of a node u to reach a node v through a sequence of adjacent vertices. In undirected graphs, the problem of temporal reachability can be solved by determining the connected components of a the graph. If a pair of vertices belongs to the same connected component then these nodes are reachable. In contrast if a graph is directed then it is imperative that there is a sequence of links from the first node s to the last node l of the path, with clearly defined source and destination nodes.

Definition 3.5. Given a directed graph $G = (V, E)$ where V is a set of nodes and E is a set of edges, and a pair of vertices s and $l \in V$. l is reachable by s if there is a sequence of vertices s,v_1,v_2,...,v_k,l such that every edge (v_{i-1},v_i) in E for 1 $\leq$ i $<$ k

The problem of temporal reachability is defined as the task to determine a temporal path that connects a source node u and a destination node v. The common property with the problem of reachability in directed graphs is that for every edge (a,b) that belongs to the temporal path the nodes of origin and termination of said edge should be clearly defined, in respect with the temporal information of the edge. As a result temporal reachability between two nodes is depended of the time interval in which the query in done.

Definition 3.6. Given the graph $G = (V, E)$ at time instance t, where V is the set of all the nodes and E is the set of all the edges that have appeared by time T. Assuming that u and v are two nodes that belong to V, that are not temporaly reachable at time T, our goal is to predict if these nodes will be reachable at a ***future*** time instance $T + 1$.

Chapter 4

Algorithms

In this thesis the main focus of study is future prediction queries on temporal graphs. We define three different types of queries in the context of future temporal shortest path prediction. The first type of queries focuses on future link prediction between pairs of nodes on a temporal graph. The second utilizes recurrent neural networks to predict the next item a node will interact with at a future point in time. Lastly a distance prediction model is proposed. Furthermore we propose an algorithm for the prediction of the future temporal reachability between pairs of nodes.

For the link prediction and temporal reachability tasks we use an off-the-shelf link predictor. Given two nodes u and v, a link predictor estimates whether an edge will appear between u and v. We implement our strategy for two types of link predictors. The first type is a binary predictor that predicts either the existence of a link or not. The second type is a probabilistic link predictor that predicts a probability for the existence of a link.

4.1 Temporal Shortest Path

For the computation of the shortest path distance of u to all other nodes in V at time instance t, we take advantage of Algorithm 5 proposed at [20] but without the λ. In the original paper given a temporal graph each edge is denoted by a quartet (u, v, t, λ), where λ is the time needed to reach from node u to v. We assume that this process happens instantly, as a result do not use in the context of this thesis. This is an efficient one-pass algorithm for the computation of the shortest path from a source node, that takes into account the time instance when each link is created.

Algorithm 5 exploits the concept of the representation of the edges of a graph as a stream. **Edge stream** is a form of representation of a temporal graph as an ordered sequence of edges, based on the time that each edge was created. The edges that are created at the same time instance are ordered arbitrarily in the stream.

A naive approach to the problem of the shortest path computation, is to compute all the temporal path combinations between two nodes u and v, in a time interval [t_a, t_ω], and choose the one with the minimum distance. A better approach is to take advantage of the hypothesis that if P is the shortest temporal path between two nodes u and v_k in a time interval [t_a, t_ω]. Then every prefix-subpath P_i is the shortest path for u to v_i in the time interval [t_a, end(P_i)].

Algorithm 4.1 Computing shortest-path distance

Result: The distance of the shortest path f[v] from x to every node v $\in$ V within a time interval [t_a, tω]
Foreach v $\in$ V:
    create empty L_v
Initialize f[x] = 0, and f[v] = ∞, for all v $\in$ V $\setminus$ x
Foreach Incoming edge e=(u, v, t) in the edge stream:
  If t $\geq$ tα and t $\leq$ tω:
    If u == x:
      If (0, t) $\notin$ L_x:
        Insert (0, t) into L_x
    Let (d $'$[u], a $'$[u]) be the element in L_u where
    a $'$[u] = max(a[u] : (d[u], a[u]) $\in$ L_u, a[u]$\leq$ t)
    d[v] = d $'$[u] + distanceOfEdge
    a[v] = t
    If a[v] in L_v:
      Update the corresponding d[v] in L_v
    else:
      Insert (d[v], a[v]) into L_v
    Remove dominated elements in L_v
    If d[v] $<$ f[v]:
      f[v] = d[v]
  If $t \geq t_\omega$:
    Break the for-loop and return
return f[v] for each v $\in$ V

Algorithm 4.1 proposes a greedy strategy that maintains both the ending time and the distance of a path, because it is possible for different shortest paths to exist between two nodes, for different time intervals. For every node v in the graph, the algorithm uses a sorted list L_v, comprised of (d[v],a[v]) elements, where the distance d of the path is used as the sorting element. With the notations d[v] and a[v] we describe the notions of shortest path distance between two nodes and the time instance that it happens.

The concept of dominated and dominating elements in L_v is crucial for the com-

putation of the shortest path. Given two elements (d_0[v],a_0[v]) and (d_1[v],a_1[v]) in L_v we can say that (d_1[v],a_1[v]) ***dominates*** (d_0[v],a_0[v]) if the following hypothesis is satisfied d_0[v] $< d_1$[v] and a_0[v] $\leq a_1$[v], or d_0[v] = d_1[v] and a_0[v] $< a_1$[v]. Therefore we can say that (d_0[v],a_0[v]) is ***dominated*** by (d_1[v],a_1[v]) in L_v.

4.2 Link Prediction

For the task of link prediction we combine node embeddings, computed by Node2vec, and a classifier that predicts the probability of a link appearing between two nodes. However link prediction is a task that requires the involvement of the embeddings of pairs of nodes. Most embedding creation strategies compute embeddings that correspond to the nodes of a graph. In order to confront this problem, we use the binary operations as presented in [8], in order to create new embeddings. These new embeddings no longer correspond to specific nodes but they define pairs of nodes in the network.

The edges of a sub-graph $G_s^{'}$ of the static representation G_s of the original temporal graph G_T are used to train the classifier. The edges that appear for the first time after a pivot point t $\leq$ T, are deleted from the edges of $G_s^{'}$. Thus the positive examples of the train and test sets are defined. The train and test sets contain both existing edges and non existing edges of the network, as positive and negative examples respectively. A static graph is defined as a snapshot of the state of the network at a specific time instance. This methodology is presented in Algorithm 4.2.

Algorithm 4.2 Training of the classifier

Input: edge stream of connected temporal graph G_T, pivot point pivotPoint
Output: edge probability classifier
ForEach edge e = (u,v,t) in the edge stream:
    If t $\geq$ pivotPoint and (u,v) not in previous time instance:
        add e to deleted edges
    else:
        add e to edge stream of G'
create static representation G'_s
create training set with edges of G'_s and equal number of pairs of nodes that do not exist until pivotPoint
create test set with the deleted edges and equal number of pairs of nodes that do not exist until T
create embeddings of the nodes in G'_s
create embeddings for the pairs of nodes in the training and test sets, using **binary operations** on node embedding features
Train a *classifier* on the pairs of nodes of the training set
Test the *classifier* on the pairs of nodes of the test set and compute *Area Under Curve*
Return the *classifier* and *Area Under Curve*

We propose a technique for the computation of the temporal shortest paths in the future that combines algorithms 4.1 and 4.2. The temporal sub-graph G'_T is used for the computation of the shortest paths before the pivot time and the classifier is used for the prediction of whether an edge will appear in the future between two presently unconnected nodes. The classifier is used on nodes w that, on the pivot time, have shorter distance from a destination node v than the source node u.

