Predictability in Temporal Networks

Alexandru Marcu

Supervisors:
Sune Lehmann Jørgensen
Jakob Eg Larsen

Kongens Lyngby 2013
IMM-M.Sc.-2013-57

Technical University of Denmark
DTU Compute — Department of Applied Mathematics and Computer Science
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk IMM-M.Sc.-2013-57

Summary

With the increase in availability of temporal datasets collected from complex networks come new possibilities for studying the dynamic patterns formed by the interactions of such networks. Meaningful networks can be observed anywhere in day-to-day life: in phone-calls and daily social interactions; in public transportation, in technology and in nature: in interactions between species or between proteins. Having temporal data about such systems allows for a temporal networks representation.

While the link prediction problem already has developed well-established methods for predicting future interactions by analyzing a network's intrinsic features, it predates the concept of temporal networks and only assumes a static network (a single state of the system), only being able to predict a single future state, of unknown temporal limits. When temporal data is available the expectations become higher, the occurrence of a new interaction has to be more precisely delimited in time, and more than a single state of the network has to be taken into account. Not much literature currently covers the prediction problem for temporal networks, and what exists is focused on certain domains and very specific approaches.

This thesis looks at the prediction problem for temporal networks from a broader angle, aiming to identify general goals for each stage of the problem, we propose an experimental framework for solving the prediction problem for temporal networks. The robustness of the framework is tested with an implementation aimed to obtain results from a temporal network of face-to-face interactions.

The results are collected from multiple experiments aimed to explore the parameter space, and are validated using state-of-the-art measures for predictive performance. These results will demonstrate that, although the specific methods were relatively simple, their implementation within the proposed framework brought relatively good results. The proposed framework is just a first step at generalizing a large problem, and the directions for further development are many: the framework could be optimized for specific domains or, by contrast, improved to provide more possibilities while keeping its generality.

Preface

This thesis was prepared at DTU Compute – Department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfillment of the requirements for acquiring a M.Sc. in Computer Science and Engineering.

The thesis deals with studying predictability of temporal networks and describes an experimental framework for solving the structure prediction problem. The framework is implemented and tested on a temporal network of face-to-face interactions.

The thesis consists of an introduction describing the problem domain, goals and related work (Chapter 1); a specification of the framework components proposed for extracting predictable patterns and the discussed implementation for our domain (Chapter 2); the proposed models for prediction and validation, and the discussed results, based on a set of experiments exploring the parameter space (Chapter 3); conclusions and possible future research (Chapter 4).

Lyngby, 26-July-2013

[illegible]

Alexandru Marcu

Acknowledgements

First and foremost, I would like to thank my supervisors, Sune Lehmann and Jakob Eg Larsen for their invaluable feedback regarding the direction of this thesis and its challenges.

I also wish to thank everyone in their group for inspiring me with their work and presentations; special thanks to Piotr Sapieżyński for his guidance with the ROC validation and to Vedran Sekara for providing a great XGMML representation of the dataset and for the inspiring presentations.

Contents

Chapter 1

Introduction

Everywhere in nature, society and technology we can find dynamic systems whose evolution is influenced by underlying entities that are interacting with each other over time.

From our own daily interactions with others, face-to-face [BC13] and via phone calls or Internet, interactions that occur between species [PD05], ball passing dynamics in a football game [HT05] to interactions that occur between molecules of a cell [Pal06]. They are all studied by a certain discipline in an effort to reach a better understanding of the system and solve problems related to it.

When we can identify certain entities and the interactions between them in a system, we can get an overview of such a system by representing it as a *graph*. A graph is a mathematical representation of a set of entities connected pairwise by links. The entities that are interconnected are named *nodes*, and the links that connect them are named *edges*.

Representing a system as a graph allows us to learn much about how groups of entities are connected with each other, what roles certain entities play in the system, what entities are most influential in the system, etc. Even so, for certain systems this framework can be improved by adding additional levels of detail, such as, the weight of an edge [BBPSV04], spatial positioning of a node [Bar11], etc., depending on the characteristics of the system.

In this thesis, we are interested in the dynamics of interactions between entities and we add a temporal dimension to the graph representation of a dataset, thus we get information about the times at which entities and interactions are active. By adding the temporal dimension, a static graph can be represented as a sequence of graphs ordered by the time, where each graph contains a subset of nodes and edges, that is, those which are active at the given time (as seen in Figure 1.1). Such a representation is a *temporal network*.

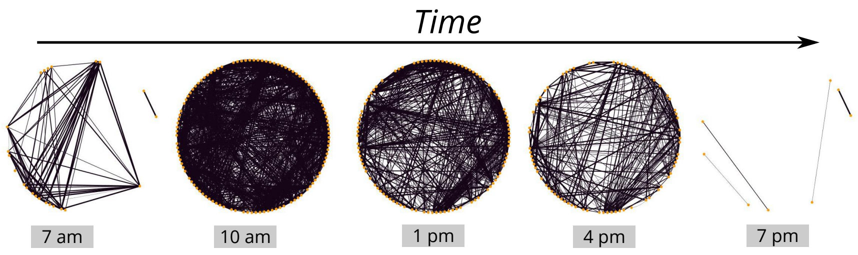


Figure 1.1: A temporal sequence showing face-to-face interactions between students at DTU Campus between 7 a.m. until 10 p.m. on a regular Monday. The graphs are arranged in circular layout, so that the activity increase during study hours is clearly visible, likely due to attending classes.

A great variety of systems can be modeled as temporal networks, yet achieving a good understanding of such systems requires some degree of preliminary understanding of the system and the domain it is part of, thus the study of temporal networks is an interdisciplinary field. Some examples of real-world problems where a temporal networks model can prove useful are – finding the path in spreading of a biological or electronic virus, determining trends in stock market prices, understanding relations between individuals in a social network and predicting future interactions. The latter example makes object of this thesis.

Having enough knowledge of the system to remove undesired noise and time intervals where the activity is known to be nondeterministic, we are left with sequences of graphs likely to share a similar topology. Such sequences can form more or less predictable patterns. For example:

- The change of seasons is known to influence the interactions between species, this can be more predictable in the case of animal species that are known to migrate on a seasonal basis. Even so, unknown factors can influence the location where they migrate, or the other species they interact with.
- Daily routine can be predictable in the case of individuals with traditional employment or education where we have well-delimited boundaries between working hours and an individual's free time, as are the interactions

with colleagues delimited from those with friends and relatives.

- The interactions between individuals at an airport on the other side, are very hard to predict accurately since individuals interacting with each other are always different. In such a network, only regular business travelers may have some predictability. The majority of travelers and tourists are not regular airport users and would be discarded as noise. On the other side, including data from individuals who are airport personnel would restrict all predictions to their daily routine (which has a much higher regularity relative to that of business travelers).

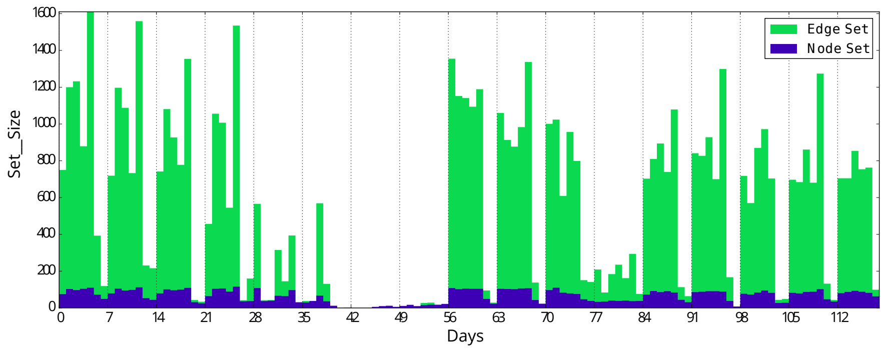
In the case of the airport interactions example, while it may be helpful to use a temporal networks model to study the outbreak of a disease, the number of relatively reliable positive predictions that can be made is very low compared to the total number of predictions (reliable or not) that can be produced, which is proportional to the size of the dataset. We denote such a dataset to yield a low *predictive value* compared to the first two examples.

In this thesis we study the predictability problem – which we present in Section 1.3 – for a proximity dataset described in Section 1.1, and for doing so, we propose an experimental framework whose goals are discussed in Section 1.2. A number of articles have been significant in inspiring the development of the framework, while others can yet bring further value as a starting point for studies that would make use or expand the proposed framework, these papers are presented in Section 1.4.

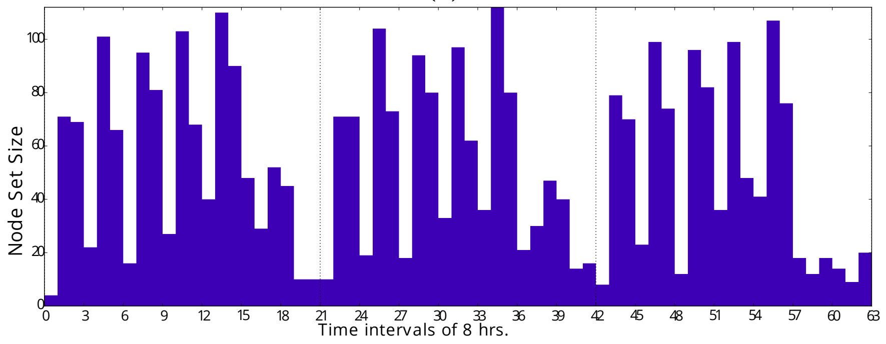
1.1 Dataset

The main dataset used in the development and testing of our predictability framework consists of proximity data collected by the SensibleDTU[1] team from the smartphones of ∼140 students via Bluetooth, using a dedicated data collection app which only became active when students were in the proximity of the DTU Campus. Due to limited range of Bluetooth visibility, and the requirement for the devices to be interconnected for at least a certain minimum period before registering an interaction, the proximity data largely corresponds to face-to-face interactions. The data was collected throughout a period of 4 months (between November 12th, 2012 until March 11th 2013) and the temporal resolution is of 1 hr. per timestep. Figure 1.2 shows an overview of the temporal data in terms of changes in cardinality of node set (individuals) and edge set (interactions) of the graphs in the time sequence.

[1] `http://sensible.dtu.dk`



(a)



(b)

Figure 1.2: The histogram in *(a)* shows cardinalities of node and edge sets for our entire dataset with graphs aggregated to 24 hours per timestep, while in *(b)* we only show the node set cardinalities within the first three weeks, from graphs aggregated to 8 hours per timestep. In *(a)*, firstly, we can observe that the number of edges presents fluctuations in their proportionality to the number of nodes. Secondly, we can get an overview of the entire temporal sequence and we can notice the low activity during weekends, the December holiday (∼42–56), and of the study break between semesters(∼77–84). Figure *(b)*, zooms in and shows only the cardinality of the node set. Thus, we can observe the activity fluctuations in different intervals of the day such as: early morning, study hours, evening.

The students taking part in data collection were enrolled in 3 different majors (Physics and Nanotechnology, Electro technology and Software engineering) thus providing a good picture of social interactions within each of these densely connected groups. Furthermore, in order to reduce the risks of participants leaving the project, the users were provided with an environment in which they could access and visualize the data collected about them.

During data collection, entries to the database were added only when an interaction (proximity) between at least two individuals took place. In consequence, a node in the dataset's corresponding temporal network will be active only when it has an active link to at least another node. Therefore it is good to note that, although the temporal networks model places no restriction on this, there are no graphs in our dataset's temporal network containing isolated nodes.

In the context of predictability in temporal networks, examples to show the value of isolated nodes are yet to be discovered, and although there may be systems where such nodes hold some value[2], our thesis is mainly focused on the predictability of interactions taking place in the system and not in predictability of entities.

1.2 Goals

The question our thesis begins from is *"can we extract the essence of a dynamic system based solely on its recurring temporal patterns?"*. The question does not have a straightforward answer: some systems may not present a variety of evolution patterns enough to be considered their essence, others may have so complex recurring patterns that extensive domain knowledge may be required, while others may indeed have accessible recurring patterns, yet – unless the right questions are asked – there is very little we can learn from.

Therefore, the primary goal of this thesis is to provide an extensible experimental framework for *(a)* exploring different data processing and training models, *(b)* finding evolution patterns in temporal networks and *(c)* determining the predictive value of a temporal network. Furthermore, the proposed framework should be domain-independent, it should not rely on assumptions which only apply to datasets coming from specific domains (e.g. social datasets).

While the domain independence cannot be completely guaranteed, we assume that a dataset that does not fit in the proposed framework cannot be meaningfully represented as a temporal network. For certain domains, additional levels of detail might be of essence and the proposed framework should be able to scale and support any additional heuristic algorithms or parameters which a certain domain might require.

The framework for solving the structure prediction problem can be split into smaller models for solving individual tasks. These smaller models should be the loose-coupled components of the framework. A component should allow for its implementation to have any kind of approach without affecting the other

[2]For example, when the activation of a node would trigger a remote interaction

components, as long as the input is correctly processed and the expected output is forwarded to the next component. Each of the actual implementations can therefore be regarded as a heuristic algorithm, for which domain specific solutions and shortcuts are desirable, and whose individual performance can be easily assessed.

It is important to specify that while aiming to keep a degree of generality the specific implementations of each of the framework components were built with the available dataset in mind. Therefore, while datasets belonging to different domains may still give adequate results, it is most likely that the current implementation will perform best with social datasets (and less so with e.g. finance or microbiology datasets). Furthermore, it should also be noted that the implementations for each component are simple solutions, meant to exemplify the possibilities of the proposed framework. Each individual component solves a complex problem for which the simple solution may be further improved with state-of-the-art methods.

For the developed Python tool which implements the framework and computes the results, the main goals were extensibility and providing an environment that allows for easy visualization and comparison of models and their results.

Throughout the thesis, additional detail will be brought to these goals as the underlying components and concepts of the proposed framework are discussed.

1.3 Definitions and Problem Statement

This section formally defines a *temporal network* as is required for defining and discussing the *structure prediction* problem for temporal networks. Definitions 1.3.1 and 1.3.2 were previously introduced in the work of Lahiri and Berger-Wolf [LBW07]. A more established and adjacent, yet substantially different problem – that of *link prediction* – is also discussed in relation to structure prediction towards preventing any future confusion.

Definition 1.3.1 A *temporal network* is a sequence of graphs $\mathcal{G} = \langle G_1, \ldots, G_T \rangle$ where $G_t = (V_t, E_t)$ is a graph of pairwise interactions at time $t \in [1, T]$, represented as *edges* in E_t connecting *nodes* in V_t.

A value for time t denotes a *timestep*, whereas an instance of the graph G_t, $\forall t$, denotes a *snapshot*.

We also informally define the *resolution* of a temporal network as Δ_t^{-1} where Δ_t

is the time difference (i.e. in seconds) between two consecutive timesteps, thus a temporal network can be said to have a high-resolution when the temporal level of detail is high, relative to another temporal network, or relative to a 24 hrs. time difference (when studying social datasets, or other systems where entities have a behavior corresponding to a circadian rhythm[3]).

Definition 1.3.2 (Structure Prediction problem) Given a temporal network $\mathcal{G} = \langle G_1, \ldots, G_T \rangle$, predict a set of edges for each G_t where $t > T$.

The *structure prediction* problem can be confused with another closely related problem of network analysis, that of *link prediction.* As the names may suggest, similar results are achieved, however at different levels of precision, and starting from different inputs. To remove all confusion, we give the definition of *link prediction* based on the definition provided in the work of Liben-Nowell *et al.* [LNK07], with minor adjustments to suit the context in discussion.

Definition 1.3.3 (Link Prediction problem) Given the snapshot G_T of a temporal network, predict the edges that will be added to the network during the interval from time T to a given future time T'.

The *link prediction* problem uses a single snapshot as input, which can be the current state of a system, and looks at the *features that are intrinsic to the network* itself, without assuming or having knowledge of the previous states of the network; a large variety of literature has been developed around the topic (since it predates the temporal networks model) and different heuristic approaches for extracting predictive network features have been developed. The model produces a list of edges likely to occur between the (present) state at time T and a certain point in the future T' without predicting the exact time of occurrence.

In contrast, the *structure prediction* problem requires an entire temporal network as input, and a solution to the problem will rely on learning from the recurrent patterns in the temporal variation of structure, however it can also make use of the features computed by a link prediction model, or by any other known models. On the other hand, having knowledge of recurrent patterns, it is expected that a solution to the structure prediction problem is capable of predicting *when* an interaction may occur with a higher precision than in the case of *link prediction.*

It is easy to remark that the two problems are designed to provide precision levels proportional to the availability of data. Knowing only a single state of

[3]A *circadian rhythm* is any biological process that displays an oscillation of $\sim$24 hours (Wikipedia)

the system, there is no way to predict when interactions will occur but only which interactions are likely to occur. Having access to a larger sequence of states, it is expected that we are also able to predict when certain interactions will occur, both by using the strategies of more simple problems as well as specialized methods for discovering the recurrent temporal patterns. We can therefore conclude that link prediction and structure prediction are *overlapping problems*, and that structure prediction is an extension to link prediction.

The framework proposed by this thesis is built as a model for solving the structure prediction problem. By default, it supports implementation of algorithms for link prediction, which may enhance the quality of recurrence based predictions, and thus, significantly improve the accuracy of the produced results.

1.4 Related Work

This section surveys the literature relating to temporal networks and to the proposed framework. We can split these papers into four categories: *1)* papers used as *starting points* – that introduced concepts significant in the design of the framework, *2)* papers that introduce *network measures* that may be experimented with and which may contribute to producing improved prediction models, *3)* papers describing *specific systems* which may inspire new experiments with the framework and *4)* papers that may be of use in *extending the framework*.

