Graduate School Form 30
Updated 1/15/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Ryan Rossi

Entitled
Improving Relational Machine Learning by Modeling Temporal Dependencies

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Sunil Prabhakar Chair	
Sonia Fahmy	
Luo Si	
Elena Grigorescu	

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Sunil Prabhakar

Approved by: Sunil Prabhakar / William Gorman 06/30/2015
Head of the Departmental Graduate Program Date

IMPROVING RELATIONAL MACHINE LEARNING BY MODELING TEMPORAL DEPENDENCIES

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Ryan A. Rossi

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2015

Purdue University

West Lafayette, Indiana

Dedicated to my family.

ACKNOWLEDGMENTS

I am extremely thankful and honored to be surrounded by a long list of truly exceptional and inspirational individuals. Although, words are not sufficient for expressing my gratitude towards these exceptional individuals, I will do my best.

First, I am extremely fortunate to have my PhD advisor Sunil Prabhakar, for his support, encouragement, and invaluable advice over the years. I am also grateful to my committee members, Sonia Fahmy, Luo Si, and Elena Grigorescu, for their invaluable feedback that improved my dissertation. I would also like to thank Assefaw Gebremedhin, David Gleich, and Jennifer Neville for their help, advice, and numerous discussions. I am also indebted to Nesreen Ahmed for her continuous support, encouragement, and collaboration throughout my time at Purdue.

I feel very grateful to have had the opportunity to work closely with top researchers in a variety of government, industrial, and academic research labs. I am particularly indebted and honored to have the opportunity to work with Rong Zhou at Palo Alto Research Center (PARC, formerly Xerox PARC). He has been a great mentor to me and I am especially thankful for his often selfless support, encouragement, and advice over the years. I would also like to thank Dan Davies, Surendra Reddy, and the rest of the High Performance Analytics (HPA) team at PARC, for their continuous support, discussions, and encouragement. In addition, I am especially fortunate to my mentors at the Naval Research Laboratory (NRL), David Aha and Luke McDowell, for their invaluable support, guidance, and advice while at NRL and also throughout the years. I am especially thankful for their help in writing and encouragement that ultimately helped me become a better researcher. Both have been a source of motivation and inspiration in the completion of this dissertation. During my time at NASA Jet Propulsion Laboratory (JPL), I am extremely fortunate and grateful to have had the opportunity to be mentored by Mark Powell. His kind nature, advice, and encouragement gave me the strength to overcome many challenging problems that I encountered at NASA JPL and beyond. I also thank Khawaja Shams and the rest of the Operations Planning Software group at NASA JPL for their help and support. In addition, I feel grateful to David Jensen and Brian Taylor at UMass Amherst KDL, Celeste M. Matarazzo and Brian Gallagher at Lawrence Livermore National Laboratory (LLNL), and Srinivas Mukkamala at New Mexico Institute of Technology for the amazing opportunity to work with them and experience these institutions.

I feel eternally indebted to my early research advisor, Jean-Louis Lassez, who not only introduced me to research, but has significantly changed my life in the process. It was Jean-Louis Lassez who inspired me to pursue a Ph.D. and has been a great source of inspiration and role model as a researcher, mentor, and friend. Throughout the years, he always challenged and helped me improve my writing, research, teaching, and communication skills. I thank him for his relentless advice to tackle difficult problems, and his unique character and perspective that has rubbed off on me over the years. His charisma and caring nature always gave me the strength to move forward. I also thank Stephen Sheel for his help and support over the years.

I thank my friends, collaborators, and colleagues for their advice, friendship, and help with research: Shujaat Ahmed, Balamurugan Anandan, Dieudonne Baributsa, Hoda Eldardiry, Tayfun Karadeniz, Anyin Li, Victoria Livinski, Varun Mithal, Pedro Pastrana-Camacho, Anant Pradhan, Nilothpal Talukder, and the many others who I have unfortunately omitted.

Most of all, I owe my deepest gratitude to my family for their endless love and encouragement. Everything was possible thanks to their strong support. Especially, my mother Ellen Ross; who has been so selfless in supporting me throughout my life and career, my grandfather John Joseph Andren; who has been a great role model and source of encouragement in my life with his hardworking, courageous, and enthusiastic character.

I am also extremely grateful to have received a number of fellowships that helped support my research and gave me the freedom to pursue difficult and challenging research problems. In particular, parts of this research is based upon work supported by a National Science Foundation (NSF) Graduate Research Fellowship (GRFP), National Defense Science and Engineering Graduate Fellowship (NDSEG), and Purdue Bilsland Dissertation Fellowship.

TABLE OF CONTENTS

Page

LIST OF TABLES x

LIST OF FIGURES xi

ABSTRACT xiii

1 INTRODUCTION 1

1.1 Motivation 2

1.2 Relational Time Series Learning 6

1.2.1 Dynamic Relational Representation Discovery and Tasks . . 7

1.2.2 Relational Time Series Prediction 10

1.3 Related Work 12

1.3.1 Tempoal Link Representation Tasks 12

1.3.2 Temporal Centrality and Analysis 12

1.3.3 Time Series Analysis 13

1.3.4 Relational Learning 13

1.3.5 Deep Learning 14

1.4 Thesis Statement and Contributions 15

1.5 Outline 19

2 DYNAMIC NODE LABELING 22

2.1 Motivation 22

2.2 Dynamic Relational Classification Framework 24

2.2.1 Temporal Granularity 25

2.2.2 Temporal Influence: Links, Attributes, Nodes 26

2.2.3 Temporal-Relational Classifiers 27

2.3 Temporal Ensemble Methods 29

2.3.1 Transforming the Temporal Nodes and Links 29

Page

2.3.2 Sampling or Transforming the Temporal Feature Space . . . 30
2.3.3 Adding Noise or Randomness 30
2.3.4 Transforming the Time-Varying Class Labels 30
2.3.5 Multiple Classification Algorithms and Weightings 30
2.4 Methodology and Data 31
2.4.1 Cora 31
2.4.2 Python Communication Network 32
2.4.3 Models 35
2.5 Unsupervised Dynamic Node Labeling 36
2.6 Experiments 40
2.6.1 Single Models 40
2.6.2 Temporal Ensemble Models 44
2.7 Related Work 48
2.8 Summary 50
3 DYNAMIC NODE WEIGHTING 51
3.1 Motivation 51
3.2 PageRank with Time-dependent Teleportation 55
3.2.1 A Generalization of PageRank 58
3.2.2 Choosing the Initial Condition 59
3.2.3 PageRank with Fluctuating Interest 59
3.2.4 Ranking from Time-series 62
3.2.5 Modeling Activity 64
3.2.6 Smoothing Empirical Activity 65
3.2.7 Choosing the Teleportation Factor 66
3.3 Methods for Dynamic PageRank 69
3.3.1 Forward Euler 69
3.3.2 Runge-Kutta 70
3.3.3 Maintaining Interpretability 71

Page

3.4 Experiments 72

3.4.1 Datasets 72

3.4.2 Rankings from Transient Scores 73

3.4.3 Difference Ranks 75

3.4.4 Predicting Future Page Views and Tweets 78

3.5 Applications 80

3.5.1 Clustering Dynamic Patterns 80

3.5.2 Discovering Causal Links 82

3.6 Related Work 86

3.7 Summary 88

4 DYNAMIC NODE PREDICTION 90

4.1 Motivation 90

4.2 Background 93

4.3 Framework for Dynamic Feature-based Roles 95

4.3.1 Feature Discovery 95

4.3.2 Dynamic Role Learning 96

4.4 Role Transition Models 100

4.4.1 Model Representation 100

4.4.2 From Global to Local Node Transition Models 101

4.4.3 Stacked Transition Model 103

4.4.4 Summary Transition Model 103

4.4.5 Remarks and Discussion 104

4.5 Experiments 105

4.5.1 Datasets and Structural Analysis 105

4.5.2 Model Validation using Ground-truth 105

4.5.3 Interpretation and Analysis 106

4.5.4 Predicting Future Behavior 110

4.5.5 Scalability and Complexity 113

Page

4.6 Applications . . . 117

4.6.1 Clustering Temporal Behaviors . . . 117

4.6.2 Anomalous Dynamic Patterns . . . 118

4.7 Related Work . . . 123

4.8 Summary . . . 124

5 SUMMARY AND CONCLUSION . . . 125

5.1 Future Work and Challenges . . . 125

5.1.1 Automatic Kernel Function Learning . . . 126

5.1.2 Space and Time Characterization . . . 126

LIST OF REFERENCES . . . 128

A Data . . . 142

B Research Codes and Libraries . . . 143

B.1 Dynamic PageRank Library . . . 143

B.2 Parallel Maximum Clique (PMC) Package . . . 143

VITA . . . 144

LIST OF TABLES

Table Page

1.1 Supervised and unsupervised techniques for relational time-series representation learning from dynamic attributed networks 16

1.2 Taxonomy of dynamic network prediction 18

1.3 Summary of applications for the proposed techniques 19

1.4 Overview of dissertation 20

2.1 Temporal-relational representation 25

2.2 Datasets used for empirical evaluation. 31

2.3 Categories of attributes computed from the temporal python communication network 34

2.4 Latent topics learned from the evolving textual information and the most significant words of each topic 48

3.1 Dynamic PageRank notation 56

3.2 Dataset properties 73

3.3 Ratio of error between base models and dynamic PageRank 80

3.4 Causal link discovery 83

4.1 Dataset characteristics 96

4.2 Feature validation 99

4.3 Role validation 99

4.4 Performance of the dynamic behavioral mixed-membership model . . . 114

LIST OF FIGURES

Figure Page

1.1 Dynamic attributed graph data 2

1.2 Overview of our approach for relational time series learning 4

1.3 Bridging the gap between relational learning and time series analysis . 5

1.4 Data model for relational time-series learning 6

1.5 Taxonomy of dynamic relational representation tasks for nodes 7

2.1 Including temporal weights for the attributes and links 28

2.2 Relational schema for the Cora network 32

2.3 Discovering the underlying latent semantics of the links 37

2.4 Defining a network from the evolving textual information 39

2.5 Comparing TVRC against RPT and DT models 41

2.6 Exploring the space of temporal relational models 42

2.7 Exploring temporal granularity models 43

2.8 Comparing temporal, relational, and traditional ensembles 45

2.9 Comparing attribute classes w.r.t. temporal, relational, and traditional ensembles. 46

2.10 Randomization tests to compare ensembles 47

2.11 Evaluation of relational time-series classifiers using the time-series of latent topic features 49

3.1 Illustration of evolving teleportation distribution 54

3.2 Evolution of the PageRank dynamical system 63

3.3 Evolution of PageRank values for one node 66

3.4 Evolution of PageRank values for one node (cont.) 67

3.5 Evolution of PageRank values for one node (cont.) 68

3.6 Forward Euler method for evolving the dynamical system 71

Figure Page

3.7 Intersection similarity of rankings 74

3.8 Top pages from the difference ranking 76

3.9 Top pages from difference rank using $s = 4$ 77

3.10 Clustering nodes using dynamic PageRank scores 81

3.11 Granger causality graph 84

4.1 Roles and communities are complimentary 94

4.2 Visualization of mixed-memberships and ground-truth 100

4.3 DRMM captures the diverse temporal behavior 108

4.4 Evolving mixed-memberships in an email network 109

4.5 Predicting future behavior of individual nodes 111

4.6 Evolution and patterns of local node structural behavior 115

4.7 Evolution and patterns of local node structural behavior (cont.) 116

4.8 Clustering nodes with similar role transition models 118

4.9 Detecting nodes with anomalous temporal behavior 119

4.10 Differences in static and dynamic behavior in the email network 121

4.11 Detecting time-varying node anomalies 122

ABSTRACT

Rossi, Ryan A. Ph.D., Purdue University, August 2015. Improving Relational Machine Learning by Modeling Temporal Dependencies. Major Professor: Sunil Prabhakar.

Networks encode dependencies between entities (people, computers, proteins) and allow us to study phenomena across social, technological, and biological domains. These networks naturally evolve over time by the addition, deletion, and changing of links, nodes, and attributes. Existing work in Relational Machine Learning (RML) has ignored *relational time series data* consisting of *dynamic graphs* and *attributes*, even despite the importance of modeling these dynamics.

This dissertation investigates the problem of *Relational Time-series Learning* (RTL) from dynamic attributed graph data, with the goal of improving the predictive quality of existing RML methods. In particular, we propose a framework for learning dynamic graph representations, as well as methods for the three representation discovery tasks of (i) dynamic node labeling, (ii) weighting, and (iii) prediction. In addition, techniques for modeling relational and temporal dependencies are proposed, along with efficient methods for discovering features, ensembles, as well as classification methods. The results demonstrate the importance of modeling both the relational *and* temporal dependencies as well as learning an appropriate graph data representation that captures these fundamental patterns. Furthermore, while previous work has focused on static graphs that are small, non-attributed, simple, or homogeneous, we instead have carefully designed *generalized relational time-series models* that are: (a) efficient with linear or nearly linear runtime, (b) scalable for big graph data, (c) flexible for a variety of data types and characteristics, and (d) capable of modeling attributed and heterogeneous relational time-series data. Finally, the proposed methods are shown to be scalable, effective, and flexible for a variety of real-world applications.

1. INTRODUCTION

This dissertation investigates the problem of learning dynamic relational representation from a time series of attributed graph data. The purpose is to achieve better predictions for relational time series classification and regression.

Recently, Statistical Relational Learning (SRL) [1] methods were developed to leverage the relational dependencies [2] between nodes [3–8]. In many cases, the SRL algorithms were shown to improve over traditional machine learning approaches [3,5,8]. This dissertation builds upon this body of knowledge by considering dynamic attributed graphs and three main representation learning tasks including dynamic node labeling, dynamic node weighting, and dynamic node prediction. In particular, we propose novel methods and tools for modeling the relational *and* temporal dependencies of the nodes, links, and attributes. Using the modeled dependencies, we automatically learn a time series feature-based representation that captures the fundamental properties of the data. The learned representation from the dynamic attributed network is then used for relational time series prediction. Our work investigates the problem of discovering a dynamic graph data representations for improving the accuracy of predictive models as well as a variety of other machine learning tasks.

In addition, this dissertation seeks to answer the following main questions: (1) How to represent, summarize, and incorporate the temporal dependencies in dynamic attributed networks? (2) How to discover effective features from dynamic attributed networks? (3) How to use these features to build accurate relational time-series forecasting models? (4) How to use them for solving other problems in machine learning such as temporal-relational pattern mining and clustering? In particular, our main focus is on discovering representations of dynamic attributed graphs that capture the fundamental relational and temporal dependencies for improving relational time series prediction.

1.1 Motivation

In recent years, relational data has grown at a tremendous rate; it is present in domains such as the Internet and the world-wide web [9–11], scientific citation and collaboration [12,13], epidemiology [14–17], communication analysis [18], metabolism [19,20], ecosystems [21,22], bioinformatics [23,24], fraud and terrorist analysis [25,26], and many others. The links in these data may represent citations, friendships, associations, metabolic functions, communications, co-locations, shared mechanisms, or many other explicit or implicit relationships.

The majority of these real-world relational networks are naturally dynamic—evolving over time with the addition, deletion, and changing of nodes, links, and attributes. Despite the fundamental importance of these dynamics, the majority of work in *relational learning* has ignored the dynamics in relational data (i.e., attributed network data). In particular, dynamic attributed graphs have three main components that vary in time. First, the attribute values (on nodes or links) may change over time (e.g., research area of an author). Next, links might be created and deleted throughout time (e.g., host connections are opened and closed). Finally, nodes might appear and disappear over time (e.g., through activity in an online social network). Figure 1.1 provides an intuitive view of these underlying dynamics.

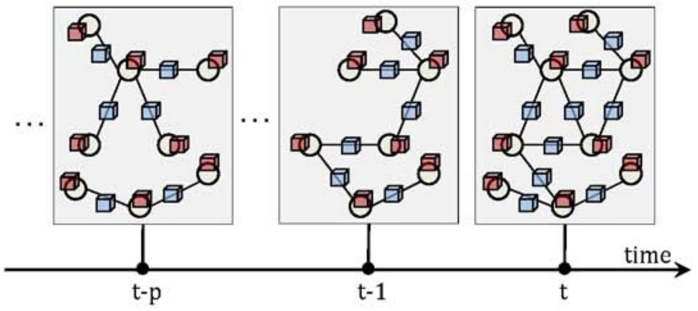


Fig. 1.1.: Dynamic attributed graph data.

Previous research in machine learning (ML) assumes independently and identically distributed data (IID) [27,28] and has ignored relational dependencies (and temporal dependencies). This independence assumption is often *violated* in relational data as (relational) dependencies among data instances are naturally encoded. More specifically, *relational autocorrelation* is a correlation or statistical dependence between the values of the same attribute across linked instances [29] and is a fundamental property of many relational data sets. For instance, people are often linked by business associations, and information about one person can be highly informative for a prediction task involving an associate of that person. Recently, SRL methods [1] were developed to leverage the relational dependencies (i.e., relational autocorrelation [2], also known as homophily [30]) between nodes [3, 4, 6–8]. In many cases, these relational learning methods improve predictive performance over traditional IID techniques [3, 5, 8].

Relational learning methods have been shown to improve over traditional ML by modeling relational dependencies, yet they have ignored temporal information (i.e., explicitly assumes the data is independent with respect to time). In that same spirit, our work seeks to make further improvements in predictive performance by incorporating temporal information and designing methods to accurately learn, represent, and model temporal *and* relational dependencies. The temporal information is known to be significantly important to accurately model, predict, and understand relational data [31, 32]. In fact, time plays a key role in many natural laws and is at the heart of our understanding of the universe, i.e., the unification of space and time in physics [33] and how time is related to space and vice-versa is fundamentally important to our understanding and interpretation of the universe and its laws [33, 34]. We make a similar argument here, that ignoring time in attributed networks can only add further uncertainty, as time places a natural ordering on the network components, including the changing of attribute-values, links, and nodes.

This dissertation formulates the problem of relational time series learning and proposes a framework that consists of two main components as shown in Figure 1.2. The first component learns a feature-based representation from a collection of dynamic relational data (i.e., a time series of graphs and attributes) given as input which incorporates the fundamental temporal dependencies in relational graph data. While the second component leverages the learned feature-based representation for relational

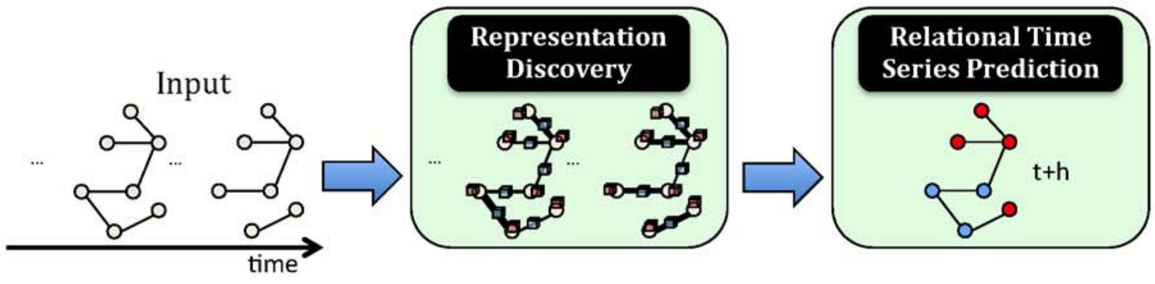


Fig. 1.2.: Overview of our approach for relational time series learning.

time series prediction, which includes both relational time series classification models and regression. In other words, this dissertation proposes techniques for learning an appropriate representation from dynamic attributed graph data for the purpose of improving accuracy of a given relational learning[1] (see [1]) task such as classification [2]. From another perspective, one may view these techniques as a system for learning a compact and representative set of dynamic graph features that capture the fundamental temporal dependencies in the data. In that light, the proposed methods may also be useful for many other applications (i.e., besides predicting node attributes) including dynamic network analysis [35, 36], anomaly detection [37] in dynamic graphs [38–41], dynamic ranking [42–44], among many other machine learning tasks and applications [45–47].

We propose a taxonomy shown in Figure 1.5 for the fundamental representation tasks for nodes in dynamic attributed networks. This dissertation focuses on the three fundamental representation tasks for dynamic attributed networks including dynamic node labeling, dynamic node weighting, and dynamic node prediction. As an aside, let us note that we do not focus on the (symmetric) link-based representation tasks since they have received considerable attention recently (in various other forms) [48–57] and therefore this work focuses on the novel dynamic representation tasks for nodes.

Ultimately, the goal of this dissertation is to help further bridge the gap between relational learning [1–4, 6–8] and traditional time series analysis [58–64]. See Figure 1.3.

[1]This area of machine learning is sometimes referred to as Statistical Relational Learning (SRL) or Relational Machine Learning (RML).

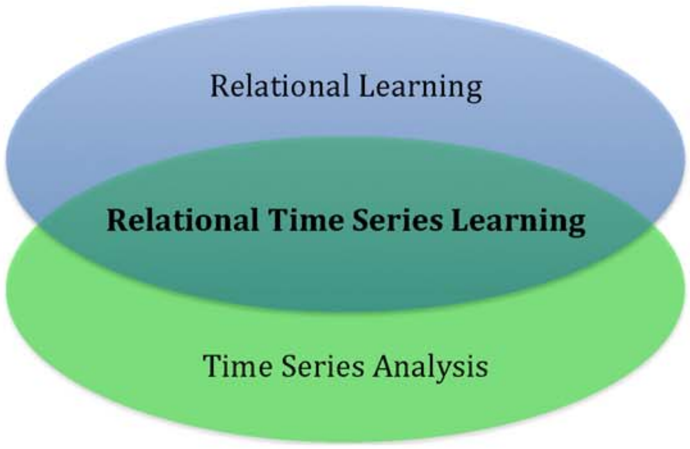


Fig. 1.3.: Toward bridging the gap between relational learning and time series analysis. The focus of this dissertation is on the intersection between these two areas which we call *Relational Time Series Learning*. Relational learning has primarily focused on static relational graph data (graphs + attributes) whereas the time series analysis literature avoids graph data and instead focuses on independent time series (i.e., modeling each of time series disjointly) or time series that are assumed to be completely dependent (i.e., clique). Despite the fact that each is independent of one another, we find that many of the known techniques from time series analysis may help in modeling time series of relational graph data and attributes.

This gap between these fundamentally important problems seemingly arose due to the fact that the majority of relational learning techniques from the machine learning community has ignored temporal or dynamic relational data [1,3,4,6–8,65,66] whereas the time series work has *ignored graph data* and has mainly focused on (i) modeling independent time series or (ii) multiple time series that are assumed to be completely correlated [58–61]. The intersection of these two areas is *relational time series learning* and differs significantly from relational learning and time series analysis in the data utilized, model and data assumptions, and their objectives. For instance, the prediction objective of the relational learning problem is mainly for within-network or across-network prediction tasks and does not predict the future node attribute-values. Relational learning does not utilize or model temporal information whereas time series analysis lacks relational information. There are many other differences discussed later.

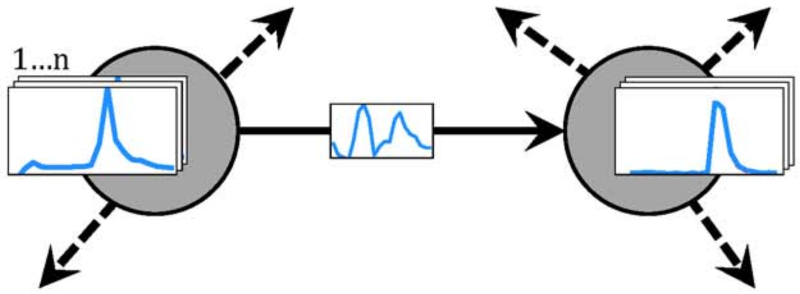


Fig. 1.4.: Data model for relational time-series learning. Each node in the network may have an arbitrary number of time series attributes such as blood pressure, number of hourly page views, etc. Further, we also assume that each edge in the network may have an arbitrary number of time series attributes as well (not shown for simplicity). For each edge, we also model the temporal dependencies, resulting in the time series on the edge in the illustration above. As an aside, if there are multiple types of edges between nodes such as friend-edges and email-edges representing friendship between two individuals and email communications, respectively, then we would model the temporal dependencies for each of the edge types resulting in learning multiple time series of edge temporal edge strengths.

1.2 Relational Time Series Learning

Relational data in the real-world is naturally dynamic—evolving over time with the addition, deletion, and changing of nodes, links, and attributes. Examples of dynamic relational data include social, biological, technological, web graphs, information networks, among many other types of networks. In particular, dynamic attributed graphs have three main components that vary in time. First, the attribute values (on nodes or links) may change over time (e.g., research area of an author). Next, links might be created and deleted throughout time (e.g., host connections are opened and closed). Finally, nodes might appear and disappear over time (e.g., through activity in an online social network). An intuitive illustration of the underlying dynamics governing relational data is shown in Figure 1.1.

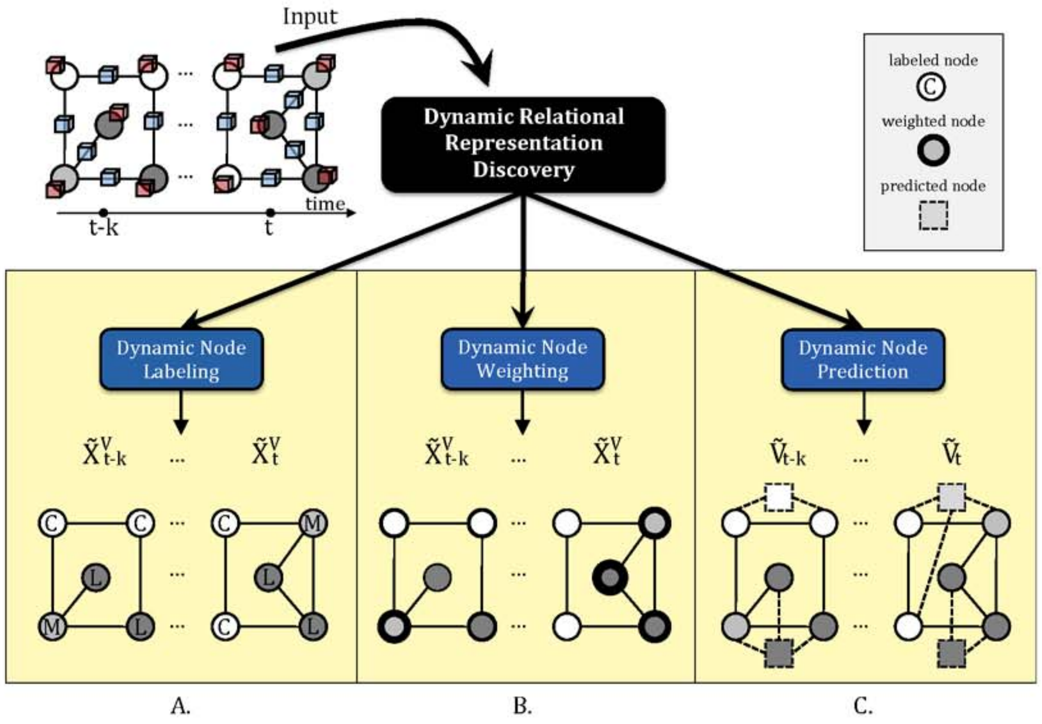


Fig. 1.5.: Taxonomy of dynamic relational representation tasks for nodes For the dynamic node representation tasks, we introduce a dynamic relational representation taxonomy focused on the representation tasks of dynamic node labeling, weighting, and predicting the existence of nodes. This taxonomy provides a fundamental basis for organizing the main techniques proposed and their key contributions. In particular, Section 2 focuses on the dynamic relational representation task of labeling while the dynamic node weighting problem is investigated in Section 3. Finally, Section 4 also proposes dynamic node prediction techniques that focus on predicting the existence of nodes.

1.2.1 Dynamic Relational Representation Discovery and Tasks

Representation issues have been at the heart of the artificial intelligence community for decades [67–69]. Most recently, relational data representations have become an increasingly important topic due to the recent proliferation of network datasets (e.g., social, biological, information networks) and a corresponding increase in the application of Statistical Relational Learning (SRL) algorithms to these domains. In particular, appropriate transformations of the nodes, links, and/or features of the data can dramatically affect the capabilities and results of SRL algorithms. See Rossi *et al.* [2]

for a comprehensive survey on relational representation discovery (for static graph data).

In this work, we use relational autocorrelation and along with two temporal dependencies in dynamic attributed networks. More precisely, we observed two fundamental temporal dependencies of dynamic relational network data including the notion of temporal locality and temporal recurrence. We define these temporal dependencies informally below since they apply generally across the full spectrum of temporal-relational information including non-relational attributes, relational node attributes, relational edge attributes, as well as edges and nodes in a graph.

Property* 1** (Temporal Locality):*** *Recent events are more influential to the current state than distant ones.*

This temporal dependency implies that a recent node attribute-value, edge attribute-value, or link is stronger or more predictive of the future than a more distant one. In terms of attribute-values on nodes (or edges) this implies that a recently observed attribute-value (e.g., number of posts) at t is more predictive than past observations at $t-1$ and more distant. Hence, if $x_i(t) = \alpha$ is observed at time t, then at time $t+1$ it is likely that $x_i(t+1) = \alpha$. In the case of edges, this implies that a recently observed edge $(v_i, v_j) \in E_t$ between v_i and v_j at time t implies that there is a high probability of a future edge $(v_i, v_j) \in E_{t+1}$ at $t+1$ will arise.

Property* 2** (Temporal Recurrence):*** *A regular series of observations are more likely to indicate a stronger relationship than an isolated event*

The notion of temporal recurrence implies that a repeated sequence of observations are more influential or have a higher probability of reappearing in the future than an isolated event. In other words, a repeated or regular sequence of node attribute-values (or edge attribute-values) are more likely to reappear in the future than an isolated node attribute-value. As an example, given a communication network and a node attribute representing the topic of communication for each node in the network, if node v_i has a regular sequence of topics, i.e., $x_i(k) = \alpha, \text{for} k = p, ..., t$ across a recent set of timesteps, then there is a higher probability that $x_i(t+1) = \alpha$ is observed than another topic of communication. In terms of edges, temporal recurrence implies that a repeated or recurring series of edges $(v_i, v_j) \in E_k, \text{for} k = p, ..., t$ between v_i and v_j implies a higher probability of a future edge $(v_i, v_j) \in E_{t+1}$ at $t+1$. As an aside, temporal recurrence is based on regular or recurring series of similar observations whereas temporal locality is based on the notion that the most recent observations are likely to persist in the future.

Learning accurate relational time series representations for nodes in dynamic attributed networks remains a challenge. Just as SRL methods were designed to exploit the relational dependencies in graph data, we instead leverage the relational dependencies *and* the temporal dependencies of the edges, vertices, and attributes to learn more accurate dynamic relational representations.

In this dissertation, we formulate the problem of dynamic relational representation discovery and propose a taxonomy for the dynamic node representation tasks shown in Figure 1.5. More specifically, the dynamic representation tasks for nodes include (i) predicting their label or type, (ii) estimating their weight or importance, and (iii) predicting their existence. We propose methods for each of the dynamic relational node representation tasks in Figure 1.5 which are defined below.

Problem 1 (**Dynamic Node Labeling**): Given a time-series of attributed graph data, we define the dynamic node labeling problem as the task of learning a time series of node labels $\mathbf{X}_p, ..., \mathbf{X}_t$ where for each timestep a given node may be assigned a single label (i.e., class label) or multiple labels (i.e., multiple topics or roles). The time series of labels may represent a known class label previously observed or a latent variable such as roles, topics, among many others.