Algorithm 4.3 Prediction of future shortest temporal path

1: **Input**: temporal graph G_T, source node u, destination node v, hash table H_u
d: the current temporal distance between u and v.
2: **Output**: new shortest path distance
3: **ForEach** node w with temporal distance d - 2:
4: **If** *link predictor* finds that (w, v) exists:
5: shortest path distance = d -1
6: **Update** shortest path (u, w) in H_u with new distance d - 1

Algorithm 4.3 presents the method for the prediction of the shortest paths at a future time instance. We assume that the distance between the neighboring nodes is of length 1. In order to reduce the computational cost to only the needed computations of the algorithm, we restrict the search only to paths that on the pivot point have a shortest temporal distance that is smaller than the shortest distance of the source node to the target node minus 2.

A similar strategy, is followed with the tNodeEmbed and Multilens embedding strategies. For the computation of the node embeddings, these methods take into account the temporal dynamics of the graph. Like in our strategy the pivot time point separates the graph into a past temporal sub-graph $G^{'}$. The pivot point is defined as the nearest timestamp to a user given ideal pivot point that belongs to the maximum time interval [t_a, t_ω] of the network.

4.3 Next Interaction Prediction Algorithm

In this section we propose a different method for the computation of the temporal shortest paths between two nodes u and v given the specific time instance T_i we want to make a prediction at. To achieve that we modify Algorithm 4.3. We replace the link prediction classifier with a system that given a source node w recommends the next node it will connect with at a specific time instance.

Algorithm 4.4 Next Iteration Prediction

Input: temporal graph G_T, source node u, destination node v, hash table H_u
d: the current temporal distance between u and v.
L_T = List of time instances to predict
Output: the time instance t' and the shortest temporal path distance d'
ForEach time instance t in L_T:
  d_t: distance between u and v at time t
  ForEach: temporal distance d' <=d_t in distance hash table:
    ForEach: node w with temporal distance d':
      z = predicted node w will interact with at t+1
      If predicted node z corresponds to the destination node v:
        t' = t+1
        d' = d+1
      Update the H_u with the nodes recommended for the next time instance

With Algorithm 4.4 we achieve specificity in our predictions. The predicted shortest temporal paths are carried out for a specific point in time. Given a source node u, a destination node v and a graph G_T at a specific point in time we want to predict distance d' of the shortest temporal path at a future point in time t'.

We use L2 distance in order to verify whether the predicted embedding z corresponds to the embedding of the destination node v. The L2 distance, also known as the Euclidean norm, in the case of this method is the shortest distance between two vectors. Jodie uses the L2 distance to validate if the predicted embedding corresponds to the embedding of the desired destination node, and then ranks the embedding based on this distance.

Proof. Given that the next interaction predictor gives the correct node predictions. And a source node u and a destination node v of the future shortest temporal path.
if p_t: the true shortest temporal path from u to v at time $t^{'}$.
if p_{pred}: the predicted shortest temporal path from u to v at time $t^{'}$.

$$\text{We assume } p_t \neq p_{pred}$$

For every future time instance t_i the algorithm predicts the nodes that will expand the shortest temporal path of the previous time instance.

Since the next interaction predictor gives the correct node predictions for every t_i all possible paths will be predicted.

Therefore, at time instance $t^{'}$, the predicted path will correspond to the real one. As a result $p_t \quad p_{pred}$

□

Due to the fact that Algorithm 4.4 predicts, at every time instance, from the final node of the shortest temporal path that is nearest to the source node u to the one that is farthest, we can be sure that when an embedding z that corresponds to the one of the destination node is predicted, the distance of the shortest future temporal path will have been found. Therefore there is no need for further iterations of the algorithm to be carried out. On the contrary if there is no correspondence between the predicted embeddings and the destination embedding, we update the hash table to account for the predicted expansion of the shortest temporal paths when the the next time instance is examined.

4.4 Distance Prediction Algorithm

In this section we describe a method for the prediction of the shortest path on a temporal network. For the distance prediction task we modify two of the models presented in [13] to account for the temporal change in the network.

4.4.1 vdist2vec

The vdist2vec model takes as input the one-hot vector h_i, h_j of two nodes v_i,v_j and creates the embedding representations of these nodes. These embeddings are then fed into a distance prediction network for the prediction of the distance.

The Loss function for the optimization of the distance prediction task is the mean square error (E_ϕ) between the true distance of the vertices d(v_i,v_j) and the predicted distance $\widehat{d}_{i,j}$.

$$L_d = E_\phi[(d(v_i, v_j) - \widehat{d}_{i,j})^2] \tag{4.1}$$

4.4.2 vdist2vec-L

The optimization of vdist2vec modifies the loss function to concentrate on shrinking the larger errors

$$L_n = E_\phi[f_\delta(d(v_i, v_j) - \widehat{d}_{i,j})] \tag{4.2}$$

The f_δ function is based on the *Huber* loss function and δ is the top 1% with the largest prediction error.

On the original paper the above models are trained on the distance of random vertex pairs of the graph. We modify the training and testing process to account for the temporal information.

4.4.3 Altering the models

In the context of this thesis the vdist2vec and vdist2vec-L models are trained on the shortest temporal paths between pairs of vertices, before a specified time point T that denotes the present, and tested on the future shortest temporal paths after T.

4.5 Extension

We consider and describe two extensions of the link prediction task. The first extension uses a probabilistic link predictor to compute the expected shortest temporal distance between a pair of nodes at a future point in time. While the second extension defines the problem of future temporal reachability using a binary link predictor.

4.5.1 Link Prediction Algorithm with Probabilities

In this section we propose a technique that constitutes a variation of algorithm 4.3 with a probabilistic classifier in place of the binary one. Given a pair of nodes u and v, a link predictor estimates the probability that an edge will appear between u and v at a future time instance.

Algorithm 4.5 Prediction of the expected distance of future shortest temporal path

1: **Input**: temporal graph G_T, source node u, destination node v, hash table H_u
d: the current temporal distance between u and v.
2: **Output**: expected shortest path distance
3: **ForEach**: temporal distance d' in distance hash table:
4: **ForEach**: node w with temporal distance d':
5: p_e = probability that (w, v) exists, given by link predictor
6: π_d = π_d * (1- p_e)
7: expected temporal distance at d' $expectedDist[d']$ = d' * π_d
8: expected shortest path distance = $\sum expectedDist[d']$

Algorithm 4.5 presents the method for the prediction of the expected distance of the shortest paths at a future time instance. The expected distance between two nodes is defined as the sum of the products of the possible distances times the probability that a path with the corresponding distance will occur. This computation strategy is presented in Equation 4.4.

