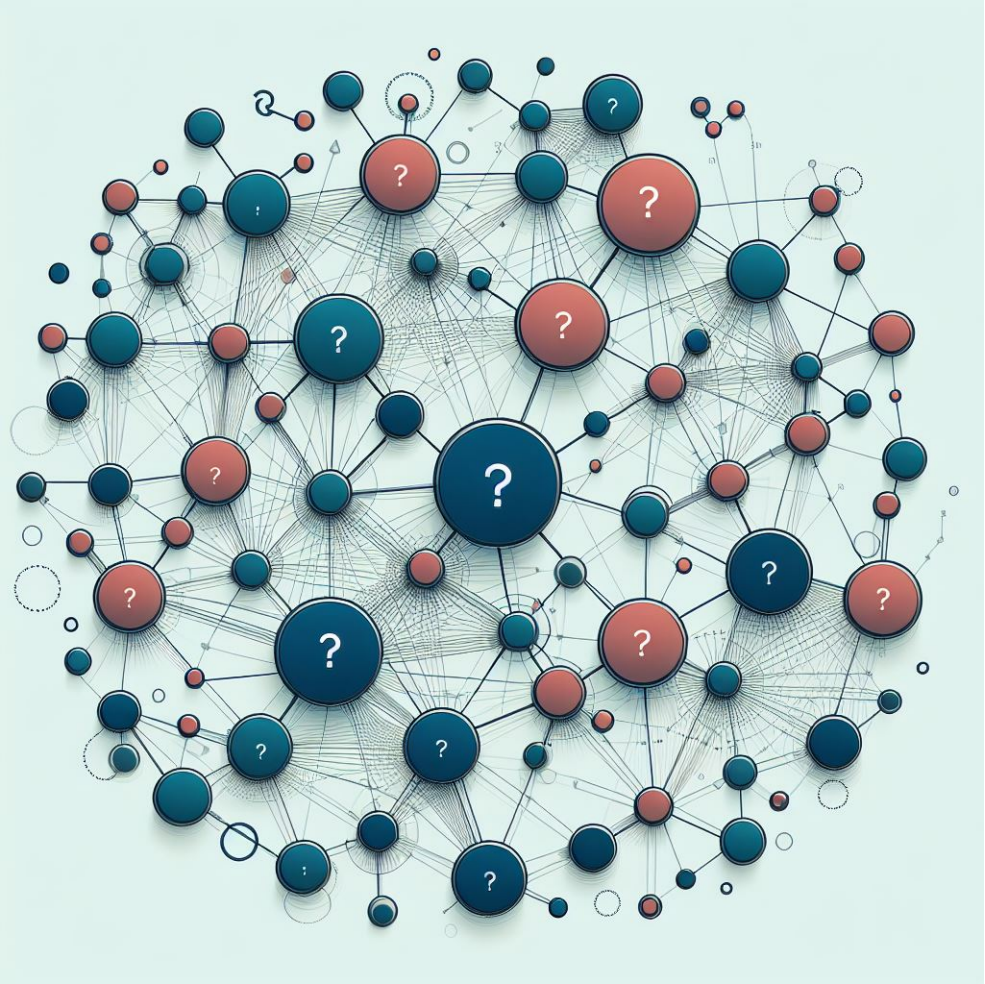
Predicting and Interpreting Bipartite Temporal Networks
MSc Thesis

Version of March 26, 2024



Stanislav Mironov

Predicting and Interpreting Bipartite Temporal Networks
MSc Thesis

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Stanislav Mironov
born in Moscow, Russia

Multimedia Computing Group
Department of Intelligent Systems
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

Predicting and Interpreting Bipartite Temporal Networks
MSc Thesis

Author: Stanislav Mironov
Student id: 4457668

Abstract

A network, is defined as a collection of nodes interconnected by links. When this topology changes through time, we call it a temporal network. A specific class of networks, with only two types of nodes with no connections between one kind, is the bipartite network. An example is a telecommunications network, where nodes represent telecommunication base station and various mobile services like web-browsing, streaming etc. A link may exist only between a base station and a service. Moreover, each link is associated with a time-evolving weight, which represents the volume of the traffic between the corresponding base station and service over time. This weight associated with each link is also called the activity weight, with the link considered active only when the weight is non-zero. Predicting such a temporal weighted network in the future is crucial for telecommunications engineers, allowing for e.g., better traffic management. Prediction of the unweighted temporal network one step ahead, at time $t+1$, based on the network observed in the past between $[t-L-1;t]$, has been studied recently in contact networks. However, the prediction of weighted temporal networks, or equivalently, predicting the activity weight of each link, in the future has not been explored yet. Moreover, we also aim to uncover the mechanisms that enable the prediction of a weighted temporal network. We achieve this by devising several strategies that help us select the most relevant links within the network, whose activity weights in the past serve as the input for the interpretable, statistical learning algorithm, LASSO Regression, to predict the activity of a given target link at time $t+1$. The focus of the strategies is to capture a relationship of activity weights between the selected and target links. These selected links range from most active links (amount of timesteps the link weight is non-zero), those with largest activity weights or most similar to the target link using several metrics. In this thesis we apply this general methodology to two bipartite networks sourced from real world data and

evaluate the performance of different strategies. Through the learned LASSO coefficients and prediction accuracy, we discover that past activity weight of a link is the best predictor for it's future weights. In terms of predicting power, most is coming from the past weights of the link we want to predict and one or two neighbouring links. Most of the selected links have minimal impact on the prediction accuracy. While different strategies of link selection excel in specific conditions, their improvement over the random link selection, is relatively low. The proposed method could be further applied to predict other weighted temporal networks with different properties to understand whether and how the the performance of link selection strategies depends on properties of the network to be predicted.

Thesis Committee:

Chair:	Dr. H. Wang, Faculty EEMCS, TU Delft
University supervisor:	Dr. H. Wang, Faculty EEMCS, TU Delft
Committee Member:	Rob Kooij, Faculty EEMCS, TU Delft

Preface

First of all, I would like to sincerely thank the Multimedia Computing Group and Dr. Huijuan Wang for giving me the opportunity to pursue a study in a field of my interest and helping me to pivot when the initial direction was being explored by another student. It allowed me to get loads of hands-on experience and explore various avenues of a quite novel and complex problem. This also forced me to really apply all the diverse knowledge that I have acquired during my MSc studies at Delft University of Technology in parctice, even though it might not have made the final version of this thesis. My supervisors Huijuan and Omar Fernández Robledo have thoroughly helped me during this stretch, as well as other phd students like Li Zhou. Despite my irregularities in working on this thesis, they alwyas found the time to brainstorm and exchange ideas, which I am very thankful for. Besides that, I want to thank my ex-classmates for helping me review certain parts of my thesis that were close to their expertise. Lastly, I would like to thank my family and friends for their support during my studies. Their faith in me helped me achieve my goals.

Stanislav Mironov
Delft, the Netherlands
March 26, 2024

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Networks serve as a fundamental framework for understanding the intricate relationships and interactions between components of a system. These components are denoted as nodes, while the relationships are represented by links. Among the vast classes of networks, bipartite networks offer a unique lens through which one can examine problems with an innate duality through the interactions between entities of two distinct sets, each contributing to and dependent on the other in a nuanced manner.

A bipartite network $G = (\mathbb{S}, \mathbb{U}, \mathbb{E})$ is defined by two disjoint sets, $\mathbb{S}$ containing S nodes and $\mathbb{U}$ containing U nodes, as well as the set $\mathbb{E}$ with E links. The key point here, is that links exist only between a node in $\mathbb{S}$ and one in $\mathbb{U}$. A weighted bipartite network can be represented by its biadjacency matrix R, an $S \times U$ rectangular matrix in which each element $R_{s,u}$ represents the weight between nodes s and u.

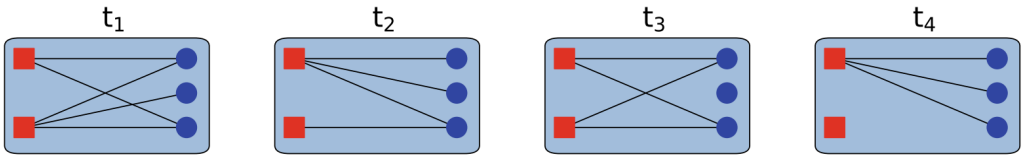


Figure 1.1 | **Schematic overview of a temporal network at $T = 4$ timesteps.**

When the topology of a network changes over time, we call it a temporal network. One example of a bipartite temporal network is the following telecommunications network where data transference between services and base stations, could be represented as a temporal weighted bipartite network[17] (see Figure 1.1). A temporal bipartite network observed at discrete time $T = [1, 2, \ldots, T]$, and composed of a set $\mathbb{S}$ of S services and a set $\mathbb{U}$ of U base stations can be represented by a $S \times U \times T$ temporal biadjacency matrix $\mathcal{R}$. Each element $\mathcal{R}_{s,u,t}$ represents the amount of data that has been transferred from service s to base station u at time t, where $s \in [1, S], u \in [1, U]$ and $t \in [1, T]$. We will refer to this as the activity of a link. It has an associated weight equal to $\mathcal{R}_{s,u,t}$ for each moment $t \in T$. Every nonzero weight

is interpreted as the link being active, hence the term activity. Links refer to the links in the time aggregated network where two nodes are connected by a link if they have at least a non-zero activity weight at one time step over all the timesteps $\in [1,T]$ in the dataset. A temporal bipartite network can be represented equally by its aggregated network and each link is further associated with a time series $v_{i_1,j}$, where each element $v_{i_1,j}(t) = \mathcal{R}_{i_1,j,t}$.

Analyzing the activity evolution of such networks can be quite valuable [33], [43]. There are many interesting properties of a network that can be examined, an example is detecting dynamic communities as shown by Lorenz-Spreen et al. [26]. However it is important to note that most of such research has been addressing unweighted unipartite temporal network prediction problems, using classical similarity indices[22]. While there exist bipartite modifications for such similarity metrics, as well as probabilistic models, the focus is always on predicting missing links Lu and Uddin [27]. The latest trend is based on learning-based solutions that learn the representation of the network e.g., through network embeddings [21]. These black-box algorithms have better accuracy but innately more challenging to explain. Since our objective is not only to predict temporal weighted bipartite network, but also being able to understand the underlying mechanisms of the prediction, new methods need to be developed.

The focus of our research, lies in predicting the activity weights of a network in the future, given the observation of the network in the past. Specifically, the objective is to predict the activity weight of each link at a time $t+1$ based on the weighted network observed in the past between $[t-L-1;t]$, with L representing the length of observation time. We consider two real-world weighted temporal bipartite networks: the traffic network between telecommunications base stations and telecommunications services, the human mobility network of individuals between the geographic locations where they live and venues they visit such as restaurants, pharmacies, fitness centers and so on. Predicting the future traffic volume from each telecommuncations base station and service and prediction the future mobility volume between each location and public venue are crucial for the management of traffic and control of epidemic spreading, respectively. Such prediction problems can be modeled as the problem of predicting a weighted temporal bipartite network.

Our approach is as follows: For each link in the networks, we collect a set of neighbouring links based on several strategies and use their past activity weights to predict the future activity of the desired (target) link. We propose several strategies to select links, which are based on properties of links in the network or their similarity to the target link. For example, one strategy selects links with the highest activity rate2.1 (the frequency with which the link is active i.e., has a non-zero weight).

Besides, we also select links that are more similar to the target link based on e.g., similarity metrics like euclidean and cosine distances between their accompanying time series. These are discussed further in 4.1. The selected subset of time series of weights is then fed into the LASSO algorithm, which learns the best regression from these weights and predicts a future weight of the target link. Reviewing the regression performance and the learned coefficients from the LASSO algorithm, lets

us compare the aforementioned properties and express the extent to which they benefit the prediction of the network. The results suggest that every link has varying degrees of influence in determining a given link's weight. Taking this into account, we analyse the networks for community structure and use it to limit the aggregated network fed to LASSO to check if this improves the prediction.

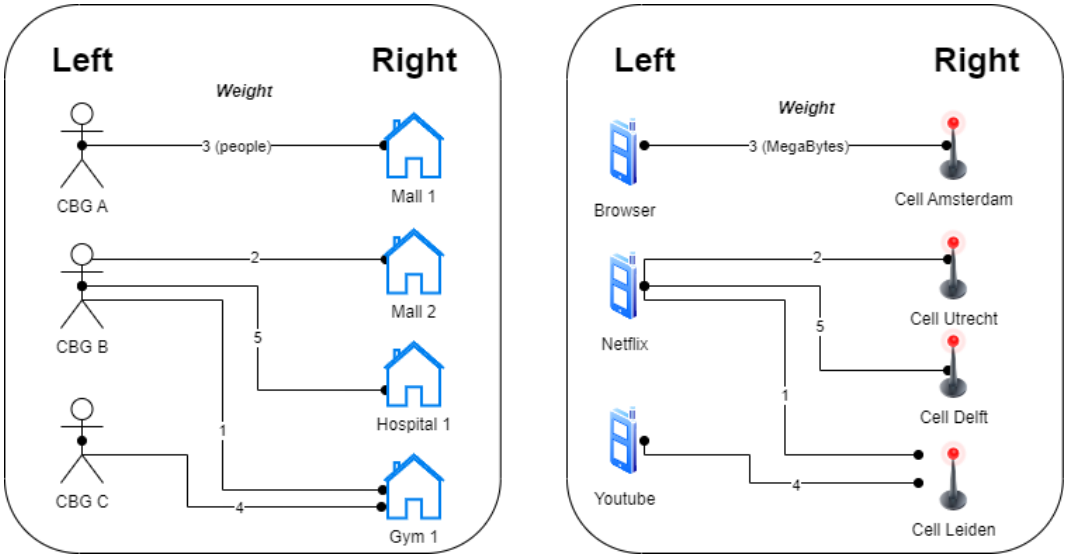


Figure 1.2 | **Two network graphs to illustrate the datasets used in our paper. Left is Stanford mobility network, right is KD telecom tower traffic..**

1.1 Problem Statement

At a high level, the picture is quite clear. Predicting link activity has been a hot topic within network science in the near past. A logical next step would be a foray into weighted variations of these networks. Having access to two weighted bipartite networks from real-world data has inspired us to explore the possibility of predicting not just the activities of links but also their accompanied weights. In the context of our real-world networks, an accurate prediction of the activity weights could help the involved experts to better spend their resources in the future based on past observations, some of which we collect in the current day. We propose a solution for this problem that is also quite interpretable. This makes it so that even if we can't achieve the most accurate prediction we can still experiment with different setups and reason on which factors benefit the prediction the most. Capturing this information and putting it in context of other problems can lead to increased insights and possible provide new solutions to previously unsolved problems.

A more mathematical formulation is as follows: The goal of this problem is to predict the network structure at a time $t+1$ based on the observed weighted network in the past between $[t-L+1;t]$, with L representing the historical observed data. This can also be done partially, where we select a subset of the network and focus

on predicting only the links within that subset. This will lessen the computational burden while still providing a blueprint for our proposed approach. This blueprint consists of the weighted bipartite network prediction model as well as the various strategies for optimizing the neighbouring links to learn the prediction from, all of which can be extended to similar problems. We want to explore a white-box model that can assist us with predicting activity weight classes in a bipartite temporal network, based on the topological aspects of the aggregated network, together with the temporal activity weight information of the whole network. Notice the mention of activity weight classes - due to the properties of the datasets, we have adjusted our preprocessing such that the links do not hold the raw activity weight but rather are classified in one of ten classes based on the percentile of the weight from the total pool of activity weights form the aggregated network. The reasoning for this is explained further in 2.3

1.2 Research Question(s)

The research question investigated in this thesis is:

To what extent is it possible, to predict the weighted network structure at time $t+1$ within a (sparse) weighted bipartite temporal network, based on historical temporal data, $t \in [1, 2, \ldots, T-L]$, given a trainingset of L timesteps?

This main research question can be further narrowed down into various sub-questions. Each of these is addressed in it's own section or chapter of the thesis.

- is it possible to apply Lasso Regression in such a way that we can predict future activity weights of each link?
- What is the impact of link activity and actvity rate on the prediction quality?
- What is the impact of memory length on the prediction quality?
- What is the impact of sparsity of such networks on the prediction quality?
- Can community detection assist in predicting the network structure?

1.3 Proposed Solutions

Using the available datasets, we streamline the inputs into weighted temporal bipartite networks of similar format. During this process we resample the link weight distribution of both networks, such that it will become a uniform distribution, containing 10 classes of active links. Each class is based on the original link weights placed in one of 10 percentiles. One extra class is used to represent the inactive links (sparse data). Then, for each link, we collect the activity weights of neighbouring

links. Using carefully crafted strategies we select a fixed subset of neighbouring links and using a fixed training period, achieved by using a sliding window on the activity weights, feed the historical weights of selected links into a linear regression algorithm known as LASSO. LASSO assigns a factor to the weight of each neighbouring link and regresses all inputs to predict the future weight of the target link. We compare performance across different datasets and try to further optimize different training periods and proposed neighbouring links selection strategies to make use of physical attributes of the links, such as the total activity rate and total weight in the aggregated network form. We also try to enhance the prediction accuracy using a stricter definition of neighbours using community detection algorithms for bipartite networks.