The literature which makes the first category can justify many of the decisions taken in the design of the framework, the second category can be of great use in creating interesting experiments, the final two categories – with a few exceptions – consist of relatively new papers which may prove useful in raising new questions and introducing new concepts that could prove to be significant in further development of the proposed framework, these will further be referred to in Chapter 4, when future work is discussed.

1.4.1 Starting points

An important starting point was the work of Lahiri and Berger-Wolf with *"Structure Prediction in Temporal Networks using Frequent Subgraphs"* [LBW07], the paper introduces the previously described structure prediction problem and provides an algorithm for discovering the recurrence pattern of interactions based on the recurrence in their relative delays.

This approach is central to our method for discovering patterns formed by interactions. To reduce the number of computed delays (which is huge, and has a negative impact on the computational performance), interactions are redefined as *maximal frequent subgraphs.* These are extracted in a preprocessing step using a data mining algorithm named *MAFIA* [BCG01]. This algorithm as well as a different data mining algorithm named *FP-Growth* [HPY00] were experimented with in the beginning of our framework's development, they were only kept as alternate implementation choices, since the computational performance of both of these *frequent itemset mining* algorithms is still very low, while the predictive performance may also be affected. This choice and the solution that was found, will be detailed in Section 2.4.

One of the challenges of adapting the algorithm proposed by Lahiri and Berger-Wolf was the resolution difference between our dataset, and the datasets on which they based their experiments: the Enron dataset [KY04], a financial dataset of stock market price changes and a biological dataset consisting of interactions between plains zebras. It became obvious that while the delay based prediction algorithm could provide meaningful information at a resolution of e.g. 24 hrs. per timestep, its results on a resolution of 1 hr. per timestep were not as meaningful. That is, because most of the recurring delays had a size of only a single timestep, thus pointing to interactions that occur at consecutive timesteps, which are in fact, the continuation of an existing interaction.

Aiming to keep the degree of generality of our implementation, we looked for models of discovering the optimal resolution. The model provided by Clauset and Eagle [CE12] looks at the length of interactions as a function of the duration of the time window. They compute the optimal resolution based on the average degree of nodes, the clustering coefficient and a measure called the *adjacency correlation* between pairs of adjacency matrices.

In another such model proposed by Sulo *et al.* [SBWG10], a time series is formed by computing a user-defined network measure to each graph in the temporal network. For different resolutions (which are iterated through), two measurements having opposite behaviors (relative to window size) are applied to the time series, these are the *variance* and the *compression ratio* of time series' string representation. When the difference between the two measurements is minimal, then the resolution for which the measurements were made is optimal.

Testing these models brought different results, which in most cases had a negative impact on predictability, with some exceptions for the model of Sulo *et al.*, where a list of optimal resolutions is yielded depending on a *goodness value* which defines what maximal difference between the two opposing measurements is accepted for a resolution to be considered optimal. By relaxing this parameter a larger list was resulted which included some of the values expected for our dataset. Due to the variety of resolutions that can be used, the framework proposed in this thesis does not assume an optimal resolution, allowing for dynamically sized timesteps instead. Research on the effect of temporal resolution used for temporal networks has been done by Ribeiro *et al.*[RPB12] who provide a framework for investigating the consequences of aggregating timesteps to a fixed resolution.

1.4.2 Network measures

A comprehensive overview of temporal networks and the available metrics was published by Holme and Saramäki [HS12], their work includes references to a wide range of papers around the topic. The metrics presented in this paper and others are detailed in the recently published work by Nicosia *et al.* [NTM$^+$13] and in the work of Tang *et al.* [TSM$^+$10]. The previously mentioned paper by Sulo *et al.* [SBWG10] also indicates some of the network measures for temporal networks, particularly those meaningful to detecting the similarity between graphs in the context of finding an optimal resolution.

While Nicosia *et al.* and Tang *et al.* are focused on those measures that have to be redefined for temporal networks (e.g. paths, distances between nodes), small-world behavior and motifs, Sulo *et al.* proposes measures that can be applied directly for the graphs corresponding to each timestep (e.g. density, eccentricity, radius).

1.4.3 Domain-specific systems

In cell biology, a recent paper by Wallach *et al.*[WSM$^+$13] identifies new protein-protein interactions (PPIs) and builds a dynamic network of the interactions, predicting that circadian PPIs "dynamically connect many important cellular processes, contributing to temporal organization of cellular physiology in an unprecedented manner".

The paper of Dechter [DMP91] describes *"Temporal Constraint Networks"*, which is a concept not directly related with our temporal networks concept. However, given its potential utility in solving problems from fields such as Distributed Computing or Artificial Intelligence, it could be interesting to look into how this type of problem-solving systems would benefit from a state-of-the-art temporal networks approach. Especially since temporal constraint networks present many common challenges with the more recent temporal networks approaches, such as verification of path consistency, and temporal problems which the more recently developed measurements may address. If such networks are indeed of use in AI, then with a temporal networks representation and a robust solution to the predictability problem, a new approach to planning could be discovered.

A number of face-to-face interaction systems, such as the one which is used by this thesis, are described in the paper of Barrat and Cattuto[BC13]. Their work brings comprehensive experiments and results in regards to the particularities of representing such systems with a temporal networks model.

Finally, as a continuation of the previously mentioned work of Nicosia *et al.* and Tang *et al.*, a very recent article of Tang *et al.*[TLS$^+$13] provides case studies from different domains and comprehensive results demonstrating the utility of the network measures presented by their previous papers.

1.4.4 Extending the framework

While this thesis focus is on structure prediction in temporal networks, our approach to feature extraction could greatly benefit from extension with methods from the more established and discussed problem of link prediction, adapted to our temporal context and cross checked with the features that we extract from the temporal sequence. Good surveys of the existing link prediction methods and new link prediction approaches can be found in the works of Liben-Nowell and Kleinberg[LNK07], of Lü and Zhou[LZ11] and of Popescul and Ungar [PU03]. A new and comprehensive approach to solving the structure prediction problem that makes use of link prediction is detailed in the more recent paper by Ouzienko *et al.*[OGO11].

A significant abstraction of the temporal networks model can be found in the work of Berlingerio *et al.*[BCG$^+$11] *"Foundations of Multidimensional Network Analysis"*. It argues that, in a multidimensional network there can be multiple edges connecting two nodes; for temporal networks, the edges would correspond to each recorded timestep. The paper further presents relevant measures for the abstraction and tests the proposed framework on a real world multi-dimensional network constructed from search queries submitted by 650,000 users.

In the paper *"Quantifying social group evolution"* Palla *et al.*[PBV07] study the evolution of communities in temporal networks by looking at two datasets, one describing the collaboration between scientists and another of phone calls between users of a mobile network. They provide an algorithm based on the clique percolation method [PDFV05] that allows investigating the dependencies and relationships that influence the evolution of communities. Their results show that while large groups may persist longer when capable of dynamically altering their membership, smaller groups tend to remain stable only while their composition remains unchanged. Community structure is determined using a different approach by Mucha *et al.*[MRM$^+$10], their paper provides a framework that extracts communities based on "combinations of individual networks coupled through links that connect each node in one network slice to itself in other network slices".

Richer datasets can say more about the meaningfulness of certain interactions, a detail of great importance to the accuracy of prediction models. Gilbert and Karahalios [GK09] describe the concept of tie strength, and obtain results by extracting the features of a Facebook data sample of 35 participants; their Facebook dataset is rich enough to allow extracting a wide range of features, from messages, using sentiment analysis, and from profile data, they can extract differences of age, education, etc. Although a dataset of this detail might be hard to acquire on a large scale and for domains other than that of online social networks, the concept of *tie strength* could be developed in a more general model even for less detailed datasets.

CHAPTER 2

Discovering Predictable Patterns in Temporal Networks

In this chapter we present our approach to solving the structure prediction problem. For this, we propose a framework which includes both solving the problem as well as collecting statistics from the results and learning the predictive value of a temporal interactions dataset. At the core of the proposed framework stands the concept of *layers*, therefore the framework will henceforth be referred to as "Layered Predictability Finder", or in short *LPF*.

Previously, in Figure 1.2b, discussed when introducing the dataset, we have shown how the nodes and the edges have a fluctuating cardinality from one snapshot to another. Knowing the nature of the dataset, one could say that showing this through a graphical illustration is not necessary, because it describes an obvious, visible phenomenon. Not all datasets benefit of the same accessibility to an uneducated observer as a social dataset does.

The proposed framework relies on the premise that, having an observable phenomenon, theory or assumption, that characterizes a system, there must always be at least one way to quantify the temporal fluctuation in that system such that, the values of the resulting discrete signal can be mapped to certain states of the known phenomenon. If this is true, then the timesteps corresponding to the mapped values can also classified based on the quantified values of their corresponding snapshots. We name such a classification of timesteps, a *layer*. A reference to a layer will be named a *label*, and the classification process will be denoted as *labeling*.

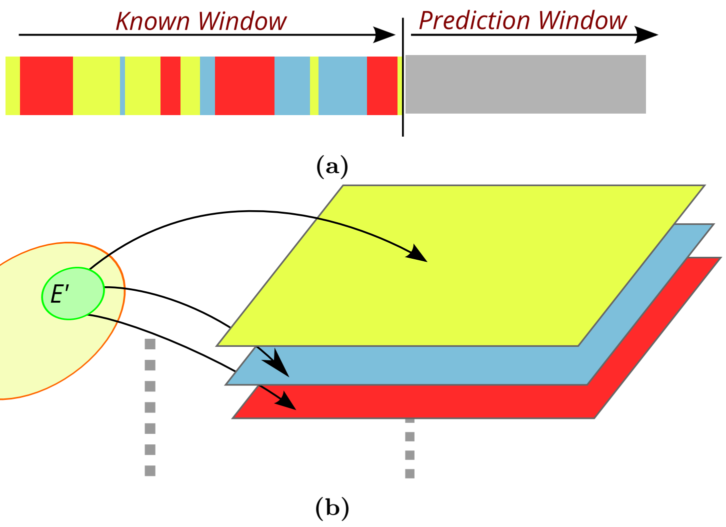


Figure 2.1: A schematic of the proposed framework. Figure *(a)* shows how the timesteps of a temporal network can be classified. For example, a timestep can be *labeled* as *yellow* if the density d of its graph is at least 0.25, *blue* if $d \geq 0.4$ and *red* if $d < 0.25$. *(b)* shows how dyadic interactions from the set E', formed by data that was kept after *noise reduction* are each assigned a layer based on their regularity. A simple example of what this would mean can be exemplified by the following scenario: *Bob usually meets Charlie during morning classes (i.e. blue layer), Charlie only meets with Alice in the afternoon (i.e. yellow layer), Charlie sometimes meets with Bob in the evening (i.e. red layer)*. While $A - C$ is always regular in *yellow*, $B - C$ can be regular in *blue*, yet the interaction may not be regular in the *yellow* layer. In other words, we classify interactions based on the time intervals where they are known to be active most regularly. This will be detailed later on in Section 2.4.

Measuring an observable phenomenon produces a set of layers, these we denote to be a *dimension*. In order for an observable phenomenon to be useful to prediction, it must have a measurement which provides a signal of relatively high variance. If relevant, multiple measurements of the same phenomenon can be combined (e.g. density with diameter), or multiple dimensions (e.g. proximity and phone calls) can be aggregated or separately used in prediction.

In the case of the temporal network formed by our dataset, the observable phenomenon is “the temporal dynamics of face-to-face interactions between students at the DTU Campus”. Had we not known the underlying context in which the dataset was extracted, then the phenomenon could have only been expressed as “the temporal dynamics of proximity interactions between individuals at a fixed location”, yet we known the test subjects of the experiment, the location, and more importantly, that the system’s dynamics are dependent on the academic calendar, furthermore we know that the proximity interactions are actually face-to-face interactions (as described in Section 1.1). The more it is known about a dataset whose predictability is studied, the better are the chances of creating a better performing prediction model.

A schematic of the basic concepts of the proposed framework is shown and described in Figure 2.1, it also introduces the aspect of edge classification. Not every component of LPF could be visually expressed in a drawing, which is why we give an overview of the framework’s main components in Table 2.1, here we also show in advance the assessment of their impact on the prediction performance, and the flexibility of implementation, these are thoroughly described in the sections dedicated to each component. The impact of components on each other is shown by the dependency graph in Figure 2.2.

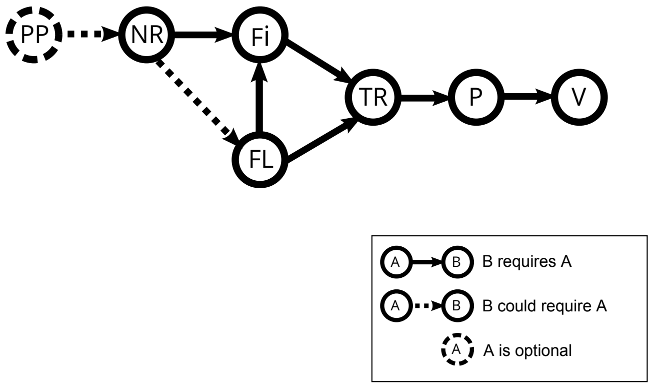


Figure 2.2: **Dependency graph between components in LPF**, node labels stand for the initials of the components presented in Table 2.1. The dependency between components means that a component requires the output produced by another. A dashed circle represents an optional component, and a dotted edge represents an optional dependency (based on the choice of implementation). A component adds its produced data to the existing framework knowledge and this is passed to the next component. The last component (Validation) has access to all the produced data, whereas the first component (Preprocessing or Noise Reduction) will only have access to the raw temporal network.

Component	Description
Preprocessing (PP)	*[Optional]* Performs any tasks that may be required, e.g. resolution adjustments, extraction of data samples, domain-specific adjustments
Noise Reduction (NR)	Discards edges and nodes from the temporal network, if they are *noise*, the definition of noise is variable.
Find Layers (FL)	Uses a variable measurement to label each timestep and create layers.
Find Interactions (FI)	Categorizes the edges into layers and applies an interaction principle. The default interaction principle is that an interaction is formed by a dyad. Changing this definition is meant to join multiple edges in a subgraph, such that interactions become more meaningful, and the computational performance of the components that follow is improved.
Training (T)	Extracts features from the labeled time sequence and from the computed list of interactions. These are trained (i.e. on the known window) using different methods. The component produces a list features with different trust levels.
Prediction (P)	Uses the list of features to create two prediction samples, one containing trusted predictions, and another with negative predictions, made from untrusted features.
Validation (V)	Tests the predictions against a validation sample and computes the predictive performance of the model which was experimented with.

Table 2.1: Overview of the framework components

The rest of this chapter will focus on pattern discovery and describe the components that facilitate it. Prediction and Validation will be covered together with the results in Chapter 3.

2.1 Preprocessing

Preprocessing (PP) has the **purpose** of *optimizing the dataset* before being used for solving a prediction problem. In itself, PP is not a component for which LPF provides a general specification; this stage is free to implement any approach that is found necessary or appropriate.

It is generally recommended that PP does not overlap with other framework components, e.g. *noise reduction*. However, this is not mandatory and may be acceptable, for example, when there is a different (or domain-specific) definition of noise compared what is implemented by the reduction component, and both need to be used.

It is important that domain-specific observations or assumptions are considered here. When a high predictive performance is required, it is better that data known to be unpredictable is removed, allowing LPF to focus on what is unknown or not easily observable, rather than learning from the raw data. By contrast, if a raw dataset with known unpredictable (or outside the scope) sections is given as input some undesired consequences may be observed. First, the processing time will be unnecessarily increased, and second, not everything which is unpredictable may be detected accurately, creating predictions from interactions outside the scope, therefore likely to be inaccurate. A list that exemplifies some of the tasks PP may carry is given below.

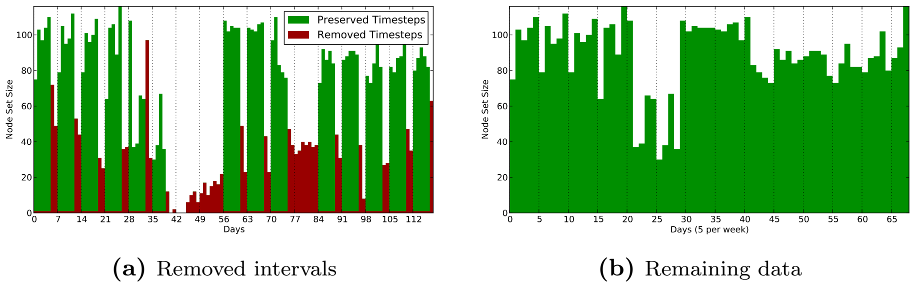
Time sequence filtering. A dataset may contain time intervals for which the underlying interactions may have a different scope than what needs to be predicted; interactions occurring in these intervals may not exhibit the same patterns as the others. If known, it may be beneficial for the results if such time intervals are discarded. If not, these intervals are eventually detected by LPF and treated separately, but this may not always be done accurately.

Extraction of data samples. In the general case, predictions should be extracted from a subset of the data, known as a *training sample*, whereas validation should be done against a distinct subset, known as a *validation sample*. Depending on the validation model, multiple samples can be used for training, or in validation, meaning to provide a better picture of the results.