The probability p_e of an edge appearing between the final node of a temporal path, with distance d from the source node, and the destination node is computed by the link predictor. The product of these probabilities gives the final probability π_d of a path with distance d, to exist.

$$\pi_d = 1 - \prod(1 - p_e) \tag{4.3}$$

Given the probabilities π_d that represent the probability of a temporal path of distance d to exist we compute the expected distance. The expected distance of the shortest path constitutes the result of the aggregation of the products between the probability of a path of distance d existing and the corresponding distance. The proposed strategy is presented in equation 4.3.

$$expecteddistance = \sum \pi_d \times d \tag{4.4}$$

4.5.2 Temporal Reachability Prediction Algorithm

In this section we propose an algorithm for the prediction of the temporal reachability between pairs of nodes at a future time instance. Two nodes are temporally reachable

when there is a temporal path that connects them. We utilize a classifier on pairs of nodes that, on the pivot time, are not connected with a temporal path.

Algorithm 4.6 Prediction of temporal reachability between nodes

1: **Input**: temporal graph G_T, source node u, destination node v, hash table H_u d: the current temporal distance between u and v.
2: **Output**: Temporal reachability prediction
3: **ForEach** node w with temporal distance d - 2:
4: **If** *link predictor* finds that (w, v) exists:
5: **return** True

Algorithm 4.6 presents the method to predict whether two nodes will be temporally reachable at a future time instance. If the link predictor finds that a link is expected to exist between at least one node w, of the temporal paths that begin with the source node u, and the destination node v we infer that v is temporally reachable from u. This way computational cost of the algorithm is restricted only to the computations that are necessary for the prediction of the future temporal reachability.

Chapter 5

Experimental Evaluation

In this chapter we evaluate our different future prediction tasks. We use temporal datasets that are derived from a vast array of networks (e.g. email, protein interaction, phone call networks). Furthermore we evaluate the predictions with metrics that are appropriate for the corresponding future prediction task.

In the first section we present the experiments and the results of our evaluation for the task of the future shortest temporal path prediction. We test our three different approaches to the problem and compare the results of each different method. In the second section we present two extensions of the link prediction algorithm, the first extension uses a probabilistic classifier to compute the expected shortest temporal distance, while the second extension tackles the problem of the future temporal reachability between nodes.

5.1 Experimental Results

In this section we will describe the datasets we used, and the experiments we performed, and the metrics used for the evaluation of the prediction methods that we propose. For all the tasks, the mean square error metric is used for the evaluation of

the distance of the predicted future shortest temporal path. Specifically, for the task of the next interaction prediction the mean reciprocal rank is used. Lastly, for the future distance prediction the mean relative error metric is also used.

In order to measure how well the link predictor can classify whether a pair of nodes will develop a link or not, we use the area under curve metric. The classifier is evaluated for the future link prediction and temporal reachability tasks.

Table 5.1: Dataset features

Dataset	mail-Eu-core	MOOC User Action	PPI	reality-call	sign-bitcoinalpha	enron-employees
Nodes	986	7,047	16,458	6,809	3775	150
Edges	332,334	411,749	144,033	38,106	24186	50,571
Static Edges	24,929	59,835		7,697	14120	1611
Diameter	6	3	8	8	10	4
Timestamps	207,880	345,600	42	51,572	24,180	16066

5.1.1 Datasets

For the evaluation of the prediction tasks and the accuracy of the future shortest path prediction, we utilize the datasets presented in Table 5.1.

email-Eu-core[1]. A network that was generated using the incoming and outgoing email communications between anonymous members of a large European research institution. The users are the nodes and the directed edges correspond to the email communication between two users at a specific timestamp.

MOOC User Action[2]. This graph corresponds to the actions taken by the users of a popular Mooc platform. The edges represent the action taken by a user on a target, while the nodes represent the users. The timestamp of each edge defines the time in seconds that an action was taken.

PPI[3]. This graph includes the protein-protein interactions (PPI). Each node corresponds to a protein, while the edges that connect the proteins are the observed biological interaction. Each edge is defined by a timestamp that corresponds to the date that the interaction was discovered.

[1]https://snap.stanford.edu/data/email-Eu-core-temporal.html
[2]https://snap.stanford.edu/data/act-mooc.html
[3]https://snap.stanford.edu/data/cit-HepPh.html

reality-call[4]. This is a subgraph of the network used for the Reality Mining experiment conducted in 2004. Nodes are the participants of the experiment and edges correspond to phone calls between the nodes. The timestamp corresponds to the time in seconds the the phone call happened.

sign-bitcoinalpha[5]. A network based on the interactions of people who trade using Bitcoin. The nodes correspond to the humans trading with bitcoin and the edges correspond to the trust rating interactions between users. The timestamp defines the time of the rating, in seconds.

enron-employees[6] A network based on the exchange of emails between the employees of Enron. The nodes represent the employees and the edges are the email sent. The timestamp is computed in seconds and corresponds to a time interval from May, 1999 to June, 2002.

5.1.2 Evaluation Metrics

We evaluated our embedding approaches with regard to the respective tasks, temporal link prediction, the prediction of the shortest temporal path between nodes, next interaction prediction, and distance prediction.

For the **Temporal Link Prediction** task, the goal is to predict whether two nodes are going to develop an edge in a future time point.

The evaluation metric we used for all the methods was the area under curve (AUC). Area under curve is an estimate of the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance.

The goal of the **Future shortest temporal path prediction** is to compute whether a new shortest temporal path will appear between two nodes in a future time point.

In order to evaluate the accuracy of shortest temporal path prediction task we use the following metrics.

1. **Mean of the mean squared errors**. We compute the mean value of the mean squared errors that are observed for the prediction task from every possible source node.

[4]https://networkrepository.com/ia-reality-call.php
[5]https://snap.stanford.edu/data/soc-sign-bitcoin-alpha.html
[6]https://networkrepository.com/ia-enron-employees.php

2. **True Positives**. The percentage of the pairs of nodes that are correctly predicted to have a path between them, compared to the total number of pairs of nodes.

3. **True Negatives** The percentage of the pairs of nodes that are correctly predicted to not have a path between them, compared to the total number of pairs of nodes.

4. **False Positives** The percentage of the pairs of nodes that are wrongly predicted to have a path between them, compared to the total number of pairs of nodes.

5. **False Negatives** The percentage of the pairs of nodes that are wrongly predicted to not have a path between them, compared to the total number of pairs of nodes.

For the **Next Interaction Prediction** task, the goal is to predict the next item that a selected object will interact with at a future time instance.

For the evaluation of the next interaction task we used the mean reciprocal rank and for the evaluation of the resulting shortest temporal path we used the mean squared error metric.

The goal of the **Distance Prediction** task, is to predict the distance of a temporal path at a future time instance. The metric used for the evaluation of this task is the mean squared error.

5.1.3 Link Prediction

In this section we present the results of the our evaluation. Our evaluation includes a comparison between the different embedding feature parsing strategies of Node2vec, Multilens, and tNodeEmbed for the link prediction.