Problem 2 (**Dynamic Node Weighting**): Given a time-series of attributed graph data, we define the dynamic node weighting representation task as the learning of a time series of weights for the nodes $\mathbf{X}_p, ..., \mathbf{X}_t$ that utilize relational and temporal dependencies in the dynamic relational data. The time series of weights may represent the importance or influence of a node in the dynamic attributed network or it may simply represent a latent variable capturing the fundamental dependencies in the dynamic relational data.

Problem 3 (**Dynamic Node Prediction**)**:** Given a time-series of attributed graph data, we define the dynamic node prediction representation task as the prediction of the existence of a node in a future timestep $t + 1$ where the learning leverages past temporal-relational data and more specifically incorporates relational and temporal dependencies in the dynamic relational data. The predicted node may represent a novel type of node, not yet discovered such as a role or topic of communication, or it may be a novel node from an already existing node type such as a Facebook user or group. Most techniques also predict the existence of edges between the predicted node and the set of nodes in the graph.

1.2.2 Relational Time Series Prediction

Using the learned representation, we demonstrate the effectiveness of these techniques for relational time series *classification* and *regression* of dynamic node attributes. We define relational time series classification and regression more precisely below.

Problem 4 (Relational Time Series Classification)**:** *Given a "known" time series of attributed graph data* $\mathcal{G} = \{\mathbf{G}_1, ..., \mathbf{G}_t\}$ *for learning, the task is to infer the class labels* $\mathbf{Y}_{t+h}$ *of the nodes at time* $t + h$ *in the graph where* L *refers to the set of possible labels.*

As an aside, if $h = 1$ then we call this one-step ahead prediction, whereas multi-step ahead prediction refers to the case when $h > 1$, and thus the prediction is across multiple timesteps. Our relational time series classification methods are also flexible for both binary (i.e., $|L| = 2$) and multi-class problems (i.e., $|L| > 2$), whereas binary classification has been the primary focus of the past relational learning methods for static graphs. Similarly, we also investigate the relational time series regression problem.

Problem 5 (Relational Time Series Regression)**:** *Given a time series of attributed graphs $\mathcal{G} = \{\mathbf{G}_1, ..., \mathbf{G}_t\}$, the task is to estimate the real-valued variable $\mathbf{Y}_{t+h} \in \mathbb{R}^n$ at time $t + h$ for the nodes in the graph.*

The prediction task investigated in this dissertation is also fundamentally different than the traditional relational learning problems/assumptions. More specifically, we define within-network (e.g., inference) as the task where training instances from a single (static) graph are connected directly to instances whose classification labels are to be predicted [5, 70]. Conversely, the task of across-network inference attempts to learn a model on a (static) network and applying the learned models to a separate network [71, 72]. For instance, we may learn a model from a static and/or aggregated graph from Facebook and use that learned model for prediction on another social network such as Google+ or Twitter. While both prediction problems for relational learning assume a static network, they also differ fundamentally in their underlying assumptions and goals. On the other hand, we focus on using the past time series of attributed graphs where the training nodes may be connected directly to nodes whose classification labels are to be predicted and similarly the past time series observations of the prediction attribute may also be directly used. The fundamental idea is that both past relational and temporal dependencies and information may be used to predict the future time series values of a given attribute. We also note that we may learn a model using some past data and use it to predict the future value at $t + h$ of an attribute time series, or we could use a technique that does "lazy learning" in the sense that the past data is determined upon prediction time and used for predicting $t + h$. Our work includes both types of methods. As an aside, most of the relational learning methods for static graph data is mainly for classification, while regression has received considerably less attention. Despite this seemingly systematic bias towards classification, we focus on the two complimentary dynamic prediction tasks, namely, relational time series classification and regression.

1.3 Related Work

This section briefly reviews and discusses the past work.

1.3.1 Tempoal Link Representation Tasks

While our dynamic relational representation discovery taxonomy shown in Figure 1.5 focuses on the labeling, weighting and prediction of nodes, there is also the symmetric dynamic graph representation tasks for links which includes link labeling, link weighting, and link prediction. Our work is not concerned with the link-based dynamic representation tasks as these have been investigated in various contexts [48–57]. For instance, link prediction and weighting has been used to improve search engines [54], recommendation systems [73] for both products [51, 52] and friends (i.e., social recommendation) [74], among many others [75–77]. We also note that other work has focused on predicting links in temporal networks using tensor factorizations [78] and predicting structure in these networks using frequent subgraphs [79].

1.3.2 Temporal Centrality and Analysis

Recently, there has been a lot of work on analyzing dynamic or temporal graphs which has focused solely on edges that change over time, and has ignored and/or discarded any attributes (both dynamic or static) [35, 80–88]. Centrality measures have also been extended for temporal networks [35,36]. While the vast majority of this work has focused only on dynamic edges (i.e., dynamic/temporal/streaming graphs), we instead focus on dynamic relational data and incorporate the full spectrum of dynamics including edges, vertices, and attributes (and their static counterparts as well).

1.3.3 Time Series Analysis

Last section discussed temporal graph analysis which lacked attribute data, whereas non-relational attribute-based time series data [62–64] is the focus of this section. In particular, traditional time series methods ignore graph data all together [58–61], and focus solely on modeling a time-dependent sequence of real-valued data such as hourly temperatures or economic data such as stock price or gross domestic product (GDP) [62–64]. In contrast, our proposed methods naturally allow for modeling time series of attributes and graphs (i.e., relational time series data) where each node and edge may have a multi-dimensional time series with arbitrary connectivity or dependencies between them as shown in Figure 1.4.

At the intersection of time series analysis and machine learning, Ahmed *et al.* [61] recently used machine learning methods such as Neural Networks [89] and SVMs [90] for time series forecasting. In particular, the authors found that many of these machine learning methods offered significant improvements over the traditional time series models [61] such as auto-regressive (AR) models and the ilk [60]. The main contribution of the work by Ahmed *et al.* [61] was there use of traditional machine learning methods for time series forecasting, which has recently attracted numerous follow-up studies [91–93]. From that perspective, our work makes a similar contribution as we formulate the problem of relational time series learning for dynamic relational graph data, and propose techniques for relational time series classification and regression, which are shown to improve over traditional relational learning and time series methods.

1.3.4 Relational Learning

The majority of research in relational learning has focused on modeling static snapshots or aggregated data [65, 66] and has largely ignored the utility of learning and incorporating temporal dynamics into relational representations. Previous work in relational learning on attributed graphs either uses static network snapshots or significantly limits the amount of temporal information incorporated into the models.

Sharan *et al.* [94] assumes a strict representation that only uses kernel estimation for link weights, while GA-TVRC [95] uses a genetic algorithm to learn the link weights. Spatial-RPTs [96] incorporate temporal and spatial information in the relational attributes. However, the above approaches focus only on one specific temporal pattern and do not consider different temporal granularities (i.e., they use all available snapshots and lack the notion of a lagged time-series). In contrast, we explore a larger space of temporal-relational representations in a flexible framework that can capture temporal dependencies over *links*, *attributes*, and *nodes*. To the best of our knowledge, we are the first to leverage the full spectrum of dynamic relational data to improve predictions.

We are also the first to propose and investigate *temporal-relational ensemble methods* for time-varying relational classification. However, there has been recent work on relational ensemble methods [97–99] and non-relational ensemble methods for evolving streams [100]. While none of the past work proposes *temporal-relational ensemble methods* for classification, there has been recent work on relational ensemble methods [97–99]. In particular, Preisach *et al.* [97] use voting and stacking methods to combine relational data with multiple relations whereas Eldardiry and Neville [99] incorporates prediction averaging in the collective inference process to reduce both learning and inference variance.

1.3.5 Deep Learning

Our work is also related to the machine learning topic of deep learning [101–107], which has recently received a considerable amount of attention from industry due to its success in a variety of real-world applications and systems [108–110]. However, nearly all of this work has focused on images and other similar types of data, whereas we focus on dynamic attributed networks. In view of our work, deep learning for dynamic relational data is informally any method that constructs a representation with varying levels of abstraction or granularity with dependencies between the various layers. For instance, our proposed DRMM method for node prediction first learns a large set of features, then we discover roles from those features using matrix factorization (i.e., capturing the essence of that set of features), and finally we model the role transitions over time. These representations form a hierarchy of layers each capturing a different level of granularity in the dynamic attributed networks.

1.4 Thesis Statement and Contributions

In this dissertation, we investigate the problem of *relational time series learning* and propose techniques for the dynamic representation tasks of dynamic node labeling and weighting. Using the learned representation, we demonstrate the effectiveness of these techniques for relational time series *classification* and *regression* of dynamic node attributes. The main thesis of this dissertation can be stated as follows:

> *Discovering an appropriate dynamic graph representation that captures the relational and temporal dependencies of the nodes, edges, and attributes, will improve the accuracy of predictive models*

In this dissertation, we proposed a variety of methods for learning dynamic relational representation from a time series of attributed graph data. Using the learned dynamic relational representation as a basis, we focus on two problems of fundamental importance: (a) how to learn a time-series of features for the graph-based forecasting tasks including classification, regression, and multivariate regression problems, (b) how to use the learned features from the dynamic relational representation methods to solve other problems in machine learning such as anomaly detection or other qualitative tasks such as clustering the main time series patterns in a dynamic attributed networks? The main contributions of this dissertation are as follows:

- ▷ We introduce an intuitive taxonomy for *dynamic relational representation discovery* that formulates the node representation tasks of (i) predicting node labels, (ii) estimating node weights, and (iii) predicting their existence.

Table 1.1.: Supervised and unsupervised techniques for relational time-series representation learning from dynamic attributed networks

		Supervised	**Unsupervised**
Dynamic Node	**Labeling**	Relational Time Series Classification (DRC [111] [18] [2])	▷ Dynamic Local Structural Behaviors (Dynamic Role Mixed-Membership Model (DRMM) [41] [112]) ▷ Dynamic Latent Topics (Latent Textual Semantics [18])
	Weighting	Relational Time Series Regression [44]	Dynamic Node Ranking and Importance (Dynamic PageRank [113])

- ▷ We propose three techniques for learning dynamic relational representations for improving SRL tasks in dynamic attributed graphs (based on the above taxonomy) In addition, the proposed methods are also particularly useful for (1) dynamic relational classification (DRC), (2) dynamic ranking and importance (DPR), and (3) modeling large dynamic networks (DRMM).

- ▷ We propose a general framework for dynamic relational representation discovery and use the various components for dynamic relational classification (DRC). More specifically, we develop a family of methods called dynamic relational classifiers (DRC) which are suitable for heterogeneous networks where links and nodes may be of different types. The proposed dynamic relational classification methods use a variety of kernel functions to model the dynamic influence of the edges, vertices, and attributes. These weights are then incorporated into the dynamic relational classifiers to moderate the influence of the edges, vertices, and attributes. We use supervised feature construction to systematically select the most significantly correlated features using the class labels. In addition, we learn the model parameters automatically learned using cross-validation. This method is suitable for both heterogeneous and homogeneous dynamic networks and utilizes a time-series of graphs and attributes of any type (continuous, discrete, etc). DRC performs time-series forecasting of a discrete class label. In all cases, the proposed dynamic relational classifiers outperform competing models that ignore temporal information.

▷ We develop *Dynamic Pagerank* to model the centrality of a vertex as external interest in those vertices vary. We demonstrate the utility of dynamic pagerank using Wikipedia where external interest is a time-series of hourly page views. The method learns a time-series of importance scores which is shown to be useful in time-series forecasting of continuous real-valued attributes. Moreover, we also demonstrate its effectiveness for identifying time-series patterns/trends, anomalies, and for identifying causal links.

▷ In addition, we propose *Dynamic Role Mixed-Membership Model* (DRMM) to learn a time-series of roles (cliques, star-centers, or star-edges) based on a set of dynamic graph features. We then compute a time-series of transition models that describe the role transitions. This model is useful for predicting future behavior, identifying patterns/trends, and anomaly detection.

▷ We reinterpret the SRL prediction tasks for relational time series data that leads to the problem formulation of relational time series classification and relational time series regression. Using the proposed dynamic relational representation methods as a basis, we systematically investigate the relational time series forecasting tasks. In particular, we focus on the forecasting tasks in dynamic attributed networks that include: (i) predicting the future label (i.e., classification), and (ii) predicting the weight or importance (i.e., regression).

Table 1.2 provides an overview of the proposed techniques indicating the temporal and relational information leveraged in each of the methods. Nearly all the past work in relational learning has ignored the dynamics and thus assumes the graph data is

Table 1.2.: The techniques are categorized by the type of dynamic information used in the various dynamic relational models.

	Static Attributes	**Dynamic Attributes**
Static Graph	✗	Dynamic PageRank [44] [113]
Dynamic Graph	DRMM [41] [112], [114]	DRC [111] [18] [2]

static. To the best of our knowledge, we are the first to (i) propose dynamic relational discovery techniques for nodes to improve accuracy, (ii) model both the relational and temporal dependencies for prediction in dynamic relational data, and (iii) introduce dynamic relational ensemble methods.

The methods proposed in this dissertation are useful for a variety of tasks beyond relational time series classification and regression. To further demonstrate the effectiveness of the dynamic relational representation discovery techniques, we use them for a variety of other graph-based machine learning tasks including anomaly detection (and its graph-based variations), clustering time-series patterns, identifying causal links, dynamic ranking, pattern mining, and network exploratory analysis. All techniques are extensively evaluated for real applications such as importance/ranking of web pages, anomaly detection, and pattern mining. See Table 1.3 for a summary of the applications used to evaluate the effectiveness of our methods. The proposed methods are shown to be scalable, effective, and flexible for use in a variety of real-world applications. We have also used our feature learning approach for evaluation of graph generators that were designed specifically to model the evolution of the Internet AS and router-level topologies, see [114] for more details. More importantly, we observed a significant transition in the structure of the Internet AS.

In addition, the proposed methods may also be viewed as dynamic feature construction techniques. Dynamic graph features may be learned by one method and used to improve the accuracy or scalability of another proposed method. For instance, the learned dynamic features may be used as input into another technique (i.e., the time series of importance scores from DPR may be used in DRC to learn additional temporal-relational features for prediction, etc). Moreover, we proposed both unsupervised and supervised methods for learning in dynamic attribtued networks.

1.5 Outline

The remainder of this dissertation is organized as follows. Chapter 2 introduces the framework for dynamic node labeling along with our proposed techniques for relational time series classification. In Chapter 3, we propose an approach for the representation task of dynamic node weighting and utilize it for relational time series regression (i.e., predicting number of hourly page views on Wikipedia). Our approach for dynamic node prediction is proposed in Chapter 4. Finally, Chapter 5 concludes with a summary of contributions.

Parts of this dissertation have been published in conferences and journals. In particular, the survey and taxonomy of relational representation transformation is described in a paper [2] published in the Journal of Artificial Intelligence Research (JAIR). The

Table 1.3.: Summary of applications for the proposed techniques. Note that ✓ indicates the applications evaluated in this dissertation.

Task	**Proposed Method**	**Attribute Prediction**	**Discovering Link Causality**	**Clustering Time-series**	**Network Anomaly Detection**	**Dynamic Importance/Ranking**	**Temporal Pattern Mining**	**Graph Similarity**
Node Labeling	Dynamic Relational Classifiers (DRC) [111] [18] [2]	✓					✓	
Node Weighting	Dynamic PageRank (DPR) [44] [113]	✓	✓	✓	✓	✓	✓	
Node Prediction	Dynamic Role Model (DRMM) [41] [112] [114]	✓		✓	✓		✓	✓

Table 1.4.: Overview of dissertation. We propose four methods to discover dynamic graph features. These methods may also be used for modeling, ranking, classification, and mining dynamic graphs.

Representation Task		**Prediction**	**Proposed Methods**
Dynamic Node Labeling	(Ch. 2)	Classification	Dynamic Relational Classifiers (DRC) [111] [18] [2]
Dynamic Node Weighting	(Ch. 3)	Regression	Dynamic PageRank (DPR) [44] [113]
Dynamic Node Prediction	(Ch. 4)	Classification	Dynamic Roles (DRMM) [41] [112] [114]

work on modeling temporal dependencies for dynamic relational classification and ensemble methods (Chapter 2) is described in a paper [111] published in the Advances in Knowledge Discovery and Data Mining (PAKDD) whereas the work on automatically learning the evolution of latent topics and incorporating their influence into a time-evolving relational learning algorithm is described in a paper [18] published in the 2010 Proceedings of the First Workshop on Social Media Analytics (SOMA SIGKDD). The work on Dynamic PageRank (DPR) from Chapter 3 is described in a paper [44] published in the 2012 Proceedings of the Workshop on Algorithms and Models for the Web Graph (WAW 2012). An extended version of this work (including additional theoretical analysis, solvers, data, and experiments) is described in a paper [113] published in the 2014 Journal of Internet Mathematics. For Chapter 4, the proposed feature-based roles along with a general framework for computing them was published in the IEEE Transactions on Knowledge and Data Engineering [115]. In addition, the work based on using dynamic roles for characterizing the structural patterns and trends in dynamic networks is described in a paper [112] in the Proceedings of the 2012 International Conference Companion on World Wide Web (LSNA WWW) whereas the work on our Dynamic Role Mixed-Membership Model (DRMM) is described in a paper [41] in the 2013 Proceedings of the Sixth ACM International Conference on Web Search and Data Mining (WSDM). Additionally, the application of this work for understanding the evolution of the AS and for the subsequent task of evaluating state-of-the-art topology generators was accepted for demonstration [114] in the 2013 IFIP Networking Conference.

In an effort to support future research, we also published a proposal for an *interactive data repository* [116] that goes beyond existing scientific data repositories by providing visual analytic techniques for real-time interactive data exploration, mining, and visualization [116]. Using that proposal as a basis, we developed the first interactive data repository for graphs [117, 118] (`http://networkrepository.com`) which currently has over 500 networks and includes the relational time-series data used throughout this dissertation [117, 118]. Finally, a visual graph analytic platform is described in a paper [119] published in the Proceedings of AAAI 2015. In particular, the platform developed visual graph mining and exploration techniques for understanding large dynamic attributed networks and included the first interactive relational machine learning methods for both (i) supervised learning (e.g., interactive relational classification methods) and (ii) unsupervised learning (e.g., interactive role discovery).

2. DYNAMIC NODE LABELING

Relational networks often evolve over time by the addition, deletion, and changing of links, nodes, and attributes. However, accurately incorporating the full range of temporal dependencies into relational learning algorithms remains a challenge. We propose a novel framework for discovering *temporal-relational representations* for classification. The framework considers transformations over *all* the evolving relational components (attributes, edges, and nodes) in order to accurately incorporate temporal dependencies into relational models. Additionally, we propose *temporal ensemble methods* and demonstrate their effectiveness against traditional and relational ensembles on two real-world datasets. In all cases, the proposed temporal-relational models outperform competing models that ignore temporal information.

2.1 Motivation

Temporal-relational information is present in many domains such as the Internet, citation and collaboration networks, communication and email networks, social networks, biological networks, among many others. These domains all have attributes, links, and/or nodes changing over time which are important to model. We conjecture that discovering an accurate *temporal-relational representation* will disambiguate the true nature and strength of links, attributes, and nodes. However, the majority of research in relational learning has focused on modeling static snapshots [65, 66] and has largely ignored the utility of learning and incorporating temporal dynamics into relational representations.

Temporal relational data has three main components (attributes, nodes, links) that vary in time. First, the attribute values (on nodes or links) may change over time (e.g., research area of an author). Next, links might be created and deleted throughout time (e.g., host connections are opened and closed). Finally, nodes might appear and disappear over time (e.g., through activity in an online social network).

Within the context of evolving relational data, there are two types of prediction tasks. In a *temporal* prediction task, the attribute to predict is changing over time (e.g., student GPA), whereas in a *static* prediction task, the predictive attribute is constant (e.g., paper topic). For these prediction tasks, the space of temporal-relational representations is defined by the set of relational elements that change over time (attributes, links, and nodes). To incorporate temporal information in a representation that is appropriate for relational models, we consider two transformations based on *temporal weighting* and *temporal granularity.* Temporal weighting aims to represent the temporal influence of the links, attributes and nodes by decaying the weights of each with respect to time, whereas the choice of temporal granularity restricts attention to links, attributes, and nodes within a particular window of time. The optimal temporal-relational representation and the corresponding temporal classifier depends on the particular temporal dynamics of the links, attributes, and nodes present in the data, as well as the network domain (e.g., social vs. biological networks).

In this work, we address the problem of selecting the most optimal temporal-relational representation to increase the accuracy of predictive models. We consider the full space of *temporal-relational representations* and propose **(1)** a temporal-relational classification framework, and **(2)** a set of temporal ensemble methods, to leverage time-varying links, attributes, and nodes in relational networks. We illustrate the different types of models on a variety of classification tasks and evaluate each under various conditions. The results demonstrate the flexibility and effectiveness of the temporal-relational framework for classification in time-evolving relational domains. Furthermore, the framework provides a foundation for automatically searching over temporal-relational representations to increase the accuracy of predictive models.

2.2 Dynamic Relational Classification Framework

Below we outline a dynamic relational classification framework (DRC) for prediction tasks in dynamic relational networks. Relational data is represented as an attributed graph $D = (G, \mathbf{X})$ where the graph $G = (V, E)$ represents a set of N nodes, such that $v_i \in V$ corresponds to node i and each edge $e_{ij} \in E$ corresponds to a link (e.g., email) between nodes i and j. The attribute set:

$$\mathbf{X} = \left(\begin{array}{l} \mathbf{X^V} = [X^1, X^2, ..., X^{m_v}], \\ \mathbf{X^E} = [X^{m_v+1}, X^{m_v+2}, ..., X^{m_v+m_e}] \end{array} \right)$$

contains m_v observed attributes on the nodes ($\mathbf{X^V}$) and m_e observed attributes on the edges ($\mathbf{X^E}$). Dynamic relational data evolves over time by the addition, deletion, and changing of nodes, edges, and attributes. Let $D_t = (G_t, \mathbf{X}_t)$ refer to the dataset at time t, where $G_t = (V, E_t)$ and $\mathbf{X}_t = (\mathbf{X}_t^V, \mathbf{X}_t^E)$. In our classification framework, we consider relational data observed over a range of timesteps $t = \{1, ..., T\}$ (e.g., citations over a period of years, emails over a period of days). Given this time-varying relational data, the task is to learn a model to predict either a static attribute Y or a dynamic attribute at a particular timestep Y_t, while exploiting both the relational and temporal dependencies in the data.

We define our temporal-relational classification framework with respect to a set of possible transformations of links, attributes, or nodes (as a function of time). The temporal weighting (e.g., exponential decay of past information) and temporal granularity (e.g., window of timesteps) of the links, attributes and nodes form the basis for any arbitrary transformation with respect to the temporal information (See Table 2.1). The discovered temporal-relational representation can be applied for mining temporal patterns, classification, and as a means for constructing temporal-ensembles. An overview of the temporal-relational representation discovery is provided below:

Table 2.1 provides an intuitive view of the possible temporal-relational representations. For instance, the TVRC model is a special case of the proposed framework

Table 2.1.: Temporal-relational representation.

1. For each Relational Component
   - Links, Attributes, or Nodes
2. Select the Temporal Granularity
   - ⋆ Timestep t_i
   - ⋆ Window $\{\mathbf{t_j}, \mathbf{t_{j+1}}, ..., \mathbf{t_i}\}$
   - ⋆ Union $T = \{t_0, ..., t_n\}$
3. Select the Temporal Influence
   - ⋆ Weighted
   - ⋆ Uniform

   Repeat steps 1-3 for each component.
4. Select the Relational Classifier
   - ⋆ Relational Bayes Classifier (RBC)
   - ⋆ Relational Probability Trees (RPT)

	Uniform			**Weighting**		
	Timestep	**Window**	**Union**	**Timestep**	**Window**	**Union**
Edges						
Attributes						
Nodes						

where the links, attributes, and nodes are unioned and the links are weighted. Below we provide more detail on steps 2-4.

2.2.1 Temporal Granularity

Traditionally, relational classifiers have attempted to use all the data available in a network [94]. However, since the relevance of data may change over time (e.g., links become stale), learning the *appropriate* temporal granularity (i.e., range of timesteps) can improve classification accuracy. We briefly define three general classes for varying the temporal granularity of the links, attributes, and nodes.

1. **Timestep.** The timestep models only use a single timestep t_i for learning.
2. **Window.** The window models use a sliding window of (multiple) timesteps $\{t_j, t_{j+1}, ..., t_i\}$ for learning. When the size of window is varied, the space of possible models in this category is by far the largest.

3. **Union.** The union model uses all previous temporal information for learning at time t_i, i.e., $T = \{0, ..., t_i\}$.

The timestep and union models are separated into distinct classes for clarity in evaluation and for understandability in pattern mining.

2.2.2 Temporal Influence: Links, Attributes, Nodes

We model the influence of relational components over time using temporal weighting. Specifically, when considering a temporal dataset $D_t = (G_t, \mathbf{X}_t)$, we will construct a weighted network $G_t = (V, E_t, W_t^E)$ and $\mathbf{X}_t = (\mathbf{X}_t^V, \mathbf{X}_t^E, W_t^X)$. Here W_t refers to a function that assigns weights on the edges and attributes that are used in the classifiers below.

Initially, we define $W_t^E(i,j) = 1$ if $e_{ij} \in E_t$ and 0 otherwise. Similarly, we define $W_t^X(x_i^m) = 1$ if $X_i^m = x_i^m \in \mathbf{X_t^m}$ and 0 otherwise. Then we consider two different approaches to revise these initial weights:
Weighting. These temporal weights can be viewed as probabilities that a relational component is still active at the current time step t, given that it was observed at time $(t-k)$. We investigated three temporal weighting functions:

Exponential Kernel. The exponential kernel weights the recent past highly and decays the weight rapidly as time passes [120]. The kernel function K_E for temporal data is defined as:

$$K_E(D_i; t, \theta) = (1-\theta)^{t-i}\theta W_i$$

Linear Kernel. The linear kernel decays more gradually and retains the historical information longer:

$$K_L(D_i; t, \theta) = \theta W_i(\frac{t_* - t_i + 1}{t_* - t_o + 1})$$

Inverse Linear Kernel. This kernel lies between the exponential and linear kernels when moderating historical information:

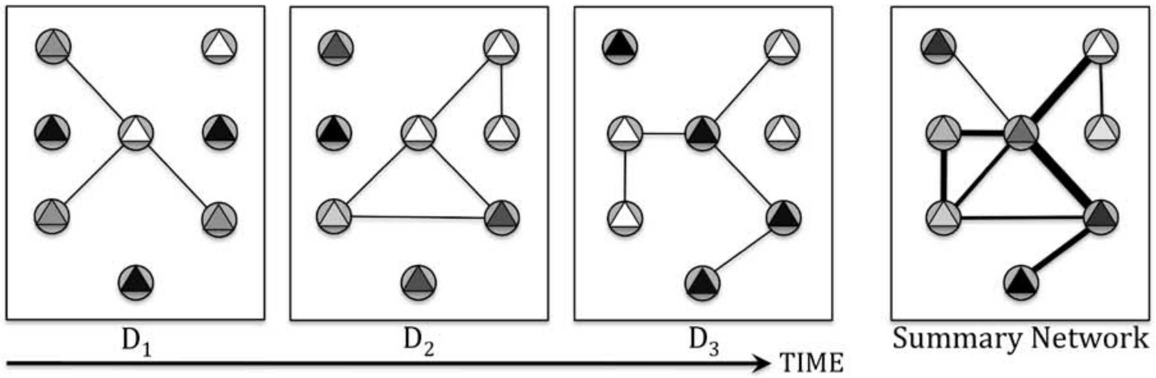
$$K_{IL}(D_i; t, \theta) = \theta W_i(\frac{1}{t_i - t_o + 1})$$

Uniform. These weights ignore the temporal influence of a relational component, and weight them uniformly over time, i.e., $W_t^E(i,j) = 1$ if $e_{ij} \in E_{t'} : t' \in T$ and 0 otherwise. A relational component can be assigned uniform weights within the selected temporal granularity or over the entire time window (e.g., traditional classifiers assign uniform weights, but they don't select the appropriate temporal granularity).
We note that different weighting functions can be chosen for different relational components (edges, attributes, nodes) with varying temporal granularities. For instance, the temporal influence of the links might be predicted using the exponential kernel while the attributes are uniformly weighted but have a different temporal granularity than the links.

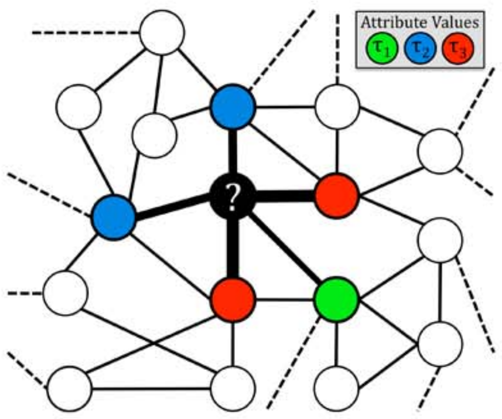
2.2.3 Temporal-Relational Classifiers

Once the temporal granularity and temporal weighting are selected for each relational component, then a temporal-relational classifier can learned. In this work, we use modified versions of the RBC [8] and RPT [121] to model the transformed temporal-relational representation. However, we note that any relational model that can be modified to incorporate node, link, and attribute weights is suitable for this phase. We extended RBCs and RPTs since they are interpretable, diverse, simple, and efficient. We use k-fold x-validation to learn the "best" model. Both classifiers are extended for learning and prediction over time.

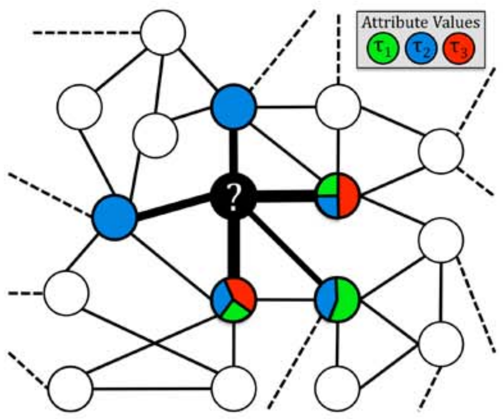
Weighted Relational Bayes Classifier. RBCs extend naive Bayes classifiers [122] to relational settings by treating heterogeneous relational subgraphs as a homogeneous set of attribute multisets. The weighted RBC uses standard maximum likelihood



(a) Graph and attribute weighting



(b) Incorporating link weights



(c) Using link and attribute weights

Fig. 2.1.: (a) Temporally weighting the attributes and links. (b) The feature calculation that includes only the temporal link weights. (c) The feature calculation that incorporates *both* the temporal attribute weights and the temporal link weights.

learning. More specifically, the sufficient statistics for each conditional probability distribution are computed as weighted sums of counts based on the link and attribute weights. More formally, for a class label C, attributes $\mathbf{X}$, and related items R, the RBC calculates the probability of C for an item i of type $G(i)$ as follows:

$$P(C^i|\mathbf{X}, R) \propto \prod_{X_m \in \mathbf{X}^{\mathbf{G(i)}}} P(X_m^i|C) \prod_{j \in R} \prod_{X_k \in \mathbf{X}^{\mathbf{G(j)}}} P(X_k^j|C)P(C)$$

Weighted Relational Probability Trees. RPTs extend standard probability estimation trees to a relational setting. We use the standard learning algorithm [121] except that the aggregate functions are computed after the appropriate links and attributes weights are included for the selected temporal granularity (shown in Figure 2.1). For prediction, if the model is applied to predict attribute Y_t at time t, we first calculate the weighted data D_t . Then the learned model from time $(t-1)$ is applied to D_t. The weighted classifier is appropriately augmented to incorporate the weights from D_t.