Resolution adjustments. An experiment should start from a resolution that is *minimally optimal* to the desired prediction scope, e.g. it might not make sense to have a resolution of 10 min. per timestep if we are only interested in predicting interactions across different weeks. This applies only when starting from relatively high resolution datasets, and should be done only when certain that the starting resolution is too high compared to that of the expected predictions. In the absence of any resolution adjustments, LPF automatically groups intervals together, attempting to find an optimal resolution based on temporal variations in network topology.

Data adjustments. Meant to handle any other specificities of the dataset, or fix problems detected in previous experiments; e.g. removing isolated nodes, removing edges with certain properties, applying certain constraints.

In our implementation, PP is used to *a)* extract one or more training samples and a validation sample, as required by the validation model which will be described in Section 3.2, and *b)* to eliminate time intervals which are outside the prediction scope, in our case these are the winter holiday, the semester break and the weekends. Although weekends are recurring intervals, interactions tend to have a different scope and are likely to exhibit different patterns, removing them from consideration provides extra predictive performance because the scope of prediction is narrowed to interactions occurring during study weeks, which is the desired scope for our experiments. The eliminated time intervals are illustrated in Figure 2.3a.



(a) Removed intervals **(b)** Remaining data

Figure 2.3: The dataset, with each bar representing a 24 hrs. interval and its height representing the number of nodes (i.e. individuals) interacting on that day. In *a)* the red timesteps are those that belong to intervals outside the scope, which have been eliminated, *b)* shows the remaining dataset. It is observable that the two weeks period before the winter holiday (before middle), which represents the exams period and is preserved in the dataset, shows a decrease in the number of nodes, around the same level as the semester break (after middle) which is, on the other side, eliminated. The decision assumes that the exams period is more likely to exhibit the same interaction patterns, than it is in the case of the semester break. This assumption will be tested later.

2.2 Noise Reduction

Noise Reduction (NR) has the **purpose** to discard edges which are considered to be *noise* from the set of edges E (of edges occurring in the raw temporal network). It should be implemented so that, after all the noisy edges are removed, the edges that remain have a high predictability potential. The most simple way an implementation would define this is by using the probability of an edge to occur, i.e. the number of known occurrences (timesteps when an edge is active) divided by the number of known timesteps. However, more complex or specialized ways of computing the predictability potential may give better results.

2.2.1 General Specification

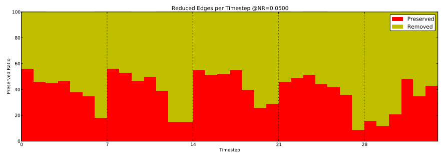
Given a temporal network $\mathcal{G} = \langle G_1, \ldots, G_T \rangle$ where every $G_t = (V_t, E_t)$ for $t \in [1, T]$ and $E = \{E_1 \cup \ldots \cup E_T\}$, we want to compute E' such that $\forall\, e \in E'$, $Pred(e) \geq NR$, where $NR \in (0, 1)$ is the predefined noise threshold and $Pred(e)$ is the *predictability potential* of an edge e. We subsequently redefine $\mathcal{G}$ such that every $E_t \subseteq E'$.

Removing edges is likely to create isolated nodes, these nodes may not always represent infrequent or unpredictable entities, however, it is minimally true that they have no frequent connections with the other nodes. Therefore the isolated nodes do not fit in the scope of the structure prediction problem and should be discarded as well.

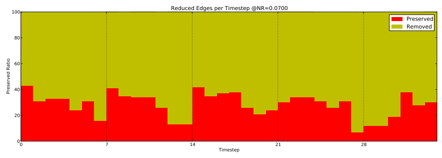
2.2.2 Implementation

For our proximity dataset, we experiment with two simple implementations. First one is to compute the probability of edge occurrence in the known window, that is, the number of *known occurrences* divided by the *number of known timesteps*. We discard edges whose probability is less than a `noise-ratio` parameter. `noise-ratio` is chosen in relation with the number of timesteps in the temporal network (influenced by size and resolution of the dataset). Therefore, *predictability potential* is defined here as the probability of an edge's occurrence.

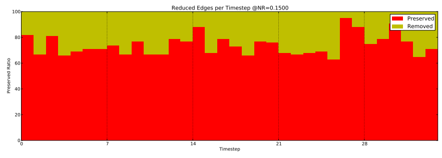
The second implementation is similar, yet we look at the number of occurrences of nodes instead, and we only discard an edge if one of its nodes is not frequent enough. We define $Pred(e) = min\left(P\left(n_i\right), P\left(n_j\right)\right)$ where $P\left(n_i\right), P\left(n_j\right)$ are the probabilities of nodes connected by e, as previously defined for edges. In other words the *predictability potential* of an edge is the occurrence probability of its least frequent node. With this definition, only edges connected to a non-frequent node are discarded, thus their number is reduced relative to the first method at the same `noise-ratio` (as shown in 2.4). This also provides control over which nodes are discarded, solving the previously mentioned problem of leaving frequently occurring nodes isolated.



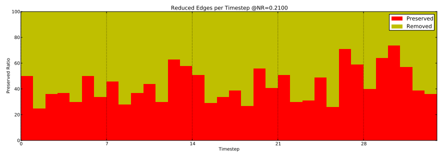
(a) $NR = 0.05$ (`edges`)



(b) $NR = 0.07$ (`edges`)



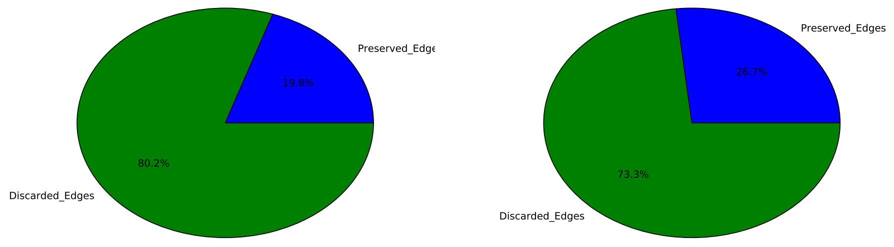
(c) $NR = 0.15$ (`nodes`)



(d) $NR = 0.21$ (`nodes`)

Figure 2.4: The ratio of *preserved* (colored red)/*discarded* (colored yellow) edges at each 24 hrs. timestep within the first five weeks. *a)* and *b)* show the number of discarded edges using edges probability at `noise-ratio` values of 0.05 and 0.07. *c)* and *d)* show the same ratio when using the least frequent nodes method and `noise-ratio` values of 0.15 and 0.21. They describe how different definitions for what is potentially predictable may work with relatively different values of `noise-ratio` should we require discarding about the same amount of edges. This may be required because it determines the number of edges that are kept, which has a high influence upon the computational efficiency of the other components. However, the implementation method plays the more important role, differences between the behaviors of the two noise reduction methods should be visible when comparing *a)* and *d)*. Although the two discard roughly the same quantities of edges (as shown in Figure 2.5), the difference between their discarding behaviors is visible at certain timesteps.

In the context of our dataset, the first method discards all dyadic interactions that do not occur frequently, whereas the second method discards interactions in which one of the individuals does not frequently occur (and therefore no interactions with him/her can be frequent), this leaves as potentially predictable, the less frequent interactions between individuals who are frequently at the campus. In other words, even though two individuals that are frequently present on campus may not meet very often, their interaction patterns may still be predictable (they may be meeting only once a week for a certain class).



(a) $NR = 0.05$ (`edges`) **(b)** $NR = 0.21$ (`nodes`)

Figure 2.5: The percentage of discarded (in green) and that of preserved (blue) edges, from the raw set E, are shown for the experiments in Figure 2.4a and Figure 2.4d.

Although the second method was not initially expected to perform better, and only to exemplify a solution to the isolated nodes problem, it does substantially perform better, as shown by the comparison made in Table 2.2 between the accuracy of positive occurrence predictions (PPV[1]) (correct occurrence predictions, divided by total occurrence predictions) in scenarios where the only different parameters are the `noise-ratio` and the `reduction-method`. We can relate this to the context of our dataset and assume that, students that frequently come to the campus (attend classes, etc.) even though they do not have frequent interactions are very likely to have regular, predictable interactions, whereas the non-frequent students are most likely to be random in their pattern of coming to the campus. We will validate this assumption further in Chapter 3.

Ref.	Method	NR	Total Pred.	True Pos.	False Pos.	PPV
1	`edges`	0.05	250	28	222	0.11
2	`edges`	0.07	527	17	510	0.03
3	`nodes`	0.15	271	90	181	0.33
4	`nodes`	0.21	95	62	33	0.65

Table 2.2: Relative prediction performance of the scenarios shown in Fig. 2.4. All scenarios use the same parameters excepting those mentioned in the table. Similarly to how the **noise-ratio** threshold needs to be appropriate for the reduction method, certain other components are influenced as well, but with substantially less impact. Which is why it becomes clear, by comparing scenarios *1)* and *3)* that the method of reduction starting from nodes outperforms the other. With approximately the same amount of predictions produced as the scenario *1)* the third scenario produces better quality predictions.

[1]We will detail *positive predictive value* along with other validation methods, in the next chapter

2.2.3 Discussion

Flexibility. More complex ways to define the predictability potential can be devised, these may suit certain domains more than others. Some concepts that may be of use in inspiring new definitions could be: the size and regularity of a clique combined with the frequency of the edge or with certain features of its community, belongingness to a certain path, degree of the nodes which are connected by an edge, etc. The number of possible approaches that can be experimented with might only be limited by the available theory.

Impact. For experiments on large datasets, defining a noise ratio that preserves a large percentage of the raw edges could affect the computational performance of the components that follow, which depends on the algorithms that are implemented. There is still progress to be made with solving problems that are relevant in this case, such as finding maximal frequent itemsets or extracting certain features optimally from big data.

Performance. For the described implementation, a frequency dictionary is built by counting the occurrences of a node (or edge) in $\mathcal{G}$, which is then used to remove the nodes or edges at each timestep, since the node set is significantly smaller, building the frequency dictionary is substantially faster using the nodes approach, therefore we could say that with such implementation the component is relatively lightweight.

Noise reduction in general can remove unnecessary load from big datasets, letting the other components focus on predicting interactions which are meaningful. Even with efficient implementations of the other processing components, having a noise reduction stage will still provide benefits. Depending on the percentage of the edges that an experiment can afford to keep, this stage can either have focus on selecting the most meaningful edges (as is our case), or if the efficiency is not important (e.g. for small datasets), the focus could be only on what is being discarded.

2.3 Finding Layers

Layers of a temporal network contain recurrent time intervals in which timesteps have a similar state in relation to a certain known phenomenon. Our proximity dataset provides good examples: at high resolutions (1-8 hrs. per timestep) layers would be based a regular circadian rhythm, and intervals would rely on the interaction patterns formed by individuals particularly during study hours, at a 24 hrs. resolution it would only be able to differentiate weekends from the rest of the weekdays, and similarly for 7 day resolutions it would be able to differentiate holidays, and trends that occur during exam periods would become more visible. One resolution can not efficiently differentiate everything, although clearly the highest resolution would perform best. Intervals of lower activity[2] such as weekends are most likely to have, on average, the same length and therefore their occurrence patterns can be learned from.

Having time intervals extracted and classified, allows us to study the interaction patterns in relation to these time layers. Whether the interactions occur at different time intervals or whether they are likely to regularly occur within the same layer should be part of domain knowledge, but in any case the intervals during which entities would interact are very likely to display certain – more or less complex – patterns. In the discussed proximity network, using a layers-based approach we can observe interactions to be mainly distributed in the layer that would correspond to study hours, and therefore most of the predicted interactions are also likely to occur within that time interval, several students are only interacting after classes, and therefore their interaction is likely to occur in that interval rather than any other.

The problem of layer-based prediction can be split in two, in the first place, the time intervals that form the layers have to be discovered and, having this done, interactions must be classified based on the layer(s) in which they occur most regularly, this section will focus on the the general concept of layers and on their discovery.

2.3.1 General Specification

Before formally introducing the concept of layers we define the basis on which it builds, an abstraction of the temporal networks concept named *labeled temporal network*.

Definition 2.3.1 (Labeled temporal network) A representation $\mathcal{G}'$ for a temporal network $\mathcal{G}$, is called a *labeled temporal network* given that $\mathcal{G}' = \langle G_{t_1}, \dots, G_{t_i}, \dots, G_{t_N} \rangle$ with $\mathcal{G}_{t_i} = (V_{t_i}, E_{t_i})$ the sets of edges and nodes occurring in a snapshot G_{t_i} where $t_i = (i, \sigma_i, \lambda_i)$ is a labeled timestep, in which i is the *ordering* of the snapshot within the time sequence, σ_i is the *size of the interval* in which $\mathcal{G}_{t_i}$ occurs, and λ_i is a label which corresponds to a certain set of results obtained from applying a predefined network measurement on G_{t_i}.

[2]When referring to *activity* in our dataset, we refer to the number interacting nodes (students).

For a labeled temporal network $\mathcal{G}'$, V' and E' are the complete sets of nodes and edges occurring at a labeled timestep in $\tau(\mathcal{G}') = \langle t_1, \ldots, t_i, \ldots, t_N \rangle$. Each labeled timestep t_i corresponds to an interval of standard[a] timesteps in $\mathcal{G}$ as follows: $t_i \equiv \left[\left(\sum_{j=1}^{i-1} \sigma_j \right) + 1, \sum_{j=1}^{i} \sigma_j \right]$, where the symbol $\equiv$ is used to express equivalence.

[a]We refer to the timesteps of a temporal network based on the standard definition as *standard timesteps*, when discussed in the same context with labeled timesteps.

Labeled timesteps can have a fixed or a dynamic size, depending on the implementation. By setting the size restriction $\sigma_{t_i} = 1,\ \forall t_i \in \mathcal{G}'$, the labeled temporal network becomes equivalent with the standard temporal network $\mathcal{G}$.

Definition 2.3.2 (Layer) In a labeled temporal network $\mathcal{G}'$, a subset of $\tau(\mathcal{G}')$ in which all timesteps have the same label λ is defined as a *layer*, $L_\lambda = \{t_i = (i, \sigma_i, \lambda_i) \mid \lambda_i = \lambda\}$.

2.3.1.1 Representing $\mathcal{G}$ as labeled temporal network

First step in converting a temporal network $\mathcal{G}$ into a labeled representation is to compute ϕ – a discrete signal for the temporal network – by measuring each snapshot using a function $m : G \to \mathbb{R}$. The network measurement should reflect the change of states in the system in relation to an observable behavior which must to be predicted. A few papers that discuss network measures that may be applied here have been previously described in Section 1.4.2. A *labeling function* is used afterwards to map the time-value pairs of the resulting signal to labels that represent the states of that observable behavior, we denote this set of labels with Ω. An overview of this process is shown in Listing 2.1.

```
Let
  $m : G \to \mathbb{R}$ be a graph measurement function
  $\tau(\mathcal{G})$ be the set of timesteps in $\mathcal{G}$ and $\Omega$ a set of labels
  $lf : (\mathbb{R}, \tau(\mathcal{G})) \to \Omega$ be a labeling function

For $\forall\ G_t \in \mathcal{G}$
  Compute $\phi_t \leftarrow m(G_t)$
  Compute $\lambda_t \leftarrow lf(\phi_t, t)$
```

Listing 2.1: An overview of the described labeling process, functions and notations

Computing a label for every timestep will result in a sequence of labels $\langle \lambda_1, \ldots, \lambda_T \rangle$. If useful, this sequence can be processed using heuristic rules (adjustments) established by the implementation. For example, when a certain label $\lambda_i = \mathcal{L}_1$ is surrounded by two subsequences $\langle \lambda_{i-x}, \lambda_{i-1} \rangle$ and $\langle \lambda_{i+1}, \ldots, \lambda_{i+y} \rangle$ where each $\lambda = \mathcal{L}_2$, than it may be acceptable for λ_i to be relabeled as $\mathcal{L}_2$ and be assimilated into the sequence, instead of splitting it in two. While this example is meant to be simple, more complex adjustments are possible and may be relevant for certain datasets.

Labeled timesteps can be computed differently, and it's up to a specific implementation to choose the best method. There are two general methods: dynamic-size and fixed-size timesteps.

Dynamic-size labeled timesteps. A subsequence of consecutive labels $\langle \lambda_i, \ldots, \lambda_j \rangle$ forms a *labeled timestep* t_k with size $\sigma_k = j - i + 1$ if all the labels in the sequence are identical, $\lambda_i = \lambda_{i+1} = \ldots = \lambda_j$. For example, having labels A and B and a label sequence $\lambda^{ex} = \langle A, A, A, B, B, A, A, A, A, B, B, B \rangle$, would allow computing a sequence of four labeled timesteps of different sizes: $\langle (A, 3), (B, 2), (A, 4), (B, 3) \rangle$.

Fixed-size labeled timesteps. In this case, each subsequence is limited to a fixed size, σ. Such a subsequence forms a labeled timestep with a label λ if λ is the most frequently occurring label in that subsequence. Computing labeled timesteps with a fixed-size requires domain knowledge in order to properly apply. For the previously exemplified sequence λ^{ex}, having a fixed size of $\sigma = 6$ would provide difficulties in computing the label of the second timestep, since both A and B occur in equal measure, by contrast a value of $\sigma = 4$ would provide three timesteps where each label can be computed, having 3 occurrences in the case of each timestep formed from $\langle A, A, A, B \rangle$, $\langle B, A, A, A \rangle$ and $\langle A, B, B, B \rangle$.

Good results can therefore be obtained with a fixed-size approach, only when certain intervals are known to have a prevailing label, thereby producing a normalization of interval sizes. A good example coming from our dataset is that, knowing that most of the activity occurs during study hours, setting a fixed interval size of 8 timesteps at resolution of 1 hr per timestep, would compute timesteps corresponding to the following intervals of the day: 0 am–8 am, 8 am–4 pm and 4 pm–0 am; the three intervals show different behaviors in relation to the activity of students at the campus.