1.4 Contributions

The main contributions of this thesis are as follows:

- Creating a preprocessing pipeline to generate a structured and consistent weighted bipartite network representation from a real-life (sparse) datasets. This includes the scaling of weights from heterogeneous distributions to one homogeneous distribution to allow for comparison of different datasets.
- Analyze ways to determine similarity between links of a network in terms of their physical attributes, such as activity rates and activity weights.
- Determine if such similarities capture information that can help in improving the prediction of (active) links.
- Research a white-box algorithm that can take sparse historical weighted bipartite temporal data and construct a model capable of predicting the activity weights (and thus also the activity rate) of the network at a future time step. This includes analysis on the impact of neighbouring links and their properties on the prediction.

1.5 Thesis Structure

In this section we will lay out the thesis structure. First, the relevant technical background needed for this report will be discussed Chapter 2. This includes a brief introduction about the datasets that we evaluate upon in this thesis with all the necessary preprocessing steps we take, a mathematical formulation of the problem, and a list of definitions used in this report, as well as a deep dive into the sparsity of the datasets. Furthermore, we guide the reader through different limitations, assumptions and decisions we have made when formulating the problem statement and modelling the datasets. This is accompanied by the analysis of the underlying data, helping us make and justify these decisions. Then, in Chapter

3, we go through the history of the link prediction problem and how it relates to our research. We gradually explore previous works showcasing the application on weighted networks and extensions focused on incorporating the temporal dimension. We cap this off by highlighting the differences when dealing with bipartite networks and showcasing state-of-the-art solutions for similar problems. We try to explore not only the weighted link prediction literature but also dive deep into the solutions that inspired this thesis due to their similarity. In Chapter 4, we take a look at our proposed method of predicting the link weights, namely the LASSO regression algorithm and provide the assumptions and limitations of this approach. Chapter 5 gives an overview of the different experiments we conducted in order to evaluate our proposed model. This includes the comparison of different error and accuracy metrics along with context as to explain the logic behind the achieved performances. Then in chapter 6, we discuss our proposed enhancement of the prediction algorithm by incorporating community structure of one of the networks. Finally, Chapter 7 concludes the work we conducted as a whole while also extracting the most important findings and outcomes and the philosophy behind them. Of course, we also mention the limitations, as well as possible future improvements at the end of this section. By the end of this paper, readers will gain a comprehensive understanding of the state-of-the-art techniques for predicting future interactions in temporal bipartite networks and will be equipped with valuable insights to contribute to ongoing research in this dynamic and evolving field.

Chapter 2

Background

Here we introduce the reader with additional background information related to the thesis. Firstly, a short introduction is given on the datasets and the mathematical formulations of the networks they represent, along with core concepts we will use in this thesis. This is followed by a more in-depth look at the data contained in the networks.

2.1 Data Analysis

In this section, an overview is given of data on which the study has been performed. Two datasets were initially selected, both representing bipartite weighted temporal networks with the links representing a certain activity accompanied expressed through a weight. To be specific, this is data (SD) used by the Stanford study [7] on the mobility patterns of humans using cellphone tracking across various physical locations in the US during the early onset of COVID-19. The dataset, derived from cellphone data, according to Chang et. al[7] maps "the hourly movements of 98 million people from neighborhoods (census block groups, or CBGs) to points of interest (POIs) such as restaurants and religious establishments, connecting 57k CBGs to 553k POIs with 5.4 billion hourly edges" (Nature, Vol 589, page 82) , creating the so-called mobility networks. The timespan of mobility networks ranges from March 1 to May 2, 2020.

This data is represented in the form of mobility networks. We use the definition from the original paper [7]: "consider a complete undirected bipartite graph $G = (V, E)$ with time-varying edges. The vertices V are partitioned into two disjoint sets $C = \{c_1, \ldots, c_m\}$, representing m CBGs, and $P = \{p_1 \ldots, p_n\}$, representing n POIs. The weight $w_{ij}^{(t)}$ on an edge (c_i, p_j) at time t represents our estimate of the number of individuals from CBG c_i visiting POI p_j at the t-th hour of simulation." (Methods section, p. 1).

Metro area	CBGs	POIs	Hourly edges	Total modeled pop	Total visits
Atlanta	3,130	39,411	540,166,727	7,455,619	27,669,692
Chicago	6,812	62,420	540,112,026	10,169,539	33,785,702
Dallas	4,877	52,999	752,998,455	9,353,561	37,298,053
Houston	3,345	49,622	609,766,288	7,621,541	32,943,613
Los Angeles	8,904	83,954	643,758,979	16,101,274	38,101,674
Miami	3,555	40,964	487,544,190	6,833,129	26,347,947
New York City	14,763	122,428	1,057,789,207	20,729,481	66,581,080
Philadelphia	4,565	37,951	304,697,220	6,759,058	19,551,138
San Francisco	2,943	28,713	161,575,167	5,137,800	10,728,090
Washington DC	4,051	34,296	312,620,619	7,740,276	17,898,324
All metro areas combined	**56,945**	**552,758**	**5,411,028,878**	**97,901,278**	**310,905,313**

Figure 2.1 | **The metro areas captured in [7].**

Due to the sheer size we limit ourselves to one of the 10 metro areas that are contained in this dataset, namely San Francisco. The choice was based on the metro area that had the smallest size of mobility network to speed up the calculations and reduce the memory footprint. Every step and/or experiment we apply from this point on wards could be easily performed on any of the other metro area's using the same methodologies.

The second dataset is a collection of up and downstream data at telecom tower sites of a dutch telecom company we will refer to as KD. This data can be modelled in the same way as the mobility networks from SD. In short, we model it as a bipartite network between the sites and types of services that initiated the transfer of data, services like music and video streaming, internet browsing, certain apps and more. An exact definition cannot be publicly disclosed due to the sensitive nature of the information. The link activity weight would represent the total amount of bytes transferred at a specific site for a specific service, either up or down. As there was no statistical difference we decided to focus on the download transfers. The total dataset spans over the end of 2019 and the whole 2020 year but due to the impact of COVID we decided to limit the data to the first 2 months (2019-11-16 until 2020-01-17).

While the raw data of SD is in form of sparse adjacency matrices, for KD this data is in the edge list form contained in csv files.

From now on we will refer to the Standford data as SD and telecom data as KD.

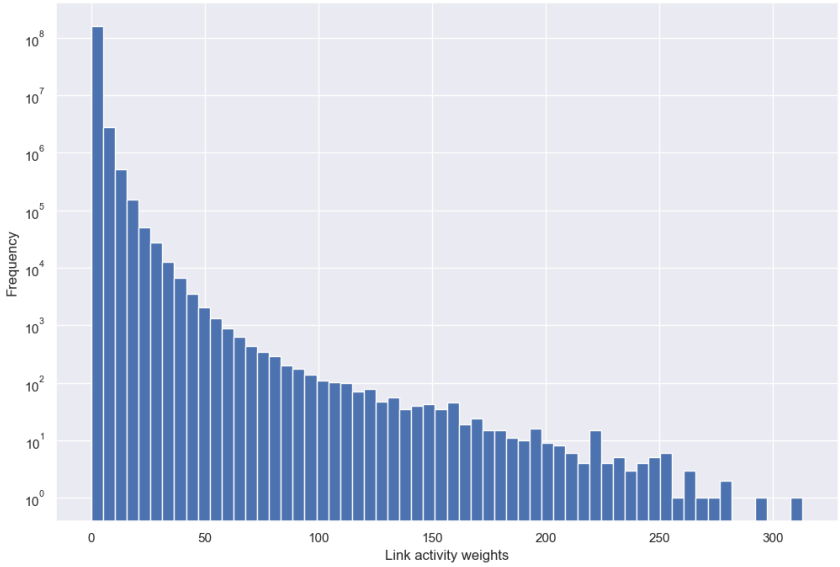
Table 2.1 Average network metrics per timestep.

Metric	SD	KD
Total number of nodes	31656	5419
Number of left-side nodes (locations)	2943	5166
Number of right-side nodes	28713	253
Total number of edges	310041	524129
Average degree	19.59	193.69
Standard deviation degree	46.90	617.81
Minimum degree	0	0
Maximum degree	1226	5161
Link density	0.04	0.42

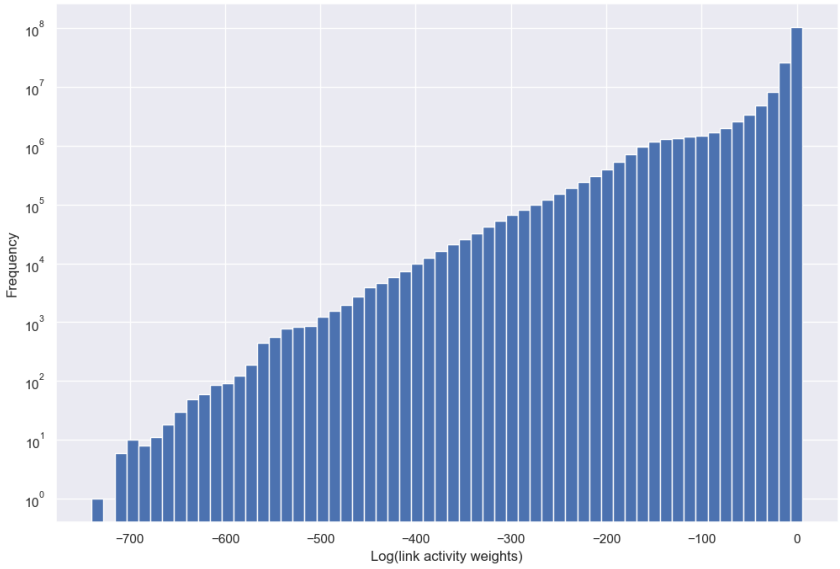
In 2.1 we show the most basic metrics of the temporal networks modelled through the raw data. The degree of a node is defined as the number of connections a node has to other nodes in the network. We see some stark differences between the two datasets: KD has more edges and less total nodes than SD which leads to a much higher degree as well as density. Keeping this context in mind, the standard deviation for both graphs is relatively high, which means that the average degree varies widely throughout. This is also expressed by the difference between the minimum and maximum degree. Lastly, the link density reminds us that we are dealing with sparse networks, with SD being an extreme case.

2.1.1 Link activity weights

First it's important to determine the distribution of the link activity weights for both networks, to know what kind of data we are working with. Below you will find two plots for each dataset, that attempt to visualise the important aspects of the distributions. We adjust the Y-scale to logarithmic in order to better visualize the distribution of activity weights and decide to omit the linear y-axis plots, as it is very hard to fit into one picture due to the effect of the sparsity of the dataset.



(a)

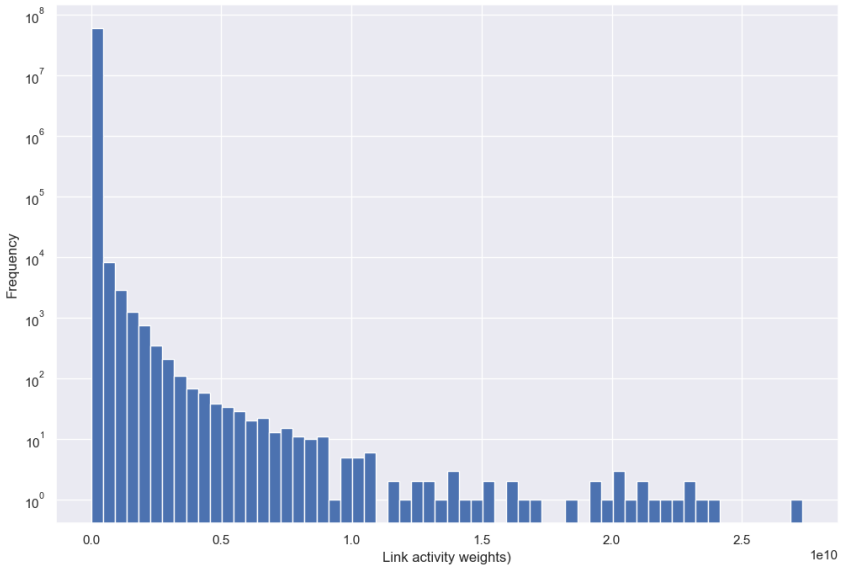


(b)

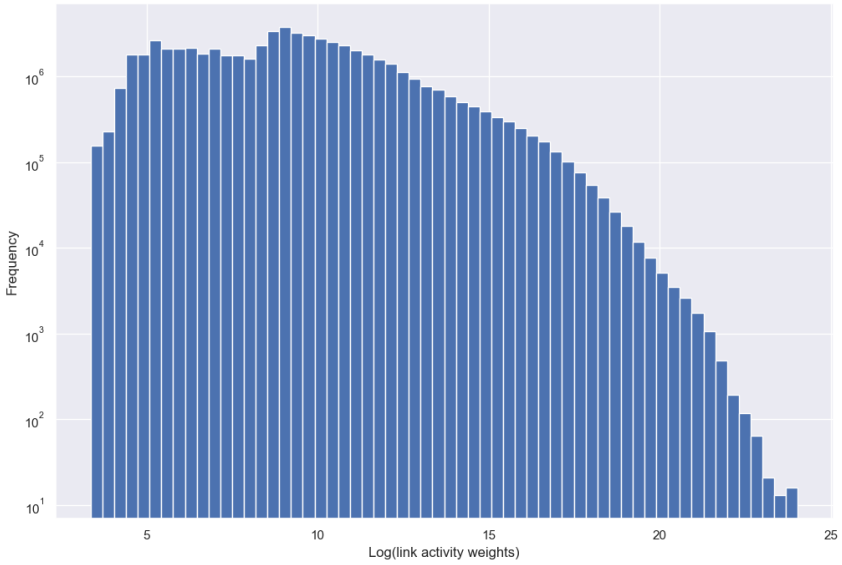
Figure 2.2 | **The link weight distribution of SD using a logarithmic Y-axis.** *A) This is the semilog distribution plotted with an logarithmic Y-axis. B) This are the log-transoformed link weights.*

The main takeaways here are that activity weight of SD are very likely following an exponential distribution. When applying statistical fits to well-known distributions, the Gamma(k,θ) (or $\Gamma(k,\theta)$ with parameters $k=0$ and $\theta=1.549$ was the best fit. This is essentially an exponential distribution with a rapid rate of decay due to the small scale indicated by θ. This is confirmed by 2.2a where we see that the

log-transformation of the weights results in a linear slope on semi logarithmic axes. This is surprising as the situation modelled by these weights (human mobility) is usually a very complex process, where distributions more akin to the power law. Yet we see a very steep slope in SD 2.2b compared to the KD dataset2.3b. The suggestion would be that the amount of visitors would occur independently and with a constant rate over time. The onset of COVID and the seasonality (daily activity) of the visits could be important factors here.



(a)



(b)

Figure 2.3 | **The link weight distribution of KD using a logarithmic Y-axis.** *A) This is the semilog distribution plotted with an logarithmic Y-axis. B) This are the log-transoformed link weights.*

For KD, the activity weight distributions are rather heterogeneous and do not necessarily adhere to one well known distribution. It does resemble a log normal or a powerlaw slope, but the left tail does not follow. This could have various explanations, e.g., a boundry effect through the fact that some services monitored by the telecom traffic have a certain required amount of data that needs to be exchanged.

On the other side, maybe the processes generating the activity weights are more complex and thus require the interaction of multiple distributions to properly model the real life scenario. Due to the large amount of datapoints we could not run a large amount of statistical tests or RSS-fits in this case, hence we can only suggest a few scenarios. This situation will impact our further processing of the data and our eventual problem-definition, as we will discuss further in the next sections.

2.2 Timestep resolution

While both datasets are roughly of the same timespan, the choice of timestep can impact the relative differences. In our case, looking at the smallest possible time unit of granularity, we have the Standford dataset tracking the mobility at every hour, while the KD data is updated every 5 seconds. That's quite the difference, so in order to streamline our experiments we decided to use an hourly granularity, as it was compatible with both datasets. Another benefit is the reduction of noisiness (randomness) of events that are often experienced when working with a timesteps in the order of seconds. We will discuss this process in more detail in 2.3, as well as our further experiments were we aggregate the data even further in larger timesteps.

2.3 Preprocessing

In following sections we will take the reader through the steps we have taken to go from the intial raw datasets and turn them into refined networks, show how we build up a model that can perform predictions on such networks and eventually explore various experiments that can further enlighten us as to what is important when predicting a weighted bipartite network.

The first step in the preprocessing pipeline, was to streamline the representation of the 2 datasets. For this the KD dataset had to be transformed into the adjacency matrix representation at each unique timestep, just like the SD format. During this conversion, consecutive matrices of lower timestep granulartiy where merged together, by using datetime aggregation, to achieve the desired hourly representation. The output now is essentially in the form of a 3d matrix with the dimenstions T, I, J. T is the total amount of (hourly) timesteps, 1512 and 1464 for SD and KD respectively. For each such timestep, there is a matrix M, such that $M = IxJ$, with I and J representing the 2 bipartite nodesets.

Both datasets can be classified as sparse due to the links not being activated most of the time, meaning their weight would be equal to 0. This can be also seen as a link being inactive, in other words the absence of a link. It is important for our model to be able to also predict the absence of links at certain timesteps (activity weight values of 0). But from our earlier findings about the link activity weights we have seen that they span quite a large numerical space with a skewed distribution. This could definitely impact a more simple model which we intend to deploy for the prediction.