2.3 Temporal Ensemble Methods

Ensemble methods have traditionally been used to improve predictions by considering a weighted vote from a set of classifiers [123]. We propose temporal ensemble methods that exploit the *temporal dimension of relational data* to construct more accurate predictors. This is in contrast to traditional ensembles that do not explicitly use the temporal information. The *temporal-relational classification framework* and in particular the temporal-relational representations of the time-varying links, nodes, and attributes form the basis of the temporal ensembles (i.e., as a wrapper over the framework). The proposed temporal ensemble techniques are drawn from one of the five methodologies described below.

2.3.1 Transforming the Temporal Nodes and Links

The first method learns an ensemble of classifiers, where each of the classifiers are learned from, and then applied to, link and node sets that are sampled from each discrete timestep according to some probability. This sampling strategy is performed after selecting a temporal weighting and temporal granularity, and transforming the data to the appropriate temporal-relational representation. We note that the sampling probabilities for each timestep can be modified to bias the sampling toward the present or the past.

2.3.2 Sampling or Transforming the Temporal Feature Space

The second method transforms the temporal feature space by localizing randomization (for attributes at each timestep), weighting, or by varying the temporal granularity of the features, and then learning an ensemble of classifiers with different feature sets. Additionally, we might use only one temporal weighting function but learn models with different decay parameters or resample from the temporal features.

2.3.3 Adding Noise or Randomness

The third method is based on adding noise along the temporal dimension of the data, to increase generalization and performance. Specifically, we randomly permute the nodes feature values across the timesteps (i.e., a nodes recent behavior is observed in the past and vice versa) or links between nodes are permuted across time, and then learn an ensemble of models from several versions of the data.

2.3.4 Transforming the Time-Varying Class Labels

The fourth method introduces variance in the data by randomly permuting the previously learned labels at t-1 (or more distant) with the true labels at t, again learning an ensemble of models from several versions of the data.

2.3.5 Multiple Classification Algorithms and Weightings

The fifth method constructs and ensemble by randomly selecting from a set of classification algorithms (i.e., RPT, RBC, wvRN, RDN), while using the same temporal-relational representation, or by varying the representation with respect to the temporal weighting or granularity. Notably, an ensemble that uses both RPT and RBC models significantly increases accuracy, most likely due to the diversity of these temporal classifiers (i.e., correctly predicting different instances). Additionally, the temporal-classifiers might be assigned weights based on assessment of accuracy from cross-validation (or a Bayesian model selection approach).

2.4 Methodology and Data

This section describes the datasets and defines a few representative temporal-relational classifiers from the framework. For evaluating the framework, we use both static (Y is constant over time) and temporal prediction tasks (Y_t changes over time).

We considered two real world datasets for our experiments. The Cora database contains authorship and citation information about computer science research papers extracted automatically from the web. The Python Communication Network (PyComm) reflects characteristics of distributed team interaction, cooperation, communication, and social relationships. Table 2.2 lists the number of objects and/or links present in each dataset. We describe each dataset in more detail below.

2.4.1 Cora

Cora is a database of computer science research papers with the respective citation and author information. The relational schema is given in Figure 2.2. The attributes associated with each object are those supplied to the relational classification model

Table 2.2.: Datasets used for empirical evaluation.

PyComm	**CORA**
Developers: 185 Emails: 13181 Bugs Msgs: 69435 Teams: 18	Papers:16,153 References: 29,603 Authors: 21,976
Time Window: Feb2007-May2008	Time Window: 1981-1998

while the summary weights on the Paper-Citation and Author-Author links, namely ‘pc_weight’ and ‘aa_weight’, are generated by the graph summarization phase. The prediction task here was to predict the whether a paper is a machine learning paper given the topic of its references and the most prevalent topics its authors are working on through collaborations with other authors.

2.4.2 Python Communication Network

The Python communication network represents communications between distributed open-source software development teams. Effort is typically distributed due to geographic dispersion of developers, thus communication among developers is critical to the success of the project. Email is a common form of communication among the developers, and often mailing lists are used to ensure timely delivery of messages to all interested parties.

We collected and analyzed email and bug communication networks extracted from the open-source Python development environment (www.python.org). In Python development, the primary location for communication is the python-dev mailing list, which is publicly available for subscription or download. Here, nodes represent developers, and edges represents an email or bug communication between two developers. Note that this network has two unique types of edges representing different actions, namely, an email or bug communication. For instance, emails being more personal,

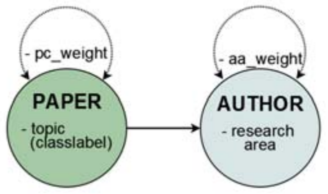


Fig. 2.2.: The relational schema for the Cora network. The link weights ‘pc_weight’ and ‘aa_weight’ are computed during graph summarization. The classlabel ‘topic’ is a binary classlabel showing whether the paper is a machine learning paper or not.

may indicate a closer relationship, whereas bug communications tend to be more formal, etc. The defect information (i.e., bugs) associated with the Python data enables us to derive measures of individual effectiveness that are consistent with the performance aspect of effectiveness measures. The prediction task is to predict whether a person has closed a bug given the people they have communicated with and various other attributes such as performance, teams, communication, centrality, and topics of communication. The dependent variable, Y_i, will measure the effectiveness of team member i. We then aim to model the influence of a number of independent variables on Y_i, including:

1. A set of intrinsic properties of the individual $X_i = \{X_{i1}, , X_{im}\}$.

2. The set of observed properties $\mathbf{X}_D$ of connected team members in both D_1 and D_2.

3. The dependent variable Y_D on team members in D_1, which is unobserved.

4. A set of variables C_ij that represent properties of communications between individuals (e.g., email frequency).

Thus our statistical models will have the following structure:

$$P(Y_i|D_1, D_2) = P(Y_i|\mathbf{X}_i, \mathbf{X}_j, Y_j, C_{ij})$$

where $j \in D_{i1} \wedge e_{ij} \in E$. The Python Communication Network consists of a collection of temporal snapshots. Let G $= \{G_1, G_2, ..., G_n\}$ be a sequence of temporal snapshots from the relational communication network. Every temporal snapshot corresponds to the events that occurred during the time period t, where $t = 1, 2, ..., n$. The size of the temporal snapshots are three month periods where $\{G_1 = (Feb07, Mar07, Apr07), G_2 = (May07, Jun07, Jul07), ..., G_7 = (Aug08, Sep08)\}$. The last temporal snapshot has only two months. The Python Communication dataset is a "pure" temporal network in the sense that the nodes, edges, class label, and most of the attributes are dynamically evolving. The class label 'has closed' represents if a developer has closed a bug in a specific time observation. Everything is evolving and therefore predictions about future events are more challenging. We believe this network where all the information is evolving at once pertains more to a real-world setting.

The features include team membership information, measures related to past and current performance, communication counts, topics of communications, and graph measures of centrality and clustering. Table 2.3 lists the details of the dataset. The four centrality measures evolve over time as the structure of the network changes. Betweenness centrality is a measurement of reachability in the network while eigenvector centrality is a measurement of connection strength between other developers.

Table 2.3.: Categories of attributes computed from the temporal python communication network

Python Communication Network Attributes		
Team Membership	Conv Tool	Build
	Demos & tools	Dist Utils
	Documentation	Doc Tools
	Installation	InterpCore
	Regular Expr	Tests
	Unicode	Windows
	Ctypes	Ext Modules
	Idle	LibraryLib
	Tkinter	XML
Performance	Assigned To	[Has Closed]
[T-1] Performance	Assigned To	Has Closed
Communication Attributes	All Comm.	Bug Comm.
	Email Comm.	
User Topics	AllTopic	EmailTopic
	BugTopic	
Centrality Attributes	Eigenvector	Cluster. Coeff.
	Betweenness	Degree
Link Attributes	EdgeCount	EdgeTopic
	EmailEdgeCount	EmailEdgeTopic
	BugEdgeCount	BugEdgeTopic
Temporal Snapshots	[Aug - Oct '07]	[Nov '07 - Jan '08]
	[Feb - April '08]	[May - July '08]
	[Aug - Sept '08]	
Predicting Individual Effectiveness on [Has Closed]		

Eigenvector centrality can be thought of as developers who communicate frequently with others whom communicate frequently (and so on) are more likely to have communications in the future and therefore given a higher weight than someone who communicates with people that barely communicate with anyone else. The clustering coefficient measures how closely connected a developer is to their neighbors.

In TVRC the relational communication attributes are moderated by the temporal edge and attribute strengths. The weighted communication attributes {ALL COMM., ASSIGNED TO, BUG COMM., EMAIL COMM.} are moderated by the link attributes All Edges, Bug Edges, Bug Edges, Email Edges respectively. The other weighted attributes are moderated by either All Edges or All Topic Edges.

2.4.3 Models

The space of temporal-relational models are evaluated using a representative sample of classifiers with varying temporal weightings and granularities. For every timestep t, we learn a model on D_t (i.e., some set of timesteps) and apply the model to D_{t+1}. The utility of the temporal-relational classifiers and representation are measured using the area under the ROC curve (AUC). Below, we briefly describe a few classes of models that were evaluated.

- **TENC**: The TENC models predict the temporal influence of both the links and attributes.
- **TVRC**: This model weights only the links using all previous timesteps.
- **Union Model**: The union model uses all links and nodes up to and including t for learning.
- **Window Model**: The window model uses the data D_{t-1} for prediction on D_t (unless otherwise specified).

We also compare simpler models such as the RPT (relational information only) and the DT (non-relational) that ignore any temporal information. Additionally, we explore many other models, including the class of window models, various weighting functions (besides exponential kernel), and built models that vary the set of windows in TENC and TVRC.

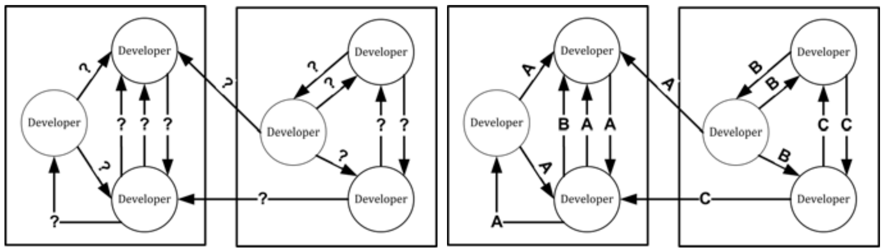
2.5 Unsupervised Dynamic Node Labeling

Given a time-series of graphs where for every edge we also have the textual data (e.g., of an email communication between two individuals), how can we leverage the textual data for dynamic node labeling to improve relational time-series prediction? To address this problem, we propose an unsupervised model called latent link semantics. Intuitively, this approach assigns each edge a label representing a general topic that best summarizes the textual content of the message. This allows us to represent each of the discoved edge labels as a feature, which are then used to discover more representative and discriminative features via a relational feature learning system. In other words, the learned edge labels representing topics are used to discover more discriminative relational topic features (based on the edge topic labels), which are then used to improve relational time-series classification.

Latent Link Semantics Many dynamic relational networks contain textual information in the form of messages or other personal information. How can we utilize this evolving information to improve relational time-series prediction?

Throughout the remainder of this section, the python developer network is used as a running example. Recall that nodes in this communication network are developers, and an edge represents either a bug or email communications between two developers.

We use techniques based on Latent Dirichlet Allocation [124] to extract topics of the communications. The latent topics are used to label the communication links between users. Let $\mathcal{T} = \{\tau_1,\ \tau_2,\ ...,\ \tau_\kappa\}$ be a set of topics extracted from the bugs and email communications. In the task of predicting effectiveness between teams and individuals we might find that $\tau_i = \{\textit{web programming}\}$ and $\tau_j = \{\textit{sports}\}$ therefore it is clear that teams and individuals communicating about sports should be penalized



(a) Before (b) After

Fig. 2.3.: Discovering the underlying latent semantics of the links. (a) Communication links with uniform semantics, and (b) latent link semantics: adding textual information through automatically labeling links in the network with the appropriate topic.

while communications about web programming might be a significant indicator of effectiveness. The topics provide context for the links instead of the uniform notion of a simple communication as shown below. We denote this technique of providing meaning for the links in a relational network as *Latent Link Semantics* (*LLS*).

The bug and email communications are from developers who work on distributed teams. Let $\mathcal{C} = \{c_1,\ c_2, ..., c_m\}$ be a set of bug and email communications and $\mathcal{W} = \{w_1,\ w_2, ..., w_n\}$ be a set of words from the communications between developers. We use the vector-space representation and define an $n \times m$ matrix denoted $\mathbf{M}$ where the coefficients represent the frequency that $w_i \in \mathcal{W}$ appears in $c_j \in \mathcal{C}$. The matrix is of size 191607 words $\times$ 82616 communications. Every communication is represented by a link between two people in the relational network. Using the text from the communication we assign the link an attribute that corresponds to the topic of communication. We introduce a latent class variable for the topics $\mathcal{T} = \{\tau_1,\ \tau_2,\ ...,\ \tau_\kappa\}$ where κ is the number of topics to be discovered. This can be thought of as a type of dimension reduction where the communications are projected into κ dimensions (or topics).

The number of topics to learn depends on many factors including the type of corpus (web content, open source development, ..) and also on the size of the collection. We chose $\kappa\ = 20$ as the majority of communications are most likely to focus on some aspect of the 18 projects under development. We removed a standard list of stopwords from the communications and also other technical words that appeared with high frequency (e.g., python). A topic can be viewed as a cluster of words that are frequently used together. We used a simple version of Latent Dirichlet Allocation [124] to model the κ topics. To estimate the parameters we used Expectation-Maximizatiom (EM) and for inference we used Gibbs sampling. Note that the model is learned from a number of past held-out data, and then used to estimate the communication topics at each timestep. That is, we construct a word-by-communication matrix for each graph in the time-series, estimate topics using the model, and then we construct a feature for each edge by assigning the topic that best fits the textual data for that email or bug communication (edge). As an aside, the initial model is recomputed if the topics no longer fit the textual data from recent communications.

We modeled the topics in three different sets of communications: (1) the email communication alone, (2) the bug communications alone, and (3) the joint set of email and bug communications. After extracting the latent topics from the email and bug communications we label the links of our relational communication network. The links are labeled by performing inference on a communication and assigning it to the topic with the largest likelihood. We also generate three object attributes {AllTopic, EmailTopic, BugTopic} where we assign a developer to the most prevalent topic in a particular set of communications. As an example if a developer most often discusses 'testing' in her bug messages for a particular timestep then she would be assigned to the BugTopic corresponding to testing.

In Table 2.4 we list the most likely words for five of the 20 topics when we combine bug and email communication together. The table contains words with both positive and negative connotation such as 'good' or 'doesnt' and also words referring to the network domain such as bugs or exception. An interesting direction to pursue would be to use sentiment analysis to automatically identify the communications with positive and negative tone and use those predictions in the analysis, to moderate relationships among developers. It is difficult to subjectively assess the high level meaning of a given latent topic from the communications. This is likely due to the fact that we are

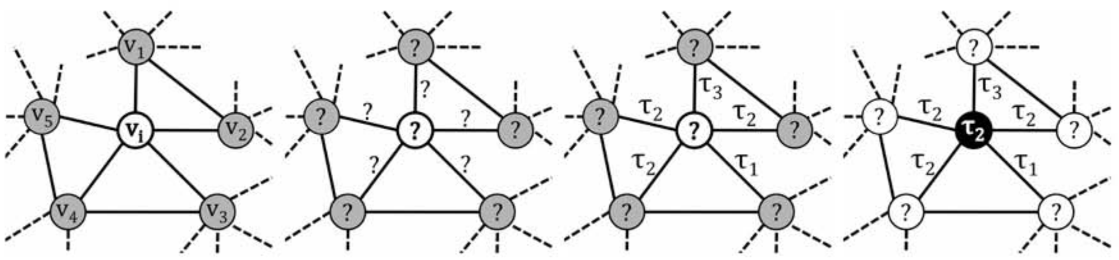


Fig. 2.4.: Defining a network using only the evolving textual information. The edges are labeled with their corresponding latent topic. **(a)** The initial network has only the textual information associated with the messages between developers. As an example, $C_{i1} = \{w_1, w_2, ..., w_z\}$. **(b)** All edges and topic attributes are initially semantically uniform. **(c)** Edges are labeled with their corresponding latent topic using maximum liklihood. **(d)** Nodal attributes are assigned a latent topic based on the mode of the neighboring latent topics.

analyzing communication in a highly specific technical domain. Nevertheless we can identify some patterns from the topics such as the first topic seems to be about open source development in a much broader sense. The word 'guido' appears significant, since that is likely a reference to the author of the Python programming language, Guido van Rossum.

Related work Recent work by Arora *et al.* [125] provides theoretical analysis for a few well-known topic modeling methods, as well a practical topic model with provable guarantees, see [126]. Dynamic topic models have also been proposed, see [127–131]. Our approach is fundamentally different from that work. In particular, we focus on the problem of learning dynamic labels from a relational time-series of graphs and textual edge information. Another key difference is that topics are represented as edge features and used for discovering more discriminatory features via relational feature learning. Finally, the initial features and the discovered relational topic features are then used for improving relational time-series classification.

2.6 Experiments

In this section, we demonstrate the effectiveness of the temporal-relational framework and temporal ensemble methods on two real-world datasets. The main findings are summarized below:

- ▷ Temporal-relational models significantly outperform relational and non-relational models.
- ▷ The classes of temporal-relational models each have advantages and disadvantages in terms of accuracy, efficiency, and interpretability. Models based strictly on temporal granularity are more interpretable but less accurate than models that *learn* the temporal influence. The more complex models that combine both are generally more accurate, but less efficient.
- ▷ *Temporal ensemble methods* significantly outperform non-relational and relational ensembles. In addition, the temporal ensembles are an efficient and accurate alternative to searching over the space of temporal models.

2.6.1 Single Models

We evaluate the temporal-relational framework using single-models and show that in all cases the performance of classification improves when the temporal dynamics are appropriately modeled.

Temporal, Relational, and Non-Relational Information. The utility of the temporal (TVRC), relational (RPT), and non-relational information (decision tree; DT) is assessed using the most primitive models. Figure 2.5 compares TVRC with the RPT and DT models that use more features but ignore the temporal dynamics of the data. We find the TVRC to be the simplest temporal-relational classifier that still outperforms the others. Interestingly, the discovered topic features are the only additional features that improve performance of the DT model. This is significant as

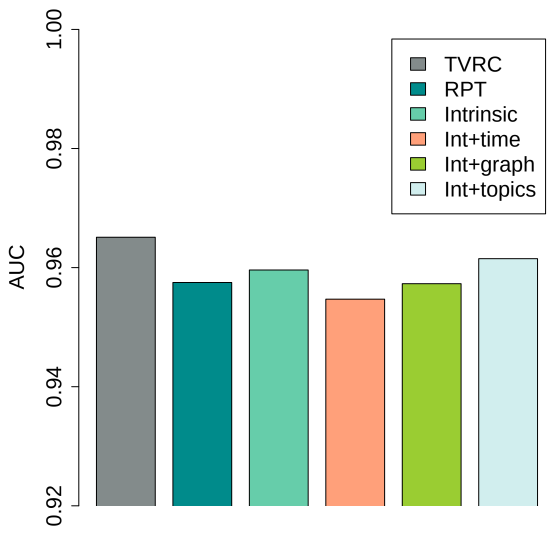


Fig. 2.5.: Comparing a primitive *temporal* model (TVRC) to competing relational (RPT), and non-relational (DT) models.

these attributes are discovered by dynamically modeling the topics, but are included in the DT model as simple non-relational features (i.e., no temporal weighting or granularity).

Exploring Temporal-Relational Models. We focus on exploring a representative set of temporal-relational models from the proposed framework. To more appropriately evaluate the models, we remove highly correlated attributes (i.e., that are not necessarily temporal patterns, or motifs), such as “assignedto” in the PyComm prediction task. In Figure 2.6, we find that TENC outperforms the other models over all timesteps. This class of models are significantly more complex than TVRC since the temporal influence of both links and attributes are learned.

We then explored learning the appropriate temporal granularity. Figure 2.6 shows the results from two models in the TVRC class where we tease apart the superiority of TENC (i.e., weighting or granularity). However, both TVRC models outperform one another on different timesteps, indicating the necessity for a more precise temporal-representation that optimizes the temporal granularity by selecting the appropriate

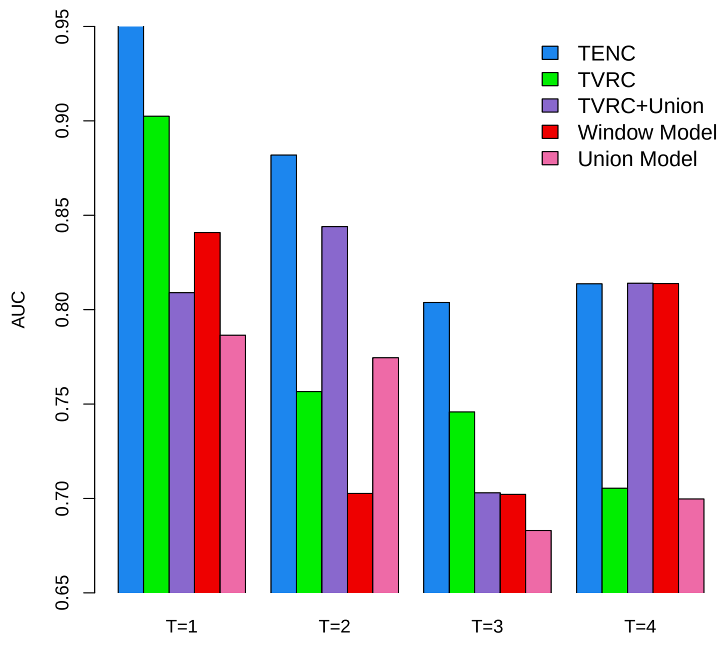
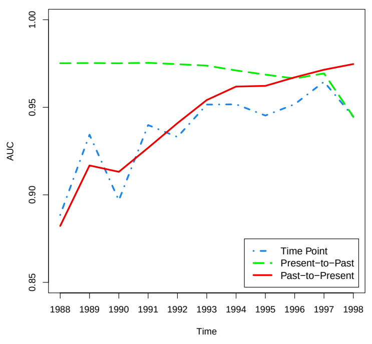


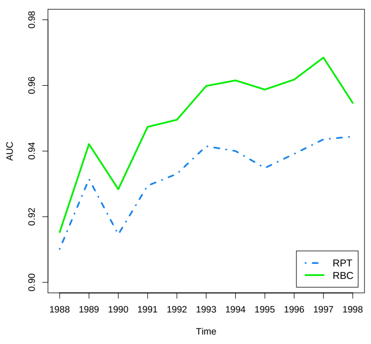
Fig. 2.6.: Exploring the space of temporal relational models. Significantly different temporal-relational representations from the proposed framework are evaluated.

decay parameters for links and attributes (i.e., TENC). Similar results were found using Cora and other base classifiers such as RBC. Models based strictly on varying the *temporal granularity* were also explored. More details can be found in [132].

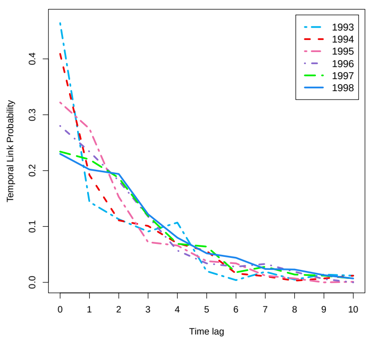
Selective Temporal Learning. We also explored "selective temporal learning" that uses multiple temporal weighting functions (and temporal granularities) for the links and attributes. The motivation for such an approach is that the influence of each temporal component should be modeled independently, since any two attributes (or links) are likely to decay at different rates. However, the complexity and the utility of the learned temporal-relational representation depends on the ability of the selective learner to select the best temporal features (derived from weighting or varying the temporal granularity of attributes and links) without overfitting or causing other problems. We found that the selective temporal learning performs best for simpler



(a) AI (RBC)



(b) Stability (AI)



(c) Link Probability (ML)

Fig. 2.7.: Exploring temporal granularity models (RPT and RBC for ML and AI tasks).

prediction tasks, however, it still frequently outperforms classifiers that ignore the temporal information. Experiments based strictly on varying the temporal granularity were also explored and can be found in [132].

Models of Temporal Granularity. In these experiments, we restrict our focus to models based *strictly* on varying the temporal granularity. In this space, there are a range of interesting models that provide insights into the temporal patterns, structure, and nature of the dataset. We first introduce three classes of models based on varying the temporal granularity and then evaluate their utility.

- **Past-to-Present.** These models consider linked nodes from the distant past and successively increases the window to include more recent information.
- **Present-to-Past.** We consider only the most recent links, nodes, and attributes and successively increase the window to include more of the past.
- **Temporal Point.** Only links, nodes, and attributes at time k are considered.

Intuitively, Figure 2.7(a) shows AUC increasing as a function of the more recent attributes and links (i.e., Past-To-Present model). Conversely, if we consider only the most recent temporal information and successively include more of the past then the AUC initially increases to a local max and then drops before increasing as additional past information is modeled. This drop in accuracy indicates a type of temporal-transition in the link structure and attributes. Overfitting may justify the slight improvement in AUC as noisey past information is added. The noise reduces bias in training and consequently increases the models ability to generalize for predicting instances in the future. More interestingly, the class of TEMPORAL-POINT models allow us to more accurately determine if past actions at some previous timestep are predictive of the future and how these behaviors transition over time. These patterns are shown in Fig. 2.7.

Temporal Stability of Relational Classifiers. In the next experiment, we use the models to compare more accurately the stability of the temporal RBC and RPT classifiers. We find the RBC to be relatively stable over time whereas RPT has significantly more variance. In particular, for the ML prediction task, the structure of the RPT trees were considerably different over the timesteps whereas they gradually evolve in the AI prediction task shown in Fig. 2.7(b).

Finally, for the papers in each time period (from CORA), the probability of citing a paper given the time-lag ℓ is shown in Fig. 2.7(c). Interestingly, the link probabilities at $\ell = 3$ for each prediction-time approximately begin to converge. Indicating a global pattern w.r.t. past links that is independent of the core-nodes initial time period. Hence, the more recent behavior of the core-nodes is significantly different than their past behavior.

2.6.2 Temporal Ensemble Models

Instead of directly learning the optimal temporal-relational representation to increase the accuracy of classification, we use *temporal ensembles* by varying the relational representation with respect to the temporal information. These ensemble

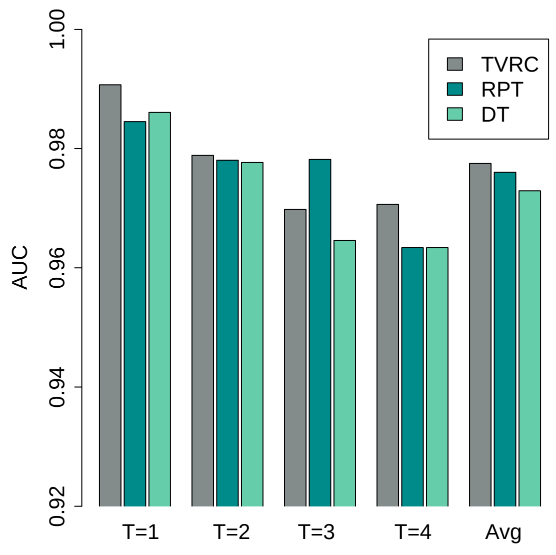


Fig. 2.8.: Comparing temporal, relational, and traditional ensembles

models reduce error due to variance and allow us to assess which features are most relevant to the domain with respect to the relational or temporal information.

Temporal, Relational, and Traditional Ensembles. We first resampled the instances (nodes, links, features) repeatedly and then learn TVRC, RPT, and DT models. Across almost all the timesteps, we find the temporal-ensemble that uses various temporal-relational representations outperforms the relational-ensemble and the traditional ensemble (see Figure 2.8). The temporal-ensemble outperforms the others even when the minimum amount of temporal information is used (e.g., time-varying links). More sophisticated temporal-ensembles can be constructed to further increase accuracy. We have investigated ensembles that use significantly different temporal-relational representations (i.e., from a wide range of model classes) and ensembles that use various temporal weighting parameters. In all cases, these ensembles are more robust and increase the accuracy over more traditional ensemble techniques (and single classifiers). Further, the average improvement of the temporal-ensembles is significant at $p < 0.05$ with a 16% reduction in error, justifying the proposed temporal ensemble methodologies.

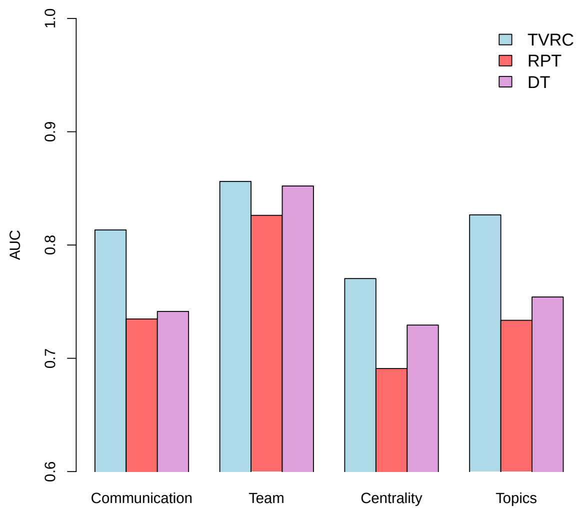


Fig. 2.9.: Comparing attribute classes w.r.t. temporal, relational, and traditional ensembles.

In the next experiment, we construct ensembles using the feature classes. We use the primitive models (with the transformed feature space) in order to investigate (more accurately) the most significant feature class (communication, team, centrality, topics) and also to identify the minimum amount of temporal information required to outperform relational ensembles.

In Figure 2.9, we find several striking temporal patterns. First, the team features are localized in time and are not changing frequently. For instance, it is unlikely that a developer changes their assigned teams and therefore modeling the temporal dynamics only increases accuracy by a relatively small percent. However, the *temporal-ensemble* is still more accurate than traditional ensemble methods that ignore temporal patterns. This indicates the robustness of the temporal-relational representations. More importantly, the other classes of attributes are evolving considerably and this fact is captured by the significant improvement of the temporal ensemble models. Similar performance is also obtained by varying the temporal granularity (see previous examples).