While the dynamic-size approach would be the preferable approach, predicting sizes of intervals may not always be simple. In the case of our dataset and domain, the intervals where the activity of students is high is in most cases easy to predict, exactly as it would make sense it can between 9 hrs (near exam periods, Mondays, etc) to 6–7 hrs (on Fridays), which makes a dynamic-size approach give a good predictive performance. Having knowledge that the activity tends to fluctuate in intervals of ∼8 hrs, similar results may seem achievable with fixed-size timesteps. Chapter 3 will, however, demonstrate that predictive performance is improved with dynamic-size timesteps.

It is also important to remark that *a)* in dynamic-size timestep sequences no two consecutive timesteps can have the same label, whereas in fixed-size timestep this is acceptable, and it may lead to predicted timesteps being mislabeled, this can be acceptable unless it's meant to be explicitly studied (not part of the structure prediction problem) or unless the prediction model explicitly uses the labels to create predictions (i.e. will not predict an interaction unless the predicted timestep has the same label as the layer in which the interaction is regular in). Also, *b)* if fixed-size timesteps create a loss of precision, the sizes of dynamic-size timesteps can be wrongly predicted, this is worse than mislabeling and can have a chain-effect on the other predictions. Generally speaking, the best method of creating labeled timesteps would therefore be that which has the least negative impact on the results.

By having obtained the labeled timesteps of $\mathcal{G}'$, a subset of timesteps where all timesteps have the same label is a layer. Section 2.4 will deal with classifying interactions into each of these labeled timesteps in order to create the actual graphs that make the snapshots of $\mathcal{G}'$.

2.3.2 Implementation

With a dataset which observes the dynamics of face-to-face interactions at the university campus, we can infer that most interactions occur when a large number of students is present on campus, that is because students are most likely to attend courses together. We can further observe that the number of interactions (edges) is not always proportional with the number of students (nodes), this may be because they are spread at different locations in the campus (perhaps during lunch time).

Also it is noticeable that the variation in the number of students at the campus is given by the 24 hours cycle, weekends, and other

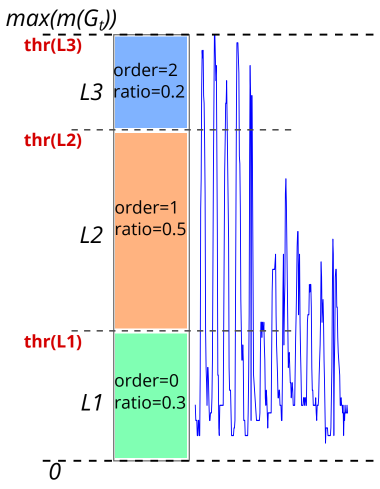
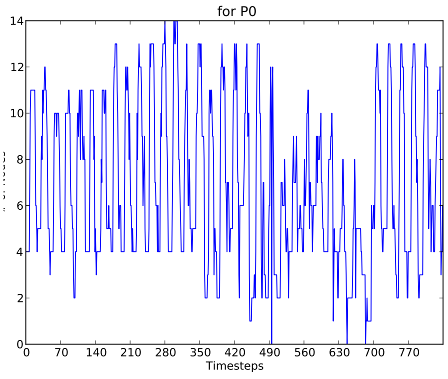
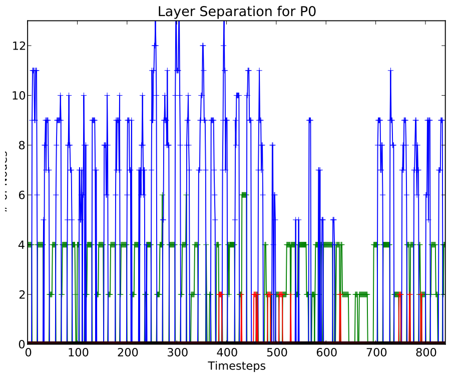


Figure 2.6: An example of how labeling is done using thresholds defined by the ratio and ordering parameters of the layer.

academic events such as holidays and exam periods. Since we can describe this variation based on the number of students, our chosen network measurement is the *number of nodes* active in a snapshot, so that $m(G_t) = |V_t| \;\forall\, G_t \in \mathcal{G}$.



(a) Raw signal



(b) Labeled timesteps

Figure 2.7: Creating labels from the raw signal. Figure *a)* shows the raw signal ϕ, based on which timesteps are labeled as illustrated in Figure 2.6, and the result of such a process is shown through the colored signal sequences of *b)*.

Having computed the discrete signal for the temporal network $\mathcal{G}$ using the given measurement, we map the values at each timestep to different states of activity. The set of states could for example be, $\Omega = \{high, mid, low\}$. Regardless of their explicit names, they are defined by a *layer ratio* and *layer ordering* attributes based on which a *layer's threshold* we denote with $thr(L_\lambda)$, is computed, as illustrated in Figure 2.6.

Given a set of labels $\Omega = \{\lambda_0, \lambda_1, \ldots, \lambda_K\}$ where the index of each label represents (in this example) the ordering of its corresponding layer and ϕ is a discrete signal based the graph measurement m, we define our labeling function lf as shown in Equation 2.1.

$$lf(m(G_t), t) = \begin{cases} \lambda_0, & \text{if } 0 \leq m(G_t) < thr(L_{\lambda_0}) \\ \lambda_1, & \text{if } thr(L_{\lambda_0}) \leq m(G_t) < thr(L_{\lambda_1}) \\ \vdots & \vdots \\ \lambda_K, & \text{if } thr(L_{\lambda_{K-1}}) \leq m(G_t) < thr(L_{\lambda_K}) \end{cases} \tag{2.1}$$

In addition to implementing both dynamic and fixed-size methods for timestep labeling, we also implemented an *automatic setting*, which in essence, uses a fixed size timestep approach based on the average size of all timesteps pre-computed with the dynamic size method. Given that for our dataset, this computed size provided realistic values (8–12, depending on the size of the known window), we could premise that the computed size is also an *optimal resolution*, and that using a labeled temporal networks representation to extracting it could be a valid alternative to the approaches presented in [SBWG10] and [CE12]. This will be further looked at in one of the experiments presented in Chapter 3.

2.3.3 Discussion

Using layers allows grouping timesteps in relation to states of the variation that has to be predicted. By zooming out to the states observed during larger intervals, it provides a good way to observe time sequence patterns. The fixed size method facilitates testing the predictability in a more traditional fashion, at different resolutions, without explicitly changing it in the data set.

2.4 Interactions

The **purpose** of this component is twofold: one is to *classify* edges into layers and the other, more complex, is to *aggregate* edges into larger subgraphs which still hold a close meaning to that of dyadic interactions.

Classifying edges is straightforward, for each edge in the complete set of edges (E) we calculate its regularity in a layer. Regularity may use a similar approach to noise reduction: computing the probability of an edge to occur within a layer as its support in that layer divided by size of the layer. Starting from nodes or using any other concepts is also a possibility. Similarly to the noise ratio parameter, the `regularity` defines the minimum threshold which must be calculated in order for an edge to be considered regular. The regularity definition is not fixed.

An edge can be regular one or more layers, or in none, in which case it is denoted as *irregular*. This happens when its activity is distributed through multiple layers, but in none of them shows regularity. Classifying edges as irregular, in essence, acts as an additional level of noise removal (with a more complex definition), since none of the edges found irregular are used (at least by our implementation) for prediction making.

For large datasets in which the set of edges is high, training pairwise dyadic interactions would require performing $|E|^2$ operations. This brings a very high price on the computational performance, one which is not always justified by an increased quality in the results. Therefore in order to solve this issue, multiple dyadic interaction should be meaningfully aggregated into a larger one. The meaning can be domain dependent and therefore the definition of how the larger interaction is created should be given by the implementation. Providing a meaningful definition with an *efficient* algorithm to compute the larger interactions might not always be trivial (see Section 2.4.3).

2.4.1 General Specification

There are no constraints regarding the order in which the two goals of this component are completed, however in most cases edges should first be classified, such that those which are irregular can be discarded allowing larger interactions to be formed only based on the edges which are regular in a given layer.

2.4.1.1 Edge classification

To classify edges, a *regularity ratio* should define the minimum value which a *regularity function* computed for an edge must return in order to consider the edge regular in a layer. The regularity function should depend on the size of the layer and the occurrences of the edge within the timesteps of that layer. The same methods used in noise reduction may not always prove suitable here. An edge may be regular in one or more layers or it can be irregular; ideally, most of the edges should have only one corresponding layer. By classifying the edges, we can compute the sets E' and V', allowing for the labeled temporal network $\mathcal{G}'$ to be completely created.

Each edge regular in a layer will be active at the labeled timesteps that correspond to the timesteps where it was originally active. When distributing edges back into the timesteps of $\mathcal{G}'$, in some cases an edge would be assigned to an interval in which it's occurrence length was small. For example, consider an edge e which has been found regular in a layer L_λ and previously occurred between timesteps $10 - 12$, yet now these timesteps are part of a timestep labeled λ of size 20. Assigning the edge to a labeled timestep when its occurrences are few compared to the size of the timestep may not always be acceptable. Unless timesteps don't generally have large sizes or unless the domain allows it, it is a good approach to verify that the edge meets a minimum number of occurrences (e.g. computed using the regularity ratio and the size of the interval) within that interval as well.

2.4.1.2 Interactions as subgraphs

In most cases, an interaction in a temporal network naturally refers to the occurrence of a connection between two nodes (dyadic interaction). However in particular domains an interaction can also be meaningfully defined by connections between more than two nodes.

When extracting predictable features from the dataset, it is most likely that the relations between interactions may need to be investigated. In most cases, there is no relation between individual dyadic interactions, and a lot of time is spent on unproductive computations. To address this problem a solution is to look at interactions as more meaningful units than individual edges. A general definition of these units is given in 2.4.1.

Definition 2.4.1 (Interaction) A subgraph G' is an interaction if it has a property $IP(G')$. This property is the *interaction principle.*

Basically, an *interaction principle* is defined by the implementation and subgraphs that satisfy it are considered to be interactions themselves. This is exemplified further by our implementation and the alternate approaches to be discussed in the rest of this section.

The timesteps at which interactions occur will be of use in studying the predictable patterns (temporal relations) formed by two interactions. Therefore we define the concept of *support set* of an interaction in 2.4.2. The definition was first introduced by Lahiri *et al.* [LBW07].

Definition 2.4.2 For any arbitrary interaction G' we define the *support set of* G' as the set of timesteps in which G' occurs, $S(G') = \{t \mid G' \subseteq G_t\}$. The cardinality of the support set, $|S(G')|$, is called *the support of* G' in the temporal network $\mathcal{G}$.

2.4.2 Implementation

2.4.2.1 Edge classification

To classify edges in our dataset we use as a regularity function the number of occurrences of that edge in the timesteps of a layer relative to the size of the layer, as defined in 2.2, where e is an edge and $S(e)$ its support set.

$$Regularity(L_\lambda, e) = \frac{|S(e) \cap L_\lambda|}{|L_\lambda|} \tag{2.2}$$

The decision whether an edge is regular or not is taken using a *regularity threshold* $R_{min} \in (0, 1)$. At a resolution of 1 hrs. per timestep, this would correspond with the total length of interactions that have occurred within the time intervals of a layer. Keeping the alternate method used by noise reduction – that is, classifying edges based on a regularity of nodes – would constrain the nodes to certain layers, which is not preferable since certain students may have different regular interactions in different layers, therefore in this case it is a better choice to look at edges instead of nodes.

Each edge found to be regular in a layer is active in the graph at a labeled timestep t_i if it is active in at least $Ceil(R_{min} \times \sigma_i)$ standard timesteps included in the labeled timestep (R_{min} is the regularity threshold). For example, with a regularity threshold of 0.2, in a layer with a size of 200, an edge would have to occur at least 40 times to be regular in the layer, and given a labeled timestep of size 10 in the same layer, the edge would have to occur at least $10 \times 0.2 = 2$ times in that interval in order to be active.

2.4.2.2 Interaction principle

Since our dataset consists of students, we can have face-to-face interactions registered during classes and during group work. Dyadic interactions are here part of more complex interactions where multiple students are all interconnected. Even in large class-rooms, where they may sit at a distance slightly larger than the Bluetooth range, students that work together as a group are more likely to sit together and be mutually interconnected during classes. Outside classes on the other hand, we have fewer interactions, likely to be fragmented into smaller components where large mutual interactions are rare.

Based on these observations, the *interaction principle* we implement first looks at the fragmentation level of a snapshot at a given time. If the snapshot is too fragmented, interactions keep their default definition and are created from *dyads*. During high activity intervals, when snapshots are more dense, interactions are defined as subgraphs formed by *cliques*. The fragmentation factor for a snapshot G_{t_i} is defined in 2.3.

$$\mathcal{F}_{t_i} = |E_{t_i}| \times \frac{1}{Conn\,(G_{t_i}) \times \alpha} \tag{2.3}$$

$Conn(G_{t_i})$ gives the number of connected components of the snapshot, and the coefficient α is the expected minimal size of the component, measured in edges. More specifically, having $\alpha = 3$ would mean that at least 3 edges per component (on average) are expected in order to have a fragmentation that is less than 1.0, which is the threshold based on which the interaction principle considers interactions to be formed by either dyads or cliques.

To keep the interaction principle uniform throughout each layer, the level of fragmentation of the entire layer can be calculated by aggregating together each of its labeled timesteps and computing it for the aggregated graph. If this does not work as expected (since aggregating fragmented graphs might result in a dense graph), the fragmentation values obtained from each snapshot in the layer can be averaged. Regardless of how the fragmentation is computed for each layer, it is important that the correct choice is made about what definition is used for its interactions (this can be verified fairly easy, by analyzing the topology of the aggregated snapshots of each layer).

If for a given layer, it is chosen to consider interactions as those being formed by cliques, the regularity of these interactions within the layer should be reevaluated. Some cliques are indeed groups of individuals working together regularly while others may be formed arbitrarily from regular interactions which are not also forming a regular clique (i.e. regular interactions which do not usually co-occur). Therefore a regularity value is computed for each clique with the same regularity function defined in 2.2, excepting that in this case, the parameter e is the subgraph formed by a clique. Furthermore, a new regularity threshold must be defined for cliques, since aggregated interactions cannot be expected to occur with the same regularity as dyadic interactions.

For the same case, each dyadic interaction can also be considered a clique but *only if* this interaction does *not belong to any regular clique with more than two nodes*, then – since the edge is most regular in the clique formed only by its own nodes – the edge can be considered a regular clique. Otherwise, a regular edge must not be considered a regular clique as long as it belongs to any larger clique that is above the minimum regularity threshold.

2.4.3 Discussion

Flexibility The interaction principle can be defined in many ways, as long as it can be verified whether or not a subgraph satisfies it. Secondly, it should be meaningful to the system that is studied. Thirdly, for a given graph – the number of subgraphs that satisfy the principle should be smaller than the number of edges. For certain domains interactions can be defined, for example, as paths, closed walks, bipartite components, or by any other rules which can possibly define a subgraph meaningful to the studied system.

In previously discussed work of Lahiri *et al.* [LBW07], an interaction is defined for multiple domains as a *frequent subgraph*, as extracted from all timesteps using an Apriori-like maximal frequent itemset mining algorithm by Burdick *et al.*[BCG01]. This was also our original approach to defining interactions. However, the mining algorithm proposed could not sufficiently scale to the size of our dataset and computational performance was very low[3]. For the same approach a different mining algorithm proposed by Han *et al.*[HPY00] was also tested yet with similar results. The problem of frequent itemsets mining has been demonstrated to be #P-complete by Guizhen Yang[Yan04]. By contrast, the algorithm for extracting maximal cliques performs in linear time, as demonstrated by Bron and Kerbosch [BK73].

Performance This component may include some of the most computationally intensive operations of solving the structure prediction problem. In our current implementation edge classification is one such operation, whose time increases with the number of edges, but a lot of space for further improvement should exist. In regards to the interaction principle, based on the experience from working on this thesis, we could premise that it may be generally better to aggregate edges based on simple, domain-specific heuristics instead of aiming to keep a general implementation with expensive algorithms that do not scale. To test this premise, other options of general algorithms should be investigated, and if one or more such algorithms that scale to large datasets are found, they should be compared with domain specific approaches for a number of different datasets.

Impact Edge classification adds a second verification level – to edges which were not discarded as noise – which is more specific to the variation of the

[3]Some of the initial experiments would take hours to complete

dataset. Furthermore, aggregating edges based on an interaction principle lowers the number of pairs for which temporal relations must be investigated, and if well chosen, predictions are also likely to become more meaningful.

2.5 Training

The *training* component extracts features from the labeled temporal network $\mathcal{G}'$ with interactions from each timestep aggregated as previously discussed. Any available characteristics of the patterns formed by interactions or by the sequence of labeled timesteps can be used to generate predictions. There are no constraints on what types of features should be extracted, but it is important that at least a category of features gives information that can be used to predict interactions (by contrast, other features may predict characteristics of the time sequence that can be used to validate a set of predicted interactions, but do not provide these interactions themselves).

Along with features extracted from the temporal patterns, features can also be extracted only from the topology of certain snapshots, using well established methods from link prediction. To exemplify a few cases, features could discover relations between interactions, such as relative influence, conditional behavior or common entities; mapping of interactions to time, particularly to smaller sections such as quarters, make it possible to learn what interactions emerge and which interactions disappear near to the end of the known window. Finally, if the labeled timesteps of $\mathcal{G}'$ are dynamically sized it is important to learn these sizes and the order in which they occur, otherwise valid interaction predictions may be placed in the wrong time intervals (i.e. if we make predictions into fixed size intervals).