Our approach here is to keep the sparsity of the data to allow the model to learn to predict the absence of links, while reshaping the activity weights to a more traditional distribution. Common choices for this transformation are often either a uniform or a normal distribution. We decided on using the first option for both datasets to keep things simple. We divide the values into 10 bins each representing the 10% percentile of the data. This way the definition of what we are trying to predict changes slightly.
We no longer will be predicting the precise activity weight but instead will try to predict a class in which the activity weight of the associated link would fall. The classes in this scenario are simply the percentiles of the actual underlying data. This slightly changes the problem definition.

Before we continue, it is important to mention that we scale the SD data by a factor of 6.9 based on the on the ratio of the US population to the number of devices used in collecting this data. Other considerations for SD are explained in A.1

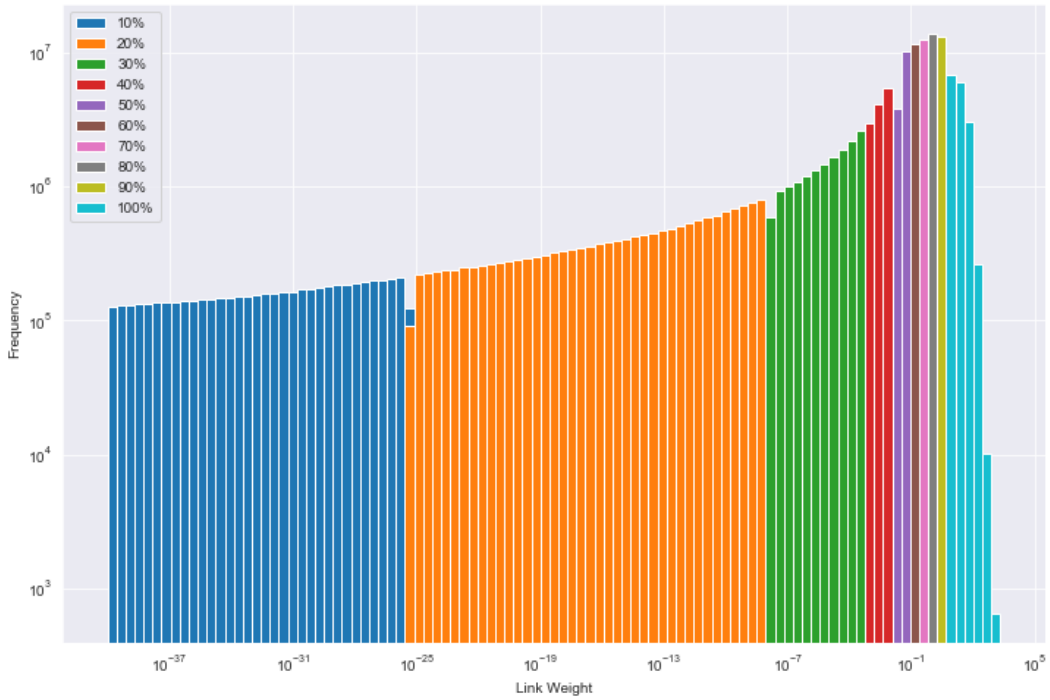


Figure 2.4 | **Activiy weight distribution in SD with color highlighted bins. Note that the x axis has been truncated in order to fit all the bins into one plot. The first class stretches to link weight values in the order of 10^{-283}.**

In the visualisations above and below, we have plotted the histograms with the bins color coded as to show in which percentile of the data the fall. This is also what the activity weight classes represent. Please note the logarithmic scale. Also, the weights that are 0 are not plotted as this would skew the plot.

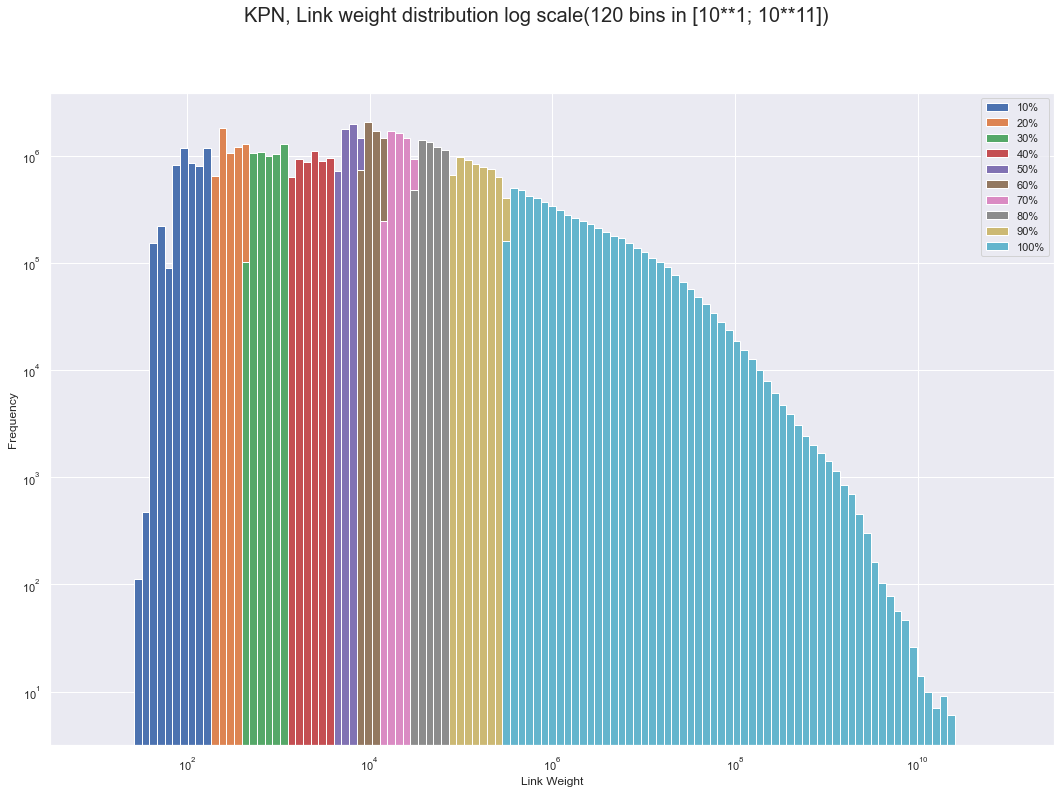


Figure 2.5 | **Activiy weight distribution with color highlighted bins.**

Table 2.2 Counts for binned link weights.

Bin number	**Bin edges**	**Value count**
0	0	532×10^6
1	$(0.000, 2.000 \times 10^2]$	5.95×10^6
2	$(2.000 \times 10^2, 4.680 \times 10^2]$	5.92×10^6
3	$(4.680 \times 10^2, 1.362 \times 10^3]$	5.93×10^6
4	$(1.362 \times 10^3, 4.443 \times 10^3]$	5.92×10^6
5	$(4.443 \times 10^3, 8.183 \times 10^3]$	5.92×10^6
6	$(8.183 \times 10^3, 1.517 \times 10^4]$	5.92×10^6
7	$(1.517 \times 10^4, 3.159 \times 10^4]$	5.92×10^6
8	$(3.159 \times 10^4, 7.877 \times 10^4]$	5.92×10^6
9	$(7.877 \times 10^4, 3.260 \times 10^5]$	5.92×10^6
10	$(3.260 \times 10^5, 2.284 \times 10^{10}]$	5.92×10^6

2.3.1 Activity rates

We have mentioned the sparsity aspect previously in this chapter as an important factor for building the model, so we will try to give more context on this using

Table 2.3 Counts for binned link weights of Stanford data.

Bin number	Bin edges	Value count
0	0	128×10^9
1	$(0.000, 4.981 \times 10^{-26}]$	16.2×10^6
2	$(4.981 \times 10^{-26}, 1.252 \times 10^{-8}]$	16.2×10^6
3	$(1.252 \times 10^{-8}, 6.451 \times 10^{-4}]$	16.2×10^6
4	$(6.451 \times 10^{-4}, 2.056 \times 10^{-2}]$	16.2×10^6
5	$(2.056 \times 10^{-2}, 1.075 \times 10^{-1}]$	16.2×10^6
6	$(1.075 \times 10^{-1}, 3.298 \times 10^{-1}]$	16.2×10^6
7	$(3.298 \times 10^{-1}, 8.433 \times 10^{-1}]$	16.2×10^6
8	$(8.433 \times 10^{-1}, 2.128]$	16.2×10^6
9	$(2.128, 6.664]$	16.2×10^6
10	$(6.664, 2.160 \times 10^3]$	16.2×10^6

activity of links as a descriptive term. Basically, due to the sparsity of our data, the edge sets of the two networks are far smaller than a simple cartesian product of the two bipartite nodesets. Here the term link density (or active links) comes in handy - when filtering only for active links we see that this is a small percentage of the total possible amount of links. For SD we have only **310041** active links, which is **0.4%** of the total possible links. For KD the number of active links is **524129**, which is equal to **42%** of total possible links. Throughout this thesis, we use a shorthand notation AR which we will define here:

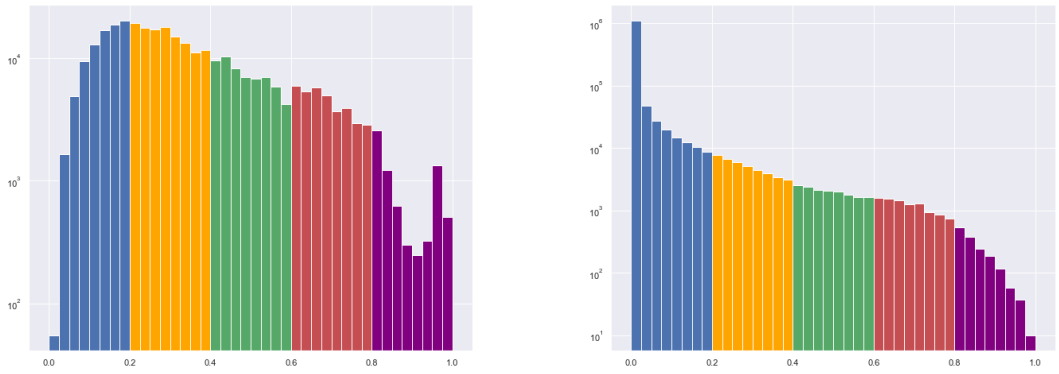
$$AR(link) \coloneqq \frac{1}{T}\sum_{t=0}^{T} act(t) \text{ — where } act(t) = \begin{cases} 1, & \text{if w}(link_t) > 0 \\ 0, & \text{if w}(link_t) = 0 \end{cases} \tag{2.1}$$

These active links are the only ones that are interesting for our thesis, as predicting links that have never been active is extremely simple, you just assign a link weight of 0 based on the historical data which is also always 0. Given this observation, we define the backbones of our networks using the active links. That is to say the backbone will include all the nodes of those links that have been active at least once during the timespan of the dataset. Another definition would be that the backbone only includes those links whose actvity rate is bigger than 0.

Keeping in mind that we want to design a simple model to predict the link weights, it is important we recognize that within this collection of active links there can be some major differences still. It is quite different to predict a link that may be active only once in all of the timesteps (a low activiy rate) compared to a link that is active most of the timesteps (high activation rate).

In our study we want to make a distinction between such links to see how the model deals with each case and hopefully extract the parameters that are most interesting for each category. As such we define 5 different classes of links in the backbone based on the activation rate. The first class covers the links that fall in

the $0-20\%$, i.e, a link $e \in AR_{20}$ if $AR(e) \in (0, 0.20]$, the second covers the links with AR between $20-40\%$ and so on ..., with the last class covering the most active links with AR of $80-100\%$. Below you can find an overview of the amount of links in each class for both networks:



(a) (b)

Figure 2.6 | **The link distribution per AR class using a logarithmic Y-axis.** *A) This is the distribution of SD links. B) This is the distribution KD links.*

Table 2.4 Counts for binned link weights of Stanford data.

AR	**SD links**	**KD links**
$(0, 0.20]$	85145	456610
$(0.20, 0.40]$	123110	40127
$(0.40, 0.60]$	59214	16196
$(0.6, 0.80]$	35420	9630
$(0.80, 1]$	7152	1566

From this we gather that most of the links are relatively less active within our backbone. Still we want to continue our experiments with our current setup so in other to facilitate meaningful results we decide to sample 1000 links from each AR class on which we will perform further experiments. However, it is much more valuable to predict the more active links in real world application, so that will be the focus when discussing the findings of our experiments.

We've done more exploratory work on the activities and their accompanying weights in both of the datasets, especially with the intention to put them in the proper real world context along with the context of the time period form which the data was collected (COVID-19). These insights can be found in the appendix A.2.

Chapter 3

Related Work

Temporal link prediction is increasingly becoming a more explored domain in computer science. This chapter discusses technologies related to the prediction of links and weighted (bipartite) graphs. It is divided into 2 sections: temporal and bipartite focused methods. Traditional methods for network prediction are omitted but can be found in appendix B, as these are the most widely studied methods that have existed for some time. Temporal methods incorporate the temporal dimension that accompanies real world data, adding a layer of complexity while also providing more insight. Then lastly, the bipartite section discusses more intricate methods that are designed with bipartite networks in mind. Within each section we will start off with link prediction methods and slowly proceed towards work directed to explore link weight predictions explicitly.

3.1 Weighted networks and link prediction

Most of the traditional methods (appendix B) are per definition intended to be used with simple undirected and unweighted graphs for simplicity's sake. However, with some modification they can be applied on weighted and directed networks. For our thesis we are mostly interested in the weight aspect and not so much the directionality as our preproccesing produces networks that are undirected. The weight is often treated as the significance of a given link and in some cases it is even used to provide an indication if a link exists or not. Given a certain threshold, if the predicted weight will exceed the threshold the model assumes it will exist and vice versa. A very popular approach is to equate the weight with similarity measures such that the methods mentioned in B.1.1 can be adopted. Bütün et al. [5] proposed a novel topological similarity metric that exploits temporal and weighted information in directed networks, which reportedly provide improvements of accuracy. Their approach is extract all possible triad pattern features and combine them with various topological similarity metrics discussed in the appendix (Common Neighbours, Jaccard Index, AA index, Ademic/Adar Index and Resource Allocation Index). These features and metrics were ingested by a supervised learning model to predict missing links.

A very cool approach is showcased in Hou and Holder [18] where the authors create Model S, a generic deep learning framework that performs link weight prediction based on node embeddings. One of the contributions is the decoupled learning process of node embedding learning and link weight prediction learning. They leverage the authors previous research where Model R was introduced and provide the possibility to use different general purpose embeddings (which we covered earlier in B.1.5) with the original link weight prediction which is based on the well-known deep learning techniques of back propagation [35] and stochastic gradient descent [24].

3.2 Temporal link prediction methods

The previous section covered fundamental methods, modified for link prediction in weighted networks. They form the basis on which extensions of these algorithms are developed to apply in more complex scenarios. A very common and important extension is to include the temporal aspect of many networks that are rooted in the real world, which we will refer to as Temporal Link Prediction (TLP). A good example would be social networks where research shows that the networks are constantly evolving and change is imminent over a given timespan. By leveraging temporal information we can predict links using historical data which is very relevant for recommender systems or information/infection spread in networks. We will discuss existing literature that tackles this subset of link prediction problems to provide a solid basis for us to build on.

Dunlavy et al. [10] utilize matrix and tensor techniques in order to compress several network snapshots into a unified matrix and compute link scores using truncated Singular Value Decomposition (SVD) and extended Katz methods. The tensor part is instrumental in computing link scores using heuristics and temporal forecasting but this comes with substantial computational burden. Additionally, [14] proposed a model grounded in latent matrix factorization, integrating content attributes with structural data to illustrate the temporal dynamics of links in various networks.

Continuing on with GCN's, we will discuss two very nice example papers that align somewhat with the goal of this paper. Both Zhao et al. [46] and Ge et al. [15] used GCNs to predict the traffic demand of a traffic network which is similar to the problem explored in this thesis. Both problems have the temporal (e.g. peak hours), as well as the spatial dependence (adjacent geographical locations). Therefore these networks are called spatio-temporal networks. In our problem case, it will be interesting to analyze if the network also contains any spatial dependence. It could be reasoned that the flow transmitted from equal locations (POI's or cell towers) could influence each other.

Zhao et al. proposed a Temporal-Graph Convolution Network (T-GCN) model, which consists of two parts. First, an GCN is used to capture the topological information. Then, the Gated Recurrent Unit (GRU) is used to capture the dynamic change of the network over time. Ge et al. proposed a model called Graph Temporal

Convolutional Network (GTCN). Besides the traffic data, this model can handle external data, such as social factors (e.g., holidays, peak hours), road network structure (e.g., bridges), and points of interest (e.g., schools, restaurants) which can influence the traffic volume of the roads nearby. Such external data is not available for our datasets and thus will be omitted in this study. However, this approach can leverage beneficial external factors for making predictions.