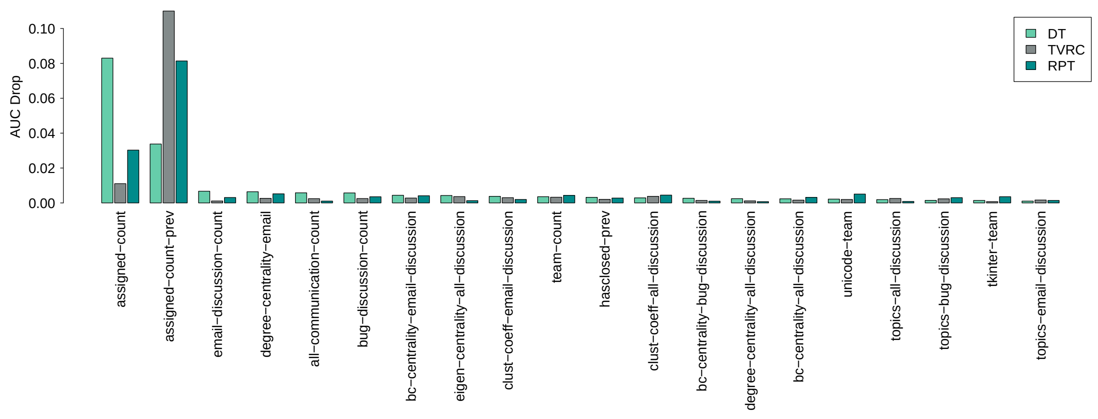


Fig. 2.10.: Randomization. The significant attributes used in the *temporal ensemble* are compared to the relational and traditional ensembles. The change in AUC is measured.

Randomization. We use randomization to identify the significant attributes in the *temporal-ensemble models*. Randomization provides a means to rank and eliminate redundant attributes (i.e., two attributes may share the same significant temporal pattern). We randomize each attribute in each timestep and measure the change in AUC. The results are shown in Figure 2.10.

Randomization is performed on an attribute by randomly reordering the values, thereby preserving the distribution of values but destroying any association of the attribute with the class label. For every attribute, in every time step, we randomize the given attribute, apply the ensemble method, and measure the drop in AUC due to that attribute. The resulting changes in AUC are used to assess and rank the attributes in terms of their impact on the temporal ensemble (and how it compares to more standard relational or traditional ensembles).

We find that the basic traditional ensemble relies on “assignedto” (in the current time step) while the temporal ensemble (and even less for the relational ensemble) relies on the previous “assignedto” attributes. This indicates that relational information in the past is more useful than intrinsic information in the present—which points to an interesting hypothesis that a colleagues behavior (and interactions) precedes

Table 2.4.: Latent topics learned from the evolving textual information and the most significant words of each topic

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
dev	logged	gt	code	test
wrote	patch	file	object	lib
guido	issue	lt	class	view
import	bugs	line	case	svn
code	bug	os	method	trunk
pep	problem	import	type	rev
mail	fix	print	list	modules
release	fixed	call	set	build
tests	days	read	objects	amp
work	created	socket	change	error
people	time	path	imple	usr
make	docu	data	functions	include
pm	module	error	argument	home
ve	docs	open	dict	file
support	added	windows	add	run
module	check	problem	def	main
things	doc	traceback	methods	local
good	doesnt	mailto	exception	src
van	report	recent	ms	directory

their own behavior. Organizations might use this to predict future behavior with less information and proactively respond more quickly. Additionally, the topic attributes are shown to be the most useful for the temporal ensembles (Fig. 2.11), indicating the utility of using topics to understand the context and strength of relationships.

2.7 Related Work

Recent work has started to model network dynamics in order to better predict link and structure formation over time [78,79], but this work focuses on unattributed graphs. Previous work in relational learning on attributed graphs either uses static network snapshots or significantly limits the amount of temporal information incorporated into the models. Sharan *et al.* [94] assumes a strict representation that only uses kernel estimation for link weights, while GA-TVRC [95] uses a genetic algorithm to learn the link weights. SRPTs [96] incorporate temporal and spatial information in the relational attributes. However, the above approaches focus only on one specific temporal pattern and do not consider different temporal granularities. In contrast, we

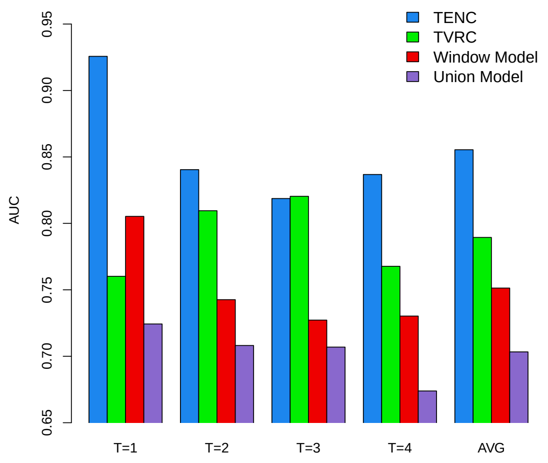


Fig. 2.11.: Evaluation of relational time-series classifiers using only the latent topics of the communications to predict effectiveness. LDA is used to automatically discover the latent topics as well as annotating the communication links and individuals with their appropriate topic in the temporal networks.

explore a larger space of temporal-relational representations in a flexible framework that can capture temporal dependencies over *links*, *attributes*, and *nodes*.

McGovern et al. [96] recently proposed Spatio-Temporal Relational Probability Trees (SRPTs) as an extension to Relational Probability Trees (RPTs) [121] to incorporate temporal variations in relational attributes and links. SRPTs are designed for relational domains where concepts vary based on small spatial and temporal scales within the data (e.g., weather prediction). They model the temporal relational information by expanding the set of features explored in the learning algorithm. For example, the feature set includes *Temporal Exists*, which assesses whether an object or a link lasted at least t time steps, and *Relative Count*, which splits data on the relative change in the number of matching items (count) within a time window. Thus, SRPTs are able to model some aspects of time-varying link structure (i.e., local degree changes) as well as time-varying attributes in relational data. Although adding temporal components to the feature space of relational decision trees provides the flexibility to model temporal variations in both attributes and relationships, it results in an exponential number of possible features which may be infeasible to explore for large datasets. McGovern et al. makes this tractable by defining a restricted set of temporal relational features manually, based on domain knowledge specific to weather prediction tasks.

To the best of our knowledge, we are the first to propose and investigate *temporal-relational ensemble methods* for time-varying relational classification. However, there has been recent work on relational ensemble methods [97–99] and non-relational ensemble methods for evolving streams [100]. Preisach *et al.* [97] use voting and stacking methods to combine relational data with multiple relations. In contrast, Eldardiry and Neville [99] incorporates prediction averaging in the collective inference process to reduce both learning and inference variance.

2.8 Summary

We proposed and validated a framework for temporal-relational classifiers, ensembles, and more generally, representations for temporal-relational data. We evaluated an illustrative set of temporal-relational models from the proposed framework. Empirical results show that the models significantly outperform competing classification models that use either no temporal information or a very limited amount. The proposed temporal ensemble methods (i.e., temporally sampling, randomizing, and transforming features) were shown to significantly outperform traditional and relational ensembles. Furthermore, the temporal-ensemble methods were shown to increase the accuracy over traditional models while providing an efficient alternative to exploring the space of temporal-models. The results demonstrated the effectiveness, scalability, and flexibility of the temporal-relational representations for classification and ensembles in time-evolving domains. In future work, we will theoretically analyze the framework and the proposed ensemble methods.

3. DYNAMIC NODE WEIGHTING

We propose a dynamical system that captures changes to the network centrality of nodes as external interest in those nodes vary. We derive this system by adding time-dependent teleportation to the PageRank score. The result is not a single set of importance scores, but rather a time-dependent set. These can be converted into ranked lists in a variety of ways, for instance, by taking the largest change in the importance score. For an interesting class of the dynamic teleportation functions, we derive closed form solutions for the dynamic PageRank vector. The magnitude of the deviation from a static PageRank vector is given by a PageRank problem with complex-valued teleportation parameters. Moreover, these dynamical systems are easy to evaluate. We demonstrate the utility of dynamic teleportation on both the article graph of Wikipedia, where the external interest information is given by the number of hourly visitors to each page, and the Twitter social network, where external interest is the number of tweets per month. For these problems, we show that using information from the dynamical system helps improve a prediction task and identify trends in the data.

3.1 Motivation

The PageRank vector of a directed graph is the stationary distribution of a Markovian *random surfer*. At a node, the random surfer either

1. transitions to a new node uniformly chosen from the set of out-edges, or
2. does something else (e.g. leaves the graph and then randomly returns) [133, 134].

The probability that the surfer performs the first action is known as the damping parameter in PageRank, denoted α. The second action is called teleporting and is modeled by the surfer picking a node at random according to a distribution called the teleportation distribution vector or personalization vector. This PageRank Markov chain always has a unique stationary distribution for any $0 \leq \alpha < 1$. In this work, we focus on the teleportation distribution vector $\mathbf{v}$ and study how changing teleportation behavior manifests itself in a dynamical system formulation of PageRank.

To proceed further, we need to formalize the PageRank model. Let $\mathbf{A}$ be the adjacency matrix for a graph where $A_{i,j}$ denotes an edge from node i to node j. To avoid a proliferation of transposes, we define $\mathbf{P}$ as the transposed transition matrix for a random-walk on a graph:

$$P_{j,i} = \text{ probability of transitioning from node } i \text{ to node } j.$$

Hence, the matrix $\mathbf{P}$ is *column-stochastic* rather than the more common row-stochastic matrices found in probability theory. Throughout this manuscript, we utilize uniform random-walks on a graph, in which case $\mathbf{P} = \mathbf{A}^T\mathbf{D}^{-1}$ where $\mathbf{D}$ is a diagonal matrix with the out-degree of each node on the diagonal. However, none of the theory is restricted to this type of random walk and any column-stochastic matrix will do. If any nodes have no out-links, we assume that they are adjusted in one of the standard ways [135]. Let $\mathbf{v}$ be a teleportation distribution vector such that $v_i \geq 0$ and $\sum_i v_i = 1$. This vector models where the surfer will transition when "doing something else." The PageRank Markov chain then has the transition matrix:

$$\alpha\mathbf{P} + (1 - \alpha)\mathbf{v}\mathbf{e}^T.$$

While finding the stationary distribution of a Markov chain usually involves computing an eigenvector or solving a singular linear system, the PageRank chain has a particularly simple form for the stationary distribution vector $\mathbf{x}$:

$$(\mathbf{I} - \alpha\mathbf{P})\mathbf{x} = (1 - \alpha)\mathbf{v}.$$

Sensitivity of PageRank with respect to $\mathbf{v}$ is fairly well understood. [134] devote a section to determining the Jacobian of the PageRank vector with respect to $\mathbf{v}$. The choice of $\mathbf{v}$ is often best guided by an application specific measure. By setting $\mathbf{v} = \mathbf{e}_i$, that is, the ith canonical basis vector:

$$\mathbf{e}_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \\ \\ \\ i\text{th row,} \\ \\ \\ \\ \end{matrix}$$

PageRank computes a highly localized diffusion that is known to produce empirically meaningful clusters and theoretically supported clusters [136, 137]. By choosing $\mathbf{v}$ based on a set of known-to-be-interesting nodes, PageRank will compute an expanded set of interesting nodes [138, 139]. Yet, in all of these cases, $\mathbf{v}$ is chosen once for the graph application or particular problem.

In the original motivation of PageRank [133], the distribution $\mathbf{v}$ should model how users behave on the web when they don't click a link. When this intuition is applied to a site like Wikipedia, this suggests that the teleportation function should vary as particular topics become interesting. For instance, in our experiments (section 3.4), we examine the number of page views for each Wikipedia article during a period where a major earthquake occurred. Suddenly, page views to earthquake spike – presumably as people are searching for that phrase. We wish to *include* this behavior into our PageRank model to understand what is now important in light of a radically different behavior. One option would be to recompute a new PageRank vector given the observed teleporting behavior at the current time. Our proposal for a dynamical system is another alternative. That is, we define a new model where teleportation is the time-dependent function:

$$\mathbf{v}(t).$$

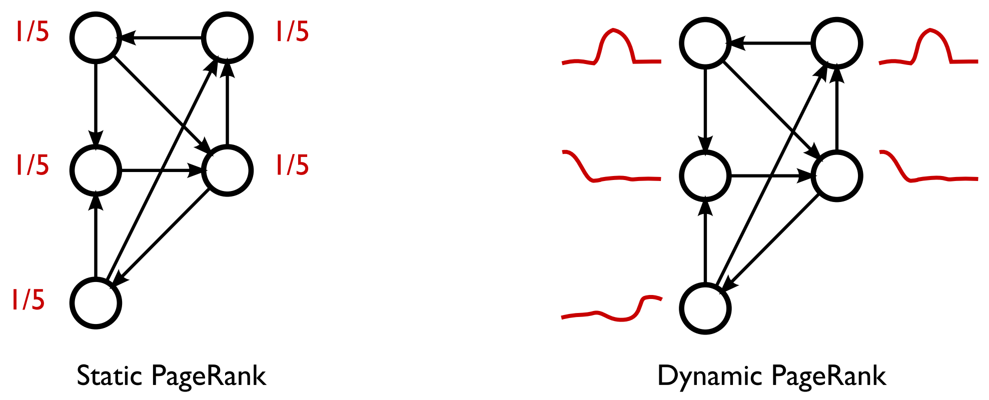


Fig. 3.1.: On the left is PageRank with static teleportation. At each step, the teleportation is to each node with uniform probability 1/5. At right, we have the PageRank model with dynamic teleportation. In this case, the teleportation distribution (illustrated in red) *changes with time*. Thus, the upper nodes are teleported to more during the middle time regime. In both cases, the graph is fixed. We study the effect of such dynamic teleportation on the PageRank scores.

At each time t, $\mathbf{v}(t)$ is a probability distribution of where the random walk teleports. Figure 3.1 illustrates this model. We return to a comparison between this approach and solving PageRank systems in section 3.6.

The dynamical system we propose is a generalization of PageRank in the sense that if $\mathbf{v}(t)$ is a constant function in time, then we converge to the standard PageRank vector (theorem 1). Additionally, we can analyze the dynamical PageRank function for some simple oscillatory teleportation functions $\mathbf{v}(t)$. Bounding the deviation of these oscillatory PageRank values from the static PageRank vector involves solving a *PageRank problem with complex teleportation* [140, 141]. This result is, perhaps, the first non-analytical use of PageRank with a complex teleportation parameter.

In our new dynamical system, we do not compute a single ranking vector as others have done with time-dependent rankings [142], rather we compute a time-dependent ranking function $\mathbf{x}(t)$, the dynamic PageRank vector at time t, from which we can extract different static rankings (section 3.2.4). There are two complications with using empirically measured data. First, we must choose a time-scale for our ODE based on the period of our page-view data (section 3.2.5). Put a bit informally, we must pick the time-unit for our ODE – it is not dimensionless. We analytically show that some choices of the time-scale amount to solving the PageRank system for each change in the teleportation vector. Second, we also investigate smoothing the measured page view data (section 3.2.6). To compute this dependent ranking function $\mathbf{x}(t)$ we discuss ordinary differential equation (ODE) integrators in section 3.3.

We discuss the impact of these choices on two problems: page views from Wikipedia and a network from Twitter. We also investigate how the rankings extracted from our methods differ from those extracted by other static ranking measurements. We can use these rankings for a few interesting applications. Adding the dynamic PageRank scores to a prediction task decreases the average error (section 3.4.4) for Twitter. Clustering the dynamic PageRank scores yields many of the standard time-series features in social networks (section 3.5.1). Finally, using Granger causality testing on the dynamic PageRank scores helps us find a set of interesting links in the graph (section 3.5.2).

We make our code and data available in the spirit of reproducible research:

`http://www.ryanrossi.com/dynamic_pagerank`

3.2 PageRank with Time-dependent Teleportation

We begin our discussion by summarizing the notation introduced thus far in Table 3.1.

In order to incorporate changes in the teleportation into a new model for PageRank, we begin by reformulating the standard PageRank algorithm in terms of changes to the PageRank values for each page. This step allows us to state PageRank as a dynamical system, in which case we can easily incorporate changes into the vector.

The standard PageRank algorithm is the power method for the PageRank Markov chain [134]. After simplifying this iteration by assuming that $\mathbf{e}^T\mathbf{x} = 1$, it becomes:

$$\mathbf{x}^{(k+1)} = \alpha\mathbf{P}\mathbf{x}^{(k)} + (1-\alpha)\mathbf{v}.$$

Table 3.1.: Summary of notation. Matrices are bold, upright roman letters; vectors are bold, lowercase roman letters; and scalars are unbolded roman or greek letters. Indexed elements are vectors if they are bolded, or scalars if unbolded.

$\imath$	the imaginary number
n	number of nodes in a graph
$\mathbf{e}$	the vector of all ones
$\mathbf{P}$	column stochastic matrix
α	damping parameter in PageRank
$\mathbf{v}$	teleportation distribution vector
$\mathbf{x}$	solution to the PageRank computation: $(\mathbf{I} - \alpha\mathbf{P})\mathbf{x} = (1-\alpha)\mathbf{v}$
$\mathbf{x}(t)$	solution to the dynamic PageRank computation for time t
$\mathbf{v}(t)$	a teleportation distribution vector at time t
$\mathbf{c}$	the cumulative rank function
$\mathbf{r}$	the variance rank function
$\mathbf{d}$	the difference rank function
$\mathbf{v}_k$	the teleportation distribution for the kth observed page-views vector
θ	decay parameter for time-series smoothing
s	the time-scale of the dynamical system
$t_{\max}$	the last time of the dynamical system

In fact, this iteration is equivalent to the Richardson iteration for the PageRank linear system $(\mathbf{I} - \alpha\mathbf{P})\mathbf{x} = (1-\alpha)\mathbf{v}$. This fact is relevant because the Richardson iteration is usually defined:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(x)} + \omega\left[(1-\alpha)\mathbf{v} - (\mathbf{I} - \alpha\mathbf{P})\mathbf{x}^{(k)}\right].$$

For $\omega = 1$, we have:

$$\Delta\mathbf{x}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} = \alpha\mathbf{P}\mathbf{x}^{(k)} + (1-\alpha)\mathbf{v} - \mathbf{x}^{(k)} = (1-\alpha)\mathbf{v} - (\mathbf{I} - \alpha\mathbf{P})\mathbf{x}^{(k)}.$$

Thus, changes in the PageRank values at a node *evolve* based on the increment $(1-\alpha)\mathbf{v} - (\mathbf{I} - \alpha\mathbf{P})\mathbf{x}^{(k)}$. We reinterpret this update as a continuous time dynamical system:

$$\mathbf{x}'(t) = (1-\alpha)\mathbf{v} - (\mathbf{I} - \alpha\mathbf{P})\mathbf{x}(t). \tag{3.1}$$

To define the PageRank problem with time-dependent teleportation, we make $\mathbf{v}(t)$ a function of time.

Definition 1: *The dynamic PageRank model with time-dependent teleportation is the solution of*

$$\mathbf{x}'(t) = (1-\alpha)\mathbf{v}(t) - (\mathbf{I} - \alpha\mathbf{P})\mathbf{x}(t) \tag{3.2}$$

where $\mathbf{x}(0)$ is a probability distribution vector and $\mathbf{v}(t)$ is a probability distribution vector for all t.

In the dynamic PageRank model, the PageRank values $\mathbf{x}(t)$ may not "settle" or converge to some fixed vector $\mathbf{x}$. We see this as a feature of the new model as we plan to utilize information from the evolution and changes in the PageRank values. For instance, in section 3.2.4, we discuss various functions of $\mathbf{x}(t)$ that define a rank. Next, we state the solution of the problem.

Lemma 1: *The solution of the dynamical system:*

$$\mathbf{x}'(t) = (1-\alpha)\mathbf{v}(t) - (\mathbf{I} - \alpha\mathbf{P})\mathbf{x}(t)$$

is

$$\mathbf{x}(t) = \exp[-(\mathbf{I}-\alpha\mathbf{P})t]\mathbf{x}(0) + (1-\alpha)\int_0^t \exp[-(\mathbf{I}-\alpha\mathbf{P})(t-\tau)]\mathbf{v}(\tau)\,d\tau.$$

This result is found in standard texts on dynamical systems, for example [143].

Given this solution, let us quickly verify a few properties of this system:

Lemma 2: *The solution of a dynamical PageRank system $\mathbf{x}(t)$ is a probability distribution ($\mathbf{x}(t) \geq 0$ and $\mathbf{e}^T\mathbf{x}(t) = 1$) for all t.*

Proof. The model requires that $\mathbf{x}(0)$ is a probability distribution. Thus, $\mathbf{x}(0) \geq 0$ and $\mathbf{e}^T\mathbf{x}(0) = 1$. Assuming that the sum of $\mathbf{x}(t)$ is 1, then the sum of the derivative $\mathbf{x}'(t)$ is 0 as a quick calculation shows. The closed form solution above is also nonnegative because the matrix $\exp[-(\mathbf{I}-\alpha\mathbf{P})] = \exp(\alpha\mathbf{P})\exp(-1) \geq 0$ and both $\mathbf{x}(0)$ and $\mathbf{v}(t)$

are non-negative for all t. (This property is known as exponential non-negativity and it is another property of M-matrices such as $\mathbf{I} - \alpha\mathbf{P}$ [143].) $\square$

3.2.1 A Generalization of PageRank

This closed form solution can be used to solve a version of the dynamic problem that reduces to the PageRank problem with static teleportation. If $\mathbf{v}(t) = \mathbf{v}$ is constant with respect to time, then

$$\int_0^t \exp[-(\mathbf{I} - \alpha\mathbf{P})(t-\tau)]\mathbf{v}(\tau)\, d\tau = (\mathbf{I} - \alpha\mathbf{P})^{-1}\mathbf{v} - \exp[-(\mathbf{I} - \alpha\mathbf{P})t](\mathbf{I} - \alpha\mathbf{P})^{-1}\mathbf{v}.$$

Hence, for constant $\mathbf{v}(t)$:

$$\mathbf{x}(t) = \exp[-(\mathbf{I} - \alpha\mathbf{P})t](\mathbf{x}(0) - \mathbf{x}) + \mathbf{x},$$

where $\mathbf{x}$ is the solution to static PageRank: $(\mathbf{I} - \alpha\mathbf{P})\mathbf{x} = (1-\alpha)\mathbf{v}$. Because all the eigenvalues of $-(\mathbf{I} - \alpha\mathbf{P})$ are less than 0, the matrix exponential terms disappear in a sufficiently long time horizon. Thus, when $\mathbf{v}(t) = \mathbf{v}$, nothing has changed. We recover the original PageRank vector $\mathbf{x}$ as the steady-state solution:

$$\lim_{t\to\infty} \mathbf{x}(t) = \mathbf{x} \text{ the PageRank vector.}$$

This derivation shows that dynamic teleportation PageRank is a generalization of the PageRank vector. We summarize this discussion as:

Theorem 1: *PageRank with time-dependent teleportation is a generalization of PageRank. If $\mathbf{v}(t) = \mathbf{v}$, then the solution of the ordinary differential equation:*

$$\mathbf{x}'(t) = (1-\alpha)\mathbf{v} - (\mathbf{I} - \alpha\mathbf{P})\mathbf{x}(t)$$

converges to the PageRank vector

$$(\mathbf{I} - \alpha\mathbf{P})\mathbf{x} = (1-\alpha)\mathbf{v}$$

as $t \to \infty$.

3.2.2 Choosing the Initial Condition

There are three natural choices for the initial condition $\mathbf{x}(0)$. The first choice is the uniform vector $\mathbf{x}(0) = \frac{1}{n}\mathbf{e}$. The second choice is the initial teleportation vector $\mathbf{x}(0) = \mathbf{v}(0)$. And the third choice is the solution of the PageRank problem for the initial teleportation vector $(\mathbf{I} - \alpha\mathbf{P})\mathbf{x}(0) = (1-\alpha)\mathbf{v}(0)$. We recommend either of the latter two choices in order to generalize the properties of PageRank. Note that if $\mathbf{x}(0)$ is chosen to solve the PageRank system for $\mathbf{v}(0)$, then $\mathbf{x}(t) = \mathbf{x}$ for all t is the solution of the PageRank dynamical system with constant teleportation (theorem 1).

3.2.3 PageRank with Fluctuating Interest

One of the advantages of the PageRank dynamical system is that we can study problems analytically. We now do so with the following teleportation function, or forcing function as it would be called in the dynamical systems literature:

$$\mathbf{v}(t) = \frac{1}{k}\sum_{j=1}^{k} \mathbf{v}_j \Big(\cos(t + (j-1)\tfrac{2\pi}{k}) + 1\Big),$$

where $\mathbf{v}_j$ is a teleportation vector. Here, the idea is that $\mathbf{v}_j$ represents the propensity of people to visit certain nodes at different times. To be concrete, we might have $\mathbf{v}_1$ correspond to news websites that are visited more frequently during the morning, $\mathbf{v}_2$ correspond to websites visited at work, and $\mathbf{v}_3$ correspond to websites visited during the evening. This function has all the required properties that we need to be a valid teleportation function. With the risk of being overly formal, we'll state these as a lemma.

Lemma 3: *Let $k \geq 2$. Let $\mathbf{v}_1, \ldots, \mathbf{v}_k$ be probability distribution vectors. The time-dependent teleportation function*

$$\mathbf{v}(t) = \frac{1}{k} \sum_{j=1}^{k} \mathbf{v}_j \Big(\cos(t + (j-1)\tfrac{2\pi}{k}) + 1 \Big),$$

satisfies the both properties:

1. $\mathbf{v}(t) \geq 0$ *for all t, and*
2. $\sum_{i=1}^{n} v(t)_i = 1$ *for all t*

Proof. The first property follows directly because the minimum value of the cosine function is -1, and thus, $\mathbf{v}(t)$ is always non-negative. The second property is also straightforward. Note that

$$\sum_{i=1}^{n} v(t)_i = 1 + \sum_{j=1}^{k} \cos(t + (j-1)\tfrac{2\pi}{k}) = 1 + \sum_{j=1}^{k} \operatorname{Re} \exp\big(\imath t + \imath (j-1)\tfrac{2\pi}{k}\big).$$

Let $r_j(t) = \exp(\imath t + \imath(j-1)\frac{2\pi}{k})$. For $t = 0$, these terms express the k roots of unity. For any other t, we simply rotate these roots. Thus we have $\sum_j r_j(t) = 0$ for any t because the sum of the roots of unity is 0 if $k \geq 2$. The second property now follows because the sum of the real component is still zero. □

For this function, we can solve for the steady-state solution analytically.

Lemma 4: *Let $k \geq 2$, $0 \leq \alpha < 1$, $\mathbf{P}$ be column-stochastic, $\mathbf{v}_1, \ldots, \mathbf{v}_k$ be probability distribution vectors, and*

$$\mathbf{v}(t) = \frac{1}{k} \sum_{j=1}^{k} \mathbf{v}_j \Big(\cos(t + (j-1)\tfrac{2\pi}{k}) + 1 \Big) = \frac{1}{k} \mathbf{V} \cos(t + \mathbf{f}) + \frac{1}{k} \mathbf{V}\mathbf{e},$$

where $\mathbf{V} = \begin{bmatrix}\mathbf{v}_1, \ldots, \mathbf{v}_k\end{bmatrix}$ *and* $f_j = (j-1)\frac{2\pi}{k} \quad j = 1, \ldots, k$. *Then the steady state solution of*

$$\mathbf{x}'(t) = (1-\alpha)\mathbf{v}(t) - (\mathbf{I} - \alpha\mathbf{P})\mathbf{x}(t)$$

is

$$\mathbf{x}(t) = \mathbf{x} + Re\mathbf{s}\exp(\imath t)$$

where $\mathbf{x}$ *is the solution of the static PageRank problem*

$$(\mathbf{I} - \alpha\mathbf{P})\mathbf{x} = (1-\alpha)\frac{1}{k}\mathbf{V}\mathbf{e}$$

and $\mathbf{s}$ *is the solution of the static PageRank problem with* complex teleportation

$$(\mathbf{I} - \tfrac{\alpha}{1+\imath}\mathbf{P})\mathbf{s} = (1-\alpha)\tfrac{1}{k(1+\imath)}\mathbf{V}\exp(\imath\mathbf{f}).$$

Proof. This proof is mostly a derivation of the expression for the solution by guessing the form. First note that if

$$\mathbf{x}(t) = \mathbf{x} + \mathbf{y}(t)$$

then

$$\mathbf{y}'(t) = (1-\alpha)\tfrac{1}{k}\mathbf{V}\cos(t + \mathbf{f}) - (\mathbf{I} - \alpha\mathbf{P})\mathbf{y}(t).$$

That is, we've removed the constant term from the teleportation function by looking at solutions centered around the static PageRank solution. To find the steady-state solution, we look at the complex-phasor problem:

$$\mathbf{z}'(t) = (1-\alpha)\tfrac{1}{k}\mathbf{V}\exp(\imath t + \imath\mathbf{f}) - (\mathbf{I} - \alpha\mathbf{P})\mathbf{z}(t)$$

where $\mathbf{y}(t) = \mathrm{Re}\mathbf{z}(t)$. Suppose that $\mathbf{z}(t) = \mathbf{s}\exp(\imath t)$. Then:

$$\mathbf{z}'(t) = \imath\mathbf{s}\exp(\imath t) = (1-\alpha)\tfrac{1}{n}\exp(\imath t)\exp(\imath\mathbf{f}) - (\mathbf{I} - \alpha\mathbf{P})\mathbf{s}\exp(\imath t).$$

The statement of $\mathbf{s}$ in the theorem is exactly the solution after canceling the phasor $\exp(\imath t)$. We now have to show that this solution is well-defined. PageRank with a complex teleportation parameter γ exists for any column-stochastic $\mathbf{P}$ if $|\gamma| < 1$ (see [140, 141]). For the problem defining $\mathbf{s}$, $\gamma = \alpha/(1+\imath)$ and $|\gamma| = \alpha/\sqrt{2}$. Thus, such a vector $\mathbf{s}$ always exists. $\square$

We conclude with an example of this theorem. Consider a four node graph with adjacency matrix and transition matrix:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \qquad \text{and} \qquad \mathbf{P} = \begin{bmatrix} 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \end{bmatrix}.$$

Let $\mathbf{v}_j = \mathbf{e}_j$ for $j = 1, \ldots, 4$. That is, interest oscillates between all four nodes in the graph in a regular fashion. We show the evolution of the dynamical system for 20 time-units in Figure 3.2. This evolution quickly converges to the oscillators predicted by the lemma. In the interest of simplifying the plot, we do not show the exact curves as they are visually indistinguishable from those plotted for $t \geq 4$. By solving the complex valued PageRank to compute $\mathbf{s}$, we can compute the magnitude of the fluctuation:

$$|\mathbf{s}| = \begin{bmatrix} 0.0216 & 0.0261 & 0.0122 & 0.0235 \end{bmatrix}^T.$$

This vector accurately captures the magnitude of these fluctuations.

3.2.4 Ranking from Time-series

The above equations provide a time-series of dynamic PageRank vectors for the nodes, denoted formally as $\mathbf{x}(t), 0 \leq t \leq t_{\max}$. Applications, however, often want a single score, or small set of scores, to characterize sets of interesting nodes. There are a few ways in which these time series give rise to scores. Many of these methods were explained by [42] in the context of ranking sequences of vectors. Having a variety of different scores derived from the same data frequently helps when using these scores as features in a prediction or learning task [141, 144].