2.5.1 General Specification

Having the labeled temporal network $\mathcal{G}'$ we use it as a training set and extract features from the temporal sequence $\langle t_1, \ldots, t_N \rangle$ and from the interactions identified in each snapshot G_{t_i}. Features can be extracted from:

- Recurring patterns formed by support sets of interactions (relations)
- Recurring patterns formed by labeled timesteps (label variation)
- Topology features in a snapshot or in aggregated snapshots (link prediction)
- Combinations between the above

The main goal for extracting features is being able to build an extension of the temporal network consisting of timesteps and interactions predicted at unseen timesteps. Features which can be linked to one or more known timesteps and explicitly describe an interaction occurring at an unseen timestep, will be denoted as *primary features* and are grouped by the timesteps they are extracted from, as a sequence $\langle F_{t_1}, \ldots, F_{t_N} \rangle$. All primary features must have a trust value $\Phi(f)$, whose definition must be provided by an implementation, such that a minimum trust threshold Φ_{min} can classify these features as either trusted or untrusted.

Auxiliary features may also be extracted, and may count as extra variables in the definition of trust. While these do not predict interactions by themselves, they may be used, for example, to identify certain trends and have a "vote" on whether an interaction is trusted or not. These can play an important role in the classification of features. If no good primary features are found, yet there are many auxiliaries that can be found, a naïve prediction model could be an appropriate solution, i.e. a model that assumes every interaction can occur in every unseen timestep, leaving the auxiliary features to decide how much of this is true.

2.5.2 Implementation

The first two feature extraction sources are used by our implementation. First, we discover features of the time sequence, which allow predicting the size (if dynamic-size timesteps are used) and label of future timesteps; trends can also be extracted since the activity of interactions varies between sections (usually quarters) of the labeled time sequence. These trends and patterns are the *auxiliary features* which are correlated with the relations between interactions. Before discussing these relations, we enumerate the features which have been identified and are extracted from the time sequence.

- Timestep sizes. For each layer it is computed what are the most common sizes that timesteps in the layer have. If a fixed-size timesteps method is used, than this is known by default with maximum reliability.
- Neighbors. For each layer it computes what are the most common labels of timesteps that surround a timestep from the layer it is computed for (i.e. what layers do succeeding and preceding timesteps commonly belong to).
- Regular differences between occurrences of timesteps in the same layer measured in *a)* labeled timesteps and *b)* size of labeled timesteps (std.

timesteps).

- Occurrences per section of the labeled time sequence and probability per section. Helps us compute trends based on how often has a timestep occurred in e.g. the last quarter, what was its probability to occur in that quarter and how is it relative to the other quarters.

The *primary features* are extracted from the *trigger-forecast relation* between interactions. This relation was first described in [LBW07] and considers that one interaction can be a good predictor for another if there is a regular time difference, a constant *delay* between them. For our dataset, examples could be extracted from how a single student interacts throughout a week: in different days the student may interact with other students from different groups, since different courses are attended. If there are best friends with whom a student regularly meets after classes then it is likely that the time differences of that interaction and its previous occurrence is regular, and we can predict it based on its own delays.

Delays are computed by a function of two support sets as shown in Listing 2.2.

```
Input: The support sets of two interactions where:
        $x_t$ considered to be the trigger
        $x_f$ considered to be a forecast

For each timestep $t_\alpha$ in $S(x_f)$:
  For each timestep $t_\beta$ in $S(x_t)$:
    Compute $\Delta(t_\alpha, t_\beta) \leftarrow t_\alpha - t_\beta$

Let $\Delta_v$ be a list of valid delays
For each $diff$ in the $\Delta$ matrix:
  if $diff > 0$:
    $\Delta_v \leftarrow diff$

Return $\Delta_{frq}$, the list of mode values of $\Delta_v$
```

Listing 2.2: The function for computing frequent delays

The size of main diagonal of matrix Δ is given by $min(|S(x_t)|, |S(x_f)|)$. This value represents the maximum number of possible occurrences of a delay between the two interactions, and we also denote it as the *size of a delay*. Therefore the probability of each valid delay δ occurring in Δ_v is the number of occurrences of δ divided by the size of the diagonal, the delays from Δ_{frq} have the highest probability and we only consider these for each pair of interactions; the probability of a delay should not be confused with their trust value, which will be defined later. The distribution Δ_v can be multi-modal or it can also be empty, if the interaction considered to be a forecast only has occurrences before those of the trigger. In such cases the two interactions are not in a trigger-forecast relation.

The delays can be computed between pairs of interactions in the same layer, or between all pairs of existing interactions. The first approach can reduce the number of computed delays but would restrict the relations to a single layer, since it would not be computed between interactions from different layers. If only one layer has the majority of interactions, this has no impact.

The probability of a delay alone may not always be an appropriate measure for the trust we assign to a delay. The trust value can be strengthened by taking additional features into consideration. Therefore, we also consider the number of times a delay occurred in the last quarter of the time sequence compared to the number of times it could have occurred, i.e. it measures how many times the forecasted interaction appears with the expected delay compared to total number of times the forecasted interactions appears in the last quarter, the function computing this feature is denoted as $LastQ(\delta)$ where δ is a delay. Another feature is the number of times the delay occurs relative to the size of its layer, this feature is denoted as $LayerOcc(\delta)$. Lastly, the probability of the delay $P(\delta)$, is also taken into consideration. Each of these features is assigned an importance coefficient (power), based on which the trust value is calculated as shown by 2.4.

$$\begin{aligned}\Phi(\delta) = {} & LastQ(\delta) \times LastQ_{power} \\ & + P(\delta) \times P_{power} \\ & + LayerOcc(\delta) \times LayerOccpower \end{aligned} \tag{2.4}$$

Generally, the sum of all power coefficients should be 1.0 and these are assigned by specific experiments based on the importance we put on each feature. Certain features, such as *LayerOcc* may never give a value of 1.0 unless there are two interactions occurring at each timestep of the layer. It may therefore be acceptable in such cases that the sum of all power coefficients is allowed (at least theoretically) to be > 1.0 and that features known to be unable to reach high values compared to the others are regarded as a bonus.

We classify these delays into trusted and untrusted using a `delay-ratio` parameter, which is multiplied with the average trust of all delays in order to obtain the minimum threshold $\Phi_{min} = DelayRatio \times AverageTrust$. This allows the trust threshold to remain proportional with the trust values, regardless of the trust definition. A fixed threshold method of delay classification (original implementation) was also kept and can be used to set an minimum in case the

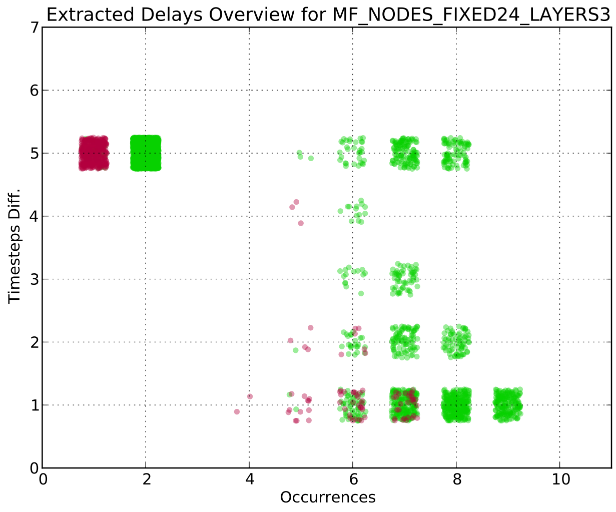


Figure 2.8: Delays from multiple layers relative to their value (time difference) and number of occurrences. In this experiment $LastQ_{power} = P_{power} = 0.5$ and $LayerOcc_{power} = 1.0$. Green dots are trusted delays and red dots are untrusted, they correspond to the closest integer coordinates. Generally, the most trusted delays are those with small time differences and many occurrences. However, a notable exception can be observed in the case of the delays at $(2, 5)$. Despite having few occurrences, delays may have been recovered by an excessively high importance given to $LastQ_{power}$, i.e. if these delays occur 2 out of 2 possible times in the last quarter, 0.5 is added to their trust value.

average trust is too low.

2.5.3 Discussion

Flexibility. As shown in Figure 2.8, there may be no perfect measure for trust but when more auxiliary features are combined and given the right importance, the trust measure is likely to classify primary features (i.e. interactions at unknown timesteps) better than the score given by a single auxiliary feature would. The use of multiple models for training based on distinct categories of features (Ensemble learning) can also bring a better predictive performance and improve the existing approach.

Performance. The implementation of the training component can be very expensive in terms of computational performance; even if interactions are combined into larger components, extracting a large number of auxiliary features can have a significant impact if the algorithms grow with the size of the dataset.

This component plays an important role in solving the prediction problem, and the perspectives from which it can be approached are unlimited. Its implementation is certainly one of the most interesting and relevant aspects of solving the structure prediction problem.

Chapter 3

Prediction Results and Validation

This section describes our method for creating predictions and the methods for validating them and discusses the results. These components are dependent on what is produced by the previous components, and aiming to keep them loose coupled required that their general specification is given with less detail; which is why in the largest part of this chapter, the focus is placed on our own dataset, implementation and results.

3.1 Predictions

A *prediction* model uses the features extracted from the known window with the **purpose** of extending the known window with the predicted snapshots $G_{t_{N+1}}, G_{t_{N+2}}, \ldots$, such a predicted sequence is called a *predicted sample.* Features that are extracted through training can have a different nature and a prediction model should be constructed to handle these features. In our training model, the main predictive features are the delays between interactions and the time sequence features, therefore our prediction model will make use of these.

For validation purposes and correctly computing the prediction performance of the model, features that are below the minimum trust threshold are also considered, and it is verified whether predictions created from these untrusted features are actually incorrect. Using trusted and untrusted features two predicted samples can be created. The one created from trusted features is called a *positive predicted sample* and holds the predictions we classified as positives, the other is the *negative predicted sample* with predictions classified as negative and formed by untrusted features. The features used for classification of predictions are the delays. In order to keep the prediction model simple and have an uniform evaluation of these samples, the features which are extracted from the time sequence and predict the labels and the sizes of unseen timesteps are regarded as a trusted assumption (having the prediction samples with different unseen timestep sizes would over-complicate the prediction model and add bias to how the predictions are evaluated).

3.1.1 General Specification

Given a sequence of features for each known timestep $F_{t_1}, \ldots, F_{t_N}$, this component predicts a set of interactions which occur at unseen timesteps $G_{t_{N+1}}, G_{t_{N+2}}, \ldots$. Each pair formed by an interaction and an unseen timestep $P_{(k,x)} = (\mathcal{I}_x, t_{N+k})$ is a prediction. If $P_{(k,x)}$ is created using a trusted feature than $P_{(k,x)}$ is classified as a positive prediction, otherwise it is negative. Two sets of predictions P^+ and P^- contain the positive and negative predictions. The time sequence $G_{t_{N+1}}, G_{t_{N+2}}, \ldots$ created with the predictions of P^+ and P^- form the positive and negative predicted samples.

Each predicted sample has its size limited by the temporal distance of the extracted features (i.e. some features may predict an interaction to occur k timesteps away from the last known timestep, k varies based on the extracted features). If it is necessary to predict timesteps at longer temporal distances than the capabilities of the extracted features, new features should be found. An alternate choice, yet perhaps a superficial one, would be to extract features formed by the predicted interactions themselves and apply a penalty to the trust value of each second generation prediction. It could only work in highly predictable systems, with efficient models and for features with high trust values.

The resolution of predicted samples should generally be the same resolution that has been used when features where extracted. This becomes more complex when using a dynamic-sized labeled timestep resolution. The labels and sizes of each predicted timestep t_{N+k} will also have to be predicted. Since these predictions will not predict interactions, but instead, the intervals in which they are expected to occur, it is acceptable that both P^+ and P^- are computed for the same intervals, which we consider to be trusted knowledge.

3.1.2 Implementation

The predictive features extracted from each timestep t_i are 3-tuples $(\mathcal{I}_{tr}, \mathcal{I}_{fc}, \Delta)$, where $\mathcal{I}_{tr}$ is an interaction which occurs at t_i and $\mathcal{I}_{fc}$ is an interaction which occurs at $t_{i+\Delta}$, with Δ being the delay between the two interactions.

We split the features in the two sets P^+ and P^- based on the trust value of the delay Δ, and compute the negative and positive predicted samples. It should be noted that Δ has a timestep difference value expressed in labeled timesteps, therefore two delays having the same values of timestep difference can actually refer to different temporal distances if labeled timesteps have dynamic sizes. To find out when do the predicted interactions actually occur, we compute the labels and sizes of unseen timesteps, as far as the timestep t_{N+k} with a maximum k for which an interaction is predicted to occur. Labels of predicted timesteps are only meaningful because they link the timesteps to a layer, and the size of the timestep can be predicted based on the most common size of labeled timesteps in that layer. If labeled timesteps are fixed-size, than the Δ differences always express the same temporal distance and the labels and sizes of unseen timesteps are known.

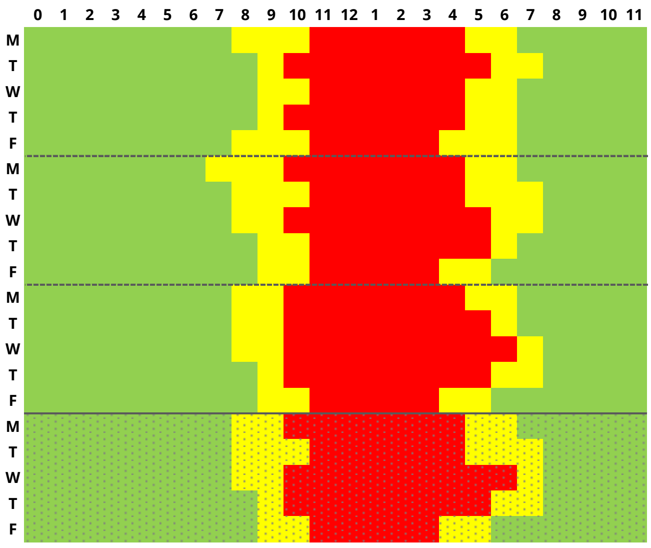


Figure 3.1: A matrix showing labeled timesteps corresponding to three layers, each interval stretches over multiple timesteps classified based on the number of interacting individuals, the fourth week (bottom five rows) is the predicted sequence of labeled timesteps, based on the most regular sizes of timesteps in a layer. For our dataset, since the variation of sizes in these intervals is low, the error caused by predicting sizes is negligible, while the prediction benefits can be significant.

3.1.3 Discussion

When correctly predicted, dynamic-sized timesteps allow better estimations of when an interaction is expected to occur. Correctly delimiting intervals in which interactions are less predictable can still provide quality predictions. For our dataset, the interval in which interactions are most predictable usually has a predictable size ($\sim$7–9 timesteps, study hours), other interactions may regularly occur outside the study hours, but the exact time is harder to estimate (interactions here are not always scheduled). If we have two layers, one for study hours and another for the remaining time, we can precisely predict regular interactions that do not occur within study hours. Although the problem of precise prediction is relaxed, it allows for correctly predicting an interaction which is regular but whose exact time is harder to estimate.

3.2 Validation

The *validation* component has the **purpose** of verifying the classification of computed predictions in P^+ and P^-. This can be trivially performed by testing if the predicted interactions occur in the interval in which they are expected to either occur or not to occur. Beyond this, it is up to each implementation to choose the best tools and algorithms for learning from these results. This section will describe the general validation method and the tools that have been used for learning from our experiments. In Section 3.3 a set of experiments will be evaluated using the described tools.

3.2.1 General Specification

In a single validation run, two sets of computed predictions P^+ and P^- classified as either positive or negative are tested against a validation sample. If an interaction occurs in its predicted interval, the prediction is considered true, otherwise the prediction is false. This binary answer is correlated to the expected result which is given by how the prediction was initially classified. This allows further classification of the validation results into four distinct classes. Multiple validation runs can be performed for different pairs of training and validation sets.

Predictions in the negative sample formed by P^- are expected to be false, and predictions here that are indeed false are *true negatives (TN)*; if they are actually true, these predictions are classified as *false negatives (FN)*. Similarly, for the positive predicted sample, true predictions are *true positives (TP)* and false predictions are *false positives (FP)*. Together these values form a *confusion matrix* as illustrated in Figure 3.2, where the columns stand for a predicted class (for each predicted sample) and rows stand for the actual class. In other words, having all predictions classified on the main diagonal and none on the secondary would mean that the classification was perfect.

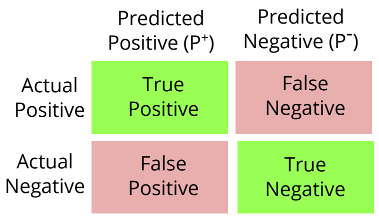


Figure 3.2: A confusion matrix where each class is obtained by testing two sets of predictions against a validation sample, and correlating the actual result with the expected result. In many cases, predictions may not be proportionally distributed in these classes, and/or the number of predictions in a certain class may have a higher importance than those from another.