For the computation of the Node2vec and tNodeEmbed embeddings we use the default parameters proposed in the respective papers. Node2Vec in particular performs 10 walks of length 80 for the creation of vectors with 128 dimensions. On the contrary tNodeEmbed parses the Network at different time instances and results in temporal embeddings of 128 dimensions.

Train and Test Subsets

To divide our dataset into current and future state, we select a pivot point T such that 80% or the closest proportion possible of the edges have a timestamp earlier or

equal to the pivot point. The statistics of the datasets are shown in Table 5.2.

Our classifier for predicting the new edges is based on embeddings. We use three methods for the creation of the embeddings, namely Node2vec [8], tNodeEmbed [6] and Multilens [7] with the ϵ-graph time-series method defined in [21]. The embeddings are created using G_T.

We train a classifier using the embeddings. The edges that have a timestamp earlier or equal to the pivot point make up the positive examples of the training set, while the edges that have a timestamp after the pivot point form the positive examples of the test set. It is imperative that the edges that belong to the test set appear for the first time after the pivot point. For the negative examples of the training set an equal number of pairs of nodes, that do not form an edge at the pivot point, are chosen randomly. Likewise the negative examples of the test set are chosen randomly from the pairs of nodes that are not connected with an edge at all.

Table 5.2: Dataset features after edge deletion

	email-Eu-core	MOOC User Action	PPI	reality-call	sign-bitcoinalpha	enron-employees
Nodes	986	7047	16458	6809	3775	150
Deleted Edges	8865	515	28806	1787	10060	623
Diameter	7	3	10	8	10	4
Time Interval	[0, 69,459,254]	[0, 2,572,086]	[1970,2015]	[1095984059, 1105142178]	[1289192400, 1453438800]	[926389620, 1024674019]
Pivot Point	48,621,477	1,800,460	2012	1,101,478,930	1,387,740,240	995,188,699

The timestamps in the email and the MOOC datasets are normalized to start from zero, and counted in seconds. The largest time interval at the email dataset begins at 0 seconds and ends at 69,459,254 seconds. Likewise the largest time interval at the MOOC actions dataset begins at 0 seconds and ends at 2,572,086 seconds. On the contrary the timestamps at the PPI dataset are counted in years and are not normalized, the largest time interval begins in 1970 and ends in 2015.

Node2Vec Experiments Binary classifier

In this section we present the results of the evaluation for the task of the prediction of the shortest temporal. For the tests we use a binary clasifier that predicts whether an edge between two nodes exists and Node2Vec embeddings. The Node2Vec embeddings are vectors of 128 dimensions, for their creation 10 walks of 80 items length are used.

Table 5.3: AUC of the classifier using Node2vec embeddings

	email-Eu-core	MOOC User Action	reality-call	sign-bitcoinalpha	enron-employees
Average	0.53	0.71	0.57	0.41	0.62
Hadamard	0.78	0.59	0.95	0.84	0.91
WeightedL1	0.80	0.56	0.96	0.91	0.93
WeightedL2	0.80	0.59	0.96	0.91	0.92

Table 5.4: Future shortest path prediction mean of the MSE using Node2vec embeddings

	email-Eu-core	MOOC User Action	reality-call	sign-bitcoinalpha	enron-employees
Average	0.64	0	0.09	0.94	0.29
Hadamard	0.70	0	0.086	0.95	0.29
WeightedL1	0.66	0	0.087	0.93	0.29
WeightedL2	0.66	0	0.087	0.93	0.29

Table 5.3 presents the area under curve (AUC) of the classifier. We test the networks on classifiers that are trained on embeddings computed with different types of binary operations, as presented in Table 2.1. When we compare the AUC results with the mean of the MSE for the future shortest path prediction, presented in Table 5.4, we find a discrepancy between successful link predictions by the classifier and the correct future shortest path prediction. We find that while the classifier works better on the email network, the future shortest path prediction is closer to the truth on the MOOC, reality-call and enron datasets.

This contradictory observation where one network is very precise in the computation of the distance for the existing shortest paths, but does not work well in the classification of whether a shortest path exists or not, may be are result of the small number of positive examples of shortest paths that exist in the MOOC network (0.014) in contrast with the email network (0.979).

Table 5.5: Future shortest path prediction embedding features computed: using the Average binary operator

Average	email-Eu-core	MOOC User Action	reality-call	sign-bitcoinalpha	enron-employees
TP	0.979	0.014	0.721	0.992	1
TN	0	0.971	0.274	0.007	0
FP	0.021	0.015	0.004	0	0
FN	0	0	0	0	0

Table 5.5 shows how well the predicted future temporal shortest paths correspond to the ground truth, when a classifier that utilizes embeddings that are created with the average operation is used. We find that our methods correctly predicts the existence of shortest temporal paths in the future, with the exception of some of the more dense networks like the email (0.021) and the MOOC (0.015) that have a small percentage of wrongly predicted shortest temporal paths.

Table 5.6 presents the results for the shortest path prediction task when the Hadamard method is used. We observe that for the MOOC network specifically the percentage of the non existing shortest temporal paths that are correctly predicted decreases (0.819) while the persentage of the shortest temporal paths that are falsely predicted to exist increases (0.166).

Table 5.6: Future shortest path prediction embedding features computed: using the Hadamard binary operator

Hadamard	email-Eu-core	MOOC User Action	reality-call	sign-bitcoinalpha	enron-employees
TP	0.979	0.014	0.721	0.992	1
TN	0	0.819	0.274	0.006	0
FP	0.021	0.167	0.004	0	0
FN	0	0	0	0	0

Table 5.7: Future shortest path prediction embedding features computed: using the WeightedL1 binary operator

WeightedL1	email-Eu-core	MOOC User Action	reality-call	sign-bitcoinalpha	enron-employees
TP	0.979	0.014	0.721	0.992	1
TN	0	0.727	0.274	0.007	0
FP	0.021	0.259	0.003	0	0
FN	0	0	0	0	0

We observe that this trend is amplified when the binary operators of WeightedL1, presented in Table 5.7, and of WeightedL2, presented in Table 5.8, are used.

Table 5.8: Future shortest path prediction embedding features computed: using the WeightedL2 binary operator

WeightedL2	email-Eu-core	MOOC User Action	reality-call	sign-bitcoinalpha	enron-employees
TP	0.979	0.014	0.721	0.992	1
TN	0	0.714	0.274	0.007	0
FP	0.021	0.272	0.004	0	0
FN	0	0	0	0	0

tNodeEmbed Experiments

Table 5.9 presents the AUC results for the classifier trained on embeddings computed with tNodeEmbed, for the embedding creation the change of the morphology of the network through time, is taken into account. We find that the results for the link prediction task are better when temporal information of the network is taken into account during the embedding creation. Furthermore we find that the results of the shortest path prediction task follow the same trend, as shown in Table 5.10.