Lei et al. [25] focus mainly on weighted graphs. This approach is named Graph Convolution Network - Generative Adversarial Network (GCN-GAN) and uses GCN and GAN to predict the weighed links. The GCN is applied in the first step to capture the topological structure of each snapshot, similar to Zhao's and Ge's approaches. This is followed by leveraging an LSTM, which captures the evolution of the dynamic weighted network. Finally, GAN is used to generate high-quality weighted links. In short, GAN is often used to deal with the sparsity and wide-value range of observed link weights. Compared to encoding methods such as DeepWalk and Struc2vec (hich have made significant strides in graph embedding sphere), GNNs can improve the richness and depth of the embeddings Zhou et al. [48]. These methods are trained to capture non-linear transformation of the dynamic networks over time. However, this also increases the complexity of these models.

3.2.1 Weighted link prediction in temporal graphs

Most existing Temporal Link Prediction (TLP) methods merely focus on the prediction of unweighted graphs. There are some approaches we mentioned earlier, that can provide the likelihood of a link appearing in future time steps, they are not modified to predict the corresponding link weights. There exist some robust methods that are rooted in classifier and matrix factorization with the use of adjacency matrices of the networks, but the results are rather lackluster. Concretely, a large section of these methods are usually optimized as such that they minimize the reconstruction error between the training ground-truth and prediction result. However, this objective fails to effectively address the challenges posed by real-world datasets used in weighted TLP problems, such as wide-value-range and link sparsity issues.

Wide-Value-Range Issue. In weighted networks, link weights may have a broad value range, e.g., [0, 1000000]. Additionally, there might be a significant number of links with small weights, e.g., $1 \times 10-10$, which cannot be dismissed. However, the error minimization objective is sensitive to large link weights which may lead to being unable to distinguish the scale difference between small weights. For instance, the disparity in scale between (1, 2) is greater than that between (990, 1000), despite the latter producing a larger error.

Sparsity Issue. In a network, there is often substantial difference between small and zero weights, especially in their physical meaning. Zero often indicates that there is no link while a small weight implies the existence of a link, albeit with a small (maybe insignificant) weight. The structure of some real-world often exhibits sparsity, with a significant portion of zero weights. Given that error minimization objectives tend to be sensitive to larger weights, methods relying on such optimiza-

tion strategies struggle to differentiate between zero and small weights. This is somewhat alleviated by more advanced approaches such as GCN-GAN we discussed earlier.

Both of these properties play a role in the networks selected for this thesis. As such we must be aware of potential pitfalls and construct our model in a way that can overcome these issues or at least take the produced results with the proper context of lower performance if we can't rely on a more advanced TLP framework due to resource scarcity.

3.3 Bipartite Link Prediction Methods

Up until this point we have been focusing on link prediction methods in unipartite networks, where links can exist between any pair of nodes. The networks we deal with in this thesis are of the bipartite class, hence we will review the link prediction problem for that specific context.We found a multitude of approaches in the literature, mainly because the solutions are some combinations of previously discussed methods. We think that this logical progression from the straightforward link prediction problems to specific and more complex scenarios will shed some important clues for the problem in this thesis.

The first category of methods is the projection-based approach. The approaches within this category create a unipartite network from the bipartite network. This created network is called the projected graph. [13], [40], [1] and [27] all proposed methods (in different contexts) which solve the link prediction problem by relying on projecting the bipartite network into a unipartite network. Based on this unipartite graph, the models try to predict which links will occur in the future which did not exist in the past. Lu et al. go as far as to apply GNN based learning methods onto the weighted projected network in order to predict disease prognosis. Predicting new links in a network is extremely popular and essential in medical and e-commerce fields. However, in this project predicting new links is not the desired outcome. Therefore we leave the projection based approach for now.

Kunegis et al. [22] examine the link prediction problem in bipartite networks and note that most common neighbour based approaches (e.g., Common Neighbours, Adamic/Adar, Resource Allocation, etc.) are not suitable for such networks. This limitation arises due to the fact that adjacent nodes typically belong to different clusters and are only connected through paths of odd lengths. As elaborated in the appendix B.1.1, it is demonstrated that only the Preferential Attachment similarity-based approach is applicable to these networks in its native form, as it considers the degree of neighbours. However, this is also the worst performing of the similarity measures.

Moving one step further, there were studies that focused on embedding the network structure and using both neighbour and path-based features along with machine learning techniques (e.g., Naive Bayes, Support Vector Machines, Random Forest) to learn on the features and produce a predicted structure at a future

timestep. Unfortunately, the authors designed the study around merely predicting the existence of new links.

Bipartite networks are common in the recommender system space, and while the ultimate goal is clearly different from this thesis, a lot of the approaches share similarities. Yoon et al. [42] have achieved positive results by extracting bipartite network embeddings using random-walks and predicting future embeddings using Kalman filtering method. The embeddings was a suitable approach to deal with the sparseness in their datasets, which is encouraging. In their study they did consider the weight of the links as the purchasing power so it was included in the embeddings, as well as the temporal dimensions as they produced these embeddings at different timesteps.

The difficult part was finding literature that represents the setup of our study. Almost all the research using temporal bipartite graphs is rooted in predicting the existence of links. The usual metric is then often AUC (Area Under ROC Curve). This is a good representation of the predictive power of an algorithm but is rooted in binary classification of positive and negative samples, e.g, presence of a link. The weight is never considered, something to note for our methodologies moving forward. From this we conclude that the problem of predicting a weighted network structure including weight, especially in bipartite graphs, is highly under-researched and means we are providing some novel insight with our setup of the experiments.

3.3.1 Summarizing

From this hefty summary of literature, one issue sticks out as a thorn. Namely, the fact that temporal link prediction specifically in bipartite networks mainly focuses on predicting new links. This could be explained by the prominence of user-item networks when research is classified as focusing on bipartite networks. In these networks, it is of high value knowing when a user will connect to an item. Another interesting observation is that applying some of the methods discussed in the previous section on bipartite networks will probably not result in the expected prediction accuracy as proved by Jin et al.. Therefore, before selecting and implementing a model for the task at hand, it is important to know which data will be used and how it is used.

To conclude, in this chapter several types of methods for (weighted) temporal link prediction have been discussed. The complexity of these approaches differs significantly. There is a glaring preference amongst the scientific community for simple solutions that can eventually be tweaked and enhanced to complete more complex tasks. By focusing on the basics of the data and networks at hand, we can more fundamentally describe network evolution and perform various experiments to acquire isights into the most valuable features of such networks.Further down the line, one might fancy implementing the more complex models and compare the performances to the simpler approaches to gain a sense of efficiency. Because the performance of the models depends highly on the network structure, it is impossible to conclude which models are most suitable beforehand.

Chapter 4

Predictive Algorithms

In this section, we introduce our methodology, which enables link activity weight prediction in weighted temporal networks and helps facilitate the exploration of the relationships between activities of links in a network G . Specifically, we aim to understand to what extent a link's activity weight at a given time step is determined by the activities of other links as well as its own activity at the previous time step.

4.1 Lasso Regression

Our approach is applicable to a generic temporal network with N nodes and M links (node pairs with at least one non-zero link weight) whose activities are recorded within a time window $[1,T]$. The activities of the M links are recorded by a $M \times T$ matrix X. The state or activity of link i at time $t+1$ is $x_i(t+1)(t \in [p,p+L-1])$, which takes on a value $w \in W$ (with W being the set of all observed link activity weights) when link i is active, and equals 0 otherwise. We assume that the activity of link i at time $t+1$ is a function of the activities of all the links at time t, i.e.,

$$x_i(t+1) = f_i(x_1(t), x_2(t), \ldots, x_M(t)) \tag{4.1}$$

The mapping function f_i is unknown and unique to each link. It can be inferred from the activities of all links, i.e., $[x_i(p), x_i(p+1), \ldots, x_i(p+L)]$ where $i \in [1,M]$ within a time window $[p,p+L]$, and denoted as $f_i^{p,L}$. In total, we construct L training samples for each link i based on the temporal network observed between $[p,p+L]$: we use link i's state at each time step $t+1 \in [p+1,p+L]$ as the target and the corresponding features are the activity weights of all links at time step t. The training samples for a nodepair i is expressed as the set $D_i(p,L)$:

$$D_i(p,L) = \{x_i(t+1); x_1(t), x_2(t), \ldots, x_M(t)\}_{t=p}^{p+L-1} \tag{4.2}$$

The objective of this predictive algorithm is to learn the function $f_i^{p,L}$ from the training set $D_i(p,L)$. By learning this function, we can determine the extent to which $x_i(t+1)$ can be estimated by the activity of each link at time t.

Lasso Regression is one such algorithm. It assumes f_i to be a linear function

$$x_i(t+1) = \sum_{j=1}^{M} x_j(t)\beta_{ij} + c_i \tag{4.3}$$

We want to minimize the following with respect to β_i:

$$\min_{\beta_i}\{ \sum_{t=p}^{p+L+1} (x_i(t+1) - \sum_{j=1}^{M} x_j(t)\beta_{ij} - c_i)^2 + \alpha \sum_{j=1}^{M} |\beta_{ij}|\} \tag{4.4}$$

where L is the number of training samples, M is the number of features which is also the number of links used for prediction, c_i is the intercept and $\beta_i = \{\beta_{i1}, \beta_{i2}, \ldots, \beta_{iM}\}$ are the regression coefficients of all the features for link i. A large coefficient β_{ij} indicates that feature $x_j(t)$ exerts influences or in other words, has a significant share in determining the target value of $x_i(t+1)$.
We use $L1$ regularization, which adds a penalty to the sum of the magnitude of coefficients $\sum_{j=1}^{M} |\beta_{ij}|$. The parameter α controls the penalty strength. The regularization forces some of the coefficients to be zero and thus lead to models with few non-zero coefficients (relevant features). If α is zero, Lasso Regression reduces to the classical linear regression algorithm. Given a training data set $D_i(p, L) = \{x_i(t+1); x_1(t), x_2(t), \ldots, x_M(t)\}_{t=p}^{p+L-1}$, the coefficients $\beta_i(p, L)$ of the Lasso Regression model for each node i can be learned.

4.2 A baseline to compare against

In order to gauge the performance we also need some kind of a baseline algorithm to provide the necessary context outside of the absolute error values. As we deal with quite a novel problem statement in this thesis, there aren't any clear favorites that we can take from existing literature for example. There are other methods that can be adapted to our problem definition, which we briefly cover in 3, but such methods are quite complex and can take a whole thesis to implement and explore. Hence for our baseline method, we keep to the mantra that we have mentioned previously - to keep things as simple as possible. We decide to predict the weight of a link by sampling from the overarching distribution of the link weights in our backbone network. Important to note that we include the absence of a link (the 0 value) in the distribution and given our sparse networks the predicted value will be heavily skewed towards 0.

This is a choice we made to not add unnecessary complexity which in turn could make it more difficult to trace and explain the results we will get from our experiments. However, other choices could definitely be made which could provide a more realistic comparison to the chosen predictive algorithm, which in our case is LASSO.

Such choices are to exclude the weights that are 0 when sampling from the distribution or fixate the probability of 0 occurring, for example based on specific

heuristic or more in general, the behaviour we want our baseline to exhibit. That is to say if we the compared algorithm will never predict the absence of link weight, it doesn't make sense to have a baseline that will, especially if that is also the median predicted value.

Another interesting limitation for this baseline is to sample only from the distribution of link weights observed in the backbone at the timestep we want to predict, or in several timesteps before which could act as an active memory window. As you might guess there are a lot of ways to design and constrict such a baseline prediction to achieve desired behaviour. For our case the simplest way is enough as first and foremost we want to see if an algorithm like LASSO is capable to provide an sizable performance boost at all, as this is a novel way as far as we know to tackle such a problem.

4.3 Training and test data

In each time interval $[p, p+L]$, where p is in the range of $[1, T-L-1]$, the observed temporal network is treated as a training set.The model function learned from this dataset is then evaluated for its predictive performance on the temporal network observed at $p+L$. When applying Lasso regression on the outline framework, one can use the learned the coefficients $\{\beta_i(p,L)\}, i = 1,2,\ldots,M$, from each training set $D_i(p,L)$ to predict the activity and the associate weights of the links in the test set $Y_i(p,L) = x_i(p+L+1); x_1(p+L), x_2(p+L), \ldots, x_M(p+L)$. In total, $T-L-1$ training sets, together with their corresponding test sets, will be considered for each of the two bipartite temporal networks we employ in our experiments.

4.4 Neighbouring link selection

In a perfect world one would use all the possible links from a temporal network during the Lasso regression runs to produce unbiased results and shed light on the possible relationships between all links. But due to the size of the temporal networks this would take much more processing power than there is available to us. It also might be a little overkill in the sense that Lasso is designed to limit the regression to handful of links (this depends on the α parameter). The results would omit most of the features used in training. Intuitively it also makes sense to look at a selection of all the possible links based on their characteristics like the topology for example. It probably doesn't make sense to predict a link with values of other links that topologically far away from the target.

Because of the above we decided to design a strategy for selecting link activity weights that would be used as features while training the Lasso algorithm, to limit the computational overhead. We propose a selection based on the concept of neighbouring links in the aggregated network. Recall that the aggregated network has a link between the nodes if that link was active (had a nonzero link weight) at least once during the total time span of the temporal network. Using this we find all the

links that have at least one node in common with the target link (which we will call neighbours) and use those as the features. If no such links exist we skip the prediction of that link entirely.

The neighbouring quality is, in essence, just one form of many measures of similarity in the topological dimension. In 3 we mentioned various other studies that successfully extracted useful information for prediction based other measures of similarity. To optimize our choice, we experiment with various similarity measures within the subset of neighbouring links.

We've also added two measures that are not based on similarity but rather maximizing a certain property. These properties are the total activity weight value and the total activity frequency of links. As these are innate, physical properties of links and in turn the networks, we wanted to see if neighbouring links that maximize these values could be a better predictor in our sparse scenario.

The intuitive explanation for the activity weight is that links that have a larger value over the total timespan are transmitting more flow or information within the network. As such these links contain more information and can be more useful for our predictions. Similar explanation also works for the activity frequencies, the higher the activity of a link the more timesteps with a nonzero activity weight for a particular link. Whenever you want to predict a link that is also of higher activity, such links should provide a good reference. Obviously this method might not perform too well when predicting lower activity links.

Table 4.1 Different link similarity measures explored.

Measure	**Short description**	**Equation**
Pearson correlation	Most common form of similarity, measures the linear correlation.	$COV(x,y)/\sigma_x\sigma_y$
Cosine similarity	The distance between the link activity weights in Euclidean space	$\frac{x \cdot y}{\|\|x-y\|\|}$
Euclidean distance	The distance between the link activity weights in Euclidean space	$\|x-y\|_2$
Highest total weight	The sum of the link activity weights over time	$max_w \sum_{t=0}^{T} w(link_t)$
Highest activity	The total activity over time	$max(AR(link))$ 2.1

In 4.1 you can see the total overview of measures we experimented with. One note on the distance based measures, we've settled on the euclidean and cosine variants as they are commonly found in literature and simple and lightweight to compute, but there are a handful other variants like the cityblock, mahalanobis and hamming

distance, all with their own usecases. The same applies to the correlation based metrics, while the pearson coefficient is most common there are others like Spearman, Kendall and Kruskall as well as the possibility to focus on cross-correlations. For our example we deal with the cross-correlation with lag 1, by offsetting the target link weight timeseries by 1 step into the future.

In the second part of the table we also list several measures that were explored during the thesis but eventually left out for the final experiments and results. These boil down to 2 main reasons: 1) These methods are more computationally expensive than the previous measures. 2) These measures perform best in special conditions, such as time offset patterns or comparing links with different time granularity. We've alleviated most of these in the preprocessing steps in 2 so this will only add complexity to our setup. One note for Node2Vec specifically, we've found good results using this technique but due to it being a deep learning model, which are essentially black-box models, we decided to leave it out of this thesis.

All in all, there are many other ways to create embeddings and find similarity of time-series based data, the methodologies listed in this chapter are only mentioned to provide a brief overview of the common measures. For different problems there are more advanced and specific measures like the Ratanamahatana et al. [34]. A good resource on similarity measures in link prediction is written by Lü and Zhou.[28]

With this setup, the intial collection of neighbours for a given link could vary significantly, if we recall the global degree values for the networks. To get a feeling for how this can impact the predictions, we construct different train and test sets where we conditionally sample 1000 links for prediction from each activity class, selecting only those which have sufficient neighbouring links and discarding those that are below a certain threshold. Considering that the smallest average degree of the two networks is 19.59 we consider a scenario where we employ a neighbour threshold of 20, with it being incremented to 60 and 100 in subsequent experiments.

Chapter 5

Model Evaluation

In this section, we will discuss the results of our experiments with the Lasso model.