3.2.1.1 Accuracy and imbalanced datasets

The *predictive accuracy* is computed as the the number of correctly classified predictions (main diagonal) divided by the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.1}$$

However, this may not always be an appropriate measure of prediction performance for *imbalanced datasets*, where the classification categories are not being near evenly represented, or when the cost of different errors significantly varies. For example, consider an experiment which gives 100 predictions of which 90 are classified as TN, the remaining 10 as FN, and no predictions are classified as TP or FP. This would give an accuracy of 90%, but the predictive performance of a model which provides no positive predictions can not be considered good. Furthermore, in many cases the cost of false positives may be significantly higher than that of true negatives. In such cases, accuracy is clearly not an appropriate measure of predictive performance, and more advanced measures for evaluating it should be used, those that are used by our implementation are discussed in Section 3.2.2.

3.2.1.2 Overfitting

Overfitting occurs when a model describes noisy data instead of, or along with generalized data, depending on the models implemented by each component, this may occur when certain thresholds are set too low, when infrequent edges are incorrectly used, or when considering features with low trust value. Although a model may perform decently on a certain training set, if the validation set is changed or when different data is provided, the results will have an unusually high fluctuation. A good method to prevent overfitting is cross-validation, discussed in Section 3.2.2.3.

3.2.2 Validation Methods

3.2.2.1 Receiver Operating Characteristic

A measure useful for data sets where classification categories are not evenly balanced, is given by the *Receiver operating characteristic (ROC)* curve, which summarizes classifier performance over a range of trade-offs between true positive and false positive error rates [Swe88]. This measure is the Area Under the ROC Curve (AUC) and computing it is a standard technique for evaluating the predictive performance of a model. An ROC curve is obtained by plotting the results of multiple tests for different values of the trust threshold used to classify prediction.

ROC curves can be seen as a family of best decision limits for relative costs of TP and FP. On an ROC curve the X-axis represents the percentage of false positives from the total predictions classified as false $FP/(TN + FP)$ (*False Positive Rate (FPR)*) and the Y-axis represents the percentage of true positives from the total predictions classified as true $TP/(TP+FN)$ (*True Positive Rate (TPR)*, also known as *Recall*). The ideal point on the ROC curve is $(0, 1)$ which indicates that all positive predictions have been classified correctly and no negative predictions are misclassified as positive. The line $y = x$ represents the scenario in which the predictions are randomly guessed [Cha05]. Therefore, an adequate level of predictive performance is considered to characterize a model when $AUC > 0.5$. These notions are illustrated by Figure 3.3.

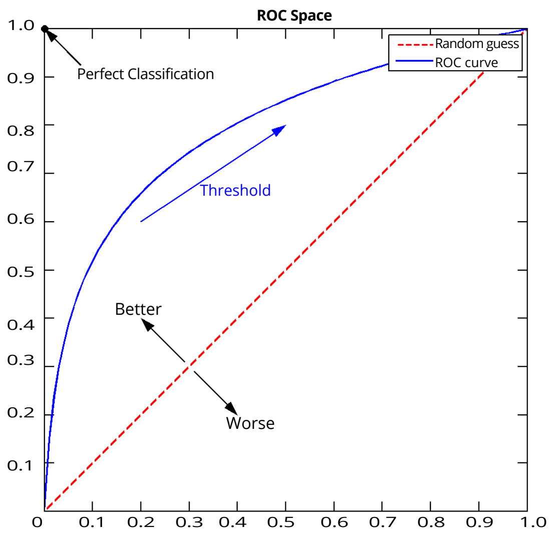


Figure 3.3: General aspects of the Receiver Operating Characteristic (Source: [Wik13])

3.2.2.2 Precision and Recall

A good way to learn if the trust threshold – that classified predictions as negative or positive – was appropriately chosen, is to study whether by lowering this threshold we can obtain a better TPR or *recall value*, without affecting the *precision value*, also known as the Positive Predictive Value (PPV). These measures are defined as follows.

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

Lowering the minimum trust requirement will result in predictions classified as false negatives becoming true positives because these predictions correctly described an occurring interaction, but their trust value was too small. On the other side, lowering this threshold also means moving true negatives into the positive sample, where they become classified as false positives. In other words, the question that is answered by studying TPR in relation with PPV is how many good predictions can we recover from the negative sample without introducing more false positives than the true positives that we recovered. An example from one of our experiments is illustrated in Figure 3.4.

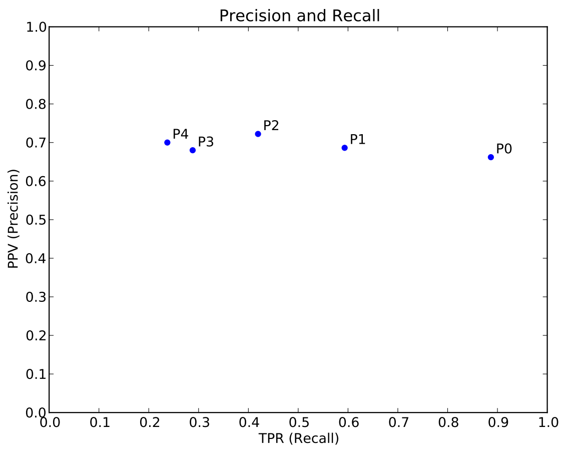


Figure 3.4: PPV relative to TPR. In this set of results the experiments have their thresholds gradually lowered from P4 (1.35× *Average-Trust*) to P0 (0.5× *Average-Trust*). It can be observed that the PPV is not significantly affected by the increase in TPR. This means that the number of true positive (relevant) predictions recovered by lowering the trust threshold is proportional to the number of false positives recovered.

3.2.2.3 Cross-validation

A standard method used to prevent overfitting is *cross-validation*. The method broadly refers to validating the data using different pairs of training and validation samples and averaging the results. If individual results for these pairs are fluctuating, then the model is likely too complex or it may at least indicate problems with the choice of parameters. Using cross validation we can average these fluctuations and get a more accurate picture of the results.

This is done by testing the results from multiple equally sized training samples against a validation sample. This method is known as *k-fold cross-validation*, and k represents the number of equally sized samples that are used, one for validation and the others for testing. This method specifies that, the process of cross-validation should be performed for k times (folds) so that each sample plays the role of validation sample once, and the results are averaged to produce a single estimation. However, in the context of temporal prediction, testing a training sample that occurs after the validation sample might not be acceptable (the future cannot be expected to "predict" the past), which is why our implemented approach only uses the final two samples as validation samples, while the others are training samples.

Once every training sample has been cross-validated with both validation samples, the relevant measures (e.g. TPR, PPV) from each test are averaged towards providing a better estimation of the results. The tested sample sizes for our dataset have so far been of two and three weeks, and as expected, the results were usually better for the larger sample size.

3.2.2.4 Matthews Correlation Coefficient

Matthews correlation coefficient (MCC) is another measure for predictive performance which can be applied to imbalanced datasets. It is an useful alternative to the AUC, for which an exact value cannot be computed as trivially as the MCC. It can also be used to verify the AUC or any other measures of predictive performance. MCC is relatively straightforward compared to AUC, it returns a value between -1 and $+1$, where a coefficient of $+1$ represents a perfect prediction while 0 corresponds to the random guess scenario. The MCC can be calculated directly using the values of the confusion matrix as defined by 3.4.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.4}$$

3.3 Experimental Results

This section discusses the results of a number of relevant experiments that were identified. It starts with a baseline experiment which uses a set of parameters considered to be *defaults*, and explores how altering these parameters can impact the results.

In order to build the ROC curve and compare how precision changes at different levels of recall, each experiment will consist of five scenarios (called *contexts* in the implementation) which only differ in the values computed for the delay trust threshold. We remind that the trust threshold is the main classifier, and is computed as `delay-ratio` × `average-trust`, where `delay-ratio` is a parameter and `average-trust` is the average trust of all delays.

These scenarios are labeled $P4, P3, \ldots, P0$ and $P4$ has the highest `delay-ratio`, of 1.5 which decreases in steps of 0.25 such that $P0$ has a `delay-ratio` of 0.5. An overview of the defaults is given in Table 3.1. The choice for each default value is justified and explained when discussing specific experiments. At this point, let's assume that these defaults provide the best known results. The following sections will test whether modifying them produces the anticipated results and will discuss whether the prediction performance is improved or not.

3.3.1 Default parameters

Parameter	Default	Range
`noise-ratio`	0.4	[0.25, 0.45]
`layers`	3 at (0.3,0.4,0.3)	2–4
Labeled timestep size	24	24, Dynamic, Auto
Layer regularity (Edges)	0.25	[0.10, 0.35]
Layer regularity (Clique)	0.15	[0.5, 0.25]
Aux. feature power: *Prob*	0.5	[0.25, 1.0]
Aux. feature power: *LastQ*	0.5	[0.25, 1.0]
Aux. feature power: *LayerOcc*	1.0	[0.5, 1.5]
`delay-ratio`	0.5–1.5	Fixed
Reduction method	Nodes	Fixed

Table 3.1: **Default parameters.** Each parameter's role was previously discussed with the implementation of its parent component. By experimenting different settings, we learn more about the how each parameter influences the results.

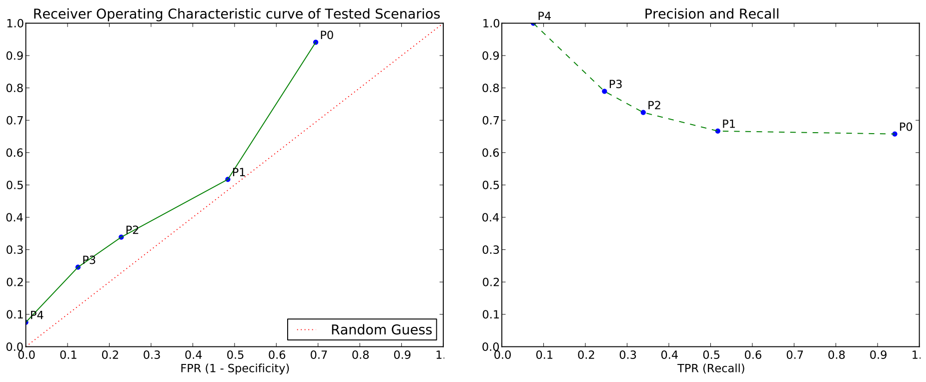
Along with providing the default parameters provided in Table 3.1 it is also important to note that – by default – weekends and holidays are removed from the data set by preprocessing.

Scenario	Delay Ratio	Predictions	PPV	MCC
P0	0.5	87	0.66	0.33
P1	0.75	89	0.67	0.031
P2	1.0	97	0.72	0.11
P3	1.25	93	0.79	0.14
P4	1.5	81	1.0	0.16

Table 3.2: The number of produced predictions, precision (PPV) and Matthews correlation coefficient (MCC) for each scenario using the default values. The average MCC is 0.15.

We also mention that by default, we use the simpler method of validation which splits the dataset in two equal halves of 7 weeks each. An experiment with cross-validated results will however, be provided in Section 3.3.8. This choice had to be made due to large processing time required by applying k-fold validation to our dataset, also – since the method was learned of towards the end of the LPF tool's development – a number of bugs were created which require refactoring. Furthermore, due to dataset size, we can at best use samples of 2–3 weeks, and due to the nature of the data, we cannot validate samples from the future to with samples from past. Simply said, this method requires further work on both how it's implemented in a temporal context, and on the actual implementation in our tool.

The results we obtain by using the default set of parameters are shown in Table 3.2 and illustrated in Figures 3.5 and 3.6.



(a) ROC Space **(b)** PPV and TPR

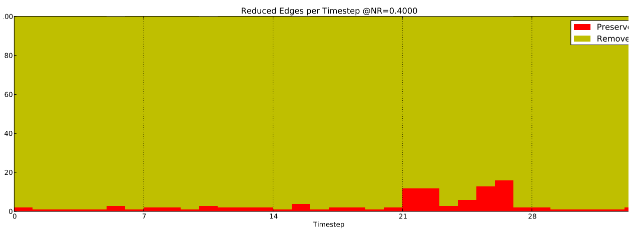
Figure 3.5: **Results for the default parameters.** It can be observed that $P1$ scenario has a relatively higher FPR. This may not seem significant when looking at AUC alone (which is > 0.5), however this deviation has considerable impact on MCC (see Table 3.2). It's also shown in *b)* that PPV is not significantly affected by the increase in TPR, which is the desirable behavior.

3.3.2 Lower noise ratio

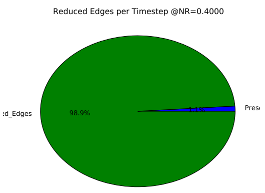
Modified Param.	New Value(s)	Default
`noise-ratio`	**0.25**	0.4

Table 3.3: **Discussed parameters.**

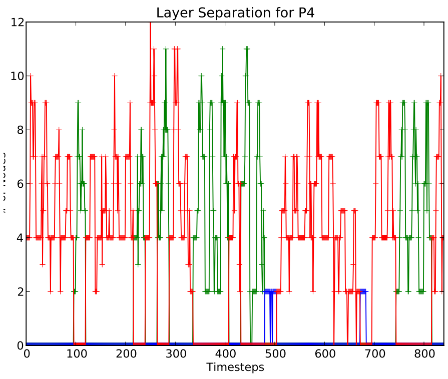
In the results obtained with the default parameters (i.e. Table 3.2), it is shown that with a `noise-ratio` value of 0.4, ∼90 predictions are usually generated and the precision (PPV) is high and it is not substantially affected when the *recall* (TPR) increases. Decreasing the `noise-ratio` to 0.25 can be expected to lower the quality of predictions and negatively impact these results, because noisy and less predictable nodes are preserved.



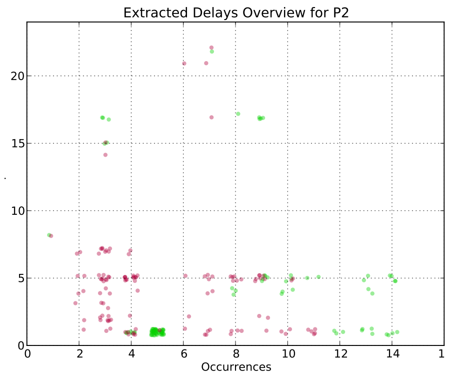
(a)



(b)



(c)



(d)

Figure 3.6: Processing statistics for the default parameters. *a)* and *b)* show the ratio of preserved (red or blue) and removed (yellow or green) edges for each timestep and the overall percentage, which is of 1% preserved edges. *c)* shows the measured signal and how timesteps are classified in layers. *d)* shows the delays, as they are colored in $P2$ at a ratio of 1.0 (where the trust threshold is the average delay trust).

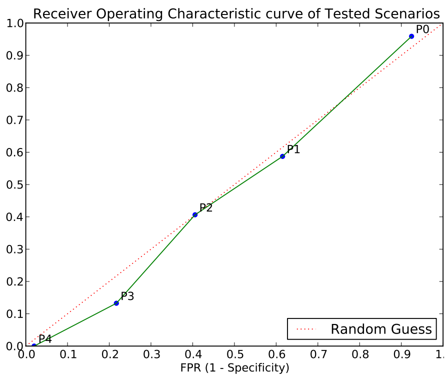
Scenario	Delay Ratio	Predictions	PPV	MCC
P0	0.5	402	0.61	0.073
P1	0.75	535	0.6	-0.02
P2	1.0	554	0.62	0.0007
P3	1.25	439	0.48	-0.111
P4	1.5	383	0.0	-0.110

Table 3.4: The number of produced predictions, PPV and MCC at lowered `noise-ratio`. Average MCC is -0.03. Not good!

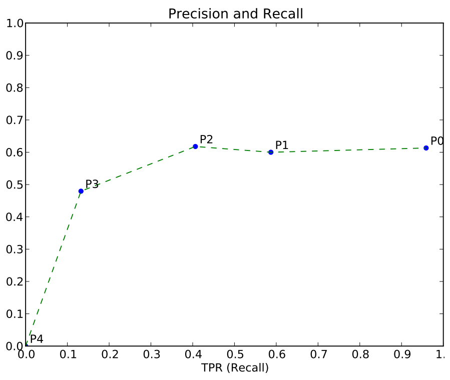
As expected, Table 3.4 shows that at a `noise-ratio` of 0.25, a significantly larger set of predictions is produced, but these predictions have a clearly lower quality. Since some of the tested scenarios yield negative values for MCC coefficients, which is also in agreement with the observation that AUC is below 0.5, we stop at 0.25 as the lower limit value for this parameter.

The decrease in quality of results is explainable because at 0.25 `noise-ratio` 24% of the edges are preserved (see Figure 3.8), which is huge compared to the default parameter value of 0.4 which only preserves 1%. As a result, previously fragmented layers become dense and interactions also increase in size, from dyads to cliques, or in the number of nodes in a clique. Since the noise reduction method affects nodes, we can describe this for our dataset as having untrusted (i.e. irregular) individuals being introduced into previously trusted cliques. It is usually the backbone of a group that is the most predictable, when irregular nodes are introduced, the group becomes only as predictable as its least predictable node.

If we experiment in the opposite direction and increase `noise-ratio` to 0.5, predictions are no longer generated by all the tested scenarios. Therefore we set that 0.4 is the maximum acceptable `noise-ratio` value when using the node frequency method. Clearly, the direction for improvement is towards obtaining a better prediction performance with a lower `noise-ratio`. Therefore, in the next experiment in 3.3.3 we keep `noise-ratio` at 0.25 and aim to improve the prediction model using other parameters.