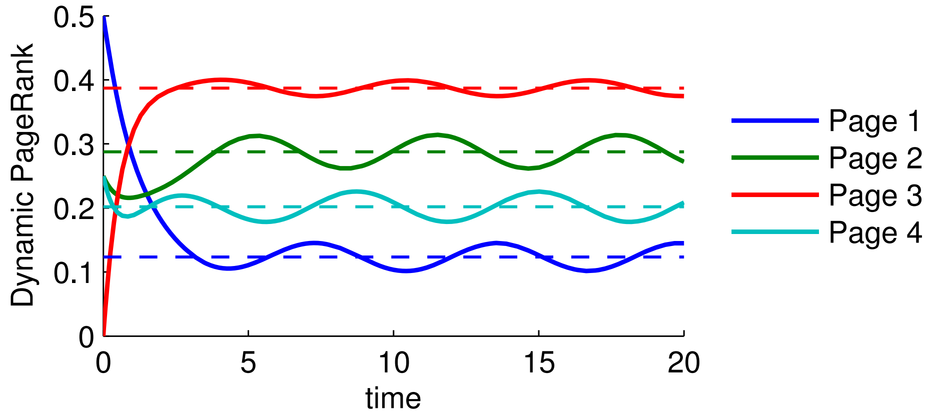


Fig. 3.2.: The dashed lines represent the average PageRank vector computed for the teleportation vectors. The curves show the evolution of the PageRank dynamic system for this example of teleportation. We see that the dynamic PageRanks fluctuate about their average PageRank vectors. Lemma 4 predicts the magnitude of the fluctuation.

Transient Rank. We call the instantaneous values of $\mathbf{x}(t)$ a node's *transient* rank. This score gives the importance of a node at a particular time.

Summary, Variance, and Cumulative Rank. Any summary function s of the time series, such as the integral, average, minimum, maximum, or variance, is a single score that encompasses the entire interval $[0, t_{\max}]$. We utilize the *cumulative rank,* $\mathbf{c}$ and *variance rank,* $\mathbf{r}$ in the forthcoming experiments:

$$\mathbf{c} = \int_0^{t_{\max}} \mathbf{x}(t)\, dt \qquad \text{and} \qquad \mathbf{r} = \int_0^{t_{\max}} (\mathbf{x}(t) - \tfrac{1}{t_{\max}}\mathbf{c})^2\, dt.$$

Difference Rank. A node's difference rank is the difference between its maximum and minimum rank over all time, or a limited time window:

$$\mathbf{d} = \max_t[\mathbf{x}(t)] - \min_t[\mathbf{x}(t)] \qquad \mathbf{d}_W = \max_{t \in W} \mathbf{x}(t) - \min_{t \in W} \mathbf{x}(t).$$

Nodes with high difference rank should reflect important events that occurred within the range $[0, t_{\max}]$ or the time window W. We suggest using a window W that omits the initial convergence region of the evolution. In the context of Figure 3.2, we'd set

W to be $[4, 20]$ to approximate the vector $|\mathbf{s}|$ numerically. In Section 3.5 and figure 3.9, we see examples of how current news stories arise as articles with high difference rank.

3.2.5 Modeling Activity

In the next two sections of our introduction to the dynamic teleportation PageRank model, we discuss how to incorporate empirically measured activity into the model. Let $\mathbf{p}_1, \ldots, \mathbf{p}_k$ be k observed vectors of activity for a website. In the cases we examine below, these activity vectors measure page views per hour on Wikipedia and the number of tweets per month on Twitter. We normalize each of them into teleportation distributions, and conceptually think of the collection of vectors as a matrix

$$\mathbf{v}_1, \ldots, \mathbf{v}_k \quad \rightarrow \quad \mathbf{V} = \begin{bmatrix} \mathbf{v}_1, \ldots, \mathbf{v}_k \end{bmatrix}.$$

Let $\mathbf{e}(i)$ be a functional form representing the vector $\mathbf{e}_i$. The time-dependent teleportation vector we create from this data is:

$$\mathbf{v}(t) = \mathbf{V}\mathbf{e}(\text{floor}\{t\} + 1) = \mathbf{v}_{\text{floor}\{t\}+1}.$$

For this choice, the time-units of our dynamical system are given by the time-unit of the original measurements. Other choices are possible too. Consider:

$$\mathbf{v}_s(t) = \mathbf{V}\mathbf{e}(\text{floor}\{t/s\} + 1) = \mathbf{v}_{\text{floor}\{t/s\}+1}.$$

If $s > 1$, then time in the dynamical system slows down. If $s < 1$, then time accelerates. Thus, we call s the time-scale of the system. Note that

$$\mathbf{x}(sj), \qquad j = 0, 1, \ldots$$

represents the same effective time-point for any time-scale. Thus, when we wish to compare different time-scales s, we examine the solution at such scaled points.

In the experimental evaluation, the parameter s plays an important role. We illustrate its effect in Figure 3.3(a) for a small subnetwork extracted from Wikipedia. As we discuss further in section 3.3, for large values of s, then $\mathbf{v}(t)$ looks constant for long periods of time, and hence $\mathbf{x}(t)$ begins to converge to the PageRank vector for the current, and effectively static, teleportation vector. Thus, we also plot the converged PageRank vectors as a step function. We see that as s increases, the lines converge to these step functions, but for $s = 1$ and $s = 2$, they behave differently.

3.2.6 Smoothing Empirical Activity

So far, we defined a time-dependent that $\mathbf{v}(t)$ changes at fixed intervals based on empirically measured data. A better idea is to smooth out these "jumps" using an exponentially weighted moving average. As a continuous time function, this yields:

$$\bar{\mathbf{v}}'(t;\theta) = \theta\mathbf{v}(t) - \theta\bar{\mathbf{v}}(t;\theta).$$

To understand why this smooths the sequence, consider an implicit Euler approximation:

$$\bar{\mathbf{v}}(t) = \frac{1}{1+h\theta}\bar{\mathbf{v}}(t-h;\theta) + \frac{h\theta}{1+h\theta}\mathbf{v}(t).$$

This update can be written more simply as:

$$\bar{\mathbf{v}}(t;\theta) = \underbrace{\gamma\mathbf{v}(t)}_{\text{new data}} + \underbrace{(1-\gamma)\bar{\mathbf{v}}(t-h;\theta)}_{\text{old data}},$$

where $\gamma = \frac{h\theta}{1+h\theta}$. When $\mathbf{v}(t)$ changes at fixed intervals, then $\bar{\mathbf{v}}(t;\theta)$ will slowly change. If θ is small then $\bar{\mathbf{v}}(t;\theta)$ changes slowly. We recover the "jump" changes in $\mathbf{v}(t)$ in the limit $\theta \to \infty$.

The effect of θ is shown in Figure 3.3(b). Note that we quickly recover behavior that is effectively the same as using jumps in $\mathbf{v}(t)$ ($\theta = 1, 10$). So we only expect changes with smoothing for $\theta < 1$.

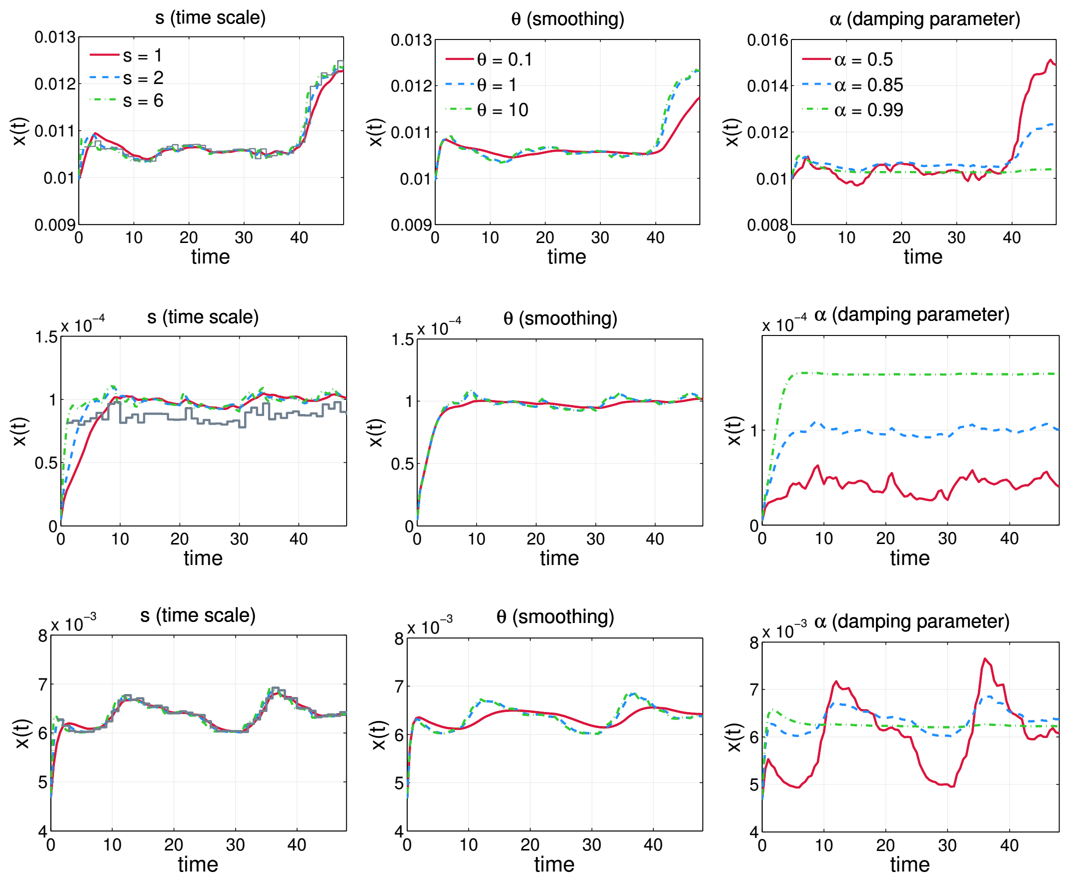


Fig. 3.3.: The evolution of PageRank values for one node due to the dynamical teleportation. The horizontal axis is time $[0, 20]$, and the vertical axis runs between [0.01,0.014]. In figure (a), $\alpha = 0.85$, and we vary the time-scale parameter (section 3.2.5) with no smoothing. The solid dark line corresponds to the step function of solving PageRank exactly at each change in the teleportation vector. All samples are taken from the same effective time-points as discussed in the section. In figure (b), we vary the smoothing (section 3.2.6) of the teleportation vectors with $s = 2$, and $\alpha = 0.85$. In figure (c), we vary α with $s = 2$ and no smoothing. We used the `ode45` function in Matlab, a Runge-Kutta method, to evolve the system.

3.2.7 Choosing the Teleportation Factor

Picking α even for static PageRank problems is challenging, see [145] for some discussion. In this manuscript, we do not perform any systematic study of the effects of α beyond Figure 3.3-3.5. This simple experiment shows one surprising feature. Common wisdom for choosing α in the static case suggests that as α approaches 1,

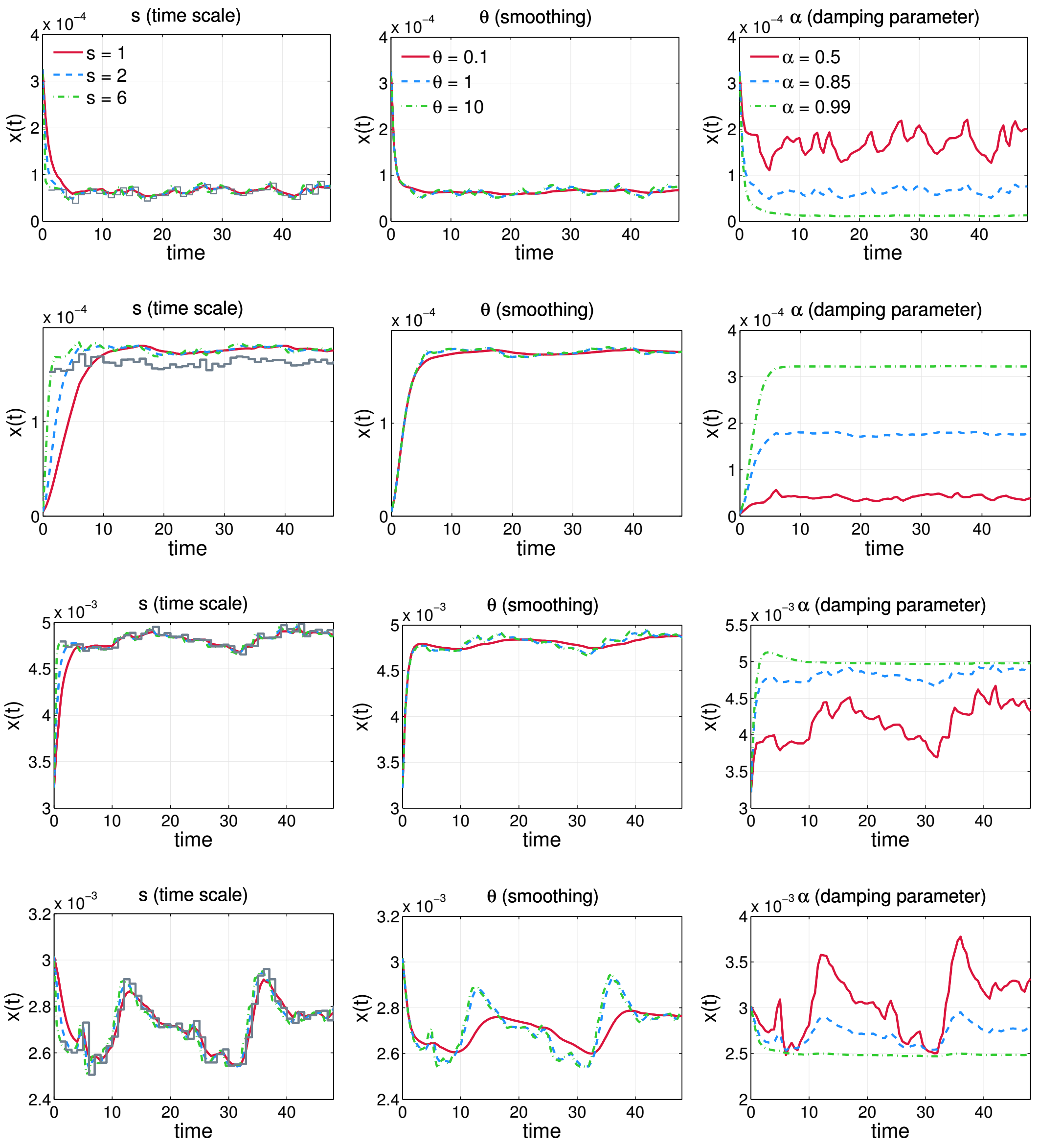


Fig. 3.4.: Evolution of PageRank values for one node (cont.)

the vector becomes more sensitive. For the dynamic teleportation setting, however, the opposite is true. Small values of α produce solutions that more closely reflect the teleportation vector – the quantity that is changing – whereas large values of α reflect the graph structure, which is invariant with time. Hence, with dynamic teleportation,

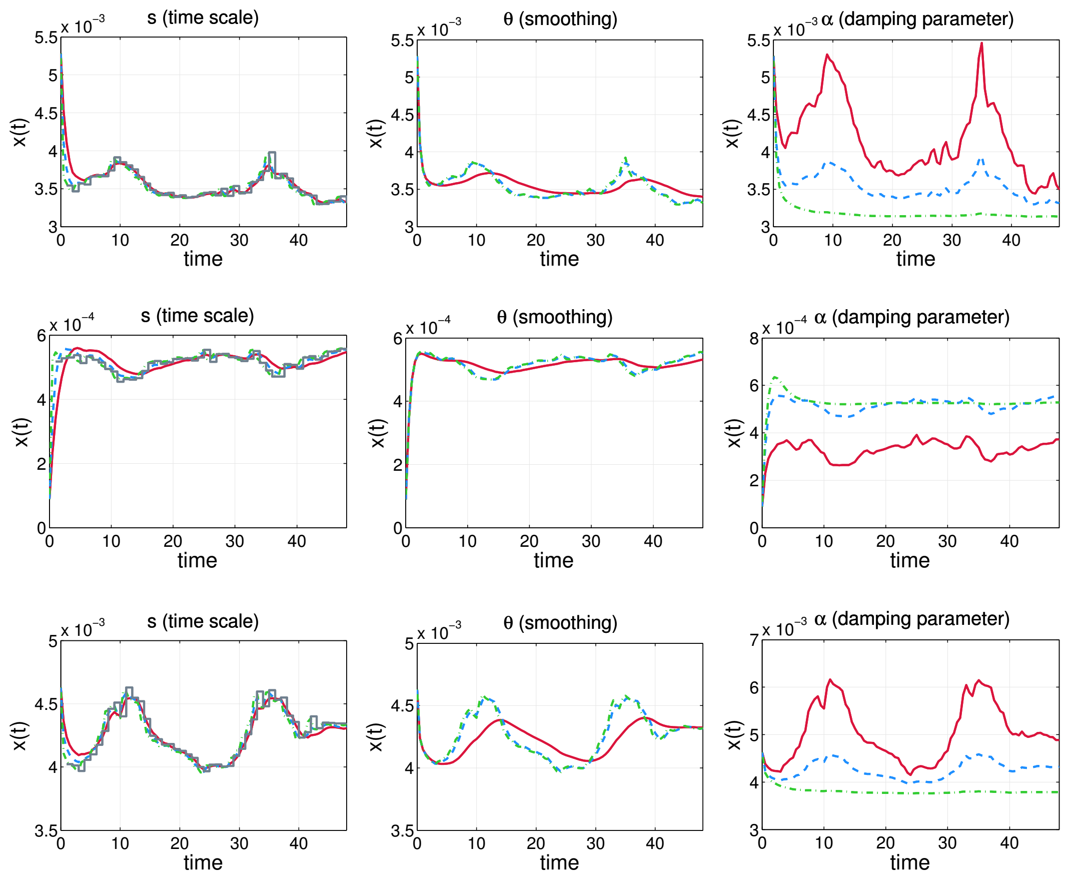


Fig. 3.5.: Evolution of PageRank values for one node (cont.)

using a small value of α is the sensitive setting. Note that this observation is a straightforward conclusion from the equations of the dynamic vector:

$$\mathbf{x}'(t) = (1 - \alpha)\mathbf{v}(t) + \alpha\mathbf{P}\mathbf{x}(t) - \mathbf{x}(t)$$

so α small implies a larger change due to $\mathbf{v}(t)$. Nevertheless, we found it surprising in light of the existing literature.

3.3 Methods for Dynamic PageRank

In order to compute the time-sequence of PageRank values $\mathbf{x}(t)$, we can evolve the dynamical system (3.2) using any standard method – usually called an integrator. We discuss both the forward Euler method and a Runge-Kutta method next. Both methods, and indeed, the vast majority of dynamical system integrators only require a means to evaluate the derivative of the system at a time t given $\mathbf{x}(t)$. For PageRank with dynamic teleportation, this corresponds to computing:

$$\mathbf{x}'(t) = (1-\alpha)\mathbf{v}(t) - (\mathbf{I} - \alpha\mathbf{P})\mathbf{x}(t).$$

The dominant cost in evaluating $\mathbf{x}'(t)$ is the matrix vector product $\mathbf{Px}$. For the explicit methods we explore, all of the other work is linear in the number of nodes, and hence, these methods easily scale to large networks. Both of these methods may also be used in a distributed setting if a distributed matrix-vector product is available.

3.3.1 Forward Euler

We first discuss the forward Euler method. This method lacks high accuracy, but is fast and straightforward. Forward Euler approximates the derivative with a first order Taylor approximation:

$$\mathbf{x}'(t) \approx \frac{\mathbf{x}(t+h) - \mathbf{x}(t)}{h},$$

and then uses that approximation to estimate the value at a short time-step in the future:

$$\mathbf{x}(t+h) = \mathbf{x}(t) + h\left[(1-\alpha)\mathbf{v}(t) - (\mathbf{I} - \alpha\mathbf{P})\mathbf{x}(t)\right].$$

This update is the original Richardson iteration with $h = \omega$. We present the forward Euler method as a formal algorithm in Figure 3.6 in order to highlight a comparison with the power and Richardson method. That is, the forward Euler method is simply running a power method, but changing the vector $\mathbf{v}$ at every iteration. However, we derived this method based on evolving (3.2). Thus, by studying the relationship between (3.2) and the algorithm in Figure 3.6, we can understand the underlying problem solved by changing the teleportation vector while running the power method.

Long time-scales. Using the forward Euler method, we can analyze the situation with a large time-scale parameter s. Consider an arbitrary $\mathbf{x}(0)$, $\alpha = 0.85$, $s = 100$, $h = 1$, and no smoothing. In this case, then the forward Euler method will run the Richardson iteration for 100 times before observing the change in $\mathbf{v}(t)$ at $t = 100$. The difference between $\mathbf{x}(k)$ and the exact PageRank solution for this temporarily static $\mathbf{v}(t)$ is $\|\mathbf{x}(k) - \mathbf{x}\|_1 \leq 2\alpha^k$. For $k > 50$, this difference is small. Thus, a large s and no smoothing corresponds to solving the PageRank problem for each change in $\mathbf{v}$.

Stability. The forward Euler method with timestep h is stable if the eigenvalues of the matrix $-h(\mathbf{I} - \alpha\mathbf{P})$ are within distance 1 of the point -1. The eigenvalues of $\mathbf{P}$ are all between -1 and 1 because it is a stochastic matrix, and so this is stable for any $h < \frac{2}{1+\alpha}$.

3.3.2 Runge-Kutta

Runge-Kutta [146, 147] numerical schemes are some of the most well-known and most used. They achieve far greater accuracy than the simple forward Euler method, at the expense of a greater number of evaluations of the function $\mathbf{x}'(t)$ at each step. We use the implementations of Runge-Kutta methods available in the Matlab ODE suite [148]. The step-size is adapted automatically based on a local error estimate, and the solution can be evaluated at any desired point in time. The stability region for Runge-Kutta includes the region for forward Euler, so these methods are stable. These methods are also fast. To integrate the system for Wikipedia with over 4 million vertices and 60 million edges, it took between 300-600 seconds, depending on the parameters.

Algorithm 1 Forward Euler method for evolving the dynamical system

```
 1: procedure FORWARD-EULER
 2:     Input:
 3:       a graph $G = (V, E)$ and a procedure to compute $\mathbf{P}\mathbf{x}$ for this graph
 4:       a maximum time $t_{\max}$
 5:     a function to return $\mathbf{v}(t)$ for any $0 \leq t \leq t_{\max}$
 6:     a damping parameter $\alpha$
 7:     a time-step $h$
 8:     Output: $\mathbf{X}$ where the $k$th column of $\mathbf{X}$ is $\mathbf{x}(0 + kh)$ for all $1 \leq k \leq t_{\max}/h$
    (or a subset)
 9:     $t \leftarrow 0$; $k = 1$
10:     $\mathbf{x}(0) \leftarrow \mathbf{v}(0)$ (or any other desired initial condition)
11:     while $t \leq t_{\max} - h$ do
12:         $\mathbf{x}(t+h) \leftarrow \mathbf{x}(t) + h\,[(1-\alpha)\mathbf{v}(t) - (\mathbf{I} - \alpha\mathbf{P})\mathbf{x}(t)]$
13:         $\mathbf{X}(:, k) \leftarrow \mathbf{x}(t+h)$
14:         $t \leftarrow t + h$; $k \leftarrow k + 1$
```

Fig. 3.6.: The forward Euler method for evolving the dynamical system: $\mathbf{x}'(t) = (1-\alpha)\mathbf{v}(t) - (\mathbf{I} - \alpha\mathbf{P})\mathbf{x}(t)$. The resulting procedure looks remarkably similar to the standard Richardson iteration to compute a PageRank vector. One key difference is that there is no notion of convergence.

3.3.3 Maintaining Interpretability

Based on the theory of the dynamic teleportation system, we expect that $\mathbf{x}(t) \geq 0$ and $\mathbf{e}^T\mathbf{x}(t) = 1$ for all time. Although this property should be true of the computed solution, we often find that the sum diverges from one. Consequently, for our experiments, we include a correcting term:

$$\mathbf{x}'(t) = (1-\alpha)\mathbf{v}(t) - (\gamma\mathbf{I} - \alpha\mathbf{P})\mathbf{x}(t)$$

where $\gamma = (1-\alpha)\mathbf{e}^T\mathbf{v}(t) + \alpha\mathbf{e}^T\mathbf{x}(t)$. Note that $\gamma = 1$ if the $\mathbf{x}(t)$ has sum exactly 1. If $\mathbf{e}^T\mathbf{x}(t)$ is slightly different from one, then the correction with γ ensures that $\mathbf{e}^T\mathbf{x}'(t) = 0$ numerically. Similar issues arise in computing static PageRank [149], although the additional computation in the Runge-Kutta methods exacerbates the problem.

3.4 Experiments

We now use dynamic teleportation to investigate page view patterns on Wikipedia and user activity on Twitter. In the following experiments, unless otherwise noted, we set $s = 1$, $\alpha = 0.85$, do not use smoothing ("$\theta = \infty$"), and use the `ode45` method from Matlab to evolve the system. We study this model on two datasets.

3.4.1 Datasets

We provide some basic statistics of the Wikipedia and Twitter datasets in Table 3.2. For Wikipedia, the time unit for $s = 1$ is an hour, and for Twitter, it is one month.

Wikipedia Article Graph and Hourly page views. Wikipedia provides access to copies of its database [150]. We downloaded a copy of its database on March 6th, 2009 and extracted an article-by-article link graph, where an article is a page in the main Wikipedia namespace, a category page, or a portal page. All other pages and links were removed.

Wikipedia also provides hourly page views for each page [151]. These are the number of times a page was viewed for a given hour. These are not unique visits. We downloaded the raw page counts and matched the corresponding page counts to the pages in the Wikipedia graph. We used the page counts starting from March 6, 2009 and moving forward in time. Although it would seem like measuring page views would correspond to measuring $\mathbf{x}(t)$ instead of $\mathbf{v}(t)$, one of our earlier studies showed that users hardly ever follow links on Wikipedia [145]. Thus, we can interpret these page views as a reasonable measure of external interest in Wikipedia pages.

Table 3.2.: Dataset properties. The page views for each page on Wikipedia or total tweets for each user on twitter is denoted as **p** and we show the maximum and average for any page at any time.

Dataset	Nodes	Edges	$t_{\max}$	Period	Average p_i	Max p_i
WIKIPEDIA	4,143,840	72,718,664	48	hours	1.4243	353,799
TWITTER	465,022	835,424	6	months	0.5569	1056

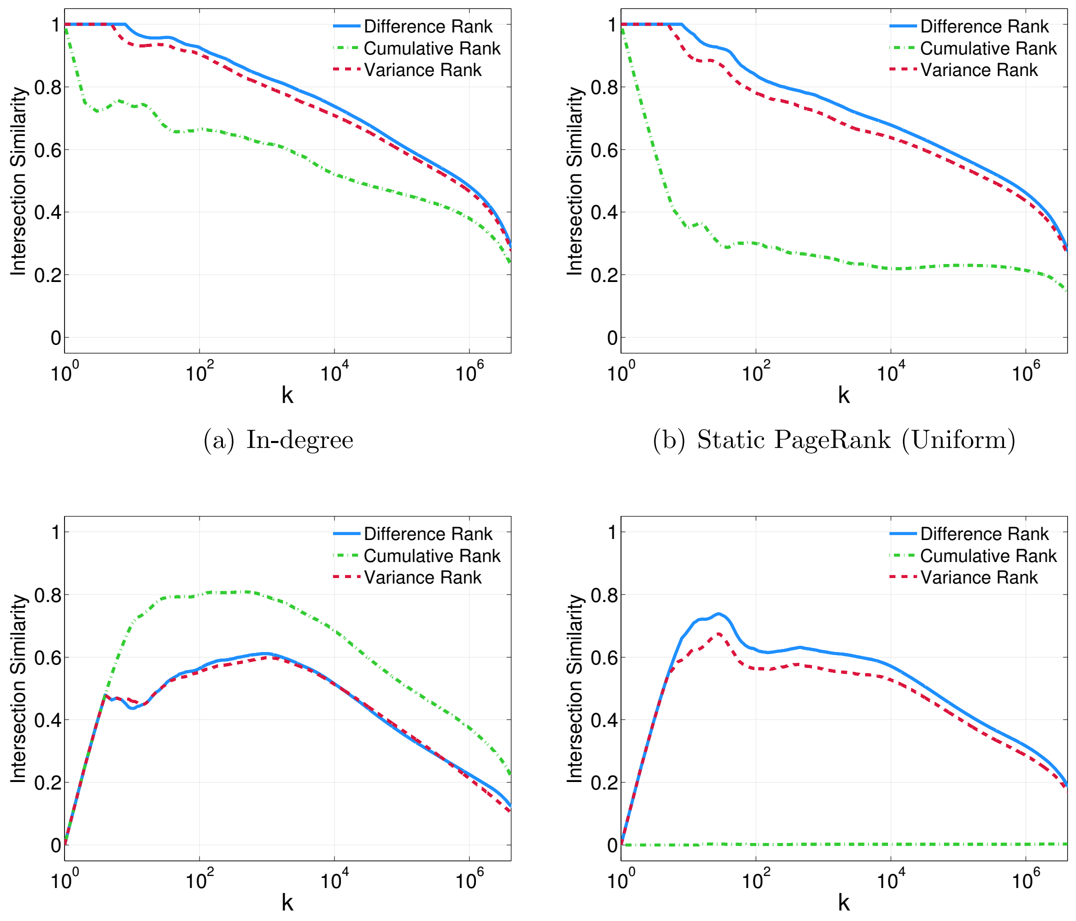
Twitter Social Network and Monthly Tweet Rates. The Twitter social networks consists of a set of users that *follow* each other's tweets, or small 140 character messages. Thus, Twitter has both a network structure, the follower graph, and activity on top of this graph, the tweet stream. We built the follower graph by starting with a few seed users and crawling follow links for several iterations. (The particular crawl we use is from 2008.) We then take the set of users from the follower graph and extract their tweets for a period of six months between 2008 – 2009. Each tweet is time-stamped, and we construct a sequence of vectors to represents the number of tweets from each user in each month. (We briefly explored finer levels of granularity for Twitter activity, but these choices led to sparser vectors.) These vectors are the basis for the teleportation time-series in our time-dependent PageRank.

3.4.2 Rankings from Transient Scores

First, we evaluate the rankings from dynamic PageRank using the intersection similarity measure [152]. Given two vectors $\mathbf{x}$ and $\mathbf{y}$, the intersection similarity metric at k is the average symmetric difference over the top-j sets for each $j \leq k$. If $\mathcal{X}_k$ and $\mathcal{Y}_k$ are the top-k sets for $\mathbf{x}$ and $\mathbf{y}$, then

$$\text{isim}_k(\mathbf{x}, \mathbf{y}) = \frac{1}{k} \sum_{j=1}^{k} \frac{|\mathcal{X}_j \Delta \mathcal{Y}_j|}{2j}$$

where Δ is the symmetric set-difference operation. Identical vectors have an intersection similarity of 0.



(a) In-degree (b) Static PageRank (Uniform)

(c) Average page views (d) Static PageRank (avg. page views)

Fig. 3.7.: Intersection similarity of rankings derived from dynamic PageRank. We compute the intersection similarity of the difference, variance, and cumulative rankings given by dynamic PageRank and compare these with the rankings given by the in-degree, average page views, static PageRank with uniform teleportation, and static PageRank with average page views as the teleportation vector. For dynamic PageRank, we set the initial value $\mathbf{x}(0)$ to be the solution of the static PageRank system which uses $\mathbf{v}(0)$ as the teleportation vector.

For the Wikipedia graph, Figure 3.7 shows the similarity profile comparing a few ranking measures from dynamic PageRank to reasonable baselines. In particular, we compare $\mathbf{d}$, $\mathbf{r}$, $\mathbf{c}$ (from §3.2.4) to indegree, average page views, static PageRank with uniform teleportation, and static PageRank using average page views as the teleportation vector. The results suggest that dynamic PageRank is different from the other measures, even for small values of k. In particular, combining the external influence with the graph appears to produce something new. The only exception is in Fig. 3.7(d) where the cumulative rank is shown to give a similar ordering to static PageRank using average page views as the teleportation.