5.1 Evaluation metrics

But first a little bit about the evaluation metrics we will use to score our models. Because we treat this link prediction problem as a simple regression problem we can use classic metrics as pertains to regression. Most of these metrics are quite common because they are simple and easily applicable, but place the burden on the interpreter to interpret them correctly. Here is a rundown:

- (Root) Mean Squared Error or (R)MSE, quantifies the average prediction error made by the model per observation. Mathematically, the RMSE is the square root of the mean squared error (MSE), which calculates the average squared deviation between observed actual outcomes and predicted values by the model. When the data consists of many outliers, then MSE could incorrectly represent the model's performance because the outlier errors would skew the evaluation. This is where taking a square root of MSE will dampen this effect, hence the popularity of RMSE.

$$MSE = \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n} \qquad (5.1) \qquad\qquad RMSE = \sqrt{MSE} \qquad (5.2)$$

- Mean Absolute Error (MAE), akin to RMSE, assesses prediction error. It quantifies the average absolute deviation between observed and predicted outcomes. MAE demonstrates robustness against outliers compared to RMSE, thus complementing it effectively for result interpretation. Additionally, its error scale matches that of the input data, making it more intuitive to understand the error's significance.

$$MAE = \frac{\sum_{i=1}^{n}|\hat{y}_i - y_i|}{n} \qquad (5.3)$$

- Accuracy, not a regression metric in nature, it does have the ability to provide a slightly different but still relevant insight for our specific problem. Because the model, predict floating values with a lot of decimal places, the predictions made by the model should be first discretized to one of the 10 classes. Then, the accuracy can be computed by 5.4, where I represents the Boolean indicator function which is equal to 1, if $\hat{y}_i = y_i$ is true, and 0 otherwise, where $\hat{y}_i$ represents the discretized prediction.

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n} I(\hat{y}_i = y_i) \tag{5.4}$$

5.2 Comparison to the Baseline

Before we proceed we want to set the baseline to which we will be comparing our model. As our model tries to be as simple as possible we wanted to compare it with even simpler methods. So for the baseline we chose to simply predict the link weights by sampling from the total distribution of the link weights. By the total distribution we mean the distribution of all link weights contained in all of the adjacency matrices of the respective dataset.

Table 5.1 Average baseline performance.

Dataset	AR	MAE	RMSE	Accuracy
SD	(0, 20]	0.602	1.974	0.904
	(20, 40]	1.334	2.800	0.763
	(40, 60]	2.396	3.721	0.548
	(60, 80]	3.026	3.957	0.332
	(80, 100]	3.057	3.677	0.135
KD	(0, 20]	0.318	1.405	0.942
	(20, 40]	1.576	3.182	0.700
	(40, 60]	2.645	4.100	0.501
	(60, 80]	3.968	5.177	0.312
	(80, 100]	4.637	5.461	0.150

Because we include the absence of link (weights of zero) in the matrices, sampling from the distribution will almost always result in a value of 0. This was intentional but can be tweaked depending on the desired properties, which we discussed in detail previously in 4.2. obviously the performance is very good in the bottom AR class as low activity frequency means a lot of zeros. The more nonzero weights we need to predict the worse the performance. Accuracy is a good metric to show this, as it drops down heavily when we are dealing with very active links. If we would macro-average this score per class i.e, calculate raw accuracy where each sample is weighted according to the inverse prevalence of its true class, we would see that the performance of all but the first AR class are similar. It would however be interesting

to see if our model can deal with the ever increasing error rate, despite the accuracy performance.

5.3 Hyperparameters

We decided to lay out our experiments and the results in such an order that we can provide an answer to a certain sub-question. In a way we are really tuning different (hyper)parameters to see what the significance might be on the overall performance. As we recall from 4.1, one natural hyperparameter is the alpha (α) parameter innate to the LASSO algorithm. It assigns a weight to all the input features (neighbouring links), with some being set to 0 i.e., not used at all. This is the shrinking parameter that dictates the rate of the penalization but in general the more features we provide the more non-zero weights given α is the same. We performed a grid search along five logarithmically spaced points within $[10^-3, 10]$ which consistently performed best for values of either 0.1 and 1, for every subset of experiments. Please not that this is not an extensive grid search and we can not with full confidence say that these values will perform the best in every scenario. What we do hop to highlight with this is that any future method relying on similar regression algorithms should properly explore a (semi-)optimal shrinking parameter for their specific data. Because this is a computationally expensive operation and we were only using a personal machine to run these experiments we decided to cut corners where possible and converge more quickly, by deciding to further zoom in between the aforementioned alphas. The eventual values for the alpha that we considered were: $[0.1, 0.25, 0.5, 0.75, 1]$. We calculate the optimal α based on the average error for a specific test set of 1000 links. This is done because of implementation specific considerations. In the sections of this chapter we will limit the shown metrics to the optimal alpha, while providing the complete results in the appendix. We will start this off by exploring the effect of the training sets we use to train our LASSO model, the so called history.

5.3.1 Training history

We settle on testing 3 different training set lengths that represent one, two and three weeks of data, which is 168, 336 and 504 individual timesteps of 1 hour respectively. We provide a simple visual explanation in 5.1.

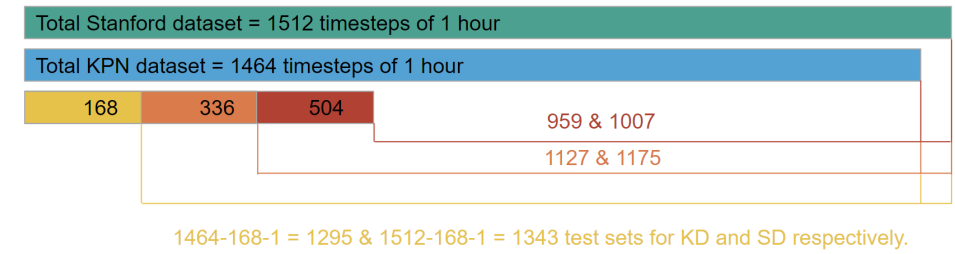


Figure 5.1 | **A visual guide to how the training and test sets relate to the whole datasets.** *Yellow block represents 1 week of training data, orange is 2 weeks and red is 3 weeks.*

Table 5.2 Comparison of mean performance for different training set lengths (20 link feature set).

Dataset	AR	1 week			2 weeks			3 weeks		
		MAE	RMSE	Accuracy	MAE	RMSE	Accuracy	MAE	RMSE	Accuracy
SD	(0,20]	0.805	1.741	0.662	0.691	1.588	0.696	**0.647**	**1.538**	**0.715**
	(20,40]	1.279	2.053	0.456	1.216	1.988	0.477	**1.182**	**1.962**	**0.487**
	(40,60]	1.435	2.087	0.352	1.396	2.059	0.358	**1.376**	**2.038**	**0.362**
	(60,80]	1.097	1.602	0.375	1.082	1.583	0.376	**1.072**	**1.572**	**0.379**
	(80,100]	0.935	1.383	0.439	0.926	1.372	0.441	**0.923**	**1.371**	**0.442**
KD	(0,20]	0.204	0.734	**0.918**	0.200	0.706	0.916	**0.199**	**0.694**	0.917
	(20,40]	1.896	2.595	**0.260**	1.798	2.490	0.221	**1.786**	**2.465**	0.205
	(40,60]	2.302	2.832	**0.158**	2.232	2.774	0.126	**2.223**	**2.757**	0.116
	(60,80]	2.269	2.808	**0.140**	2.247	2.787	0.124	**2.238**	**2.776**	0.120
	(80,100]	1.883	2.420	0.180	1.843	2.390	0.184	**1.840**	**2.387**	**0.185**

Here we decided to show off the whole range of our experiments in terms of the AR classes and it can be clearly seen that the performance is better than the baseline 5.1 in all cases except for the (20,40] AR class, where both datasets dip under the baseline's MAE. This class is still mostly sparse for most of the timesteps yet has increasingly more nonzero weights which the model struggles to predict. It is important to note that the model in general struggles more with outliers given the RMSE values being larger than MAE, due to the fact that RMSE is more sensitive to outlier weights.

Next if we isolate our attention on the performance progression along a bigger training set, we can see that the performance does improve in terms of the errors, the more training data we provide for the model, but this is in the order of 2×10^{-2} for MAE and 5×10^{-2} for RMSE at each level of training set length. Given that we want to predict different classes of weights instead of precise weights, these improvements seem negligible. Interestingly the accuracy for the KD is actually worse the more training data we provide, although the errors do shrink. This indicates that the model is coming closer to predict the true weight class on average yet has less correct predictions. This could be due to the greater variance of link weights which cause the regression to produce more nuanced weights which fall in the wrong weight class in the discretisation step at the end of the prediction.

When performing further the grid search experiments, we will notice that the spread of errors between choices of the shrinkage parameter is bigger than the memory length difference although not as consistent. One trend we can pick up is the fact that with bigger training sets, performance improves for stricter shrinkage settings, while 1 week of memory tends to prefer looser shrinkage. That has an intuitive explanation due to the amount of data points available to train on. Please remember our earlier remark on the validity of the chosen shrinkage parameter alpha. With a more extensive grid search, bigger improvements could be had in general for all experiments we performed. It is also important to bring up the notion of our dataset constraints in terms of the total timespan we can use, which hampers our ability to really pinpoint where exactly the performance increase might fall off in term of the error statistics and maybe even increase in accuracy. The caveat obviously being that the bigger the training set will be the higher the space, but also time, complexity will be.

5.3.2 Feature set length

For the next few experiments we will be looking at how the different strategies to select the features (neighbouring links), as well as, how the overall size of this set will impact the performance. We will start off with the latter. In the previous experiment we shown the optimal values for a feature set of 20 neighbouring links chosen randomly. Let's see how it compares to the same setup but now evaluating links that have 100 neighbouring links as features. This way we can also test if our hypothesis earlier holds for the 100 link feature set.

Table 5.3 Comparison of mean performance for different training set lengths (with 100 link feature set).

Dataset	AR	1 week			2 weeks			3 weeks		
		MAE	RMSE	Accuracy	MAE	RMSE	Accuracy	MAE	RMSE	Accuracy
SD	(0,20]	0.800	1.695	0.661	0.677	1.539	0.700	**0.677**	**1.539**	**0.700**
	(20,40]	1.207	1.917	0.476	1.134	1.844	0.495	**1.134**	**1.844**	**0.495**
	(40,60]	1.310	1.929	0.380	1.266	1.889	0.398	**1.266**	**1.889**	**0.398**
	(60,80]	0.991	1.460	0.410	0.964	1.429	0.426	**0.935**	**1.407**	**0.435**
	(80,100]	0.800	1.214	0.492	0.783	1.198	0.504	**0.767**	**1.183**	**0.509**
KD	(0,20]	0.204	0.734	0.918	0.198	0.719	0.919	**0.191**	**0.698**	**0.922**
	(20,40]	1.896	2.595	0.260	1.852	2.552	0.263	**1.820**	**2.524**	**0.268**
	(40,60]	2.302	2.832	0.158	2.251	2.770	**0.167**	**2.228**	**2.759**	0.162
	(60,80]	2.269	2.808	0.140	**2.219**	2.753	**0.143**	2.220	**2.747**	0.139
	(80,100]	1.883	2.420	0.180	1.842	2.375	0.184	**1.838**	**2.369**	**0.185**

As we can see, the prediction quality overall has improved a bit but the differences are rather minuscule. It is important to consider the optimal alpha impact on these results as well. More feature links lead to a smaller error, given the shrinkage parameter is strict enough. The looser this parameter gets, the better the results get for lower feature links (20) as we can see in 5.2. Otherwise the bigger feature set is the better performing strategy, although obviously the complexity is increased.

Regarding the KD results, we see that the accuracy of the 1 week training set is being overtaken by the 2 week training set results in the 40% to 80% range and 3 weeks in other cases. Obviously the performance still is quite close with the largest training set providing us the best results in terms of the error metrics. In case of SD everything points in favor the larger training set experiments. We conclude that increasing training set has positive impact on the performance, albeit with a complexity cost to pay for performance. Further experiments we will show, all have a training set of 2 weeks unless stated otherwise. This is because on a personal machine where these experiments were executed, the gain in prediction quality wasn't worth the increased ruinning times.

In further experiments 2 weeks training set has been chosen as the performance is comparable (and sometimes better) than the 3 week case while being less resource intensive. We will also focus on the higher AR classes, as these are the most valuable to predict in the real world scenarios we described in 2.

5.3.3 Feature set selection

Up until this point the feature sets where just randomly sampled based on the neighbours of the target link. In 4.4 we have mentioned other strategies. Below you will find an overview of these strategies for the training set size of 2 weeks and a feature set of 20 links.

Table 5.4 Comparison of mean performance for different feature selecting strategies (2 weeks).

Dataset	AR	Random			Max total AR			Max total weight		
		MAE	RMSE	Accuracy	MAE	RMSE	Accuracy	MAE	RMSE	Accuracy
SD	(60,80]	1.082	1.583	0.376	**1.034**	**1.540**	**0.405**	1.047	1.562	0.402
	(80,100]	0.926	1.372	0.441	0.926	1.375	0.443	**0.923**	**1.370**	**0.444**
KD	(60,80]	2.247	2.787	0.124	2.234	2.784	0.136	**2.224**	**2.778**	**0.141**
	(80,100]	1.843	2.390	0.184	1.837	2.392	0.185	**1.822**	**2.377**	**0.186**

Table 5.5 Comparison of performance for different feature selecting strategies contd. (2 weeks).

Dataset	AR	Euclidean			Pearson correlation			Cosine		
		MAE	RMSE	Accuracy	MAE	RMSE	Accuracy	MAE	RMSE	Accuracy
SD	(60,80]	1.140	1.650	0.339	1.146	1.657	0.336	1.127	1.635	**0.354**
	(80,100]	0.959	1.426	0.429	0.960	1.424	0.426	0.994	1.447	0.408
KD	(60,80]	**2.159**	**2.727**	0.140	2.181	2.741	0.133	2.167	2.740	**0.151**
	(80,100]	1.803	**2.355**	0.189	**1.802**	2.355	**0.190**	1.804	2.357	0.189

From these we can clearly see that for the higher AR class links, strategies that incorporate physical properties of the neighbouring links tend to perform better, atleast for SD. Here both the total activity as well as it's weight come out on top. As far as KD goes, total activity weight is well performing approach in general for this dataset (especially when also considering the lower AR classes, not shown in the table 5.5). But the actual best performance comes when selecting neighbouring links that have either the lowest euclidean distance or the highest correlation to the activity weight vector of the target link.

While there are winners and losers in this exercise, we must also keep the bigger picture in mind. The improvements are numerically quite modest. It's hard to claim that the performance significantly impacts the eventual predicted activity weight in most cases, at least not for the current setup and chosen datasets.

Having said that, we still want to continue with showcasing other experiments. Let us cover the comparison between the random sampling base strategy and those of selecting based on the activity, namely neighbouring links with biggest weight and AR, as these were the best performing strategies for SD and still quite good for KD but now selecting a bigger feature set of 100 links.

Table 5.6 Comparison of mean performance for different feature selecting strategies (100 links, 2 weeks).

Dataset	AR	Random			Max total AR			Max total weight		
		MAE	RMSE	Accuracy	MAE	RMSE	Accuracy	MAE	RMSE	Accuracy
SD	(60,80]	0.964	1.429	0.426	0.933	1.413	**0.442**	**0.933**	**1.407**	0.439
	(80,100]	0.783	1.198	0.504	0.778	1.199	0.508	**0.771**	**1.187**	**0.509**
KD	(60,80]	2.219	2.753	0.143	2.202	2.748	0.147	**2.194**	**2.745**	**0.148**
	(80,100]	1.842	2.375	0.184	1.835	2.379	0.185	**1.822**	**2.368**	**0.186**

Overall we see similar behaviour for SD: the more active the links become the better the other strategies become. Middling AR classes lend themselves more towards selecting features based on the top AR neighbours while in the upper AR classes, selecting based on top weight is best. As a result, we see that LASSO is able to predict the correct activity class of the link in question with roughly 50% accuracy. But we also see that performance is very similar to the random selection strategy. The one thing both strategies always have in common is the inclusion of the previous timestep of the link we are tryiung top predict. We will expand on this in the next section 5.4. For KD, the copnclusions aren't too different. The top weight selecting strategy is always just a bit better then the random and top AR ones. The overall accuracy for this dataset is quite poor however, a reoccurring observation following from all the experiments we have shown.

5.4 Discussion

In order to provide more context to the raw error and accuracy metrics, we decided to keep track of the internal weights of the LASSO model which could help us explain how the features are contributing to the end result. For this we have created a list of metrics:

Since we saw that using a feature set of 100 neighbouring links based on those that have the highest total activity weight, leads to a slightly better accuracy and smaller errors we will focus on this configuration from now on, while still using 2 weeks of training data and the optimal alpha values explored earlier in this chapter.

Lasso is also quite strict (in the optimal configuration) in terms of involving neighbouring links as om average it only assigns a coefficient weight to 12 and 9, for links of their respective AR classes, selected from the 100 link feature set. It seems that a lot of information is not really related to the target link's activity weight. Furthermore, we can conclude that the higher the activity rate is of a link, the higher the influence of it's past timestep activity weight becomes. In the highest AR class, over 50% of the timestamps the links past weight had the highest lasso coefficient. When looking at the actual Lasso parameter weight maximum value is more than twice the size of the average parameter weight assigned to the link we are predicting.

Table 5.7 Context metrics for results.

Measure	Short description
Number of nonzero Lasso coefficients	Equal to the amount of features lasso selects to fit the regression
The value of the TL coefficient	The weight Lasso assigns to the past data of the TL itself (feature). This helps to track the importance of the TL as a feature.
The maximum of all the Lasso coefficients	The maximum weight of the coefficients. This gives more context to the TL coefficient weight and gives a bit more insight into what features were most important in the regression.
% of timesteps TL is the max coefficient	Using the above 2 metrics we can track how often TL was the most important feature with the fit.