(a) ROC Space

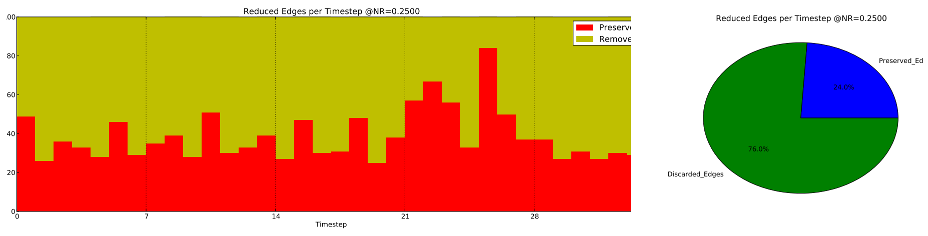


(b) PPV and TPR

Figure 3.7: Results at lower `noise-ratio`.

3.3.3 The power of layers

The default parameter for the size of a labeled timestep is fixed at 24 hrs, therefore layers are not yet made full use of. The default was chosen because timesteps



(a) (b)

Figure 3.8: Noise reduction statistics for `noise-ratio` of 0.25. The percentage of preserved edges is clearly better and more meaningful than for the default `noise-ratio`. But, can the predictions also be more meaningful?

Modified Param.	New Value(s)	Default
`layers`	**4 at (0.2,0.3,0.2,0.3)**	3
Labeled timestep size	**Dynamic**	24
`noise-ratio`	0.25	0.4

Table 3.5: Discussed parameters.

of 24 hrs seem to make sense given our dataset, furthermore, it was important that we started with a default model closer to the traditional approach. In this experiment we stay at the previously discussed 0.25 `noise-ratio`, but we classify timesteps into four distinct layers (an additional one is added) and we use dynamically sized timesteps.

Timesteps are classified into layers based on the number of nodes (activity) measured at each timestep, and the layers are delimited as follows: *1)* the *lower* layer holding timesteps with activity lower than 0.2 of the maximum registered activity, above it *2)* the *mid-lower* layer with 0.3 ratio, *3)* the *mid-high* layer with 0.2 ratio and, at the top is *4)* the *high* activity layer with a ratio of 0.2. Given the decreased level of `noise-ratio` the majority of the edges is grouped in the upper layer as shown in Table 3.6.

Layer	Std. Timesteps	Labeled Ts.	Edges
`upper`	278	31	348
`midup`	241	69	8
`midlow`	0	0	0
`low`	321	43	3
irregular	-	-	629

Table 3.6: Edge classification statistics. The regularity threshold by which edges are classified may act as an additional noise reduction measure, removing interactions which are not regular from prediction. When having a dynamic size, timesteps generally have a smaller size and edges are classified more precisely.

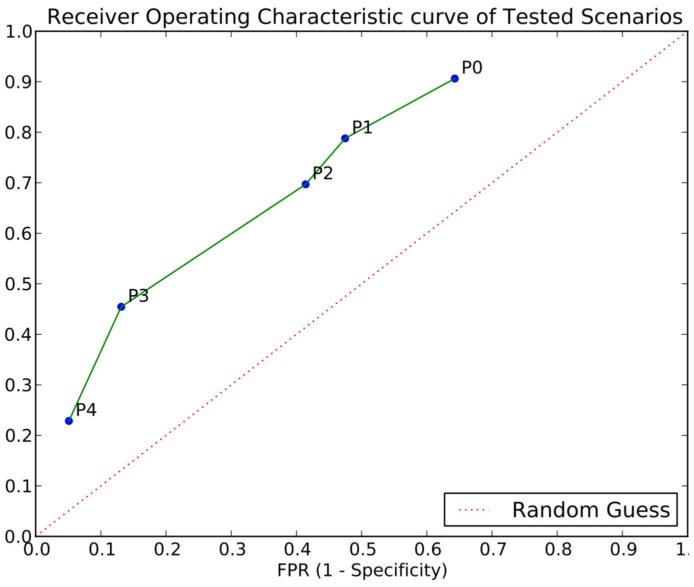
Because dynamic size labeled timesteps are usually smaller than 24 hrs (in our dataset), edges are more accurately classified. Layers now have different sizes and present a more fluctuating pattern than it previously did. The classification of edges is less balanced (at least given our chosen measure and dataset), but what is classified in a layer with dynamic sized timesteps is usually more reliable, because timesteps are labeled in the order in which they naturally occur. On the other side, a large number of interactions are irregular. In this manner, edge classification acts as an additional level of noise reduction. Because of that we have fewer predictions than in the previous experiment but more than we would have at 0.4 `noise-ratio`.

It can also be observed from Table 3.6 and in Figure 3.9b, that adding an extra layer and splitting the middle layer in two layers did not have any effect, there are no timesteps that are labeled as `midlow`. `midup` also has a low average size of $69/241 =\sim 3$. This means that the activity (number of nodes) in our dataset drops very fast from high to low, as it is expected when study hours are over.

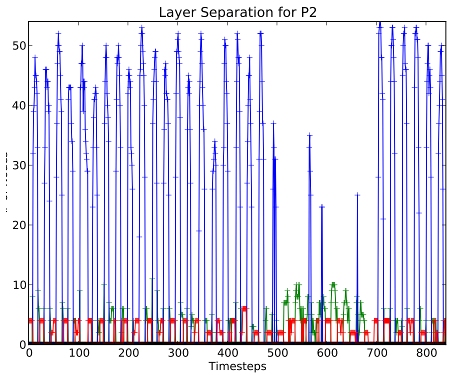
Scenario	Delay Ratio	Predictions	PPV	MCC
P0	0.5	130	0.32	0.24
P1	0.75	132	0.36	0.27
P2	1.0	132	0.36	0.24
P3	1.25	132	0.54	0.34
P4	1.5	133	0.62	0.26

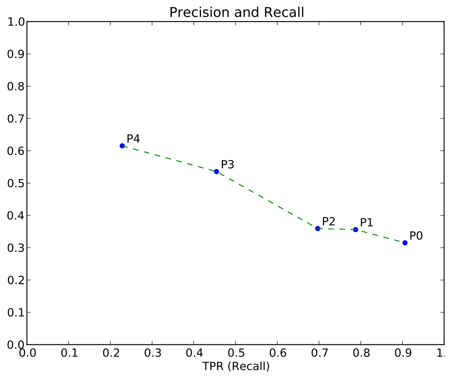
Table 3.7: Produced predictions, PPV and MCC for dynamic-sized timesteps. Average MCC is of 0.27.

What gives the clear advantage of making full use of layers is that instead of 1% of the edges, we can *consider* 24%, giving edges that are not in the top 1% "a chance" to be regular within a certain layer (i.e. time interval). It is also likely that the results are boosted by the fact that timestep size is now lower, and predictions are no longer made that far into the "future". This will be verified in Section 3.3.6, where the experiment will involve smaller fixed-size timesteps. As the results in Table 3.7 and Figure 3.9 show, the model for `noise-ratio` of 0.25 has been improved to an average MCC of 0.27 and an $AUC > 0.5$. Therefore, we keep these parameters for the next experiment, in which we look at what happens if we decrease the number of irregular edges.



(a) ROC Space





(b) Layers

(c) PPV and TPR

Figure 3.9: Results and timestep classification for dynamic-size timesteps.

3.3.4 Lower regularity requirements

Modified Param.	New Value(s)	Default
Layer regularity (Edges)	**0.15**	0.25
Layer regularity (Clique)	**0.1**	0.15
`layers`	4 at (0.2,0.3,0.2,0.3)	3
Labeled timestep size	Dynamic	24
`noise-ratio`	0.25	0.4

Table 3.8: **Discussed parameters.**

This experiment's goal is to study the impact of lowering the regularity requirements for interactions, this allows for previously irregular interactions to be considered regular in a layer. The requirement is lowered by decreasing the regularity thresholds which are applied to dyadic and to aggregated interactions. We lower the two thresholds as shown in Table 3.8 and we obtain the classification shown in Table 3.9.

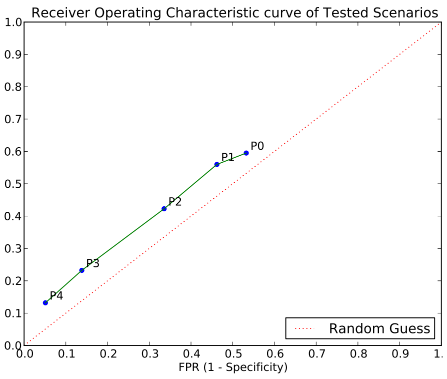
Layer	Std. Timesteps	Labeled Ts.	Edges
`upper`	278	31	421
`midup`	241	69	26
`midlow`	0	0	0
`low`	321	43	3
irregular	-	-	556

Table 3.9: **Edge classification statistics for lowered regularity thresholds.**

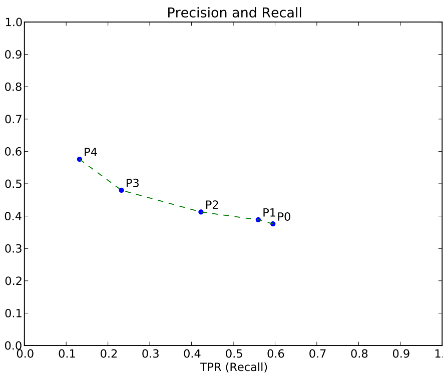
By lowering the thresholds we recover 73 edges, which are added as regular in the `upper` and `midup` layers. As expected, the quality of predictions is negatively affected. While the number of recovered edges is not significant, the average MCC is lowered from 0.27 to 0.09. On the positive side, we can see in Table 3.10 that this produced almost as many predictions as the first experiment from Section 3.3.2, but with relatively better predictive performance, which means that using layer regularity thresholds as an extra filtering method for edges is a good choice, which allows for decent results even when starting with a low `noise-ratio`.

Scenario	Delay Ratio	Predictions	PPV	MCC
P0	0.5	451	0.38	0.06
P1	0.75	462	0.39	0.09
P2	1.0	469	0.41	0.08
P3	1.25	437	0.48	0.11
P4	1.5	418	0.58	0.14

Table 3.10: Produced predictions, PPV and MCC for lowered regularity thresholds. Average MCC is of 0.09.



(a) ROC Space



(b) PPV and TPR

Figure 3.10: Results obtained with lowered regularity thresholds.

3.3.5 Measures for trust

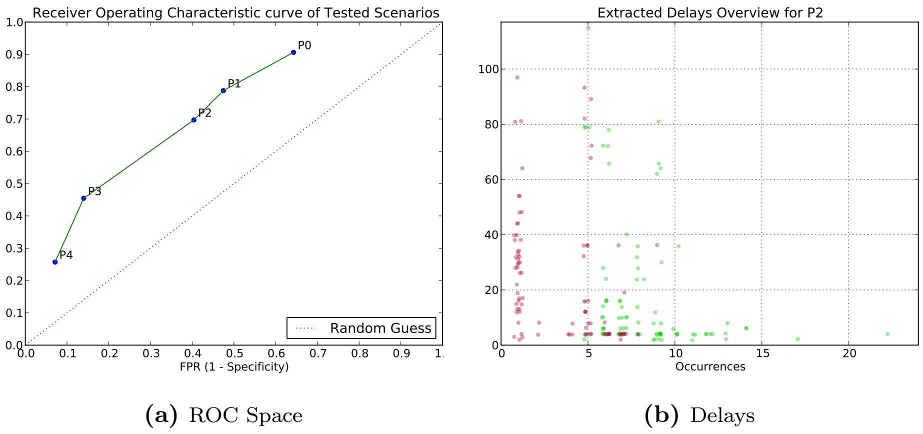
Modified Param.	New Value(s)	Default
Aux. feature power: *Prob*	**0.3**	0.5
Aux. feature power: *LastQ*	**0.7**	0.5
Aux. feature power: *LayerOcc*	**1.5**	1.0
`layers`	4 at (0.2,0.3,0.2,0.3)	3
Labeled timestep size	Dynamic	24
`noise-ratio`	0.25	0.4

Table 3.11: Discussed parameters.

Trust is calculated using the probability of a delay and two auxiliary features, as discussed in Section 2.5.2. By default, we start with a balanced set of importance ratios. In this experiment we lower the importance of a delay's raw probability and give more importance to the auxiliary features which measure the number of times a delay occurs relative to the size of its parent layer and to the size of the last quarter of the time sequence. It is expected that the quality of predictions will be further improved.

Scenario	Delay Ratio	Predictions	PPV	MCC
P0	0.5	130	0.32	0.25
P1	0.75	132	0.36	0.27
P2	1.0	132	0.37	0.25
P3	1.25	133	0.52	0.33
P4	1.5	133	0.56	0.25

Table 3.12: Produced predictions, PPV and MCC with the new trust definition. Average MCC is of 0.27.



(a) ROC Space **(b)** Delays

Figure 3.11: Results and delays map obtained with the new trust definition. The delay map shows no significant changes from that produced by previous experiments.

Results in Table 3.12 are almost the same as those from Section 3.3.3. The expected improvement in prediction quality is hardly observable, and the average MCC is the same. The map of delays in Figure 3.11b is also close to that from the previous experiment. Although the expectations were higher, this can perhaps be explained. By default, the primary features (delays) only have predictive ability if they occur close to the end of the time sequence, that is generally the last quarter (or other subdivision). In the case of delays *LastQ* must inherently be high in order to generate a prediction, thus it is a neutral statistic. Perhaps the only minor fluctuation is given by *LayerOcc*, but since layers also have different sizes and since the value is generally smaller than 0.1 for most delays, this feature does not bring very much impact either. The conclusion is that, while using auxiliary features may bring substantial quality to the trust measure, the auxiliary features chosen by our implementation do not provide any substantial impact because the primary feature indirectly takes them into account. Further work should take this into account and identify relevant auxiliary features which are not directly connected with the primary feature.

3.3.6 Optimal timestep size

Modified Param.	New Value(s)	Default
Labeled timestep size	**Auto**	24
`layers`	4 at (0.2,0.3,0.2,0.3)	3
`noise-ratio`	0.25	0.4

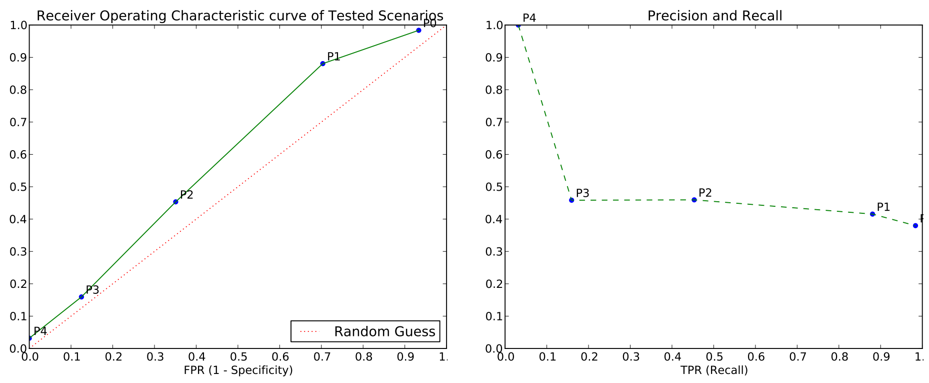
Table 3.13: Discussed parameters.

For this experiment we set the size of a labeled timestep to *auto*, which as discussed in Section 2.3.2 is computed by averaging the size of dynamic sized timesteps. The purpose is to find out whether having timesteps at the fixed-size value which is considered optimal by our method will give any better predictive performance than the previous fixed-size experiment when we lowered the `noise-ratio` to 0.25, with a timestep size of 24.

Scenario	Delay Ratio	Predictions	PPV	MCC
P0	0.5	166	0.38	0.11
P1	0.75	185	0.42	0.20
P2	1.0	189	0.46	0.10
P3	1.25	173	0.46	0.04
P4	1.5	166	1.0	0.14

Table 3.14: Produced predictions, PPV and MCC for optimal fixed-size timesteps (of 6 hrs). Average MCC is of 0.11.

The *optimal timestep size* found by our method is 6. We can observe that the results are not as bad as those obtained for Section 3.3.2 but not as good as those obtained in Section 3.3.3. The fact that the average size of dynamic timesteps is much smaller than 24 may suggest that, in part, the results obtained with dynamic size timesteps are positively influenced by smaller timestep sizes. This is likely, because delays describing short term differences (less than 24 hrs) were previously removed when assimilated by the 24 hrs large timesteps.



(a) ROC Space **(b)** PPV and TPR

Figure 3.12: Results for optimal fixed-size timesteps (of 6 hrs).

3.3.7 Predicting weekends

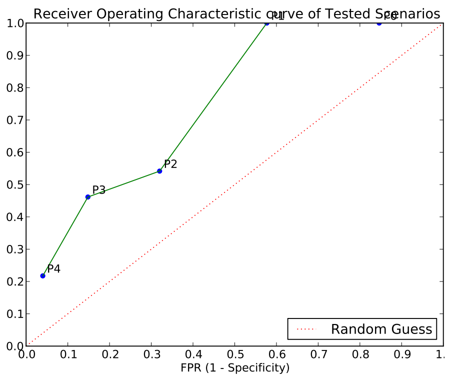
Modified Param.	New Value(s)	Default
Included Interval	**Sem. Break**	-
Included Interval	**Weekends**	-
`layers`	4 at (0.2,0.3,0.2,0.3)	3
Labeled timestep size	Dynamic	24
`noise-ratio`	0.25	0.4

Table 3.15: Discussed parameters.