3.4.3 Difference Ranks

Figure 3.8 and figure 3.9 show the time-series of the top 100 pages by the difference measure for Wikipedia with $s = 1$ and $s = 4$ without smoothing. Many of these pages reveal the ability of dynamic PageRank to mesh the network structure with changes in external interest. For instance, in figure 3.9, we find pages related to an Australian earthquake (43, 84, 82), the "recently" released movie "Watchmen" (98, 23-24), a famous musician who died (2, 75), recent "American Idol" gossip (34, 63), a remembrance of Eve Carson from a contestant on "American Idol" (88, 96, 34), news about the murder of a Harry Potter actor (60), and the Skittles social media mishap (94). These results demonstrate the effectiveness of the dynamic PageRank to identify interesting pages that pertain to external interest. The influence of the graph results in the promotion of pages such as Richter magnitude (84). That page was not in the top 200 from page views.

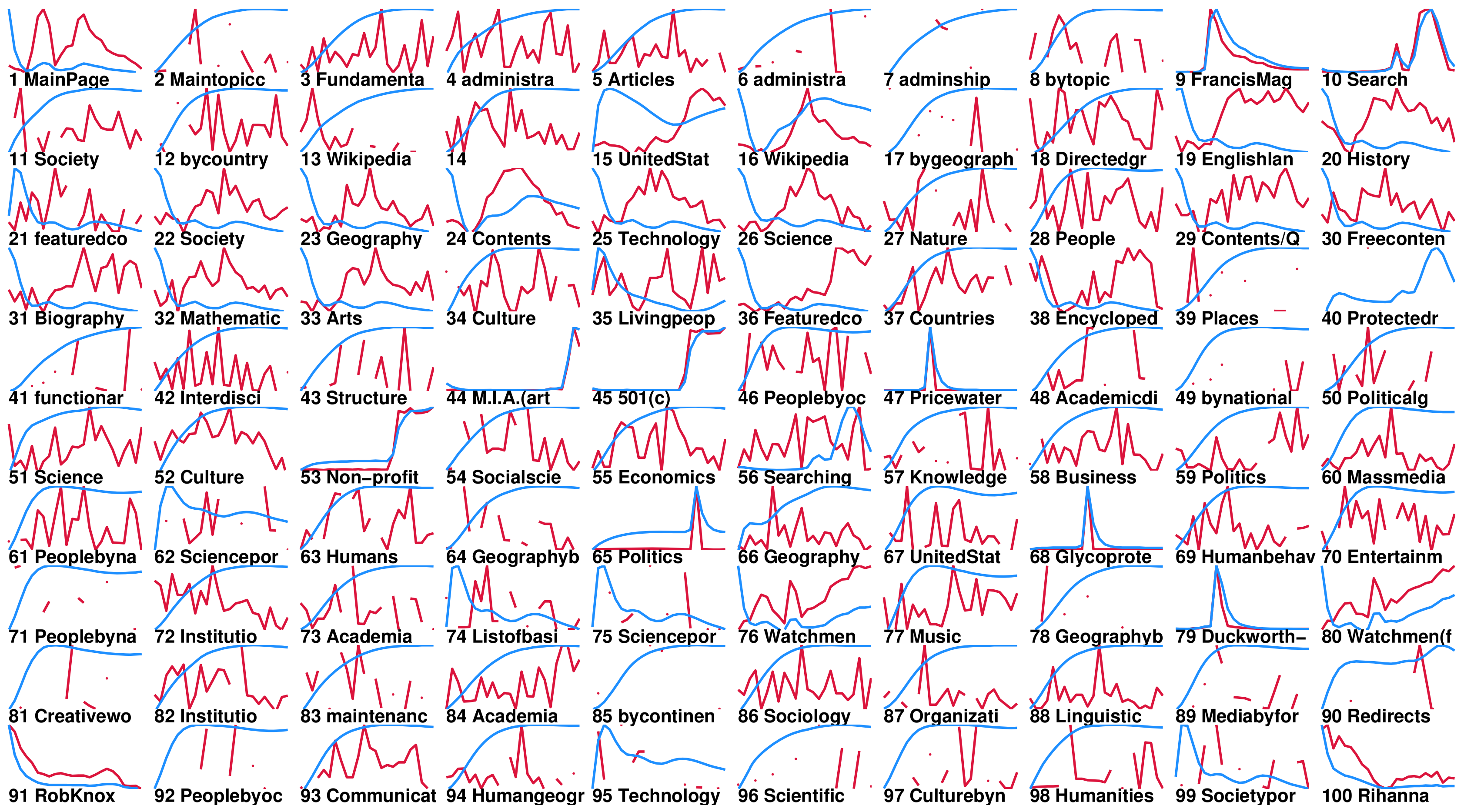


Fig. 3.8.: The top-100 Wikipedia pages that fluctuate the most as determined by the difference ranking from our dynamic PageRank approach. The blue curves are the transient scores, and the red-curves are the raw page view data. The horizontal axis is 24 hours, and the vertical axes are normalized to show the range of the data.

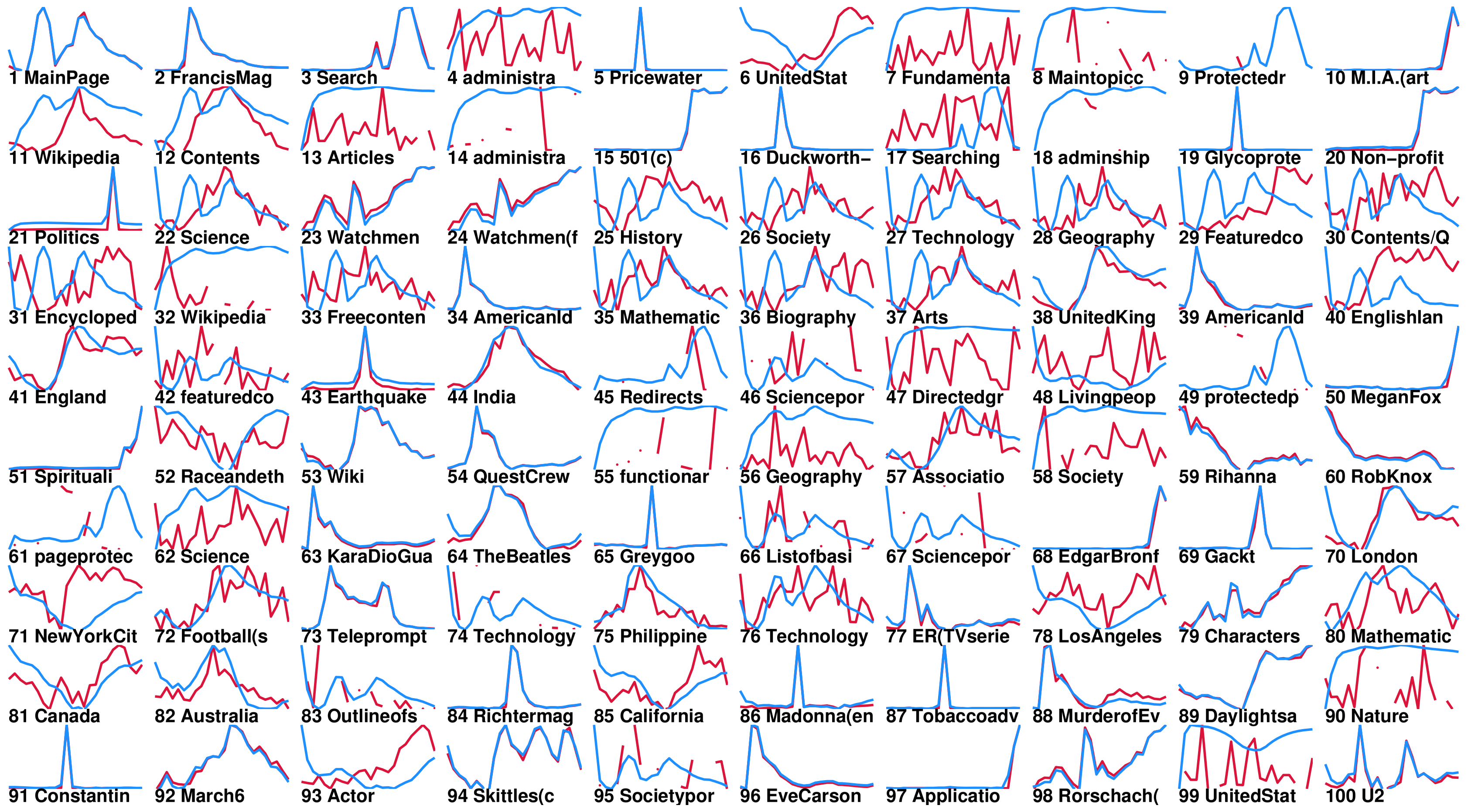


Fig. 3.9.: We again plot the top 100 pages from the difference rank, now using $s = 4$. The blue curves are the transient scores, and the red-curves are the raw page view data. The horizontal axis is 24 hours, and the vertical axes are normalized to show the range of the data. This choice gives a similar ordering as our previous forward Euler iteration method from [44]. Note the large change between the transient scores for "MainPage" between this figure and figure 3.8.

3.4.4 Predicting Future Page Views and Tweets

We begin by studying how well the dynamical system can *predict* the future. Formally, given a lagged time-series $\mathbf{p}_{t-w}, ..., \mathbf{p}_{t-1}, \mathbf{p}_t$ [61], the goal is to predict the future value $\mathbf{p}_{t+1}$ (actual page views or number of tweets). This type of temporal prediction task has many applications, such as actively adapting caches in large database systems, or dynamically recommending pages.

We performed one-step ahead predictions $(t+1)$ using linear regression. That is, we learn a model of the form:

$$\left[\begin{array}{cccc} \bar{\mathbf{f}}(t-1) & \bar{\mathbf{f}}(t-2) & \dots & \bar{\mathbf{f}}(t-w) \end{array} \right] \mathbf{b} \;\approx\; \mathbf{p}(t)$$

where w is the window-size, and $\bar{\mathbf{f}}(\cdot)$ is either page views or both page views and transient scores. After fitting $\mathbf{b}$, the model predicts $\mathbf{p}(t+1)$ as

$$\left[\begin{array}{cccc} \bar{\mathbf{f}}(t) & \bar{\mathbf{f}}(t-1) & \cdots & \bar{\mathbf{f}}(t-w+1) \end{array} \right] \mathbf{b}$$

We use the symmetric Mean Absolute Percentage Error (sMAPE) [61] measure to evaluate the prediction:

$$\text{sMAPE} = \frac{1}{|T|} \sum_{t=1}^{|T|} \frac{|p_t - \hat{p}_t|}{(p_t + \hat{p}_t)/2}.$$

This relative error measure averages all the relative prediction errors over all the time-steps. We then average it over nodes.

For evaluation of Wikipedia and Twitter, we vary w in order to use all the past information for each one-step ahead prediction. As an example, we start by setting $t = 2$, and use $w = 1$ so that we use all past information to predict $t + 1$. After we forecast $t + 1$, we repeat the above by setting $t = 3$ and use $w = 2$ in order to predict $t + 1$. We repeat this until we have predicted all timesteps (up to $t = t_{max} - 1$ and $w = t_{max} - 2$). This corresponds to learning a model for each $t = 2, ..., t_{max} - 1$ and performing one-step ahead forecasting where for each t we set $w = t - 1$. Hence, we vary w in order to use all available past information.

We note that we also tried setting w to a fixed constant and also used AIC to learn the value of w for each one-step ahead prediction. However, there appeared to be no significant difference, and thus we chose to simply use all past information. This choice also allowed us to better understand the effectiveness of using dynamic pagerank for prediction. For other applications, one may wish to set w accordingly.

We study two predictive modes. The *base model* uses only the time-series of page views or tweets to predict the future page views or tweets. The *dynamic teleportation model* uses both the transient scores with smoothing and page views to predict the future page views (or tweets).

We evaluate these models for prediction on *stationary* and *non-stationary* time-series. Informally, a time-series is weakly stationary if it has properties (mean and covariance) similar to that of the time-shifted time-series. We consider the top and bottom 10,000 nodes from the difference ranking as nodes that are approximately non-stationary (volatile) and stationary (stable), respectively. Table 3.3 compares the predictions of the models across time for non-stationary and stationary prediction tasks. Our findings indicate that the Dynamic PageRank time-series provides valuable information for forecasting future tweet rates; however, it adds little (if any) accuracy in forecasting future page views on Wikipedia.

For Twitter, the dynamic teleportation model improves predictions the most with the non-stationary nodes. The diffusion of activity captured by the model allows our model to detect, early on, when the external interest of vertices will change, before that change becomes apparent in the external interest of the vertices. This is easiest to detect when there is a large sudden change in external interest of a neighboring vertex.

Table 3.3.: The ratio between the base model and the model with dynamic teleportation scores with $s = 1, 2, 6$, and ∞, for three smoothing parameters. (Here, $s = \infty$ corresponds to solving the PageRank problem exactly for each change in teleportation.) If this ratio is less than 1, then the model with the dynamic teleportation scores improves the prediction performance. We also distinguish between prediction problems with highly volatile nodes (non-stationary) and nodes with relatively stable behavior (stationary). The results show a much stronger benefit for Twitter than for Wikipedia

Dataset	**Type**	θ	**Error Ratio**			
			s (timescale)			
			1	**2**	**6**	∞
TWITTER	stationary	**0.01**	0.450	0.898	0.836	0.967
		0.50	0.258	0.611	0.858	0.775
		1.00	0.527	0.531	0.849	0.791
	non-stationary	**0.01**	0.500	0.874	0.662	1.240
		0.50	0.461	0.499	0.658	0.835
		1.00	0.458	0.489	0.652	0.848
WIKIPEDIA	stationary	**0.01**	0.978	0.991	0.989	0.978
		0.50	1.140	1.130	1.004	0.990
		1.00	1.084	0.976	1.010	0.990
	non-stationary	**0.01**	0.968	1.011	0.968	1.004
		0.50	1.218	0.994	1.030	1.031
		1.00	1.241	0.996	0.957	0.998

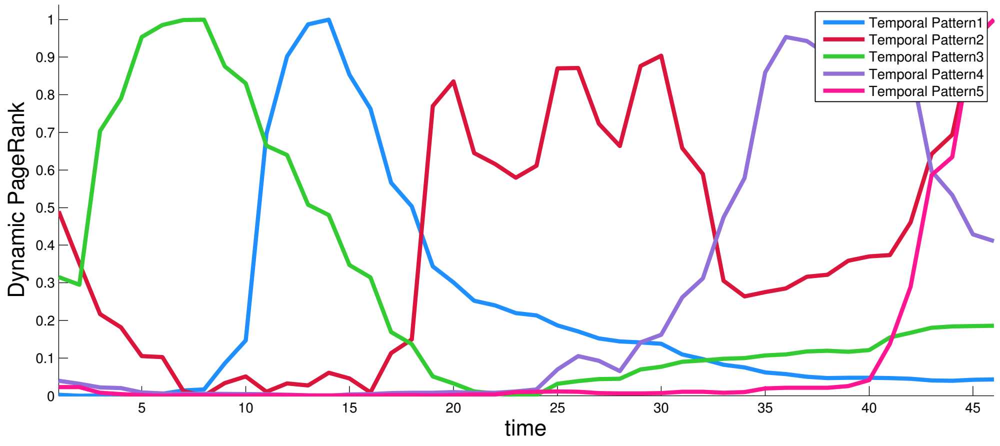
3.5 Applications

This section explores the opportunity of using Dynamic PageRank for a variety of applications outside of the context of ranking.

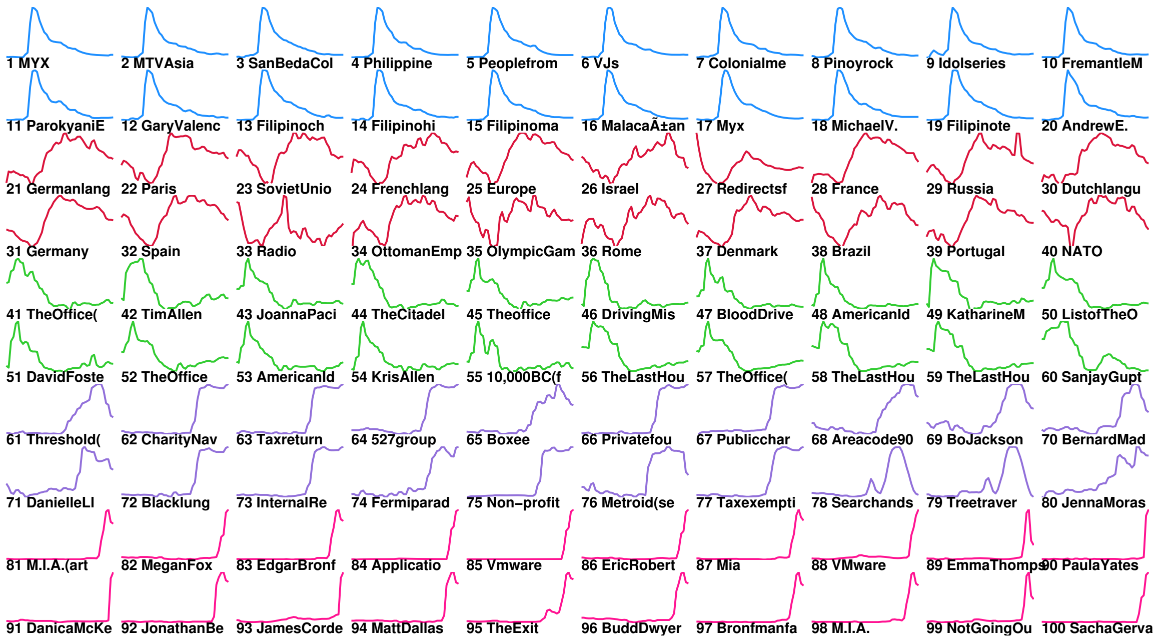
3.5.1 Clustering Dynamic Patterns

Identifying vertices with similar time-series is important for modeling and understanding large collections of multivariate time-series. We now group vertices according to their transient scores. By using the difference rank measure $\mathbf{d}$ for $s = 4$, we cluster the top 5,000 vertices using k-means with $k = 5$, repeat the clustering 2,000 times, and take the minimum distance clustering identified.

The cluster centroids are temporal patterns, and the main patterns in the dynamic PageRanks are visualized in figure 3.10(a). Pattern 2 represents European-centric behavior, whereas the others correspond to spikes or unusual events occurring within



(a) Temporal Patterns



(b) Vertices with Similar Dynamics

Fig. 3.10.: Clustering nodes using dynamic PageRank scores. Vertices with similar dynamical properties are grouped together. The visualization reveals the important dynamic patterns (spikes, trends) present from March 6th, 2009 in our large collection of time-series from Wikipedia. For each hour, we sample twice from the continuous function $\mathbf{x}(t)$ and utilize these intermediate values in the clustering.

the dynamic PageRank system. Figure 3.10(b) plots the 20 closest vertices matching the patterns above. A few pages from the five groups are consistent with our previously discussed results from figure 3.9. One such unusual event is related to the death of a famous musician/actor from the Philippines (see pages 1-20). The pages from the third cluster (41-60) are related to “American Idol” and other TV shows/actors. Also some of the pages from the fourth cluster relate to Bernard Madoff (63, 66, 67, 70, 73), six days before he plead guilty in the largest financial fraud in U.S. history. This grouping reveals many of the standard patterns in time-series such as spikes and increasing/decreasing trends [153].

3.5.2 Discovering Causal Links

In this section, we use Granger causality tests [154] on the collection of transient scores to attempt to understand which links are most important. The Granger causality model, briefly described below, ought to identify a causal relationship between the time-series of any two vertices connected by a directed edge. This is because *there is a causal relationship* between their time-series in our dynamical system. However, due to the impact of the time-dependent teleportation, only some of these links will be identified as causal. We wish to investigate this smaller subset of links.

Intuitively, if a time-series X causally affects another Y, then the past values of X should be helpful in predicting the future values of Y, above what can be predicted based on the past values of Y alone. This is formalized as follows: the error in predicting $\hat{y}_{t+s}$ from $y_t, y_{t-1}, \dots$ should be larger than the error in predicting $\hat{y}_{t+s}$ from the joint data $y_t, y_{t_1}, \dots, x_t, x_{t-1}, \dots$ if X causes Y. As our model, we chose to use the standard vector-autoregressive (VAR) model from econometrics [58]. This is implemented in a Matlab code by [155]. The standard p-lag VAR model takes the form:

$$\begin{bmatrix} y_t \\ x_t \end{bmatrix} = \mathbf{c} + \sum_{i=1}^{p} \mathbf{M}_i \begin{bmatrix} y_{t-i} \\ x_{t-i} \end{bmatrix} + \mathbf{e}_t$$

where $\mathbf{c}$ is a vector of constants, $\mathbf{M}_i$ are the $n \times n$ coefficient (or autoregressive) mixing matrices and $\mathbf{e}_t$ is the unobservable white-noise. For the results shown below, $p = 2$. We then use the standard F-test to determine significance.

Table 3.4.: Causal link discovery. Example of Causality in Wikipedia. We only consider pages with a *pval* < 0.01 as statistically significant. The page with values "caused" by Earthquake represent ideas related to earthquakes. All pages below are significant with *pval* < 0.01.

Earthquake Granger causes	**p-value**
Seismic hazard	0.003535
Extensional tectonics	0.003033
Landslide dam	0.002406
Earthquake preparedness	0.001157
Richter magnitude scale	0.000584
Fault (geology)	0.000437
Aseismic creep	0.000419
Seismometer	0.000284
Epicenter	0.000020
Seismology	0.000001

Definition 1 (Causal Link Discovery): *Given a vertex u, the neighbors of u denoted $N(u)$ and the matrix $\mathbf{X}_u$ representing the time-series of the neighbors, compute a weight for each edge from u to its neighbors $N(u)$.*

In Table 3.4, we show the causal relationships identified among the out-links of the article *Earthquake*. Recall that there was a major earthquake in Australia during our time-window. We wish to understand which of the out-links appeared to be sensitive to this large change in interest in Earthquake. We use a significance cutoff of 0.01 and test for Granger causality among the time-series with $s = 4$.

Instead of assigning only causal weights to the edges of the neighbors of a vertex, we predict edges based on causality using vertices that are not known to directly influence the target vertex (used in the query).

Definition 2 (Temproal Causal Link Prediction): *Given a vertex u and a set of vertices W such that $W \bigcap N(u) = \emptyset$ where $\mathbf{X}_W$ represents their time-series, compute a causal weight X^E_{uw} for each vertex $w \in W$.*

The above definition is less restrictive than the previous as causality weights are assigned to vertices even if they are not connected in G. This definition assumes that two vertices influence each other (i) either indirectly through a causal path in the graph or (ii) through some external event or variable (i.e., news or event). Since here, we are not interested in causal links between neighbors of a vertex, but all other vertices that are not directly connected.

This task can be viewed as the *causal matrix completion problem* and hence the worst case results in a complete graph (pairwise granger causality). Thus the graph goes from sparse to perfectly dense. For big graphs, this is impossible to store and costly to compute complex functions over. Furthermore, if $W = |V| - |N(u)|$, then we must learn $|V| - |N(u)|$ models and test each independently against $\mathbf{x}_u$ (time-series of u). Hence, it is slow even if parallelized.

Below, we propose a few techniques to avoid filling in the entire matrix by restricting the size of the vertex set W. The goal is that the granger causality graph must remain

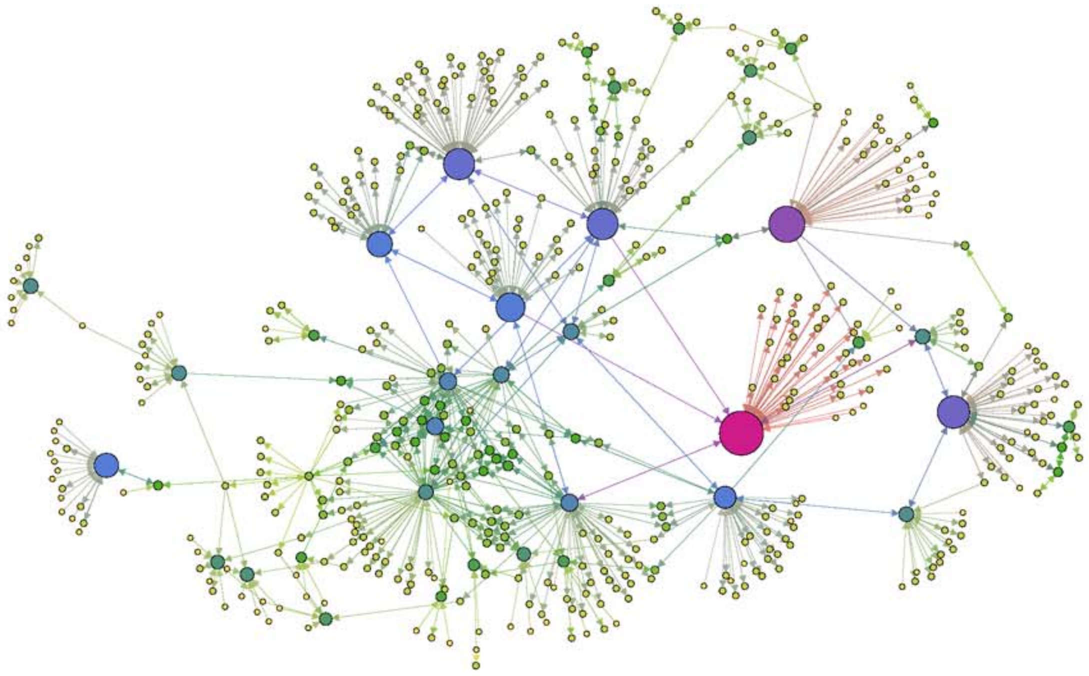


Fig. 3.11.: Granger causality graph of the top fifty vertices from the difference rank and the vertices causal (or caused by) that set. We only include causal relationships of at least $pval < 0.01$ significance. The larger vertices are those with more causal edges and intuitively represent hub-like pages such as Category or Portal pages.

sparse. Note that once the p-val significance cutoff has been chosen, edges may be pruned on the fly by simply testing against it, and discarding the edge if the causality weight is insignificant. This approach avoids the extra storage cost needed to store insignificant causality weights.

- **Two-hop paths**. Let $N_{\ell=2}(u)$ denote the vertices exactly two hops away and $\ell = 2$ denotes the path length. We the set $W = N_2(u) \bigcap N(u)$, or the neighbors two hops away from u that are not neighbors of u (cycles of size 2).
- **Community detection**. Use community detection methods, but avoid direct neighbors of u.
- **Use additional external information**.

In both the above problem definitions (Link Causal Discovery and Temporal Causal Link Prediction), the result is a causality graph where edges represent significant causal relationships.

Definition 3 (**Granger Causality Graph**): *Given a graph $G = (V, E)$ and a sequence of time-series for each vertex in V represented as a matrix $\mathbf{X}^V$ where $n = |V|$ is the number of vertices and t_{max} is the last timestep. Let $G_C = (W, F)$ denote the granger causality graph where W is the set of vertices found to have significant causality with other vertices in G and F is the set of causal edges found to be significant.*

Note $|W| < |V|$, since some vertices in G may not have a single causal edge with another vertex. This graph can then be used in a variety of graph mining techniques to reveal latent information useful for making critical business decisions or a variety of other applications. A smaller significance cutoff $(p - val)$ implies more significant causality. In general, this means a more accurate representation of the true granger causality graph. The time and space complexity can be reduced by setting the significance cutoff to be very small. The good news is that we do not sacrifice accuracy for time and space (like many problems). The effect of using the significance cutoff to prune edges simply reduces the possible noise, while also reducing the storage costs.

The above definition is in terms of a single vertex query, but can be repeated for each vertex. Clearly, this definition lends itself to a straight-forward parallelization over the neighborhood of each vertex. This strategy is used for each of the above causal link tasks.

3.6 Related Work

The relationship between dynamical systems and classical iterative methods has been utilized by [156] to study eigenvalue solvers. It was also noted in an early paper by [157] that there is a relationship between the PageRank and HITS algorithms and dynamical systems.

In the past, others studied PageRank approximations on graph streams [43]. More recently, [158] studied how accurately an evolving PageRank method could estimate the true PageRank of an evolving graph that is accessed only via a crawler. The method used here solved each PageRank problem exactly for the current estimate of the underlying graph. A detailed study of how PageRank values evolve during a web-crawl was done by [159]. In the future, we plan to study dynamic graphs via similar ideas.

As explained in section 3.3 and figure 3.6, our proposed method is related to changing the teleportation vector in the power method as it's being computed. Bianchini et al. [160] noted that the power method would still converge if either the graph or the vector $\mathbf{v}$ changed a few times during the method, albeit to a new solution given by the new vector or graph. Our method capitalizes on a closely related idea and we utilize the intermediate quantities explicitly. Another related idea is the Online Page Importance Computation (OPIC) [161], which integrates a PageRank-like computation *with* a crawling process. The method does nothing special if a node has changed when it is crawled again.

While we described PageRank in terms of a random-surfer model, another characterization of PageRank is that it is a sum of damped transitions:

$$\mathbf{x} = (1-\alpha)\sum_{k=0}^{\infty}(\alpha\mathbf{P})^k\mathbf{v}.$$

These transitions are a type of probabilistic walk and Grindrod et al. [142] introduced the related notion of dynamic walks for dynamic graphs. We can interpret these dynamic walks as a *backward Euler* approximation to the dynamical system:

$$\mathbf{x}'(t) = \alpha\mathbf{A}(t)\mathbf{x}(t) \qquad \mathbf{x}(0) = \mathbf{e}$$

with time-step $h = 1$ and $\mathbf{A}$ is a time-dependent adjacency matrix. This relationship suggests that there may be a range of interesting models between our dynamic teleportation model and existing evolving graph models.

Outside of the context of web-ranking, O'Madadhain and Smyth propose EventRank [42], a method of ranking nodes in dynamic graphs, that uses the PageRank propagation equations for a sequence of graphs. We utilize the same idea but place it within the context of a continuous dynamical system. In the context of popularity dynamics [162], our method captures how changes in external interest influence the popularity of nodes and the nodes linked to these nodes in an implicit fashion. Our work is also related to modeling human dynamics, namely, how humans change their behavior when exposed to rapidly changing or unfamiliar conditions [163]. In one instance, our method shows the important topics and ideas relevant to humans before and after one of the largest Australian Earthquakes (figure 3.9).

In closing, we wish to note that our proposed method *does not involve* updating the PageRank vector, a related problem which has received considerable attention [164,165]. Nor is it related to tensor methods for dynamic graph data [78, 166].

3.7 Summary

PageRank is one of the most widely used network centrality measures. Our dynamical system reformulation of PageRank permits us to incorporate time-dependent teleportation in a relatively seamless manner. Based on the results presented here, we believe this is an interesting variation on the PageRank model. For instance, we can analyze certain choices of oscillating teleportation functions (Lemma 4). Our empirical results show that the maximum change in the transient rank values identifies interesting sets of pages. Furthermore, this method is simple to implement in an online setting using either a forward Euler or Runge-Kutta integrator for the dynamical system. We hope that it might find a use in online monitoring systems.