Table 5.8 Lasso weight based metrics for SD (100 links).

Measure (averaged)	$AR(60,80]$	$AR(80,100]$
Number of nonzero Lasso coefficients	11.807	8.642
The value of the TL coefficient	0.042	0.070
The maximum of all the Lasso coefficients	0.197	0.167
% of timesteps TL is the max coefficient	0.384	0.533

In other words, the links past weight history always plays a role in the prediction but may not be the strongest predictor on a case by case basis.

Table 5.9 Lasso weight based metrics for SD (20 links).

Measure (averaged)	$AR(60,80]$	$AR(80,100]$
Number of nonzero Lasso coefficients	6.132	5.085
The value of the TL coefficient	0.166	0.204
The maximum of all the Lasso coefficients	0.291	0.278
% of timesteps TL is the max coefficient	0.487	0.642

Compare this to the 20 link LASSO numbers and we can make an interesting observation: The extra links selected by LASSO when we expand the feature set to 100 links, seemingly increase the regression performance. Even though, in 20 neighbouring links case, we have more emphasis put onto the historic activity weights of the target link (TL) itself, the inclusion of more neighbouring activity weights is beneficial to the prediction. This is what inspired us to delve deeper into the neighbouring link selection process, more on this in 6

Flipping over to KD, we see a bit of a different story. First of all, here Lasso

Table 5.10 Lasso weight based metrics for KD.

Measure (averaged)	$AR(60,80]$	$AR(80,100]$
Number of nonzero Lasso coefficients	24.011	18.418
The value of the TL coefficient	0.270	0.290
The maximum of all the Lasso coefficients	0.281	0.297
% of timesteps TL is the max coefficient	0.879	0.897

includes a little more neighbouring predictor links for the regression. The rest of the metrics are univocally in agreement on the situation for this dataset. The past timestep activity weight of the link we are trying to predict is the most influential in close to 90% of the timestamps. The value of the parameter weight is really close to the maximum average value as well, meaning that the reliance on the prediction is heavily inspired by a link's past observations. All the extra neighbouring links don't contribute much even though we acknowledge that for this dataset Lasso involves more links.

Chapter 6

Community Structure

From our preliminary experiments we found that there is little benefit to selecting feature links based on similarity to it's neighbouring links or innate weight and activity rates. The random strategy is dominant in most cases. Given all the literature that we discussed in the 3 it is possible that we need to extract the similarity in a different way. A working strategy seems to be to include structural information of the networks. One popular approach is to group the links (or the nodes of links) based on some factor. There exist plenty of community structure discovery algorithms, both for simple [11] and bipartite graphs [6][45] specifically. Fortuitously, a fellow Phd student from the same research group had already explored one of such methods. A good overview/survey is explored by Fortunato and Hric [12],so for the interested reader we recommend to follow up there, as we will merely explore the specific algorithm we ended up using, which is routed in modularity optimisation proposed by Newman [31]. While this was intended for unipartite graphs, we will show how it is possible to apply the method onto a weighted temporal bipartite graph, which is the class of networks we are dealing with in this study.

6.1 Community detection in projected networks

Community detection (CD) algorithms intended for static unipartite graphs can be applied to the static projected graphs of a bipartite graph, as it is also unipartite. When adding the temporal dimension, one could simply apply the CD algorithm at each time step, on a specific projected graph to detect the communities. Let's consider an undirected weighted graph G that is composed of only one type of nodes. It can be represented by a weighted adjacency matrix A. Given a weighted graph, if we partition all the nodes into non-overlapping communities, the quality of this community partition can then be measured by the modularity

$$Q = \frac{1}{2L} \sum_{i,j} [A_{i,j} - \frac{k_i \cdot k_j}{2L}] \delta_{c_i,c_j}, \tag{6.1}$$

where $k_i = \sum_j A_{i,j}$ is the sum of weights of all the links that are connected to

the node i, also called the node strength; c_i is the community label to which the node i belongs; the Kronecker delta $\delta_{c_i,c_j} = 1$, if $c_i = c_j$, and 0 otherwise; at last $L = \frac{1}{2}\sum_{i,j} A_{i,j}$ represents the total weight in the network. In essence, the modularity of a partition aims to describe the extent to which the weight of links within each community is bigger than the weight of those between communities. The modularity $Mod(G)\epsilon[0,1]$ of a graph is the maximal modularity that could be obtained via community detection. However, computing the modularity of a given graph is a NP-hard problem. The good news is that there exist various methods to obtain an approximate optimal modularity, a classical one being the Louvain algorithm[4].

6.1.1 Louivain

The Louvain algorithm[4] is a popular community detection algorithm used to identify communities or clusters in complex networks. The algorithm is based on a heuristic method that locally optimizes unipartite modularity. Modularity measures the difference between the actual number of edges within communities and the expected number of edges in a random network. Higher modularity values indicate a stronger community structure. Louvain is a two-phase iterative process. In the first phase, each node is assigned to its own community. Then, the algorithm iterates over all nodes and evaluates the modularity gain that would result from moving the node to its neighboring communities. The node is moved to the community that maximizes the modularity gain. This process is repeated until the modularity can not be improved further.

In the second phase, the communities obtained in the first phase are treated as nodes, and the process is repeated to identify new communities at a higher level of hierarchy. This hierarchical approach allows the algorithm to detect communities at different scales. One of the advantages of Louvain, is that it is computationally efficient and can handle large-scale networks with millions of nodes and edges. It is also deterministic, meaning that it will produce the same results for a given network.

Due to Louvain's popularity, it has been widely used and applied in fields like social and biological network analysis, recommendation systems, and network visualization. It provides insights into the modular structure of complex systems and helps understand the organization and functionality of networks.

6.2 Community detection in temporal bipartite networks

In the previous section, we have shown how to detect the communities of a temporal bipartite graphs through their projected graphs by using CD algorithms for unipartite graphs. This is not the only approach though, as we can also apply CD algorithms intended for static bipartite networks to each snapshot G_t of the temporal bipartite network. The modularity definition for a static bipartite weighted network has been adapted by Barber [3] with the idea of redefining the null model

to which we compare the weights within each community. It is expressed as

$$Q = \frac{1}{L}\sum_{i=1}^{S}\sum_{j=1}^{U}[R_{i,j} - \frac{k_i \cdot d_j}{L}]\delta_{c_i,c_j}, \tag{6.2}$$

which considers a random weighted bipartite network, with the same node strength as the given bipartite network, as the null model.

6.2.1 Bi-Louvain

Zhou et al. [47] have proposed a community detection algorithm for static bipartite networks based on the Louvain method and the modularity definition 6.2. It optimizes the aforementioned modularity in a greedy manner. Just like the unipartite Louvain algorithm it encompasses two steps; the assignment step and and aggregation step. These steps are iterated until convergence. The Bi-Louvain algorithm performs the same steps but twice, for each bipartite nodeset iterating after another. These steps are repeated until convergence. The assignment step follows the principle of the unipartite Louvain, except that the gain of modularity is computed according to6.2, and that the neighbors of a node are of the opposite nodeset. For the aggregation step,it is imporant to note that only the nodes of the same community and of the same nodeset are merged together.

6.3 Experiments followup

With the performance of our model being sub optimal in various scenarios and with the fact that the LASSO coefficients predominantly focus on the target links previous history for the prediction at the next timestep, we decided to try to improve the features we select. Instead of selecting from the whole link neighbour space, we now first partition the backbone into various communities, before then limiting the neighbour space to only the links that have at least one of the nodes of the same community as the one of the nodes of the target link. The idea behind it is that we would shrink the neighbour space to more relevant links which in turn could provide LASSO with better input in order to apply the regression and have a better prediction.

First we sample our target links in 5 AR classes with each having 1000 target links, just as before. This includes only sampling the links that have at least 20, 60 and 100 neighbouring links within their community or communities (depending on if the nodes of a the target link share their communities) for their respective classes. Using this target set we also generate the a target set using the old approach without taking the communities into the account, which will serve as our control group. For these experiments we again perform multiple grid searches with the same set of hyperparameters as mentioned in 5.3. We focus on 3 feature selecting strategies of random, maximum AR and maximum weight. Each link will be tested with the same alphas as before.

The results for the 20 and 100 neighbour link sets are shown in the tables below 6.1 .

Table 6.1 Comparison of mean performance for community restricted feature selection (2 weeks).

Dataset	AR	Random			Max total AR			Max total weight		
		MAE	RMSE	Accuracy	MAE	RMSE	Accuracy	MAE	RMSE	Accuracy
20 links	(60,80]	1.106	1.603	0.360	1.081	1.584	0.375	1.084	1.590	0.374
	(80,100]	0.937	1.384	0.436	0.938	1.387	0.437	0.935	1.382	0.437
100 links	(60,80]	0.982	1.444	0.417	0.962	1.436	0.426	0.962	1.430	0.424
	(80,100]	0.800	1.212	0.493	0.792	1.205	0.497	0.794	1.205	0.496

We can conclude that the performance in every case is very similar to the unrestricted experiments we conducted at first in 5. Yet, still numerically the community restricted version consistently scored lower in terms of error. If we look at the results through the accuracy metric we still see a worse performance irrespective of the chosen strategy. Regarding feature selection strategies, it is interesting to note that the top AR strategy outperforms the top weight strategy slightly, contrary to the non community based experiments where top weight was slightly better. It is difficult to draw overarching conclusions with respect to other experiments as we only experimented with 2 weeks of training data. Another gap in this exploratory analysis is the fact that we calculate predetermined clusters based on the structure of the backbone network. It would be interesting to calculate dynamic clusters as the training data changes but such calculations are computationally extremely taxing and thus unfeasible for this paper. Dynamic community detection is actually a really novel branch of community detection and there is very limited amount of literature on this topic, but could definitely be interesting as streaming data becomes the norm.

Table 6.2 Lasso weight based metrics for 100 links SD (community).

Measure (averaged)	$AR(60,80]$	$AR(80,100]$
Number of nonzero Lasso coefficients	12.650	9.544
The value of the TL coefficient	0.063	0.092
The maximum of all the Lasso coefficients	0.210	0.171
% of timesteps TL is the max coefficient	0.431	0.598

If we compare the LASSO based metrics of the community enhanced version to the first experiment back in 5.8 we notice that LASSO included roughly 1 more link from the total pool of 100 links. The average weight TL weight as well as the maximum weight are both higher then in the non community restricted experiment. Lastly we see that the weight assigned to the TL is more often the maximum weight

from all the links with weights. The same observations hold for the 20 link feature set experiments as well.

How can we interpret this? It seems to us that by only including links that are in the same community, LASSO sees more relationships between the predictor activities and target activities, hence the increase in non-zero coefficients. This could also be the reason why the prediction accuracy, as seen in 6.1, is worse than the non-CD version. Another example can be found in Vervoorn [38] where the author also observes this phenomenon - RMSE is inversely related tot the number of non-zero LASSO coefficients.

All in all, it seems that restricting the possible feature space based on evident community structure in the backbone network does not necessarily provide any tangible benefit. Possibly, with networks that exhibit even more excessive community structures this method would be more beneficial. Perhaps a more sophisticated community detection algorithm akin to the Leiden algorithm from Traag et al. [37] is more suitable for these specific datasets. It is known that Louvain algorithm can lead to arbitrarily badly connected communities, whereas the Leiden algorithm guarantees communities that are well-connected. The algorithm is designed to output a partition in which all subsets of all communities are locally optimally assigned. There are also runtime gains due to Leiden making use of the fast local move routine.

Chapter 7

Conclusions and Future Work

This chapter serves as the summary of the conducted research with all the forthcoming conclusions presented. In addition, pointers for possible future angles of research are suggested.

7.1 Conclusions

The core purpose of this project was to investigate if a simple model such as LASSO regression could be used to predict weighted temporal bipartite networks. But even more then that, we wanted to investigate how different configurations would impact the prediction result, in order to get a better understanding of the most useful link properties to consult when predicting the future activities. We defined an activity weight class prediction problem and constructed a model that is able to process traditional (sparse) data structures used for temporal bipartite networks and perform a network prediction at a given time step. This prediction not only includes the specific activity weight class but also per definition the existence (activity) of said link in the future.

- To get the elephant out of the room, we see that the overall prediction power and accuracy is not in a usable spot. The metrics show that for the most interesting links, i.e., those with the highest activity, we are more often than not 1 or 2 weight classes away from the actual weight class. In a real world scenario this would obviously be unacceptable. Of course, the caveat is that we have used a relatively small dataset compared to the data standards of current time. We clearly can conclude that training set length improves the performance proportionally irrespective of the dataset. The caveat being that one needs to be on the lookout for possible sings of over-fitting when pumping a lot of data into the model but this is alleviated with proper feature selection process.
- On the flip side, the LASSO model proved quite instrumental by helping us quantify and express the impact of what type of links lend themselves the best

to base the prediction on. As we have seen, the performance is definitely impacted by the way we select neighbouring links to train the LASSO model. We have explored a total of 6 different strategies which explore various similarities between the links, some of which rooted in their physical properties. All have their use and their effectiveness also varies based on the dataset which implies that network structure and properties are important to the prediction.

- Furthermore, we also verified that the hyperparameters used by LASSO regression have a significant impact on the performance, emphasizing the importance of hyperparameter optimization. We also conclude that training set length improves the performance proportionally irrespective of the dataset. The caveat being that one needs to be on the lookout for possible sings of over-fitting when pumping a lot of data into the model but this is alleviated with proper feature selection process.

- Because the networks provided by Stanford showed signs of community structure when explored using the Louvain algorithm, we tried to improve the quality of the neighbouring link selection by selecting only activity weights from those links, which resided in the same backbone community. Unfortunately this did not lead to any improvements in terms of the prediction quality. Like we mentioned in 6, a better community forming strategy or a dynamically changing community structure could hold the key to unlock better predictions. But it did change the internal coefficients of our LASSO model which might lend to further inspection using techniques like stability selection [30].

7.2 Future work

In this study, we continuously made choices based on the data we wanted to base our experiments on. These datasets definitely impacted the problem statement and as such the eventual design of our solution. This results in a slightly specific deep dive but we like to think that the thought process we outline in the thesis can be use as a guiding principle in any (bipartite) weighted network prediction problem.

Because of this limited scope of this project, there are several research topics that could be potentially explored to further the understanding of predicting temporal bipartite networks. Therefore, in this section, a list of potential directions is given to the interested reader which can build on this thesis. The following recommendations are:

- First of all, the algorithm used for the prediction was LASSO regression. This is a relatively lightweight method of regression which can be probably substituted for more advanced methods. With the advancements in Deep Learning space, more and more models are designed and trained on graphs specifically. One could keep the framework described in the 2 and substitute the method for something like a GCN [25]. This also applies to the choice of the baseline. When putting more performance-oriented methods against each other,

comparing with our simple baseline has no added value. Our LASSO implementation could take the spot for example but many different approaches like markov chains or LSTM's are possible.

Furthermore, the shrinkage parameter optimization of the model was not as extensive as it could be. Given more time and/or resources, a larger grid-search can be conducted with the possibility to compute and save an optimal parameter for each link individually instead of a value that is based on average performance over all the target links within an AR class.

- If LASSO regression is kept as the prediction algorithm, one could also explore with various other strategies to uncover "similar" links and use them for prediction to improve the error scores. One such way would be to again leverage some Deep Learning Frameworks, like the Node2vec model. While outside of the scope of this project, the author has implemented a version of this framework and conducted a few experiments. The results were very promising despite keeping the same setup of this thesis.

- Continuing building on LASSO is also a valid future direction. Stability selection[30] is an extension which utilizes multiple runs of LASSO with the same parameters, in order to capture the most stable features chosen in the regression. Applying such a technique could lead to a better selection of predictive links and their weights. Promising results using this algorithm which can directly be translated to more simple link prediction problems like shown by Kwon et al. [23].

- Performance of other existing approaches to community detection in bipartite networks could be explored as to assert a more concrete conclusion whether the prediction quality systemically is worse or dataset specific. We single out the Leiden algorithm[37] as it has proven performance for bipartite networks but other algorithms could be considered as well, even more traditional methods that could be mapped onto the bipartite structure using projection for example.

- Speaking of datasets, obvious extension could include a more vast set of real world bipartite networks on which this model could be ran to see how the performance stacks up against existing solutions in literature. Other telecom datasets as well as the scientific publications and traffic data are prime examples that exhibit flow that could be represented with weighted links. Given a more diverse set of results it would be possible to induce the impact of network-specific properties on the prediction result. Another benefit would be exploring the link weight class abstraction performance on different link weight distributions.

- Given datasets that span longer periods of time, different prediction granularities could be analysed against each other. Hourly predictions could still be

considered too noisy for many processes, especially if we have data that spans years. More robust trends could be picked up and exploited to improve the performance.

- We limit the prediction of existence of links to a set limited from the backbone of a network. This could be expanded to incorporate the more popular methods in the field of link prediction that concern themselves predicting new links to explore the possibility of assigning weight to hypothetical links in the future network structure.

- While the combination of our LASSO model together with sampling from detected communities did not improve the raw prediction quality, we did notice it had an effect on how LASSO extracted the related predictors. We think this can be further explored by more stable feature selection algorithms like the one by Meinshausen and Bühlmann [30] which has proven to help researches select variables more intuitively with less parameter overhead [23].