Table 5.9: AUC of the classifier using tNodeEmbed embeddings

	email-Eu-core	PPI	reality-call	sign-bitcoinalpha	enron-employees
tNodeEmbed	0.653	0.778	0.5	0.644	0.5

Table 5.10: Future shortest path prediction mean of the MSE using tNodeEmbed embeddings

	email-Eu-core	PPI	reality-call	sign-bitcoinalpha	enron-employees
tNodeEmbed	0.116	0.082	0.818	0.521	0.532

We observe that embeddings computed with tNodeEmbed lead to more precise predictions, that correspond to the true expected distance of the shortest temporal path. On the contrary, as presented in Table 5.11, we find that the percentage of the temporal paths that are correctly predicted to exist is consistent with the percentage found with the use of Node2vec. This means that the temporal information that is utilized for the creation of tNodeEmbed affects how efficient the prediction of the length of the shortest temporal path is, but it does not affect the prediction about the existence of the shortest temporal path.

Table 5.11: Future shortest path prediction embedding features computed: tNodeEmbed

tNodeEmbed	email-Eu-core	PPIs	reality-call	sign-bitcoinalpha	enron-employees
TP	0.971	0.886	0.749	0.988	1
TN	0.001	0	0.015	0.008	0
FP	0.027	0.113	0.235	0.004	0
FN	0	0	0	0	0

Multilens Experiments

In this section we present the results of the evaluation of the link prediction and temporal shortest path prediction tasks. For the creation of the embeddings and the training and testing of the link predictor we use the Multilens embedding method for the ϵ-graph time-series. The Multilens embeddings are vectors of 128 dimensions. We test this method for 100 source nodes and 50 destination nodes for each of the source.

The metrics used for the evaluation are the area under curve (AUC) and the mean squared error (MSE) and mean. The area under curve is used to evaluate the accuracy

of the link predictor. The mean squared error is used to assess the accuracy of the predicted distance.

We use two networks for the evaluation of the future link prediction task with the use of Multilens embeddings. The email-Eu-core Network that is derived from n email network and the sign-bitcoin-alpha network that is a social network of bitcoin users.

Table 5.12: AUC of the classifier using Multilens embeddings for ϵ-graph time-series

	email-Eu-core	sign-bitcoinalpha
Average	0.54	0.62
Hadamard	0.78	0.59
WeightedL1	0.68	0.54
WeightedL2	0.66	0.56

Table 5.12 presents AUC results for the classifier trained on embeddings computed with Multilens. We find that the use of the temporal information at the embedding creation leads to better link prediction results than Node2Vec, but similar to tNodeEmbed.

Table 5.13: Future shortest path prediction mean of the MSE using Multilens embeddings for ϵ-graph time-series

	email-Eu-core	sign-bitcoinalpha
Average	0.29	0.27
Hadamard	0.30	0.17
WeightedL1	0.33	0.28
WeightedL2	0.34	0.26

Table 5.13 presents MSE shortest temporal path prediction when a classifier trained on embeddings computed with Multilens is used. We find that the use of the temporal information at the embedding creation leads to better link prediction results than Node2Vec.

Table 5.14: Future shortest path prediction embedding features computed: using the Average binary operator

Average	email-Eu-core	sign-bitcoinalphas
TP	0.985	0.958
TN	0.013	0.034
FP	0.002	0.008
FN	0	0

Table 5.14 presents the results for the existence of a shortest temporal path when a classifier trained on embeddings computed with Multilens, is used. We find that the use of the temporal information at the embedding creation leads to better link prediction results than Node2Vec.

From these evaluations we come to the conclusion that when the temporal information is taken into account for the creation of the embeddings, the future shortest temporal path prediction is more accurate.

5.1.4 Next Interaction Experiments

For the next interaction prediction we use the Jodie framework. Given a source node, Jodie predicts the next object embedding that this node will interact with at a future time instance. With the use of Jodie specificity of the time instance of the prediction is achieved. We test this method for 100 source nodes and 50 destination nodes for each of the source. Furthermore we evaluate the shortest temporal distance for the next five time instances.

The metrics used for the evaluation are the mean squared error (MSE) and mean reciprocal rank (MRR). The mean squared error is used to assess the accuracy of the predicted distance. The mean reciprocal rank is used to evaluate the effectiveness of Jodie at predicting the next interaction at various time instances.

We use three different networks for the evaluation of the next interaction prediction task . The email-Eu-core Network that is derived from n email network, the reality-call that is created from the interactions of a phone call network and the sign-bitcoin-alpha network that is a social network of bitcoin users.

Table 5.15: Evaluation metrics of the next interaction prediction

	MSE	MRR
email	0.9	0.17
bitcoin	0.8	0.25
reality-call	0.4	0.42

Table 5.15 presents the results of the next interaction prediction task. We find that the proposed method works best on the reality-call network, the mean square error of the shortest temporal paths is 0.4 and the mean reciprocal rank is 0.42. This leads to the hypothesis that the next interaction prediction is highly dependent of the function of the original network.

Furthermore, we present figures that display the change of the effectiveness of the next interaction prediction over time. To achieve this we plot the change of the mean reciprocal rank over the course of the ten next time instances of the Networks.

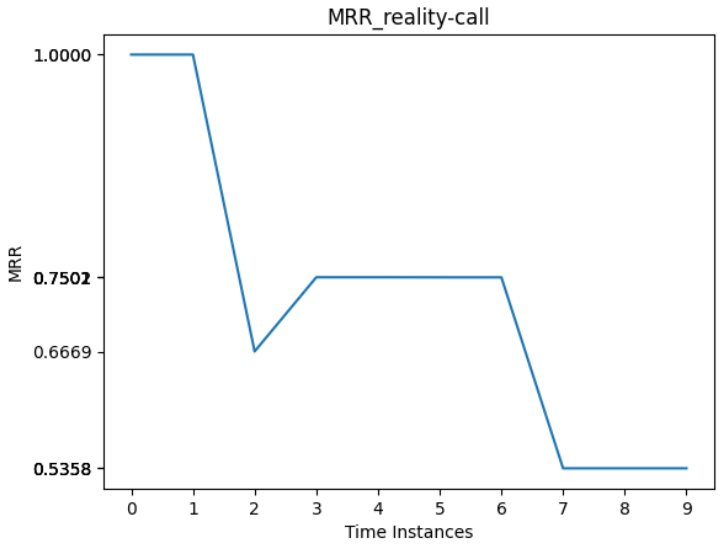

Figure 5.1: Mean Reciprocal Rank for different time instances, reality-call dataset

Figure 5.1 displays the change of the mean reciprocal rank over the course of ten time instances for the reality-call network. As it is expected we find that the quality of the next interaction prediction worsen the bigger the time difference from the present time instance becomes.