One important direction for future work is to treat the inverse problem. That is, suppose that the observed page views reflect the behavior of these random surfers. Formally, suppose that we equate page views with samples of $\mathbf{x}(t)$. Then, the goal would be to find $\mathbf{v}(t)$ that produces this $\mathbf{x}(t)$. This may not be a problem for websites such as Wikipedia, due to our argument that the majority of page views reflect search engine traffic. But for many other cases, we suspect that $\mathbf{x}(t)$ may be much easier to observe. In addition, our method also has the following useful properties:

- **Flexible**. Dynamic PageRank is flexible for many application-specific requirements. The time and space complexity can also be reduced by adjusting the time-scale used to sample, and the number of iterations, and tolerance, among many other options.

- **Effectiveness**. Our approach was shown to be effective for a variety of applications such as dynamic ranking of nodes, relational time series regression (i.e., predicting the future page views), discovering causal links, and clustering dynamic time series patterns.

- **Scalable for Big Data**. Dynamic PageRank has a time complexity of linear in terms of edges in the graph and thus great for billion-edge graphs. The storage cost depends on the time-scale and number of vertices, but in general, it is also quite reasonable and flexible for specific requirements. A sparsification technique may be used to set very small values to zero, which we can then ignore in the computation.

- **Incremental.** Dynamic PageRank is incrementally computed and good for near Real-time systems.

4. DYNAMIC NODE PREDICTION

Given a large time-evolving graph, how can we model and characterize the temporal behaviors of individual nodes (and network states)? How can we model the behavioral transition patterns of nodes? We propose a temporal behavior model that captures the "roles" of nodes in the graph and how they evolve over time. The proposed *dynamic behavioral mixed-membership model* (DRMM) is scalable, fully automatic (no user-defined parameters), non-parametric/data-driven (no specific functional form or parameterization), interpretable (identifies explainable patterns), and flexible (applicable to dynamic and streaming networks). Moreover, the interpretable behavioral roles are generalizable and computationally efficient. We applied our model for (a) identifying patterns and trends of nodes and network states based on the temporal behavior, (b) predicting future structural changes, and (c) detecting unusual temporal behavior transitions. The experiments demonstrate the scalability, flexibility, and effectiveness of our model for identifying interesting patterns, detecting unusual structural transitions, and predicting the future structural changes of the network and individual nodes.

Let us note that there are various connections between the techniques used for this representation task and our previous work. For instance, one may utilize the framework from Chapter 2 for modeling and leveraging temporal and relational dependencies. In addition, one may also use the previous feature learning system from Chapter 2 for learning a feature-based representation. Additional connections are discussed later.

4.1 Motivation

In recent years, we have witnessed a tremendous growth in both the variety and scope of network datasets. In particular, network datasets often record the interactions and/or transactions among a set of entities—for example, personal communication (e.g., email, phone), online social network interactions (e.g., Twitter, Facebook), web traffic between servers and hosts, and router traffic among autonomous systems. A notable characteristic of these *activity* networks, is that the structure of the networks change over time (e.g., as people communicate with different friends). These temporal dynamics are key to understanding system *behavior*, thus it is critical to model and predict the network changes over time. An improved understanding of temporal patterns will facilitate for example, the development of software systems to optimally manage data flow, to detect fraud and intrusions, and to allocate resources for growth over time.

Although some recent research has focused on the analysis of dynamic networks [167–172], there has been less work on developing *models* of temporal behavior in large scale network datasets. There has been some work on modeling temporal events in large scale networks [153, 173] and other work that uses temporal link and attribute patterns to improve predictive models [111]. In addition, there is work on identifying clusters in dynamic data [169, 174] but these methods focus on discovering underlying communities over time—sets of nodes that are densely connected together. In contrast, we are interested in uncovering the *behavioral* patterns of nodes in the graph and modeling how those patterns change over time.

The recent work on dynamic mixed-membership stochastic block models (dMMSB: [175, 176]), is to our knowledge, one of the only methods suitable for modeling node-centric properties over time. The dMMSB model identifies groups of nodes with similar patterns of linkage and characterizes how group memberships change over time. However, dMMSB assumes a specific parametric form where the groups are defined through linkage to specific nodes (i.e., in particular types of groups) rather than more general forms of node behavior over dynamic node sets. More importantly the dMMSB estimation algorithm is not scalable, which makes the method unsuitable for analysis of large graphs.

In this work, we aim to develop a descriptive model to answer the following questions for dynamic network datasets:

- **Identify dynamic patterns in node behavior.** What types of high level temporal patterns and trends do the data exhibit? Are behaviors cyclical or predictable? Do nodes have different behavioral patterns?

- **Predict future structural changes.** Can we predict when a node's role will change (e.g., a node with high in-degree transitions to a node with high betweenness)? Is the overall structure of the graph becoming more or less predictable over time?

- **Detect unusual transitions in behavior.** Are there nodes and/or points in time with significantly different behavioral patterns?

To facilitate the investigation of these questions, we propose to model node "roles" and how they change over time. Informally, *roles* can be viewed as sets of nodes that are more structurally similar to nodes inside the set than outside whereas communities are sets of nodes with more connections inside the set than outside. Specifically, to focus on *node behavior* (rather than the complimentary concept of community finding) we use non-parametric feature-based roles.

Using these non-parametric roles, which can generalize to new unseen nodes, we propose a novel *dynamic behavioral mixed-membership model* (DRMM) suitable for large, unbounded, time-evolving networks. The DRMM discovers features (i.e., using the graph and intrinsic attributes), extracts these features for all timesteps, and automatically learns behavioral "roles" for nodes at each timestep. The number of behavioral roles are selected automatically using MDL or AIC. Afterwards, we learn behavior transition matrices for each node (i.e., given a node role r_i, what is the probability of transitioning to r_j at the next point in time).

Our proposed DRMM technique allows us to investigate the properties of temporal networks and understand both global and local behaviors, detect anomalies, as well as predict future structural changes. The main strengths of the approach include:

- ▷ **Automatic.** The algorithm doesn't require user-defined parameters.
- ▷ **Scalable.** The learning algorithm is linear in the number of edges in the time-interval under consideration. It is also easily parallelizable as features, roles, transition models can be learned independently at each time.
- ▷ **Non-parametric and data-driven.** The model structure (i.e., number of parameters) and more generally the parameterization depends on the properties of the time-evolving network.
- ▷ **Interpretable and intuitive.** The DRMM is based on an intuitive behavioral representation (structural properties) of the network and individual nodes. It identifies explainable patterns, trends, and aids in understanding the underlying dynamic process.
- ▷ **Flexible.** The definition of behavior in our model can be tuned for specific applications. The algorithm is applicable for all types of time-evolving networks.

We demonstrate the application our model on several real world datasets, showing that it both accurately predicts future structural changes as well as identifying interesting temporal patterns and anomalies. We discuss the scalability of the approach and notably we apply the DRMM to networks with up to 300,000 nodes and 4 million edges—datasets that are orders of magnitude larger than could be modeled with dMMSBs.

4.2 Background

Roles have been mainly of interest to sociologists [177, 178], but more recently, roles have shown to be a valuable general graph concept that are naturally complimentary [179–181, 181–184] to the widely studied problem of community identification (for community detection, see [167–172, 174, 185–187]). See Figure 4.1 for an illustration. Communities are sets of vertices with more connections inside the set than outside.

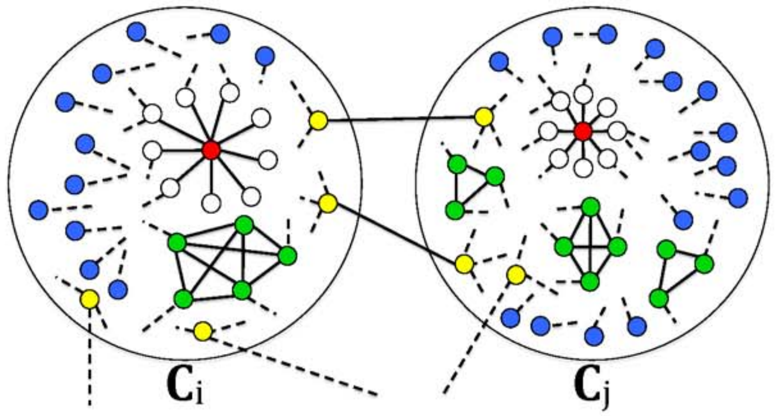


Fig. 4.1.: Roles and communities are *complimentary* graph concepts. Roles are based on structural similarity and can be globally distributed across the graph, whereas communities are defined by cohesion, density, and proximity.

Modularity maximization is one example of an objective function that formalizes this concept. Roles are sets of vertices that are more structurally similar to nodes inside the set than outside. Roles represent node-centric connectivity patterns such as near-cliques or nodes that act as bridges to different communities. Intuitively, two nodes belong to the same role if they are structurally equivalent (or more practically, similar).

Our work makes several significant contributions over the prior work on roles. In particular, we introduce the concept of dynamic roles using a *feature-based representation* and describe a general and flexible framework for computing them. Traditionally, the notion of roles have been defined based on graph equivalences such as structural [188], automorphic [189], regular [190, 191], and stochastic equivalence [179–182, 189, 192]. Methods based on these notions are inefficient and not flexible for practical applications. Therefore, we reinterpret roles using the fundamental notion of similarity on a feature-based representation and introduce a flexible framework for roles that consists of two main steps: (1) feature construction and (2) role assignment. This reinterpretation gives rise to a large family of potential methods for computing dynamic roles which many are fast, scalable, interpretable, and extremely flexible for application-specific roles. In addition, our work stands out from the earlier work in the following ways.

First, the previous work has mainly investigated static roles whereas we primarily focus on dynamic networks. Second, the previous approaches were restricted to extremely small offline social networks [181–184]. Third, we investigate the notion of roles using a feature-based representation, which allows us to utilize some key ideas from Chapter 2 and Chapter 3. Finally, roles has been traditionally defined using only the graph structure, whereas our feature-based representation naturally allows us to incorporate additional features.

4.3 Framework for Dynamic Feature-based Roles

Our goal is to model the behavioral roles of nodes and their evolution over time. Given a sequence of network snapshots (graphs and attributes), we propose the Dynamic Role Mixed Membership Model (DRMM) consisting of (1) automatically learning a set of representative features, (2) extracting features from each graph, (3) discovering behavioral roles (4) iteratively extracting these roles from the sequence of network snapshots over time and (5) learning a predictive model of how these behaviors change over time. As an aside, let us note that DRMM is a *scalable general framework* for analyzing temporal behavior as the model components can be replaced by others and each component can be appropriately tuned for any application (e.g., for the feature set, any feature construction system from [2] can conceivably be used).

The framework has two fundamental parts. In this section, we focus on discovering the feature-based representation and learning dynamic roles whereas Section 4.4 proposes global and local transition models that leverage temporal dependencies to accurately represent how the behavioral roles of the network and individual (local) nodes change over time.

4.3.1 Feature Discovery

The idea is to discover a set of underlying roles, which together describe the behaviors observed in the network, and then assign a probability distribution over these roles to each node in the network, which explain that node's observed behavior. Roles are extracted via a two-step process. The first step is to represent each active node in a given snapshot graph $\mathbf{G}_t$ using a set of representative features. For this task, we leverage [193]. The method constructs degree and egonet measures (in/out, weighted, ...), then aggregates these measures using sum (or mean) creating recursive features. After each aggregation step, correlated features are pruned using logarithmic binning. The aggregation proceeds recursively, until there are no new features. Formally, we discover a set of features at time t denoted $\mathbf{F}_t$ such that $\mathbf{F}_t$ is an $n_t \times f$ matrix where n_t is the number of active nodes and f is the number of features learned from the snapshot graph $\mathbf{G}_t$. The features are extracted for each network snapshot resulting in a sequence of node-feature matrices, denoted $\{\mathbf{F}_t : t = 1, ..., t_{max}\}$.

Table 4.1.: Dataset characteristics. The number of learned features and roles provide intuition about the underlying generative process and also indicates the amount of complexity present in the network.

Dataset	**Feat.**	**Roles**	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	**length**
Twitter	1325	12	310K	4M	41	1 day
Twitter-Cop	150	5	8.5K	27.8K	112	3 hours
Facebook	161	9	46.9K	183K	18	1 day
Email-Univ	652	10	116K	1.2M	50	60 min
Network-Tra	268	11	183K	1.6M	49	15 min
Internet AS	30	2	37.6K	505K	28	3 months
Enron	173	6	151	50.5K	82	2 weeks
IMDB	45	3	21.2K	296K	28	1 year
Reality	99	5	97	31.6K	46	1 month

4.3.2 Dynamic Role Learning

The next step is to automatically discover groups of nodes (representing common patterns of behavior) based on their time series of features. In this work, we use Non-negative Matrix Factorization (NMF) to learn roles over a time series of features from the dynamic attributed graphs, though other low-rank approximation methods such as SVD may also be used in the same fashion. More specifically, the following components of DRMM's role learning may be fine-tuned or customized for a specific application, including the (i) similarity/objective function (i.e., Frobenius norm, KL-divergence), (ii) solver/algorithm (i.e., Multiplicative update, CCD, ANLS), and (iii) regularization terms if warranted (i.e., sparsity constraints), among others. Future work will evaluate the tradeoffs between these components.

Role Learning and Assignment Given a sequence of node-feature matrices, we generate a rank-r approximation $\mathbf{Z}_t\mathbf{H} \approx \mathbf{F}_t$ where each row of $\mathbf{Z}_t \in \mathbb{R}^{n \times r}$ represents a node's membership in each role and each column of $\mathbf{H} \in \mathbb{R}^{r \times f}$ represents how membership of a specific role contributes to estimated feature values. For constructing the "closest" rank-r approximation we use NMF (multiplicative update method) because of interpretability and efficiency, though any other method for constructing such an approximation may be used instead (SVD, spectral decomposition). More formally, given a non-negative matrix $\mathbf{F}_t \in \mathbb{R}^{n_t \times f}$ and a positive integer $r < \min(n_t, f)$, find non-negative matrices $\mathbf{Z}_t \in \mathbb{R}^{n_t \times r}$ and $\mathbf{H} \in \mathbb{R}^{r \times f}$ that minimizes the functional,

$$f(\mathbf{Z}_t, \mathbf{H}) = \frac{1}{2}||\mathbf{F}_t - \mathbf{Z}_t\mathbf{H}||_F^2$$

We use an iterative method to estimate the node-role memberships for each network snapshot $\mathbf{Z} = \{\mathbf{Z}_t : t = 1, ..., t_{max}\}$ given $\mathbf{H}$ and $\mathbf{F} = \{\mathbf{F}_t : t = 1, ..., t_{max}\}$ using NMF. Afterwards, we have a sequence of matrices $\mathbf{Z}_1, \mathbf{Z}_2, ..., \mathbf{Z}_t, ..., \mathbf{Z}_{t_{max}}$ where each active node at time t is represented with their current role memberships.

To measure the difference between the node feature matrix $\mathbf{F}$ and the approximation $\mathbf{ZH}$, we define a general similarity measure $D(\mathbf{F}||\mathbf{ZH})$ to be used as an objective function for the optimization model. While there are many options for the objective functions (e.g., $\ell_p - norm$ Minkowski family, β-divergence, Bregman divergence), we primarily focus on Frobenius norm and Generalized KL-divergence:

$$D_{Fro}(\mathbf{F}||\mathbf{ZH}) = \frac{1}{2}||\mathbf{F} - \mathbf{ZH}||_{Fro} = \frac{1}{2}\sum_{ij}\Big(\mathbf{F}_{ij} - [\mathbf{ZH}]_{ij}\Big)^2 \tag{4.1}$$

$$D_{KL}(\mathbf{F}||\mathbf{ZH}) = \sum_{ij}\Big(\mathbf{F}_{ij}\log\frac{\mathbf{F}_{ij}}{[\mathbf{ZH}]_{ij}} - \mathbf{F}_{ij} + [\mathbf{ZH}]_{ij}\Big) \tag{4.2}$$

Both 4.1 and 4.2 are lower bounded by zero which occurs if and only if $\mathbf{F} = \mathbf{ZH}$. Using $D_{Fro}(\mathbf{F}||\mathbf{ZH})$ and $D_{KL}(\mathbf{F}||\mathbf{ZH})$ as a basis, we consider two alternate formulations of NMF as optimization problems: Minimize $D_{Fro}(\mathbf{F}||\mathbf{ZH})$ with respect to $\mathbf{Z}$ and $\mathbf{H}$, subject to the non-negativity constraints $\mathbf{Z}, \mathbf{H} \geq 0$. Similarly for $D_{KL}(\mathbf{F}||\mathbf{ZH})$.

Regularization terms may also be added for additional flexibility. For simplicity, these can be unified under the following extended objective function:

$$D_U(\mathbf{F}||\mathbf{ZH}) = D(\mathbf{F}||\mathbf{ZH}) + \alpha J_1(\mathbf{Z}) + \beta J_2(\mathbf{H})$$

where $\alpha J_1(\mathbf{Z})$ and $\beta J_2(\mathbf{H})$ are penalty terms whereas α and β are the regularization parameters balancing the tradeoffs between the goodness of fit and the constraints. For instance, to achieve better sparsity we can use the L1-norm penalty as follows:

$$\min_{\mathbf{Z},\mathbf{H}\geq 0} \frac{1}{2}||\mathbf{F} - \mathbf{ZH}||_{Fro} + \Big(\alpha\sum_{ir}\mathbf{Z}_{ir} + \beta\sum_{rj}\mathbf{H}_{rj}\Big)$$

Instead of the L1-norm penalty on $\mathbf{Z}$ and $\mathbf{H}$, one may also replace $\sum_{ir}\mathbf{Z}_{ir}$ and $\sum_{rj}\mathbf{H}_{rj}$ with the Frobenius norm of $\mathbf{Z}$ and $\mathbf{H}$, respectively.

Likewise, DRMM may also leverage a variety of solvers/algorithms for the above objective functions such as Multiplicative update, Cyclic Coordinate Descent (CCD), Alternating Non-negative Least Squares (ANLS), among many others. Given the objective function $||\mathbf{F} - \mathbf{ZH}||_{Fro}$, the multiplicative update rules are:

$$\mathbf{H}_{kj} \leftarrow \mathbf{H}_{kj}\frac{(\mathbf{Z}^T\mathbf{F})_{kj}}{((\mathbf{Z}^T\mathbf{Z})\mathbf{H})_{kj}}, \text{ for } 1 \leq k \leq r \text{ and } 1 \leq j \leq f$$

Table 4.2.: Feature validation. Validating DRMM's ability to distinguish patterns. Note **C** is row-normalized.

Features	S-CENTER	S-EDGE	BRIDGE	CLIQUE
S-CENTER	**0.08**	0.25	0.34	0.33
S-EDGE	0.27	**0.11**	0.25	0.37
BRIDGE	0.29	0.20	**0.17**	0.34
CLIQUE	0.24	0.24	0.29	**0.23**

Table 4.3.: Role validation. Validating DRMM's ability to distinguish patterns. Note **C** is row-normalized.

Roles	S-CENTER	S-EDGE	BRIDGE	CLIQUE
S-CENTER	**0.07**	0.25	0.33	0.35
S-EDGE	0.28	**0.10**	0.22	0.40
BRIDGE	0.29	0.18	**0.16**	0.37
CLIQUE	0.24	0.25	0.29	**0.22**

$$\mathbf{Z}_{ik} \leftarrow \mathbf{Z}_{ik} \frac{(\mathbf{F}\mathbf{Z}^T)_{ik}}{(\mathbf{Z}(\mathbf{H}^T\mathbf{H}))_{ik}}, \text{ for } 1 \leq k \leq r \text{ and } 1 \leq i \leq n$$

We have tested a variety of these configurations for DRMM and use NMF with multiplicative update and no regularization, unless otherwise mentioned.

Model Selection: The number of structural roles r is automatically selected using Minimum Description Length (MDL) criterion. However, AIC or any model selection may be used instead. Intuitively, learning more roles, increases model complexity, but decreases the amount of errors. Conversely, learning less roles, decreases model complexity, but increases the amount of errors. In this way, MDL selects the number of behavioral roles r such that the model complexity (number of bits) and model errors are balanced. Naturally, the best model minimizes, $\textit{number of bits} + \textit{errors}$.

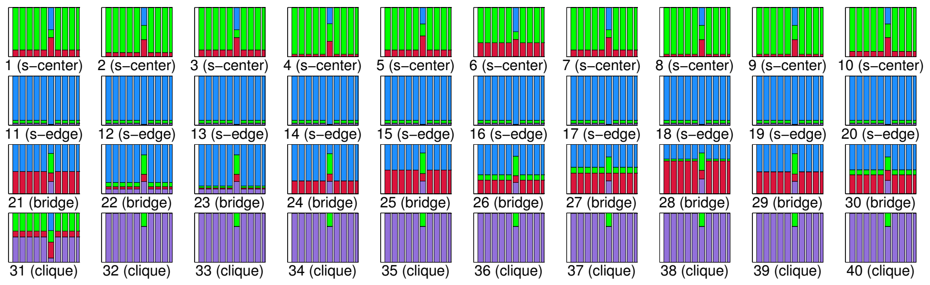


Fig. 4.2.: The pattern of each node is listed below the mixed-membership plot whereas the colors represent roles learned from our model. For simplicity, the node's pattern-type is kept stable over time. Strikingly, the DRMM clearly reveals the underlying patterns of the nodes as each pattern has a distinct signature in terms of the role distribution. For instance, the blue role of a bridge node indicates the local similarity with that of a star-edge node (low degree,...) while the red role captures the bridges more global and intrinsic property of acting as a backbone for the other nodes. The other patterns are even more straightforward to interpret. We also inject a type of global anomaly at $t = 6$ (bridges connecting to each other) which is clearly revealed as such in the plots.

4.4 Role Transition Models

How can we model the time series of behavioral roles and their changes over time? How can we do this for both the network as a whole and at the level of individual nodes? Section 4.4.1 introduces the model representation for discovering how roles change, then Section 4.4.2 discusses the use of this representation for modeling role transitions at the global network level (global role transition models) as well as at the individual node level (local role transition models). In Sections 4.4.3-4.4.4, we propose model variations that incorporate the temporal information in different ways. Finally, closing remarks are made in Section 4.4.5.

4.4.1 Model Representation

Given a sequence of dynamic behaviors $\mathbf{Z} = \{\mathbf{Z}_t : t = 1, ..., t_{max}\}$, we can learn a model of how behavior in our network changes over time. More formally, given two behavioral snapshots, $\mathbf{Z}_{t-1}$ and $\mathbf{Z}_t$, we learn a transition matrix $\mathbf{P} \in \mathbb{R}^{r \times r}$ that approximates the change in behavior from time $t-1$ to t. The transition matrix $\mathbf{P}$ represents how likely a node is to transition from role r_i to role r_j for that particular time interval:

$$\mathbf{P} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,r} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,r} \\ \vdots & \cdots & \ddots & \cdots \\ p_{r,1} & p_{r,2} & \cdots & p_{r,r} \end{bmatrix}$$

where $\mathbf{P}$ is estimated using NMF such that $\mathbf{Z}_{t-1}\mathbf{P} \approx \mathbf{Z}_t$. In the simple form of the model presented above, we learn $\mathbf{P}$ using only a single transition (i.e., $t-1$ to t). However, we also propose variations that leverage more available data by considering multiple transitions (*stacked model*) or that smooth over a sequence of transitions using kernel functions (*summary model*). We discuss these in detail next.

4.4.2 From Global to Local Node Transition Models

Using one of the above representations for the transition model, we then propose three types of transition models. Each of the three types below have various tradeoffs. Learning a node transition model is most accurate (when enough data is available), however, it is also costly to learn each model independently. The global graph transition model is efficient to compute, but less meaningful. The clustered transition model balances these two tradeoffs (speed vs. accuracy) by clustering the vertices using the evolving mixed-memberships into a small number of groups.

Local Node Transition Models A transition model is learned for each local node independently and thus we use only the nodes past mixed memberships for learning. For instance, the stacked class of models uses the training examples from the k previous timesteps such that,

$$\begin{bmatrix} \mathbf{z}_i(k-1) & \cdots & \mathbf{z}_i(t-1) \end{bmatrix} \mathbf{P}_i(t) \approx \begin{bmatrix} \mathbf{z}_i(k) & \cdots & \mathbf{z}_i(t) \end{bmatrix}$$

where $\mathbf{P}_i(t) \in \mathbb{R}^{r \times r}$ is a role transition matrix for node i and the rows sum to one. Using this learned model at time t, we can predict the future roles at time $t+1$ using $\mathbf{z}_i(t)\mathbf{P}_i(t)$. Let $\hat{\mathbf{z}}_i(t+1)$ be the predicted roles from $\mathbf{z}_i(t)\mathbf{P}_i(t)$, then the goal is that $\hat{\mathbf{z}}_i(t+1) \approx \mathbf{z}_i(t+1)$ measured by some objective/error function (i.e., Frobenius loss).

These models capture node-level transitions and thus depend only on their past structural behavior whereas the global models rely on the overall transition behavior. However, these models can have a large variance due to limited training data (past observations). The other issue is that in some datasets, many nodes are ianctive for extended periods and thus in many cases there may only be a few observations for learning. One way of dealing with this may be to use the observed examples to find nodes that have similar behavior and use their past examples for learning. The other way would be to use the neighbors of that node as a good indicator of their future behavior. The assumption here is that my structural behvaior is influenced by my immediate neighbors.

Global Graph Transition Model A single global transition model using the evolving mixed-memberships from all the vertices is learned and used for various tasks. These models capture the overall role transition/dynamic structural changes, and provide a good summary of the overall dynamic role changes/behavior. Useful for detecting anomalies at the network level. However, these models are unable to detect localized node specific anomalous behavior and role transitions and are less useful for prediction. In addition, these models utilize the past data/observations and thus there are plenty observations for learning.

Vertices with similar mixed-memberships over time are grouped together using a clustering technique. A transition model is learned independently for each group of vertices.

4.4.3 Stacked Transition Model

The stacked model uses the training examples from the k previous timesteps. More formally, the stacked model is defined as,

$$\begin{bmatrix} \mathbf{Z}_{t-1} \\ \mathbf{Z}_{t-2} \\ \vdots \\ \mathbf{Z}_{k-1} \end{bmatrix} \mathbf{P} \approx \begin{bmatrix} \mathbf{Z}_{t} \\ \mathbf{Z}_{t-1} \\ \vdots \\ \mathbf{Z}_{k} \end{bmatrix}$$

where $k = t - w$ and w is the window size; typically $w = 10$. Let us denote the stacked behavioral snapshots as $\mathbf{Z}_{k:t}$ where $k : t$ represents all the training examples from timestep k to timestep t.

4.4.4 Summary Transition Model

This class of models uses k previous timesteps to weight the training examples at time t using some kernel function. The exponential decay and linear kernels are used in this work. The temporal weights can be viewed as probabilities that a node behavior is still active at the current time step t, given that it was observed at time $(t - k)$. We define the summary behavioral snapshot $\mathbf{Z}_{S(t)}$ as a weighted sum of the temporal role-memberships up to time t as follows, $\mathbf{Z}_{S(t)} = \alpha_1 \mathbf{Z}_k + ... + \alpha_{w-1}\mathbf{Z}_{t-1} + \alpha_w \mathbf{Z}_t = \sum_{i=k}^{t} K(\mathbf{Z}_i; t, \theta)$ where α determines the contribution of each snapshot in the summary model.

In addition to exponential and linear kernels, we experimented with the inverse linear and also tried various θ values. Overall, we found the linear kernel (and exponential) to be the most accurate with $\theta = 0.7$. Nevertheless, the optimal θ will depend on the type of dynamic network and the volatility.

4.4.5 Remarks and Discussion

For each type of transition model (e.g., stacked or summary), we may learn a *global transition model* that describes how the behavior of the network as a whole changes over time or we may learn a *local transition model* for each individual node. The local transition model describes more precisely how the behavioral roles of that individual node change over time. We can estimate the local transition model for a node i as $\mathbf{Z}_{t-1}^{(i)}\mathbf{P}^{(i)} \approx \mathbf{Z}_t^{(i)}$ using NMF. The global transition model for the network is estimated in exactly the same way as described above in §4.4.

We have found the summary model to be the best performer for prediction tasks because of its ability to smooth over multiple timesteps. However, for precisely this reason, the summary model is more difficult to interpret. Therefore, we use the summary model for prediction tasks and the stacked representation for data analysis tasks, due to its interpretability. Let us note that to achieve better accuracy in predictions, one may also estimate local transition models for each node and use these for predicting a node's future role memberships. All of these options make our model flexible for use in a variety of applications. We also experimented with other variants of the DRMM transition model, including a stacked-summary hybrid and multi-state models, which make an explicit distinction between transitions from activate states and transitions from inactive states. However, we opted in favor of the simpler stacked and summary models because none of these other models provided an obvious advantage.

While our model currently assumes the role definitions are somewhat stationary, we have found that these roles generalize and can even be applied across different networks. Nevertheless, to remove this assumption, we could simply track the loss over time and recompute the roles when it surpasses some threshold.

4.5 Experiments

This section first validates DRMM using synthetic data, and then explores the effectiveness of DRMM for predicting future structural patterns.

4.5.1 Datasets and Structural Analysis

We apply the DRMM model using a variety of dynamic networks from different domains [194]. See Table 4.1 for details. Interestingly, we find a relationship between the complexity of DRMM and the complexity present in the graph. This is clearly clearly shown in Table 4.1 by analyzing simple measures generated from the DRMM behavioral representation such as the number of learned features and the number of roles. For instance, the Internet AS topology has some hierarchical structure or recurring patterns of connectivity among ISPs and therefore our model discovers only 30 features. This is in contrast to networks with more complex patterns of connectivity such as twitter and other transaction networks like the email network. In these cases, the links are instantaneous and might only last for some duration of time, thus making more complex structures more likely.

4.5.2 Model Validation using Ground-truth

In this section, we demonstrate the ability of DRMM to distinguish between common graph patterns (and consequently recover the synthetic roles). For this task, we design a simple graph generator that constructs graphs probabilistically with four main patterns: 'center of a star', 'edge of a star', bridge nodes (connecting stars/cliques), and clique nodes. After constructing the graph, we validate that the DRMM model captured these patterns by measuring if the extracted features and roles represent the known probabilistic patterns. We do this by computing the pairwise euclidean distance matrix $\mathbf{D}$ using the initial feature matrix $\mathbf{F}$ (and role-membership matrix

$\mathbf{Z}$). Let r_i denote the actual pattern of node i, and $\mathcal{P} = \{(i,j) | r_i = p,\ r_j = q\}$ then $C_{p,q} = \sum_{(i,j)\in\mathcal{P}} D_{i,j}$.

Clearly, the roles and features from nodes of the same pattern are shown to be more similar than the others (smaller values along the diagonal). See Table 4.2–4.3. Additionally, the patterns that are structurally similar to one another are represented as such by our model (star-center and clique). In Fig. 4.2, we visualize the mixed-memberships of 10 randomly chosen nodes from each pattern-type. Each pattern has a distinct and consistent signature in terms of the role distribution.

4.5.3 Interpretation and Analysis

We start with an illustrative example of applying to a large IP trace network, shown in Figure 4.3. We first plot the time-evolving mixed-memberships from four nodes shown in Figure 4.7 and then visualize their corresponding transition models in Figure 4.3(a). In the time-evolving mixed-memberships, inactivity is represented by white bars whereas in the transition models inactivity corresponds to the last row/column. The transition models are learned using the stacked representation which aids in the understandability and interpretation of the roles and their modeled transitions.