Bibliography

[1] Serpil Aslan and Buket Kaya. Time-aware link prediction based on strengthened projection in bipartite networks. *Information Sciences*, 506:217–233, January 2020. doi: 10.1016/j.ins.2019.08.025. URL https://doi.org/10.1016/j.ins.2019.08.025.

[2] A.L Barabási, H Jeong, Z Néda, E Ravasz, A Schubert, and T Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, 311(3):590–614, 2002. ISSN 0378-4371. doi: https://doi.org/10.1016/S0378-4371(02)00736-7. URL https://www.sciencedirect.com/science/article/pii/S0378437102007367.

[3] Michael J. Barber. Modularity and community detection in bipartite networks. *Physical Review E*, 76(6), December 2007. doi: 10.1103/physreve.76.066102. URL https://doi.org/10.1103/physreve.76.066102.

[4] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, October 2008. doi: 10.1088/1742-5468/2008/10/p10008. URL https://doi.org/10.1088/1742-5468/2008/10/p10008.

[5] Ertan Bütün, Mehmet Kaya, and Reda Alhajj. A new topological metric for link prediction in directed, weighted and temporal networks. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 954–959, 2016. doi: 10.1109/ASONAM.2016.7752355.

[6] Furong Chang, Bofeng Zhang, Yue Zhao, Songxian Wu, and Kenji Yoshigoe. Overlapping community detecting based on complete bipartite graphs in micro-bipartite network bi-egonet. *IEEE Access*, 7:91488–91498, 2019. doi: 10.1109/ACCESS.2019.2926987.

[7] Serina Chang, Emma Pierson, Pang Wei Koh, Jaline Gerardin, Beth Redbird, David Grusky, and Jure Leskovec. Mobility network models of covid-

19 explain inequities and inform reopening. *Nature*, 589(7840):82–87, 2020. doi: 10.1038/s41586-020-2923-3. URL https://www.nature.com/articles/s41586-020-2923-3.

[8] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, jul 2019. doi: 10.1145/3292500.3330925. URL https://doi.org/10.1145%2F3292500.3330925.

[9] Yugchhaya Dhote, Nishchol Mishra, and Sanjeev Sharma. Survey and analysis of temporal link prediction in online social networks. In *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, August 2013. doi: 10.1109/icacci.2013.6637344. URL https://doi.org/10.1109/icacci.2013.6637344.

[10] Daniel M. Dunlavy, Tamara G. Kolda, and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data*, 5(2):1–27, February 2011. doi: 10.1145/1921632.1921636. URL https://doi.org/10.1145/1921632.1921636.

[11] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, February 2010. doi: 10.1016/j.physrep.2009.11.002. URL https://doi.org/10.1016/j.physrep.2009.11.002.

[12] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, November 2016. doi: 10.1016/j.physrep.2016.09.002. URL https://doi.org/10.1016/j.physrep.2016.09.002.

[13] Man Gao, Ling Chen, Bin Li, Yun Li, Wei Liu, and Yong cheng Xu. Projection-based link prediction in a bipartite network. *Information Sciences*, 376:158–171, January 2017. doi: 10.1016/j.ins.2016.10.015. URL https://doi.org/10.1016/j.ins.2016.10.015.

[14] Sheng Gao, Ludovic Denoyer, and Patrick Gallinari. Temporal link prediction by integrating content and structure information. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, October 2011. doi: 10.1145/2063576.2063744. URL https://doi.org/10.1145/2063576.2063744.

[15] Liang Ge, Hang Li, Junling Liu, and Aoli Zhou. Temporal graph convolutional networks for traffic speed prediction considering external factors. In *2019 20th IEEE International Conference on Mobile Data Management (MDM)*, pages 234–242, 2019. doi: 10.1109/MDM.2019.00-52.

[16] Roger Guimerà and Marta Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of*

Sciences, 106(52):22073–22078, December 2009. doi: 10.1073/pnas.0908366106. URL https://doi.org/10.1073/pnas.0908366106.

[17] Petter Holme and Jari Saramäki, editors. *Temporal Network Theory*. Springer International Publishing, 2019. doi: 10.1007/978-3-030-23495-9. URL https://doi.org/10.1007/978-3-030-23495-9.

[18] Yuchen Hou and Lawrence B. Holder. Link weight prediction with node embeddings, 2018. URL https://openreview.net/forum?id=ryZ3KCy0W.

[19] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263.

[20] Artus Krohn-Grimberghe, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, page 173–182, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450307475. doi: 10.1145/2124295.2124317. URL https://doi.org/10.1145/2124295.2124317.

[21] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. Link prediction techniques, applications, and performance: A survey. *Physica A: Statistical Mechanics and its Applications*, 553:124289, 2020. ISSN 0378-4371. doi: https://doi.org/10.1016/j.physa.2020.124289. URL https://www.sciencedirect.com/science/article/pii/S0378437120300856.

[22] Jérôome Kunegis, Ernesto W. De Luca, and Sahin Albayrak. The link prediction problem in bipartite networks. In *Proceedings of the Computational Intelligence for Knowledge-Based Systems Design, and 13th International Conference on Information Processing and Management of Uncertainty*, IPMU'10, page 380–389, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3642140483.

[23] Yonghan Kwon, Kyunghwa Han, Young Joo Suh, and Inkyung Jung. Stability selection for LASSO with weights based on AUC. *Sci. Rep.*, 13(1):5207, March 2023.

[24] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient BackProp. In *Lecture Notes in Computer Science*, pages 9–48. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-35289-8_3. URL https://doi.org/10.1007/978-3-642-35289-8_3.

[25] Kai Lei, Meng Qin, Bo Bai, Gong Zhang, and Min Yang. Gcn-gan: A non-linear temporal link prediction model for weighted dynamic networks. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 388–396, 2019. doi: 10.1109/INFOCOM.2019.8737631.

[26] Philipp Lorenz-Spreen, Frederik Wolf, Jonas Braun, Nataša Conrad, and Philipp Hövel. Capturing the dynamics of hashtag-communities. pages 401–413, 11 2018. ISBN 978-3-319-72149-1. doi: 10.1007/978-3-319-72150-7_33.

[27] Haohui Lu and Shahadat Uddin. A weighted patient network-based framework for predicting chronic diseases using graph neural networks. *Scientific Reports*, 11(1), November 2021. doi: 10.1038/s41598-021-01964-2. URL https://doi.org/10.1038/s41598-021-01964-2.

[28] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011. doi: 10.1016/j.physa.2010.11.0. URL https://ideas.repec.org/a/eee/phsmap/v390y2011i6p1150-1170.html.

[29] Amel Ben Mahjoub and Mohamed Atri. An efficient end-to-end deep learning architecture for activity classification. *Analog Integrated Circuits and Signal Processing*, 99(1):23–32, August 2018. doi: 10.1007/s10470-018-1306-2. URL https://doi.org/10.1007/s10470-018-1306-2.

[30] Nicolai Meinshausen and Peter Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010. doi: https://doi.org/10.1111/j.1467-9868.2010.00740.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2010.00740.x.

[31] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, June 2006. doi: 10.1073/pnas.0601602103. URL https://doi.org/10.1073/pnas.0601602103.

[32] Jiajie Peng, Guilin Lu, and Xuequn Shang. A survey of network representation learning methods for link prediction in biological network. *Current Pharmaceutical Design*, 26(26):3076–3084, August 2020. doi: 10.2174/1381612826666200116145057. URL https://doi.org/10.2174/1381612826666200116145057.

[33] Lucas J. J. M. Peters, Juan-Juan Cai, and Huijuan Wang. Characterizing temporal bipartite networks - sequential- versus cross-tasking. In *Studies in Computational Intelligence*, pages 28–39. Springer International Publishing, December 2018. doi: 10.1007/978-3-030-05414-4_3. URL https://doi.org/10.1007/978-3-030-05414-4_3.

[34] Chotirat Ratanamahatana, Eamonn Keogh, Anthony J. Bagnall, and Stefano Lonardi. A novel bit level time series representation with implication of similarity search and clustering. In *"Advances in Knowledge Discovery and Data Mining"*, pages 771–777. Springer Berlin Heidelberg, 2005. doi: 10.1007/11430919_90. URL https://doi.org/10.1007/11430919_90.

[35] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986. doi: 10.1038/323533a0. URL https://doi.org/10.1038/323533a0.

[36] Tomasz M. Rutkowski, Rafal Zdunek, and Andrzej Cichocki. Multichannel EEG brain activity pattern analysis in time–frequency domain with nonnegative matrix factorization support. *International Congress Series*, 1301:266–269, July 2007. doi: 10.1016/j.ics.2006.11.013. URL https://doi.org/10.1016/j.ics.2006.11.013.

[37] V. A. Traag, L. Waltman, and N. J. van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1), March 2019. doi: 10.1038/s41598-019-41695-z. URL https://doi.org/10.1038/s41598-019-41695-z.

[38] Amy Vervoorn. An alternative to standardizing predictors in the lasso with an eye on selection psychology. URL https://www.universiteitleiden.nl/binaries/content/assets/science/mi/scripties/statscience/2019-2020/final-version_thesis_amy-vervoorn.pdf. Accessed: 22-6-2022.

[39] Haixia Wu, Chunyao Song, Yao Ge, and Tingjian Ge. Link prediction on complex networks: An experimental survey. *Data Science and Engineering*, 7 (3):253–278, June 2022. doi: 10.1007/s41019-022-00188-2. URL https://doi.org/10.1007/s41019-022-00188-2.

[40] Tsunghan Wu, Sheau-Harn Yu, Wanjiun Liao, and Cheng-Shang Chang. Temporal bipartite projection and link prediction for online social networks. In *2014 IEEE International Conference on Big Data (Big Data)*, pages 52–59, 2014. doi: 10.1109/BigData.2014.7004444.

[41] Li Xu, Jimmy SJ Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/1c1d4df596d01da60385f0bb17a4a9e0-Paper.pdf.

[42] Yiyeon Yoon, Juneseok Hong, and Wooju Kim. Item recommendation by predicting bipartite network embedding of user preference. *Expert Systems with Applications*, 151:113339, August 2020. doi: 10.1016/j.eswa.2020.113339. URL https://doi.org/10.1016/j.eswa.2020.113339.

[43] Xiu-Xiu Zhan, Alan Hanjalic, and Huijuan Wang. Information diffusion backbones in temporal networks. *Scientific Reports*, 9(1), May 2019. doi: 10.1038/s41598-019-43029-5. URL https://doi.org/10.1038/s41598-019-43029-5.

[44] Muhan Zhang and Yixin Chen. Weisfeiler-lehman neural machine for link prediction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, August 2017. doi: 10.1145/3097983.3097996. URL https://doi.org/10.1145/3097983.3097996.

[45] Peng Zhang, Jinliang Wang, Xiaojia Li, Menghui Li, Zengru Di, and Ying Fan. Clustering coefficient and community structure of bipartite networks. *Physica A: Statistical Mechanics and its Applications*, 387(27):6869–6875, December 2008. doi: 10.1016/j.physa.2008.09.006. URL https://doi.org/10.1016/j.physa.2008.09.006.

[46] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9): 3848–3858, 2020. doi: 10.1109/TITS.2019.2935152.

[47] Cangqi Zhou, Liang Feng, and Qianchuan Zhao. A novel community detection method in bipartite networks. *Physica A: Statistical Mechanics and its Applications*, 492:1679–1693, February 2018. doi: 10.1016/j.physa.2017.11.089. URL https://doi.org/10.1016/j.physa.2017.11.089.

[48] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications, 2018. URL https://arxiv.org/abs/1812.08434.

[49] Yajian Zhou, Jiale Li, Junhui Chi, Wei Tang, and Yuqi Zheng. Set-cnn: A text convolutional neural network based on semantic extension for short text classification. *Knowledge-Based Systems*, 257:109948, 2022. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2022.109948. URL https://www.sciencedirect.com/science/article/pii/S0950705122010413.

Appendix A

Supplemental notes and experiments

A.1 Implausible activity weights in SD

It is important to recognize the actual numerical data found in the Stanford mobility network dataset, which is publicly made available. There a large proportion of SD link activity weights are decimal values between 0 and 1. These are represented by the first few AR classes we introduce in 2.3.1 Physically that doesn't make sense within the context of the data, as there can't be a third of a human present at a certain timestep - so it is important to know the explanation on why such values are contained within the dataset.
In order to not go too deep in to the details (see appendices on Iterative Proportional Fitting Procedure (IPFP) in [7] for more information), the algorithm used estimate the visitors matrix (from which we construct the bipartite network for SD), also known as biproportional fitting, is a mathematical algorithm used to adjust joint probability distributions to match given marginal distributions. IPFP is particularly useful in situations where only marginal distributions are known or observed, but the joint distribution needs to be estimated. These marginal distributions describe visitng patterns of citizens from a certain demographic group (CBG) to physical geolocations in US cities (POI), which are the two nodesets in SD. This algorithm iterates on data which time period exceeds that of the mobility data we construct our bipartite networks from, so we treat this phenomenon mostly as noise produced by the algorithm used. Moreover, the authors themselves note that despite the effectiveness of IPFP in estimating the visitors, this comes with challenges, particularly in cases where the sparsity of data makes it difficult to reconcile the underlying SafeGraph data.

A.2 Exploratory deep dive on activity weights

Due to our data being source from real world situations it is important to take into account any factors that are specific to that time period. As such we know that COVID-19 was spreading around the world in early 2020. We can actually see the impact of quarantine when looking at the activity. One interesting angle we can take when analysing the activity of the links, is to see if we can spot any patterns by aggregating on a larger timescale. Below we have plotted the aggregated activity rates (ARs) over the days of the week. We also took a look at how the ARs evolve during consecutive weeks.

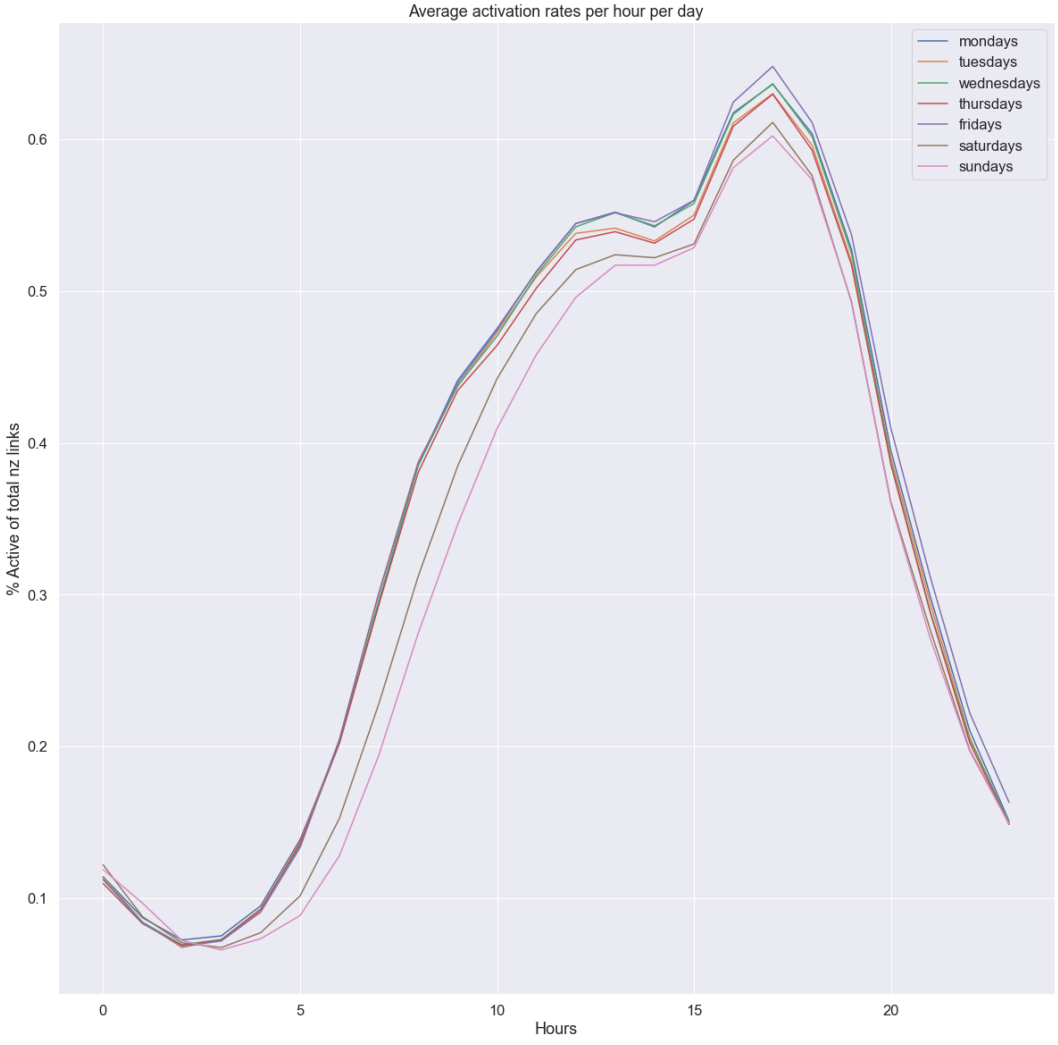
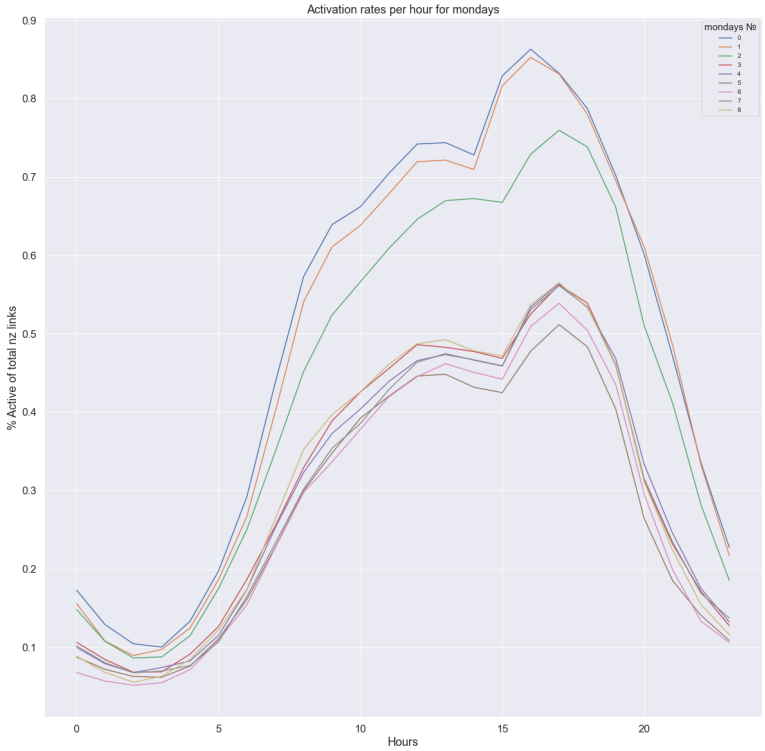
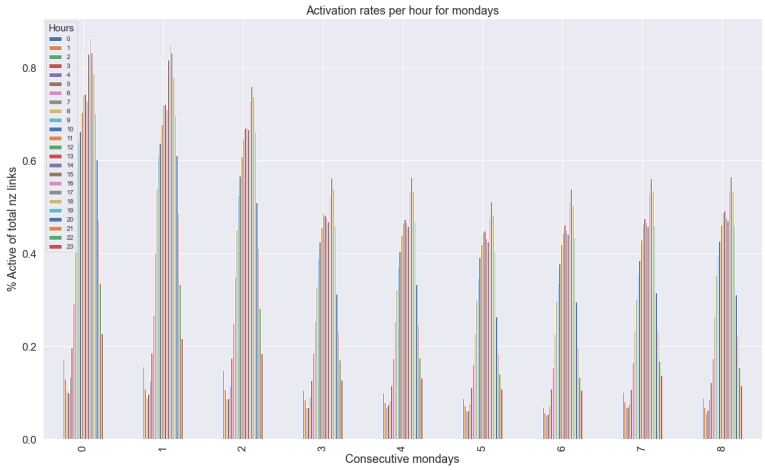