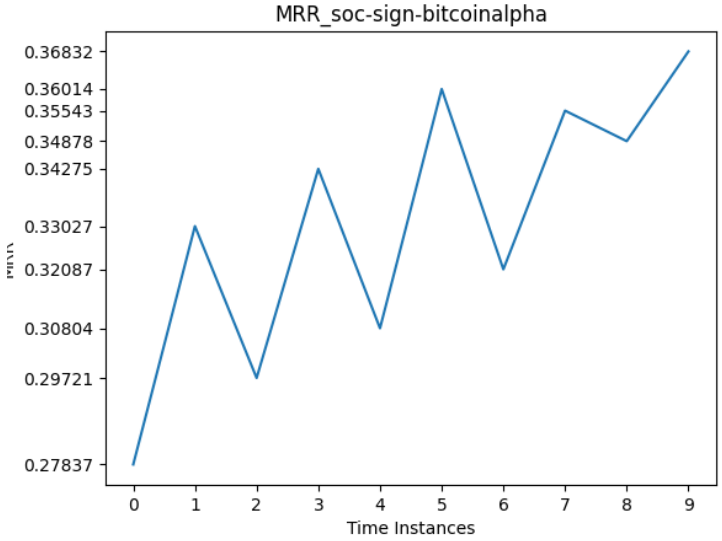

Figure 5.2: MRR for different time instances, bitcoin dataset

Figure 5.2 displays the change of the mean reciprocal rank over the course of ten time instances for the bitcoin network.

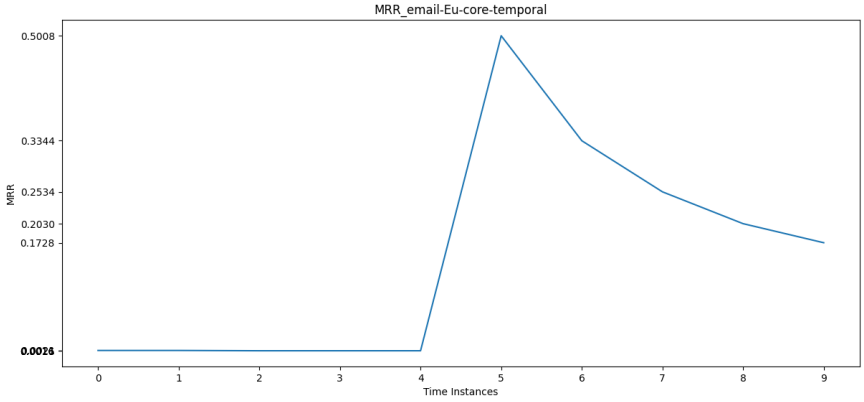

Figure 5.3: MRR for different time instances, email dataset

Figure 5.3 displays the change of the mean reciprocal rank over the course of ten time instances for the email network.

5.1.5 Distance Prediction Experiments

Two networks are used for the evaluation of the distance prediction method. The vdist2vec and and vdist2vec-L, an optimization of the first aproach that uses a novel loss function, are used.

The MLP distance predictor of these models, canstitutes of two hidden layers of 100 and 20 nodes respectively. The activation function used, for the hidden layers, is ReLU and sigmoid for the output layer. The model is trained for 20 epochs.

The metrics used for the evaluation of the accuracy of the predicted distance are the mean relative error (MRE) and the mean square error (MSE).

Table 5.16 presents the maximum, minimum and average temporal shortest path distance between the nodes of the networks we use. We find that on average the temporal shortest distance is nearer to the minimum distance of the network.

Table 5.16: Shortest temporal distance metrics

	email	bitcoin
maximum distance	5.0	8.0
average distance	1.7	1.92
minimum distance	1.0	1.0

Table 5.17 presents the mean squared error of the prediction of the shortest temporal distance for the paths of the networks. We find that the prediction task is more accurate than the strategy we propose. Furthermore we find that the basic model, vdist2vec works better on temporal queries than the optimized one, vdist2vec-L.

Table 5.17: Mean square Error of the distance prediction on temporal networks using Vdist2vec and Vdist2vec-L

	Vdist2vec	Vdist2vec-L
email	0.01	0.004
bitcoin	0.001	0.002

Table 5.18 displays the relative error results of the evaluation of the vdist2vec method. We find that for the network based on the mail interacions the distance prediction is closer to the truth than on the bitcoin interaction network.

Table 5.18: Relative Error of the distance prediction on temporal networks using Vdist2vec

	email	bitcoin
max relative error	1.985	4.0
mean relative error	0.023	0.452
min relative error	0	0

5.1.6 Experimental Results Comparison

Table 5.19: Comparative table of the mean square error for the models

	Node2vec	tnodeEmbed	Multilens	Vdist2vec	Vdist2vec-L	Jodie
email	0.70	0.116	0.30	0.01	0.004	0.9
bitcoin	0.95	0.521	0.59	0.001	0.002	0.8

Table 5.19 presents the results of the different shortest temporal path distance prediction strategies. We find that the models that directly predict the distance are the most accurate, followed by the link prediction strategies that utilize temporal embeddings.

Table 5.20: Models training time

Dataset	node2Vec	tNodeEmbed	Multilens	vdist2vec	vdist2vec-L	Jodie
mail-Eu-core	3.17	6.65	3.32	4h	4h	71h
MOOC User Action	1.13					
PPI	2.14	7.34				
reality-call	1.29	2.17				4h
sign-bitcoinalpha	0.22	0.67	0.46	9h	9h	12h
enron-employees	0.33	0.55				

Table 5.20 presents the time needed for the training of each model based on the method used. With the exception of the distance prediction methods (vdist2vec,

vdist2vec-L) and the next interaction prediction method (Jodie), which are calculated in hours, the calculation of the training time for the rest of the models is done in seconds.

The comparison of tables 5.19 and 5.20 leads us to the conclusion that even the distance prediction methods lead to predictions that are close to the real temporal distances. Nevertheless, the long training time that is needed to train them work negatively for the models. Furthermore the vdist2vec and vdist2vec-L models are methods that are specific to the distance prediction tasks. On the contrary Jodie and the methods that use classifiers can tackle a vast array of different problems, with few alterations in their architecture. Additionally, we believe that even though Jodie demands a lot of time for the training of the model and the distance prediction results are worse than the other methods there is room for improvement and further expansion of this model. We believe that this deviation from the actual distances is a result of the datasets we tested the method on. Networks with a lot of interactions are needed for the training and testing of this model. The use of networks with few interactions can lead to errors that accumulate the further away from the current time we try to predict, as it is observed here.

Qualitative evaluation:

The link prediction methods and Jodie are easily generalized to different types of problems, like reachability. On the contrary the distance prediction methods are limited to distance prediction problems. Furthermore, unlike the other strategies Jodie consists of the *Update* operation which means that there is no need to retrain the model.

Quantitative evaluation:

The distance prediction methods (v-dist2vec and vdeist2vec-L) lead to the most accurate distance predictions, but the disadvantage of this strategy is the time needed for the training of the models. On the contrary Jodie, has the update operation, which means that even though it has long training times there is no need for retraining. Furthermore the link prediction methods may be less accurate but the time needed for their training and retraining is small.

In conclusion, we believe that the link classifier methods that use temporal embeddings should preferred. Short periods of time are needed for the training of the classifiers and the predicted future distances correspond to the ground truth. We believe that the AUC results are better from the other embedding methods due to

the fact that the appearance of the same edges in different time instances is utilized for the creation of the embeddings. Additionally, we find that the use of probabilistic classifier for the prediction of the expected distance of the shortest temporal paths leads to accurate results, but we believe that further testing should be conducted on this method.