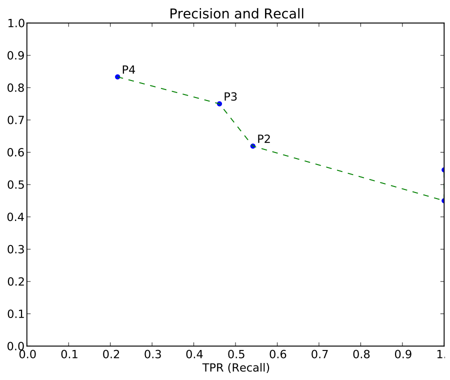
In this experiment, we test the predictive performance when keeping two sets of timesteps which by default were removed in preprocessing (see Figure 2.3a). The two sets are those corresponding to weekends and the semester break, as illustrated in Figure 3.13c both have timesteps with about average levels of activity (much better than the winter holiday, which is discarded). With this, we are looking to determine whether the defined layers model can handle the new timesteps, and whether or not, the performance is negatively affected. Fewer predictions are likely to be produced since the size of the training sample is increased, but the noise threshold defined with the same `noise-ratio` also increases.

Scenario	Delay Ratio	Predictions	PPV	MCC
P0	0.5	44	0.45	0.26
P1	0.75	44	0.55	0.48
P2	1.0	49	0.62	0.22
P3	1.25	53	0.75	0.34
P4	1.5	48	0.83	0.26

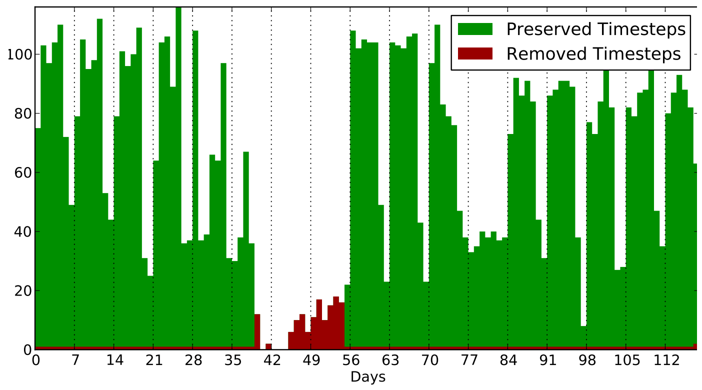
Table 3.16: Produced predictions, PPV and MCC for the extended dataset. Average MCC is of 0.31.



(a) ROC Space



(b) PPV and TPR



(c) Excluded Intervals

Figure 3.13: Results for the extended dataset. For comparison with *c)*, the timesteps that were previously removed by default are illustrated in Figure 2.3a

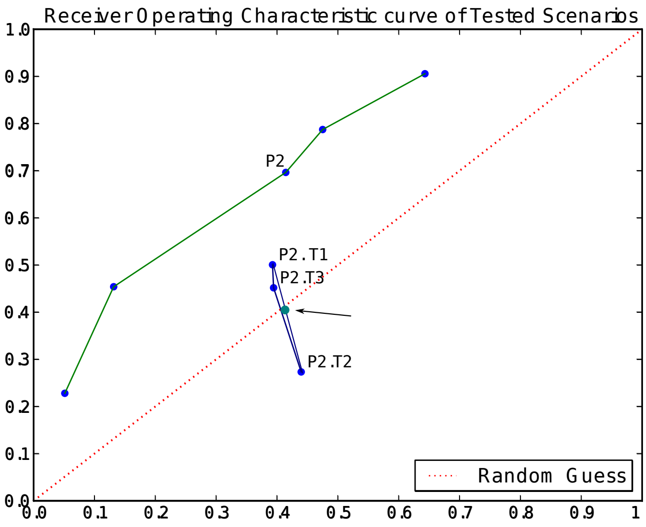
The obtained results are most likely caused by the increase in the noise threshold, even so – the fact that the results describe an even better performance than previously suggests that the use of dynamic sized timesteps is appropriate and the new timesteps are correctly treated. Furthermore we can learn that the patterns of interaction in weekends and in the semester break remain close to those from regular week days, and that the measure of number of interacting nodes, is still a valid measure for the introduced timesteps.

3.3.8 Cross validation

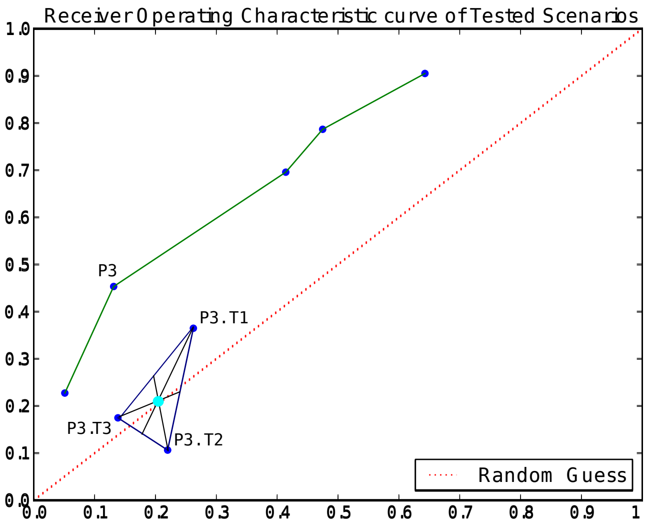
To increase the confidence in our results we perform cross-validation by splitting the temporal window extended by the previous experiment in 4 samples of equal sizes, of which the first 3 are used as training sets and the last one is the validation sample. Although cross-validation was not fully implemented in our tool, we still can test whether for a single scenario the three training sets give the same results on the validation sample. Two scenarios are tested: P2 and P3, with the parameters from the previous experiment, shown in Table 3.3.7. We expect that, since the samples fluctuate the results for each sample should have a slight fluctuation, and that since we use the extended dataset, the results may not be as good as those obtained from Section 3.3.3.

Scenario	Delay Ratio	Predictions	PPV	MCC
P2-T1	1.0	66	0.19	0.08
P2-T2	1.0	122	0.12	-0.13
P2-T3	1.0	1423	0.29	0.05
P3-T1	1.25	72	0.2	0.08
P3-T2	1.25	119	0.08	-0.10
P3-T3	1.25	1356	0.33	0.04

Table 3.17: Produced predictions, PPV and MCC for samples of P2 and P3. Average MCC for P2 is 0.27, and for P3 is 0.0.



(a) P2



(b) P3

Figure 3.14: Cross-validation results for P2 and P3

In terms of MCC the results certainly do not describe the same predictive performance shown by previous experiments. This can be justified by the reduced size of samples and the extended timesteps. If we would eliminate the consideration of the reduced size of samples and its impact, we could say that the improved results obtained in the previous experiment (Section 3.3.7) were a sign of overfitting. However, the results for the tested samples do not show more than the expected fluctuation, the samples T1 and T2 generate significantly fewer predictions than T3, and this could also be justified by the temporal distance between these samples and the validation sample as well as their variation. A lower level of activity (a week is cut in half by the winter holiday) in the timesteps of T2 (which corresponds to the intervals 28–42, 56–70 in Figure 3.13c) makes the sample have a lower predictive performance compared to T1 and T3.

As shown in Figure 3.14, the results of tested samples form a triangle which allows us to extract results using simple geometrical features, the averaged result of the trained samples would be the geometric center of the triangle, and the error would be the radius of the circle which borders the triangle. Overall, the results are both near the random guess line but would have been higher had it not been for T2. Since, with the exception of this sample there are no major fluctuations, we have no evidence to suspect overfitting. It is important to remark that a model for cross-validation in the temporal context needs to be further investigated and developed, since the temporal distance between samples has impact on the results, furthermore our results come from only 3 training sets whereas cross-validation is usually performed with 10.

3.4 Supporting tool

The discussed results were obtained with the supporting LPF tool developed for this thesis, and included with it. The tool is implemented in Python 2.7 (32-bit) and allows testing a data set provided as input in Cytoscape[1] DynNetwork[2] format (XGMML). Our dataset is included with the thesis in the `datasets` directory and can be visualized using the mentioned plugin for Cytoscape 3.0.

The logs and generated visualizations from the previously discussed experiments are included in the `thesis-results` folder. Additional experiments can be further configured using the base context configuration file from the `tests` folder.

The tool has only been tested on a 32-bit Python 2.7 environment running on Windows 7. A number of packages used which the tool depends on are enumerated in the tool's readme file. Given that these packages are installed

[1] `http://www.cytoscape.org/`
[2] `https://code.google.com/p/dynnetwork/`

the tool can be run from command line with the following command in the tool's root directory:

```
python main.py [-c <a_context_file>] [-v <int>]
```

`-v` is the level of verbosity on the console, where 1 is the least detailed and 5 is the most detailed, the default is 3. `-c` is the name of a context file (test scenario) specified in the `tests` folder. The context files are specified in YAML and, as mentioned, the `_base.yml` file is a configuration file with a default set of parameters. Not specifying a context file leads to all contexts defined in the `tests` folder being processed and comparative statistics and plots are created along with individual log files, this way the results of multiple contexts can be compared and plotted together.

The `core` package contains the main implementation of the tool, with components specified in the `core.comps` package, the `ext` folder contains functions for producing plots and statistics and extensions of such nature are fairly simple to create.

CHAPTER 4

Conclusions and Further Work

This section wraps the work done for this thesis, describes the remaining challenges and the potential of future work.

4.1 Conclusion

In this thesis we explored the domain of the structure prediction problem in temporal networks, identified primary tasks involved, and developed a set of loose-coupled components as an experimental framework for exploring the solution space, we provided methods for validating it and inherently for studying the predictability of a given temporal networks representation.

Furthermore, the framework was tested for a temporal network of face-to-face interactions, and implemented methods from existing literature which proved to give good results within the proposed framework. Through the conducted experiments we have explored the impact of different parameters on the results, using validation methods such as the Receiver Operating Characteristic, Matthews Correlation Coefficient, Precision vs Recall. We have identified that similarly to how certain static network measures do not apply to temporal networks, neither are validation methods such as cross-validation fully compatible to a temporal context, yet problems such as overfitting remain considerably relevant.

Even though further research is still required, looking at a temporal network as a set of different temporal layers allows for a strong binding between an actual interaction and the context when it occurs, layers could be seen as a special set of features which we can premise that are likely to be consistent with all domains in which a system can benefit from a temporal networks representation. Layers can work together with applied domain knowledge which can remove the observable variations and let layers focus on those that are microscopic, narrowing the range and focus of predictions to very specific intervals. But also, layers can automatically adapt to unspecified observable variations, being able to treat each of them individually, at the expense of a lower focus.

We have observed that, particularly in social temporal networks, the behavior of individuals is more dependent on time than on relations between entities, but relations also can be better understood from the temporal context in which they occur, e.g. an interaction occurring during evening hours may suggest friendship better than interactions during working hours.

Temporal networks prediction can be applied to a wide range of real problems, e.g. predicting information spread in social networks, disease spreading, optimizing transportation infrastructure or large enterprise server networks; these are all problems where a temporal networks representation does much of the work by itself.

An important future application could be looking at the predictions as a source of artificial intelligence. Many systems provide behavior patterns which show some intelligence, but clearly the ultimate dataset will consist of brain activity patterns[1]. This has been an important source of inner motivation in writing this thesis. Even with our dataset, an intelligent persona could be built given the predictions and a single additional dimension, e.g. a personality trait or course that each student liked the most. The possibilities of combining models created by temporal networks with artificial intelligence are indeed very large, and their exploration is only just beginning.

4.2 Future Work

There may be no general implementation for a solution to structure prediction in temporal networks, further work should expand the components identified by the

[1]See, for example `http://milab.imm.dtu.dk/eeg`

proposed framework and most importantly, test their boundaries of generality, which can be done, for example, using synthetic temporal networks generated with random patterns; or with any other kinds of big data. Once this has been done, the large components may be further divided into smaller and more specialized components which should remain loose-coupled.

Specifically, the training component could be divided in smaller individual components that split the problems of extracting features, training them on known data and computing their trust value. The component could be further improved with a more general way of computing the primary feature and leaving specific auxiliary features and training models that compute them to have a "vote" on whether the general primary features can be trusted or not.

Overfitting is also an important issue that needs to be investigated further using methods more appropriate than cross-validation, which tends to require large datasets and which – although, due to time constraints has not been fully implemented and tested – we premise that it might not be completely appropriate for temporal networks. Further work is certainly required in devising a cross-validation model that remains impartial in a temporal context.

The framework uses a configuration over convention approach, which allowed for investigating the impact of each parameter on predictive performance, the developed tool allows for a simple way of learning the impact of parameter changes through the logs, comparison tables and visualizations produced for individual experiments. Since validation methods are likely to be general to a wide range of results, a better structured and automated tool for learning from these results could be developed, along with a better structured supporting library for implementing the proposed framework. An important feature for such a future library would be to allow for caching of results, such that different scenarios can be more efficiently tested without running each component every time. This was half-implemented by the developed tool, but did not sufficiently scale as the set of parameters and the level of data produced grew.

When the framework reaches a more mature state, the focus can than be moved towards creating optimized instances of the framework for specific domains and sets of problems. One such problem, and perhaps the most interesting: that of using the predictable patterns in an AI context.

Bibliography

[Bar11] Marc Barthélemy. Spatial networks. *Physics Reports*, 499(1-3):1–101, 2011.

[BBPSV04] A. Barrat, M. Barthelemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *PNAS*, 101(11):3747–3752, March 2004.

[BC13] A. Barrat and C. Cattuto. Temporal networks of face-to-face human interactions. *ArXiv e-prints*, May 2013.

[BCG01] Doug Burdick, Manuel Calimlim, and Johannes Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *In ICDE*, pages 443–452, 2001.

[BCG$^+$11] Michele Berlingerio, Michele Coscia, Fosca Giannotti, Anna Monreale, and Dino Pedreschi. Foundations of multidimensional network analysis. In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*, pages 485–489. IEEE, 2011.

[BK73] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, September 1973.

[CE12] Aaron Clauset and Nathan Eagle. Persistence and periodicity in a dynamic proximity network. *eprint arXiv:1211.7343*, 2012.

[Cha05] Nitesh V Chawla. Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook*, pages 853–867. Springer, 2005.

[DMP91] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial intelligence*, 49(1):61–95, 1991.

[GK09] Eric Gilbert and Karrie Karahalios. Predicting tie strength with social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 211–220. ACM, 2009.

[HPY00] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, May 2000.

[HS12] Petter Holme and Jari Saramäki. Temporal networks. *Physics Reports*, 519(3), 2012.

[HT05] S. Hirano and S. Tsumoto. Grouping of soccer game records by multiscale comparison technique and rough clustering. In *Hybrid Intelligent Systems, 2005. HIS '05. Fifth International Conference on*, pages 6 pp.–, 2005.

[KY04] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *Machine learning: ECML 2004*, pages 217–226. Springer, 2004.

[LBW07] Mayank Lahiri and Tanya Y. Berger-Wolf. Structure prediction in temporal networks using frequent subgraphs. *Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007)*, 2007.

[LNK07] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[LZ11] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

[MRM$^+$10] Peter J Mucha, Thomas Richardson, Kevin Macon, Mason A Porter, and Jukka-Pekka Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878, 2010.

[NTM$^+$13] Vincenzo Nicosia, John Tang, Cecilia Mascolo, Mirco Musolesi, Giovanni Russo, and Vito Latora. Graph metrics for temporal networks. In *Temporal Networks*, pages 15–40. Springer, 2013.

[OGO11] Vladimir Ouzienko, Yuhong Guo, and Zoran Obradovic. A decoupled exponential random graph model for prediction of structure and attributes in temporal social networks. *Statistical Analysis and Data Mining*, 4(5):470–486, 2011.

[Pal06] BO Palsson. Properties of reconstructed networks. *Cambridge: Systems Biology*, 2006.

[PBV07] Gergely Palla, Albert-László Barabási, and Tamás Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, 2007.

[PD05] Mercedes Pascual and Jennifer A Dunne. *Ecological networks: linking structure to dynamics in food webs.* Oxford University Press, 2005.

[PDFV05] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[PU03] Alexandrin Popescul and Lyle H Ungar. Statistical relational learning for link prediction. In *IJCAI workshop on learning statistical models from relational data*, volume 2003. Citeseer, 2003.

[RPB12] Bruno Ribeiro, Nicola Perra, and Andrea Baronchelli. Quantifying the effect of temporal resolution in time-varying network. *arXiv preprint arXiv:1211.7052*, 2012.

[SBWG10] Rajmonda Sulo, Tanya Berger-Wolf, and Robert Grossman. Meaningful selection of temporal resolution for dynamic networks. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, MLG '10, pages 127–136, New York, NY, USA, 2010. ACM.

[Swe88] John A Swets. Measuring the accuracy of diagnostic systems. *Science*, 240(4857):1285–1293, 1988.

[TLS$^+$13] John Tang, Ilias Leontiadis, Salvatore Scellato, Vincenzo Nicosia, Cecilia Mascolo, Mirco Musolesi, and Vito Latora. Applications of temporal graph metrics to real-world networks. In *Temporal Networks*, pages 135–159. Springer, 2013.

[TSM$^+$10] John Tang, Salvatore Scellato, Mirco Musolesi, Cecilia Mascolo, and Vito Latora. Small-world behavior in time-varying graphs. *Physical Review E*, 81(5):055101, 2010.

[Wik13] Wikipedia. Receiver operating characteristic — wikipedia, the free encyclopedia, 2013. [Online; accessed 25-July-2013].

[WSM$^+$13] T. Wallach, K. Schellenberg, B. Maier, R. Kalathur, and P. Porras. Dynamic circadian protein-protein interaction networks predict temporal organization of cellular functions. *PLoS Genet*, 2013.

[Yan04] Guizhen Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 344–353, New York, NY, USA, 2004. ACM.