Figure A.1 | **Lineplot of average activation rates for links at a certain day, in SD, across the whole time period.**



(a)

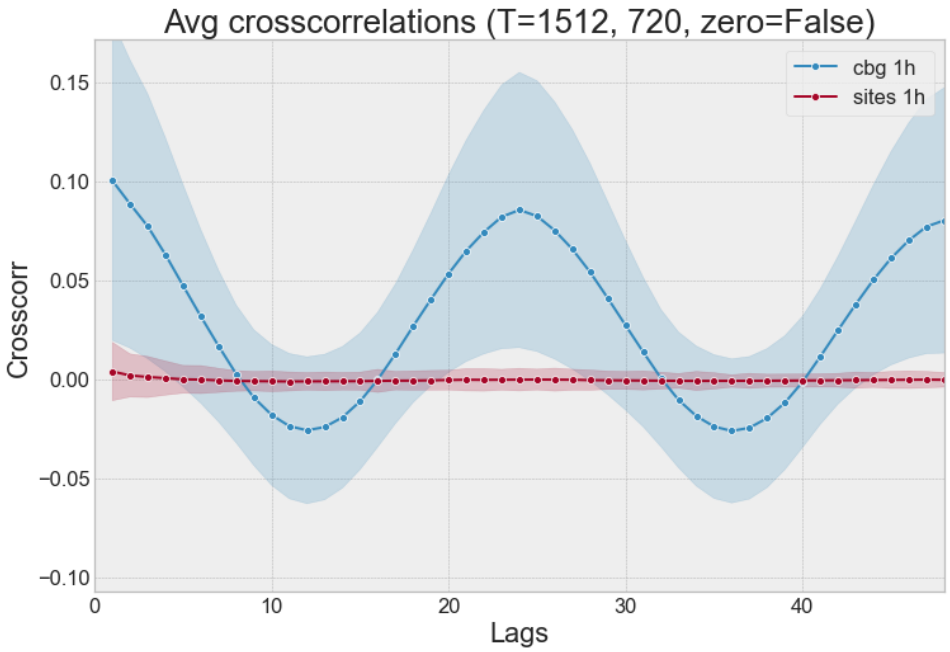


(b)

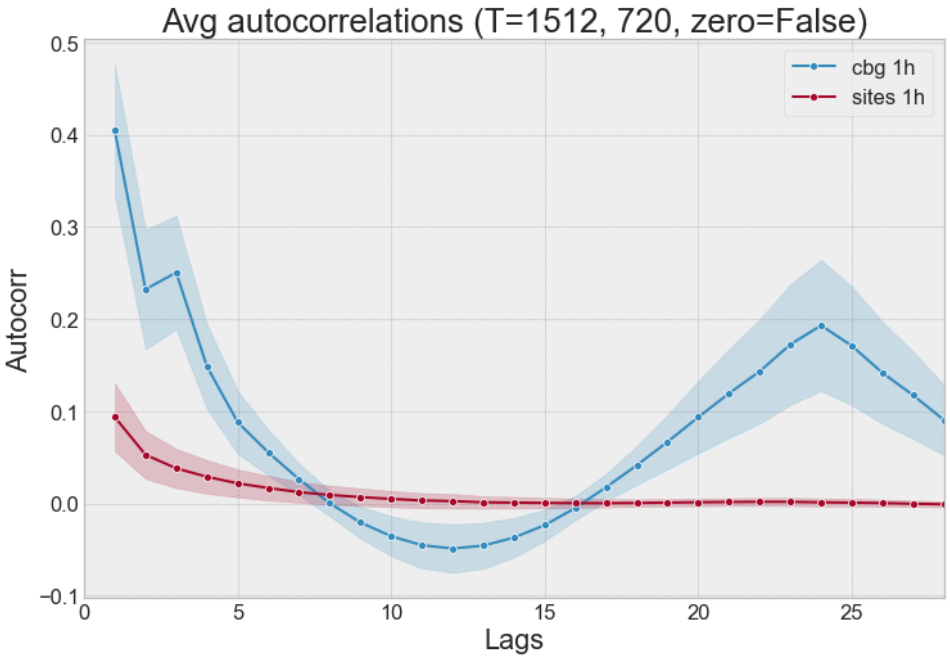
Figure A.2 |**Activation rates over all the mondays in KD**
A) shows us the ARs on mondays through the KD timeframe. B) shows us the same plots but in histogram form, where a monday is broken down in 24 hours.

We've also explored the notion of cross- and autocorrelations of the timeseries vectors of the links. For this we decided to approach the question from a physical point of view - by aggregating the correlations along the CBG (SD) and cell

towers/sites (KD).



(a)



(b)

Figure A.3 |**Correlation of activity weight in links aggregated over the physical locations.**
A) shows the aggregated cross-correlations for links that share the same physical location as a node. B) shows the autocorrelations (crosscorrelations of a link's activity weight time-shifted with itself.

We can clearly see the daily pattern for SD come into play here again. The correlations spike at multiples of 24 hours, meaning that the activity weight at a point in time is similar to the activity weight at the exact time, in previous days. This is also what inspired us to proceed with the experiments and exploration of one-step prediction trained on multi-week training sets.

What is a little puzzling is the lack of correlations in the KD dataset. Although it is not out of the question that data traffic at cell towers is a lot more random by nature, you still would expect some correlation day to day due to the natural cycle of day and night.

A.3 Dynamic feature selection

All the experiments up until this point have been facilitated by a feature selection strategies that select neighbours based on the backbone network of the datasets over the total timespan. This allows us to determine the feature links in advance, but it does diminish the application of this technique to real time situations where you might not have enough data to extract a proper backbone. It might be the case that you have a continuous stream of data and you want to dynamically update the features you select in hopes of capturing more relevant information. This is obviously a more computationally expensive approach as one has to repeatedly generate the backbone for the network over the training history timespan. Given the time constraints we have not performed this process on all the strategies we outlined in our single backbone experiments. The results are also aggregated among all the AR's due to a different methodology when these experiments were executed.

Table A.1 Mean performance for dynamically updated features (20 link feature set).

Dataset	AR	top AR		top weight	
		MAE	RMSE	MAE	RMSE
SD	$(0,20]$	0.845	1.684	0.846	1.684
	$(20,40]$	1.296	2.019	1.265	1.976
	$(40,60]$	1.512	2.141	1.481	2.094
	$(60,80]$	1.158	1.593	1.157	1.587
	$(80,100]$	0.619	1.484	0.619	1.487
KD	$(0,20]$	0.265	0.697	0.269	0.716
	$(20,40]$	1.855	2.480	1.850	2.480
	$(40,60]$	2.289	2.764	2.225	2.686
	$(60,80]$	2.283	2.772	2.271	2.763
	$(80,100]$	1.874	2.372	1.872	2.373

Appendix B

Background on traditional link prediction methods

B.1 Traditional link prediction methods

In recent years, there has been a growing interest in link prediction within networks, particularly complex networks. Link prediction is a fundamental problem that attempts to guage the likelihood of a link existing between two nodes Lü and Zhou [28], which makes it easier to understand the associations between nodes and shed light on the evolution of networks. Over time, researchers have introduced a multitude of link prediction techniques, leading to numerous surveys and literature reviews exploring this broad spectrum of methods, often within specific contexts such as social [9] or biological networks[32]. This taxonomy gets updated the more aThese methods span from simple heuristic approaches, like common neighbor counting, to the more contemporary network embedding-based methods. Many of these techniques compute similarities or probabilities of link formation by capturing the structural attributes of the network. In this section we will touch upon techniques like similarity-based, path-based, probabilistic and statistical models-based, classifier-based, and network embedding-based methods.

Before 2010, the traditional link prediction methods, such as similarity-based and path-based methods, were widespread because of their simplicity, interpretability, efficiency, and high accuracy. One downside of this approach however, is that these methods are not designed to make full use of network structure information.

B.1.1 Similarity based

Similarity-based metrics rely on a similarity score $S(x, y)$ between nodes x and y, which based on the structural or node's properties of the chosen link. This score is proportional to the probability that an edge exists between x and y. One can intuitively assume that two nodes x and y are more likely to form a link in the future, if their neighbors have large overlap. Not surprisingly, the simplest technique of measuring similarity is counting the shared neighbors directly which is called

Common Neighbors, but there are many others similarity indices that can be used, like the Jaccard Index, the Ademic/Adar Index or the Resource Allocation Index. One point of note is that these measures are all rooted in local measurements around the link and it's nodes.

To end this section, we want to highlight Preferential Attachment (PA) [2]. The idea is that a new link associated with node x is proportional to the degree of the node, $deg(x)$. This metric is quite simple as it requires the least information. Due to this aspect, it can be used in a non-local context (Common Neighbours can't for example. The downside is that it has the worst performance on most networks, per the authors. This point will be relevant later in 3.3.

B.1.2 Path based

The similarity measures discussed earlier focus solely on limited, local structural information and do not consider global similarities between nodes. In contrast, path-based methods incorporate similarity measurements based on paths between nodes, thus accounting for higher-order information and capturing more of the network's structural features compared to the previous methods. Hence the term "global" similarity is often with these methods, as we are still trying to find similarities between links, only now using information from the entire network The downside is that the computational complexity of these methods is higher than the local based metrics. Most of these methods rely on transforming the relation between the paths (often the length of the path) between two nodes into a similarity score. Common measures are the Katz Index, Leicht–Holme–Newman Global Index, Average Commute Time and Matrix Forest Index.

There also several methods of similarity that try to restrict the complexity of the aforementioned global metrics by limiting the scope. They create a trade-off between performance and complexity. Such "quasi-local" metrics include Local Path Index, Local Random Walk and Path of Length 3.

B.1.3 Probabalistic and statistical models

Moving on, probabilistic and statistical methods come in to the frame, as they provide a way to extract information of the underlying network structure. Their modus operandi consists of building a model that optimizes an objective function, which then will help simulate the network. This is done by estimating the parameters which best fit the observable data of the network. Using the model's parameters it is possible to deduce the probability of forming new or missing links. Because of this reliance on a model, a large chunk of this approach resides in training the model, rendering it impractical for large networks due to the time investment involved. Most common model in this class of methods is the stochastic block model (SBM). Here, the nodes are organized into distinct groups or communities, where the likelihood of connection between two nodes depends solely on the groups to which they belong. Generally, a combination of mechanisms is at play when looking at

(complex) networks - factors like modularity, role structure, community structure, and more. In SBM, partitioning nodes of a network can be catered to one or more of such mechanisms, such that different block models will capture different correlations of the network. An interested reader can follow up on these methods in Guimerà and Sales-Pardo [16].

B.1.4 Classifier based

Now, when we move closer to the present time, we encounter a slightly different approach for addressing the problem of predicting missing links. Rather than calculating a score based on similarity or probability, learning-based models are employed to leverage the topological features of the network. These models typically take the form of supervised or semi-supervised learning tasks. Numerous classification algorithms have been tailored specifically for link prediction. Selecting appropriate features is considered crucial in these supervised learning algorithms, akin to many other machine learning solutions. Again we must note, the time complexity and the space complexity of these type of approaches are quite high, making it unsuitable for large networks.

B.1.5 Network embeddings

As Internet technology and big data continue to advance, the scale of the networks continues to expand, leading to what is often termed as ”dimensionality explosion”. Network embedding-based methods are designed for this problem, as they aim to reduce the dimensionality while also capturing the characteristics of the network at the same time. Different from the traditional adjacency matrix, network embeddings aim to effectively preserve rich topological and structural information, including links, neighbouring nodes, and high-order proximities by embedding nodes into a low-dimensional space to predict the potential future links. Transforming high-dimensional but sparse feature vectors into low-dimensional, dense embedding vectors, facilitate more efficient representation of network properties.This class can be subdivided further into matrix factorization based, random walk based, graph neural network based, and other methods.

Matrix factorization

Matrix factorization is one of the older approaches, which has been applied in a lot of papers on link prediction problem in the last decade. It is a technique used to capture the underlying patterns of a network, embedded in the form of a matrix (often the adjacency matrix). By decomposing the matrix into the product of two lower dimensional matrices which represent the embedding vectors. Typically, the latent features are extracted and using these features, each node is represented in the latent space. These representations are then used in a supervised or unsupervised frameworks built for link prediction. It is proven to be successful in many

different domains (e.g., social network analysis [20], recommender systems [19], neuroscience [36]). Dunlavy et al. Dunlavy et al. [10] used the above-defined techniques to predict links in temporal networks. In their paper, the performance of matrix factorization is compared with tensor factorization. The main difference between these two techniques is the dimensionality of the input matrix. Since matrix factorization can only deal with a matrix of the form $m \times n$, it is less suitable when dealing with temporal networks . On the contrary, tensors can incorporate an extra temporal dimension. The conclusion of the comparison was that the tensor factorization approaches slightly outperform the matrix factorization approaches. In practical applications, nonnegative matrix factorization (NMF) and singular value decomposition (SVD) are the go-to methods to get the approximation of the original matrix, mainly because of the lesser time-complexity due to the use of extra constraints.

Random walks

Decomposition based on the adjacency matrix has the limitation of encoding only the information of direct neighbors of the node in question. Another approach is to incorporate random walk based techniques. Herein, a random walk is used to generate the context of nodes where the appearance of two nodes within the same random walk, leads to more similarity between the embeddings of those nodes. The node sequences can be treated as sentences to take advantage of natural language processing methods to get node embeddings. A few notable methods within this category are Deepwalk, Node2vec and Struct2vec.

The approaches discussed here merely provide the embedding vectors which still need to be leveraged for subsequent analysis tasks. Steps where similarity is calculated between the embeddings are necessary for link prediction. We discussed the different distance metrics to achieve this in 4.4. According to Wu et al. [39] there is very little notable difference among the distance metrics.

GNN

A very modern approach is rooted in Graph neural networks (GNNs), which are based on convolutional neural networks (CNNs) and graph embeddings. GNNs have demonstrated remarkable success in analyzing Euclidean input data, such as images [41] and text[49]. However, traditional CNNs cannot handle non-Euclidean input data, while GCNs can and thus have received much attention thanks to this aspect. These neural networks learn hidden layer representations that encode both local graph structure and node features, enabling the exploitation of these characteristics for tasks like node and graph classification, as well as link prediction, among others. Several specific frameworks tailored for link prediction include WLNM [44], DCGNN [29] and Cluster-GCN [8].