5.2 Extensions

In this section we present two extensions of the Link prediction algorithm. The first extension tackles the problem of the shortest path prediction using a probabilistic classifier for the computation of the expected distance between two nodes u and v.

The second extension focuses on the problem of temporal reachability. Given two nodes u and v that are not temporally reachable at the present, we want to find if they will be reachable in the future.

5.2.1 Link Prediction Experiments Expected Length

In this section we present results of the future shortest temporal path prediction task, the classifier used predicts a probability of an edge appearing between two nodes. We use this probability to predict the expected shortest temporal distance, for the future temporal paths.

Table 5.21: AUC of the probabilistic classifier using Node2vec embeddings

	email-Eu-core	reality-call	sign-bitcoinalpha
Average	0.50	0.50	0.50
Hadamard	0.50	0.53	0.58
WeightedL1	0.71	0.81	0.83
WeightedL2	0.72	0.81	0.84

Table 5.21 presents the area under curve (AUC) of the classifier. We test the networks on classifiers that are trained on embeddings computed with different types of binary operations, as presented in Table 2.1

Table 5.22: Future shortest path prediction mean of the MSE using Node2vec embeddings

	email-Eu-core	reality-call	sign-bitcoinalpha
Average	0.104	0.138	0.167
Hadamard	0.104	0.138	0.164
WeightedL1	0.105	0.135	0.161
WeightedL2	0.106	0.136	0.161

Table 5.22 presents the evaluation of the distance of the predicted shortest temporal paths. Compared with Table 5.4, we find that the probabilistic classifier leads to more accurate results than the binary one. We find that there is a big improvement in the predicted distance for all the embedding binary operations. Specifically for the email and bit coin datasets there is a big improvement for the prediction of the correct distance.

Table 5.23: Future shortest path prediction embedding features computed: using the Average binary operator

Average	email-Eu-core	reality-call	sign-bitcoinalpha
TP	0.978	0.989	0.997
TN	0.019	0.009	0.002
FP	0.002	0.001	0
FN	0	0	0

Table 5.24: Future shortest path prediction embedding features computed: using the Hadamard binary operator

Hadamard	email-Eu-core	reality-call	sign-bitcoinalpha
TP	0.978	0.989	0.997
TN	0.019	0.008	0.002
FP	0.002	0.002	0
FN	0	0	0

Table 5.25: Future shortest path prediction embedding features computed: using the WeightedL1 binary operator

WeightedL1	email-Eu-core	reality-call	sign-bitcoinalpha
TP	0.978	0.989	0.997
TN	0.019	0.009	0.002
FP	0.002	0.001	0
FN	0	0	0

Table 5.26: Future shortest path prediction embedding features computed: using the WeightedL2 binary operator

WeightedL2	email-Eu-core	reality-call	sign-bitcoinalpha
TP	0.978	0.989	0.997
TN	0.019	0.009	0.002
FP	0.002	0.001	0
FN	0	0	0

5.2.2 Temporal Reachability Results

In this section we present the results of the evaluation of the task of temporal reachability. For the link prediction task, link classifiers that were trained on the Node2Vec embeddings are used. We find that the proposed methodology leads to good classification of a pair of nodes being temporally reachable in the future.

We tested this method on three different temporal Networks. An email Network (email-Eu-core), a phone call network (reality-call) and a social network (sign-bitcoinalpha). We evaluate the temporal reachability for 100 randomly chosen source nodes and 100 randomly chosen destination nodes for each source node.

Table 5.27: Temporal reachability prediction: using the Average binary operator

Average	email-Eu-core	reality-call	sign-bitcoinalpha
TP	0.978	0.989	0.997
TN	0.019	0.009	0.002
FP	0.002	0.001	0
FN	0	0	0

Table 5.27 presents the results of the future temporal reachability for a classifier trained on embeddings that are created using the average binary operation. The best classification happens on the sign-bitcoinalpha, where 0.997% of the predicted pairs are correctly defined as temporally reachable. The same observation can also be made for Table 5.28 where the Hadamard binary operation is used.

Table 5.28: Temporal reachability prediction: using the Hadamard binary operator

Hadamard	email-Eu-core	reality-call	sign-bitcoinalpha
TP	0.978	0.989	0.997
TN	0.019	0.009	0.002
FP	0.002	0.001	0
FN	0	0	0

The best classification results are found on Table 5.29. When the WeightedL1 binary operation is used for the creation of the embeddings of the pairs of nodes, the classifier is the most precise in predicting the existence of the temporal reachability.

Table 5.29: Temporal reachability prediction: using the WeightedL1 binary operator

WeightedL1	email-Eu-core	reality-call	sign-bitcoinalpha
TP	0.978	0.989	0.997
TN	0.021	0.010	0.002
FP	0	0	0
FN	0	0	0

The use of the WeightedL2 binary operation, presented in Table 5.30 leads to similar results as the ones presented on the Tables 5.27 and 5.28.

Table 5.30: Temporal reachability prediction: using the WeightedL2 binary operator

WeightedL2	email-Eu-core	reality-call	sign-bitcoinalpha
TP	0.978	0.989	0.997
TN	0.019	0.009	0.002
FP	0.002	0.001	0
FN	0	0	0

In conclusion we find that, for the problem of temporal reachability, the proposed method leads to accurate results that correspond to the true temporal paths that we find on the graph.

Table 5.31: Shortest temporal path and reachability classification for the email dataset

Dataset	node2Vec	tNodeEmbed	Multilens	Reachability
TP	0.979	0.971	0.985	0.978
TN	0	0.001	0.013	0.021
FP	0.021	0.027	0.002	0
FN	0	0	0	0

Table 5.31 presents the comparative results for the shortest temporal path and reachability prediction tasks. We observe that the temporal path classification task leads to better results when we estimate the future reachability prediction than the prediction of the future shortest temporal path distance. Our assumption is that this happens due to the fact that the reachability problem is easier to solve than the problem of the prediction of the distance of a temporal path.

Chapter 6

Conclusions and Future Work

Previous research considers present-time shortest path queries (what is the current shortest path between two nodes), or historical shortest path queries (what was the shortest path in some time instance in the past). This thesis focuses on a less researched area of queries, future-time shortest path queries.

We propose three different strategies for the prediction of the future temporal shortest path. The first strategy is based on future link prediction between pairs of unconnected nodes. The second strategy utilizes a recurrent neural network to predict the next interaction of a node at a future point in time and the last predicts the distance between two nodes.

Link prediction utilises a binary link classifier and three different embedding methods. Node2Vec which is a non temporal embedding strategy, and tNodeEmbed and Multilens that use the temporal information for the creation of the node embeddings. For the task of the next interaction prediction we use the Jodie framework, that given a source node predicts the next object this node will interact with in the future. Jodie consists of two operations an update operation, that updates the embedding of a node, and a projection operation, that predicts the embedding of the node that the source node will interact with next. For the task of distance prediction we use vdist2vec and vdist2vec-L.

We found that the distance prediction methods lead to the most accurate prediction of the distance but the long training times of the models combined with the fact that it is limited only on distance prediction queries, makes this strategy not optimal.

On the contrary Jodie may be less accurate and need long training times but the update operation makes it a better solution that does not need retraining. Lastly link prediction is less accurate in the prediction of the distance of the shortest temporal paths than the distance prediction task, but the short training and the mostly accurate results when temporal embeddings are used makes it preferable from the other methods. Lastly Jodie and link prediction are strategies that can be easily generalised for different types of queries.

Furthermore, we propose two extension that exploit the link prediction strategy with node2Vec embeddings. In the first extension the binary link classifier is replaced with a probabilistic one and the expected distance of the temporal path is predicted. The second extension tackles the problem of temporal reachability using a binary link predictor. We found that that the probabilistic link predictor leads to more accurate predictions about the distance of a shortest temporal path. Lastly, for the temporal reachability task we found that it is more accurate at predicting the existance of a temporal path between two nodes than the other methods.

We believe that the algorithms we propose in this report, constitute a satisfactory first step in the management of this type of queries, but it is imperative to be expanded further. We want to carry out further tests, with different embedding methods, on the link prediction task using the probabilistic classifier. In the near future we want to test our methodology with more static embedding methods, such as P-GNN [12] and temporal embedding strategies, such as HTNE[22]. Furthermore we want to perform further tests on our algorithms using it on temporal graphs with different characteristics and morphologies.

Additionally, we believe that there is a future time point where the classifier stops giving effective predictions about the future state of the graph. Therefore there is a time interval, starting at the time point the classifier was trained until the point that the classifier stops making effective predictions. The ending point of this interval is the last time instance in which the classifier can be trained with a new state of the graph so that it continues making effective predictions.

The majority of the research that focuses on queries regarding the future state of a network, predict a future state of the graph in general. On the contrary we want to achieve time specificity in our predictions. This means that we want to be able to predict the changes that will happen in a specific time moment in the future. In order to achieve this we propose the use of a recurrent neural network (RNN) for

the role of the classifier. The RNN is trained on a sequence of the embeddings of the existing edges and the non existing pairs of nodes, on consecutive time states of the network. This way we believe that the classifier will be able to differentiate the edge embeddings at different time points and be able to better specify the state at a specific time.

A problem that is presented with this approach to the future prediction problem is that of the capacity of the memory that is needed to use these embeddings. What we found in methods like tNodeEmbed is that due to the nature of the embeddings, long vectors of integers, for large and dense networks the memory needed for the computation of the final embeddings exceeds the capabilities of current systems. This means that embeddings need to be saved on the permanent storage of the system and indexing methods are needed in order to retrieve the embeddings and sometimes even answer queries based on the graph. For the retrieval of the node or edge embeddings we can use hash tables where the key is the node and the value is the line in the file that the embedding is saved on. Lastly since the embedings are vectors we can use them to answer community detection and classification queries. We can modify the concept of Minimum Bounding Rectangles (MBR) used in R-trees to answer queries based on community detection or the prediction of an attribute.

Bibliography

[1] V. Kostakos, "Temporal graphs," *Physica A: Statistical Mechanics and its Applications*, vol. 388, no. 6, pp. 1007–1023, 2009.

[2] T. Erlebach, M. Hoffmann, and F. Kammer, "On temporal graph exploration," *Journal of Computer and System Sciences*, vol. 119, pp. 1–18, 2021.

[3] A. Debrouvier, E. Parodi, M. Perazzo, V. Soliani, and A. Vaisman, "A model and query language for temporal graph databases," *The VLDB Journal*, vol. 30, no. 5, pp. 825–858, 2021.

[4] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," *CoRR*, vol. abs/2006.10637, 2020.

[5] W. He, M. N. Vu, Z. Jiang, and M. T. Thai, "An explainer for temporal graph neural networks," *arXiv preprint arXiv:2209.00807*, 2022.

[6] U. Singer, I. Guy, and K. Radinsky, "Node embedding over temporal graphs," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, S. Kraus, Ed. ijcai.org, 2019, pp. 4605–4612. [Online]. Available: https://doi.org/10.24963/ijcai.2019/640

[7] D. Jin, R. A. Rossi, D. Koutra, E. Koh, S. Kim, and A. Rao, "Bridging network embedding and graph summarization," *CoRR*, vol. abs/1811.04461, 2018.

[8] 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, San Francisco, CA, USA, August 13-17, 2016*, B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen,

and R. Rastogi, Eds. ACM, 2016, pp. 855–864. [Online]. Available: https://doi.org/10.1145/2939672.2939754

[9] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, and H. Liu, "Attributed network embedding for learning in a dynamic environment," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, ser. CIKM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 387–396.

[10] P. Goyal, N. Kamra, X. He, and Y. Liu, "Dyngem: Deep embedding method for dynamic graphs," 2018.

[11] 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*, ser. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2018, p. 969–976.

[12] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 7134–7143. [Online]. Available: http://proceedings.mlr.press/v97/you19b.html

[13] J. Qi, W. Wang, R. Zhang, and Z. Zhao, "A learning based approach to predict shortest-path distances," in *23rd International Conference on Extending Database Technology (EDBT)*, 2020, pp. 367–370.

[14] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," *CoRR*, vol. abs/1908.01207, 2019.

[15] K. Semertzidis, E. Pitoura, and K. Lillis, "Timereach: Historical reachability queries on evolving graphs." in *EDBT*, vol. 15, 2015, pp. 121–132.

[16] K. Semertzidis and E. Pitoura, "Durable graph pattern queries on historical graphs," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 2016, pp. 541–552.

[17] K. Semertzidis, E. Pitoura, E. Terzi, and P. Tsaparas, "Finding lasting dense subgraphs," *Data Mining and Knowledge Discovery*, vol. 33, no. 5, pp. 1417–1445, 2019.

[18] H. Wu, J. Cheng, Y. Ke, S. Huang, Y. Huang, and H. Wu, "Efficient algorithms for temporal path computation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 11, pp. 2927–2942, 2016.

[19] N. M. Ahmed and L. Chen, "An efficient algorithm for link prediction in temporal uncertain social networks," *Information Sciences*, vol. 331, pp. 120–136, 2016.

[20] H. Wu, J. Cheng, S. Huang, Y. Ke, Y. Lu, and Y. Xu, "Path problems in temporal graphs," *Proc. VLDB Endow.*, vol. 7, no. 9, pp. 721–732, 2014. [Online]. Available: http://www.vldb.org/pvldb/vol7/p721-wu.pdf

[21] D. Jin, S. Kim, R. A. Rossi, and D. Koutra, "On generalizing static node embedding to dynamic settings: A practitioner's guide," 2021.

[22] Y. Zuo, G. Liu, H. Lin, J. Guo, X. Hu, and J. Wu, "Embedding temporal network via neighborhood formation," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Y. Guo and F. Farooq, Eds. ACM, 2018, pp. 2857–2866. [Online]. Available: https://doi.org/10.1145/3219819.3220054