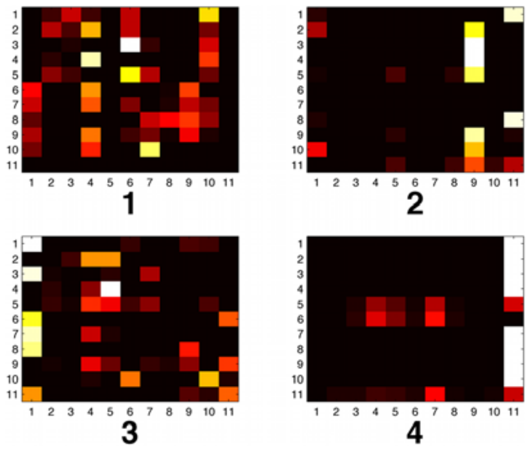
The time-evolving mixed-memberships for each of the four example nodes in Figure 4.7 show distinct patterns from one another which are easy to identify. The four patterns represented by these nodes can be classified as having the following patterns of structural behavior,

- **Structural Stability.** This node's structural behavior (and communication pattern) doesn't is relatively stable over the time.
- **Homogeneous.** The node for the most part takes on a single behavioral role.

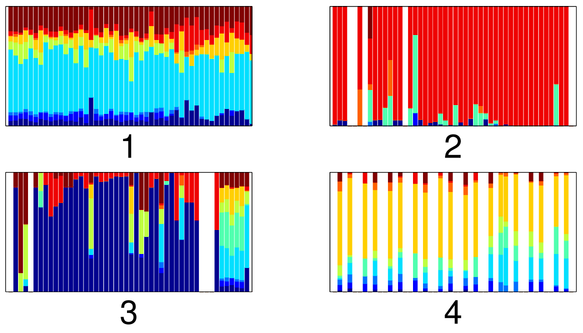
- **Abrupt transition.** Their structural behavior changes abruptly. In the IP-network, it could be that the IP was released and someone was assigned it or perhaps that the machine was compromised and began acting maliciously.
- **Periodic activity.** The node has periodic activity, but maintains similar structural behavior. In the case of the IP-communication network, this machine could be infected and every 30 minutes sends out a communication to the master indicating it's connected and "listening".

For the four example nodes, we show their transition models in Figure 4.3(a). The transition models represent the probability of transitioning or taking on the structural behavior of role j given that your current role (or main role) is role i. For instance, node 2 homogeneously takes on the red role over time as discussed previously. From Figure 4.3(c), we see that the red role is "role 9", and looking back at the node's learned transition model, we find that column 9 contains most of the mass, which represents that their is a high probability of transitioning from any other role to the red role. As shown in the mixed-memberships over time, this is exactly what is expected. As another example, we find that node 4 usually transitions from a mix of active roles to the inactive role (i.e., the inactive role is represented by column/row eleven). Therefore, we would expect our learned transition model to capture this by placing most of the mass on the last column, representing the probability of going inactive after having a mix of active roles in the previous timestep, which is exactly what we see in the fourth transition model.

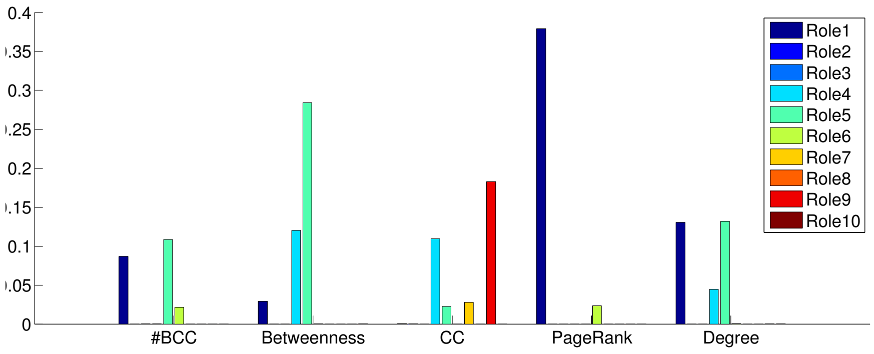
Instead of providing subjective or anecdotal evidence for what the roles represent, we interpret the roles of the DRMM with respect to well-known node measurements (e.g., degree, clustering coefficient, betweenness,...). We extend the analytical tools from [195] for use in interpreting the role dynamics. The first technique interprets the roles using the dynamic node-role memberships $\mathbf{Z}_t$ and a node measure matrix $\mathbf{M}_t \in \mathbb{R}^{n \times m}$ to compute a non-negative matrix $\mathbf{E}_t$ such that $\mathbf{Z}_t \mathbf{E}_t \approx \mathbf{M}_t$. The node measurements used are betweenness, biconnected components, PageRank, clustering



(a) Transition Models



(b) Time-evolving Mixed-Memberships

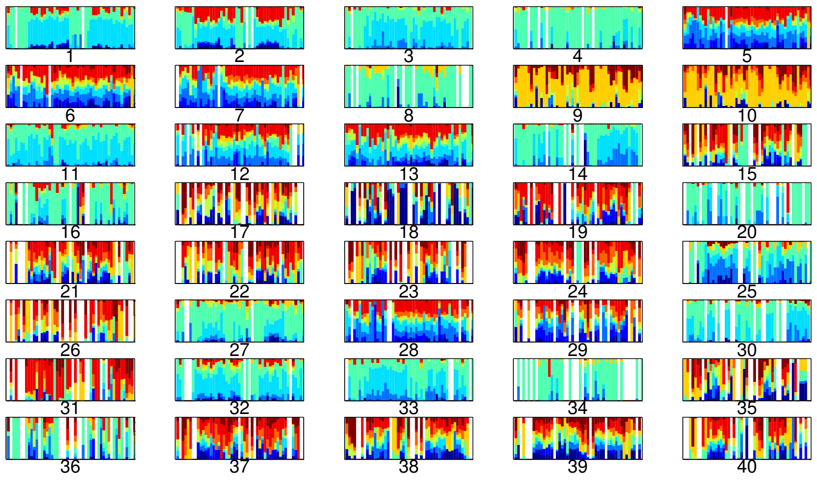


(c) Role Interpretation

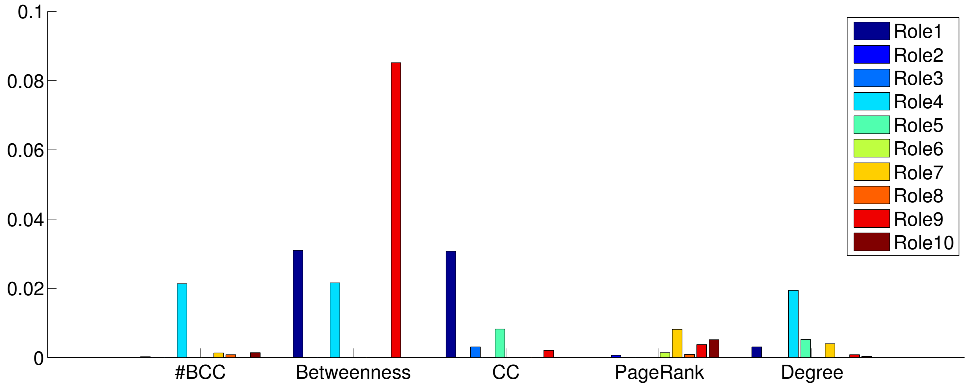
Fig. 4.3.: The DRMM transition model effectively captures the diverse temporal behavior of hosts in a computer network. (a) Transition matrices for 4 hosts. The y-axis represents the role the node transitions from, the x-axis is the role we transition to. Inactivity is represented by the last row/column. (b) Corresponding role-memberships over time. The x-axis represents time while the y-axis represents the role distribution at each point in time. Each distinct color represents a learned role. (c) Characteristics of individual roles.

coefficient, and degree. The matrix $\mathbf{E}_t$ represents the contributions of the node measures to the roles at time t. We report average contributions over time.

Figure 4.3(c) shows this quantitative interpretation of roles for the IP network. Intuitively, the first role represents nodes with high PageRank, while role five represents nodes with high betweenness, whereas role nine represents nodes with large clustering



(a) Time-evolving Mixed-Memberships (Email)



(b) Email Role Interpretation

Fig. 4.4.: The DRMM model allows us to uncover patterns of behavior in an email network. (a) evolving memberships for a group of nodes and (b) the characteristics associated with the roles.

coefficient. The other roles represent more specialized structural motifs that were not captured by the set of traditional measures used for interpretation.

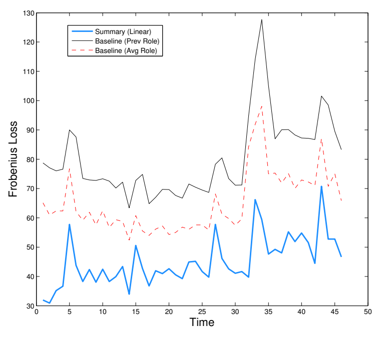
The DRMM can be used to understand the temporal behavior across a variety of time-evolving networks. Figure 4.4 shows another example for an email communication network. Just as before, we can identify significant trends and patterns and interpret these using the role interpretations from Figure 4.4(b). One notable behavioral pattern in the email communications is that most users have a set of roles for the daytime and a different set for night. Intuitively, one set of roles is work related and the other is more personal/family related (e.g., nodes 1, 2, among others). We also find nodes that have inconsistent or unstable behavior over the time, such as 17, 18, 19, among others. Additionally, some nodes have relatively stable structural behavior over the two days, such as node 4. This is also unusual, since one might expect a user's behavior to change from the work hours to the evening/night. However, users that are consistently dominated by multiple active roles are of importance (may serve in managerial or leadership roles), since they connect to groups of nodes with different types of structural patterns (see nodes 5-7).

4.5.4 Predicting Future Behavior

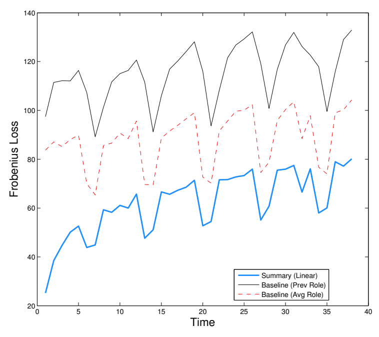
In this section, we further validate the utility of the DRMM model by demonstrating its ability to predict the future behavior of nodes. This could be useful for optimizing caches on the Web, or for improving dynamic social recommendation systems, among many others.

Models. The goal is to accurately predict $\mathbf{Z}_{t+1}$ given $\mathbf{Z}_{s(t)}$, the summary behavioral snapshot described in Section 4.4.4. Our primary means of predicting $\mathbf{Z}_{t+1}$ is using our DRMM summary transition model $\mathbf{P}$ as follows: $\hat{\mathbf{Z}}_{t+1} = \mathbf{Z}_t \, \mathbf{P}$. We compare this summary model to two sensible baselines: *PrevRole* and *AvgRole*. PrevRole simply assigns each node the role distribution from the previous time t. That is, $\hat{\mathbf{Z}}_{t+1} = \mathbf{Z}_t$. AvgRole assigns each node the average role distribution over all nodes at time t. The AvgRole model can be expressed as $\hat{\mathbf{Z}}_{t+1} = \mathbf{Z}_t \, \mathbf{P}_A$ where $\mathbf{P}_A$ is estimated from $\mathbf{Z}_t = [1] \, \mathbf{P}$. Essentially, PrevRole assumes node behavior does not change from each point in time to the next and AvgRole assumes that all nodes exhibit the average behavior of the network.

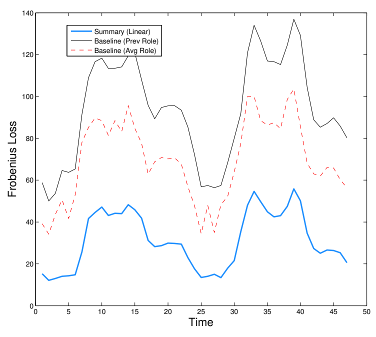
Evaluation We consider two strategies for evaluating our predictive models: (a) compare the predicted $\hat{\mathbf{Z}}_{t+1}$ to the true $\mathbf{Z}_{t+1}$ using a loss function (we use the Frobenious norm) and (b) Use $\hat{\mathbf{Z}}_{t+1}$ to predict the modal role of each node at time $t + 1$ and



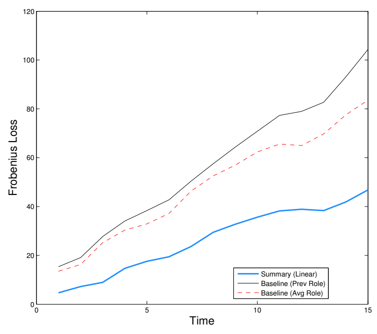
(a) IP-Trace



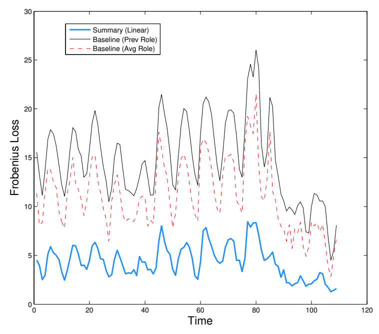
(b) Twitter Relationships



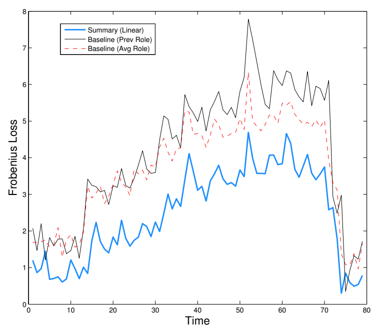
(c) Email Univ



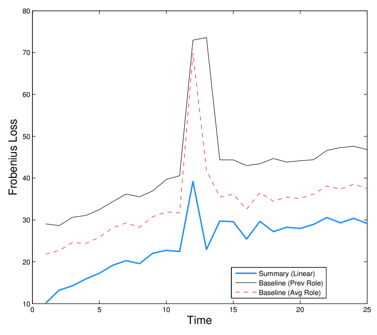
(d) Facebook



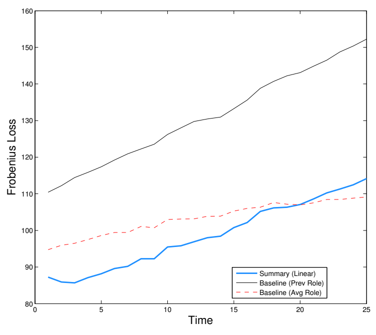
(e) Twitter Copenhagen



(f) Enron



(g) IMDB



(h) Internet AS

Fig. 4.5.: The DRMM transition model accurately predicts future behavior of individual nodes (i.e., mixed role membership) compared to sensible baseline models.

evaluate these predictions using a multi-class AUC (Area Under the ROC curve) measure. We describe each of these strategies more formally below.

Frobenious Loss: The goal here is to estimate $\mathbf{Z}_{t+1}$ as accurately as possible. The approximation error between the estimated node memberships $\hat{\mathbf{Z}}_{t+1} = \mathbf{Z}_t\mathbf{P}_{t+1}$ and the true node memberships $\mathbf{Z}_{t+1}$ is defined as $||\mathbf{Z}_{t+1} - \hat{\mathbf{Z}}_{t+1}||_F$

Structural Prediction with Multi-class AUC: This is a multi-class classification task where the true class label for node i is the modal role from the ith row of $\mathbf{Z}_{t+1}$ (i.e., the role with maximum membership for this node). The predicted class label for node i is the modal role from the ith row of $\hat{\mathbf{Z}}_{t+1}$.

We evaluate the predictions using a generalization of AUC extended for multi-class problems. In particular, we compute the AUC of all combinations of labels and take the mean (also known as Total AUC) [196]. The difficulty of the prediction task varies based on the number of roles discovered, complexity of the network evolution, and the type of time-evolving network (e.g., transactional vs. stable).

Results Figure 4.5 demonstrate that the DRMM summary transition model is an effective predictor across the range of experiments. With few exceptions, DRMM outperforms both baselines for all data sets and timesteps. This is even true for the more complex time-evolving networks such as Twitter, email, and the IP-traces, which are more transactional with rapidly evolving network structure. For brevity, some findings were omitted, for others see [194].

In addition to validating the DRMM model, both figures offer some interesting insights into the characteristics of time-evolving networks. For example, an increase in loss over time may indicate a ”concept drift” where behavior in the network has evolved to the point where the current roles can no longer adequately explain node behavior. This effect is seen most prominently in Figures 4.5(b), 4.5(d), 4.5(g) and 4.5(h). Interestingly, the drift we see in Figure 4.5(h) agrees with the current understanding that the underlying evolutionary process of the Internet AS is not constant, as was previously believed [197,198]. Most notably, there is recent evidence of the Internet topology transitioning from hierarchical to a flat topological structure [199]. Furthermore, the figures provide insights into behavioral anomalies, such as the spike we see in Figure 4.5(g). The spike in loss indicates the significant deviation of the node roles at a specific time.

Finally, in the large Twitter Relationships network, we see seasonality among the role transitions. In particular, we find that the users generally behave significantly different over the weekends, seen by the increase in loss on these days. Intuitively, we would expect users to be tweeting about different topics and using the system in different manner than they do during the work days. The Twitter Copenhagen network captures the more locally-temporal seasonality; that is users behave differently during the daytime and the nighttime hours.

4.5.5 Scalability and Complexity

Most importantly, our dynamic role model is linear in the number of edges. Let n be the number of nodes, f be the number of features, r be the number of roles and t be the number of timesteps. The feature discovery is $O(t(mf+nf^2))$ [193]. For the NMF step, we use the multiplicative update method which has worst case $O(tnfr)$. The transition models is $O(tnr^2)$ using the multiplicative update method. Thus, the running time of DRMM is linear in the number of edges, specifically, $O(t(mf + nfr + nr^2))$. The time-scale t is usually small compared to the edges (even when the time-scale corresponds to minutes or seconds in the IP-trace data). A more accurate bound can be stated in terms of the maximum number of edges at any given timestep.

Our model is capable of handling realistic networks such as social and technological networks consisting of millions of nodes and edges. This is in contrast to a similar dynamic mixed-membership models that have been recently proposed such as the dMMSB [175, 176]. These models are quadratic in the number of nodes and therefore unable to scale to the realistic networks with the number of edges in the millions. Furthermore, these models have been typically used for visualizing trivial sized networks of 18 nodes up to 1,000 nodes. This is in contrast to our work where we apply DRMM not only for visualizations, but for a variety of analysis tasks using large dynamic networks. Moreover, the dMMSB can handle 1,000 nodes in a day [176] (See page 30), while our model handles ≈8,000 nodes in 506.61 seconds (or 8 minutes and 26 seconds)

Table 4.4.: Performance analysis of the dynamic behavioral mixed-membership model. The dMMSB takes a day to handle 1,000 nodes [176], while our model takes only 8.44 minutes for 8,000 nodes.

Dataset	**Nodes**	**Edges**	**Performance**
Enron	151	50,572	117.51 sec
Twitter (Copen)	8,581	27,889	506.61 sec
Facebook	46,952	183,831	1,468.65 sec
Internet AS	37,632	505,772	1,922.85 sec
Network-Trace	183,389	1,631,824	16,138.71 sec

shown in Table 4.4. We provide performance results for other larger datasets of up to 183,389 nodes and 1,631,824 edges. In all cases, even for these large networks with over a million edges, our model takes less than a day to compute and the performance results show the linearity of our model in the number of edges. For the scalability experiments, we recorded the performance results using a commodity machine Intel Core i7 @2.7Ghz with 8Gb of memory.

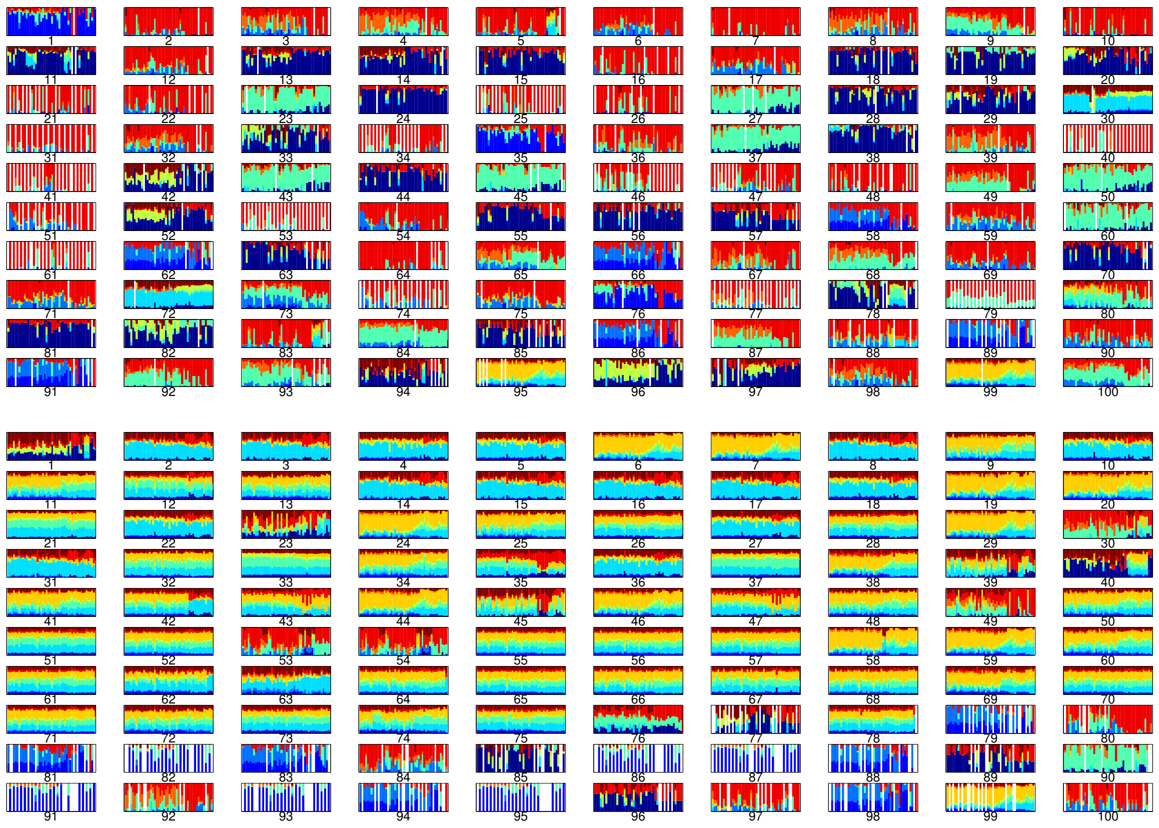


Fig. 4.6.: Evolution of node behaviors. The structural dynamics framework allows us to uncover important patterns of behavior in a large IP communications network. The roles are interpreted with respect to traditional structural properties and the role dynamics of 300 nodes are visualized where each color represents a specific behavioral role. The x-axis is time and the y-axis is the mixed-memberships.

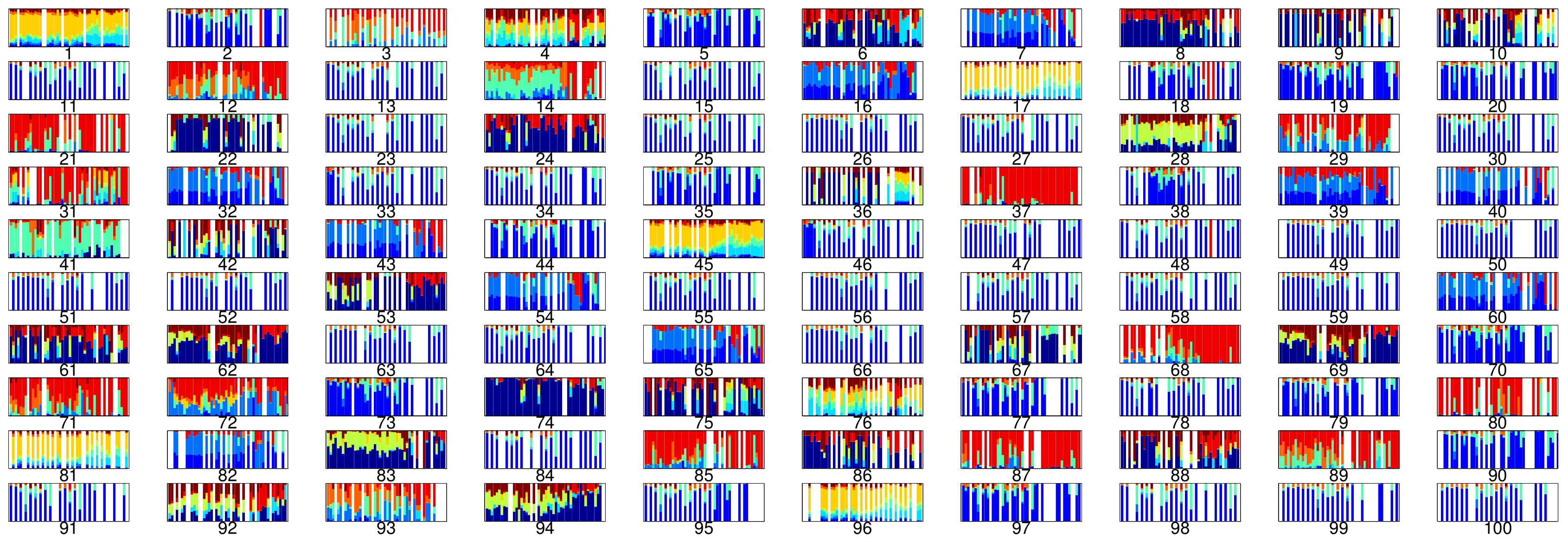


Fig. 4.7.: Evolution of node behaviors. The structural dynamics framework allows us to uncover important patterns of behavior in a large IP communications network. The roles are interpreted with respect to traditional structural properties and the role dynamics of 300 nodes are visualized where each color represents a specific behavioral role. The x-axis is time and the y-axis is the mixed-memberships.

4.6 Applications

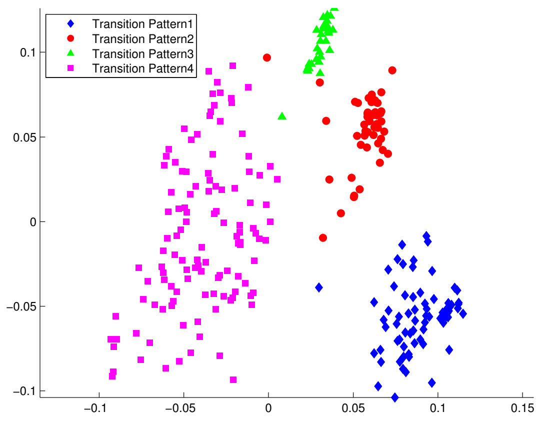
Besides prediction, the dynamic role mixed-membership model is useful for a variety of other applications. In this subsection, we explore using the dynamic role mixed-membership model and the time-series of dynamic roles for clustering/visualization of major trends in the roles, anomaly detection, pattern mining, and for graph similarity.

4.6.1 Clustering Temporal Behaviors

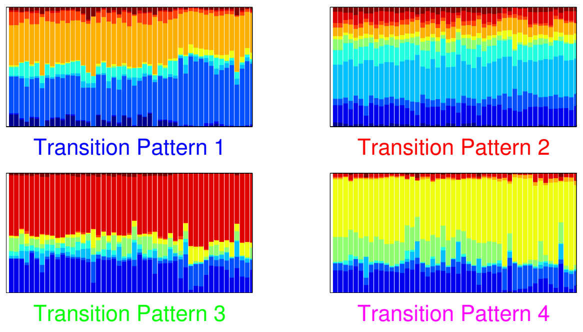
To show the patterns of the learned transition matrices, we cluster nodes based on their temporal behaviors. We find that this clustering reveals the underlying structural patterns of the evolving mixed-memberships. Formally, let $\mathbf{P}^{(i)}$ and $\mathbf{P}^{(j)}$ be the transition matrices of two nodes i and j. Then we create an $r \times r$ vector from each of the node transition models and define a similarity function between these vectors.

First we estimate a single transition model $\mathbf{P}$ for each node using the stacked model. We then compute an $n \times n$ similarity matrix using Frobenious loss between the transition matrices from the nodes. Next, we apply the classical k-means clustering algorithm to cluster the nodes by their transition matrices. Afterwards, we compute the closest rank-k approximation (k = 2 or 3) of the similarity matrix. The nodes are plotted using the low-rank approximation and labeled using the previous clustering algorithm. To reveal the structural transition pattern, we then compute the average dynamic mixed-membership for each cluster using only the nodes from that cluster.

This clustering method reveals common structural trends and patterns between nodes. For instance, this technique may group nodes together that share similar transitional patterns such as nodes with stable roles vs. nodes with more dynamic roles or nodes with high activity vs. nodes with low activity. An example is provided in Figure 4.8. For clarity in the visualization, we randomly selected a small subset of nodes from the 183,389 candidates and identified common transition patterns among them. The first visualization in Figure 4.8(a) identifies four distinct well-separated clusters of nodes with similar transition models. Figure 4.8(b) shows the average



(a) Transition Clustering



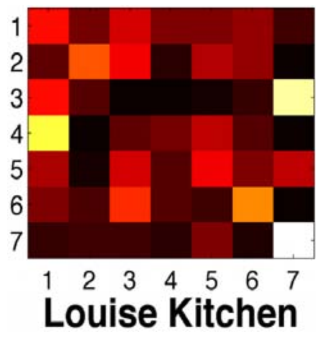
(b) Dynamic Mixed-Membership Patterns

Fig. 4.8.: The DRMM model provides an intuitive means of clustering nodes that exhibit similar patterns of behavior over time. (a) identifies four distinct clusters of nodes with similar transition patterns. (b) provides a sense of the behavior of each cluster in terms of the average role-membership over time. Again, we see that DRMM captures differences in both overall static behavior (i.e., the specific roles that dominate) and in patterns of how behaviors (i.e., roles) change over time.

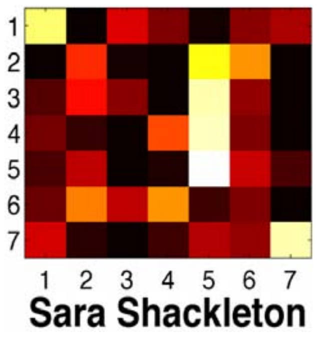
dynamic behavioral mixed-membership for each cluster. This visualization shows that each cluster represents a unique structural transition pattern between the nodes. The structural patterns can be interpreted using the previous role interpretation from Figure 4.3(c). This technique can be used for general exploratory analysis such as characterizing the patterns and trends of nodes or eventually used as a means to detect anomalies or nodes that do not fit any transition pattern.

4.6.2 Anomalous Dynamic Patterns

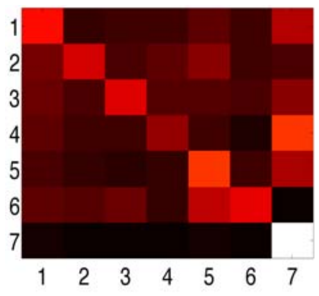
We further demonstrate the use of DRMMs for detecting anomalies in time-evolving networks. In particular, we formulate this problem with respect to identifying nodes that have *unusual structural transition patterns*. For instance, a node might transition from being a hub (i.e., a node with many people linking to it) to a node with low degree.



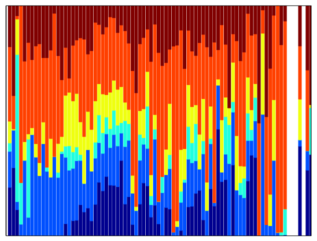
(a) T Model



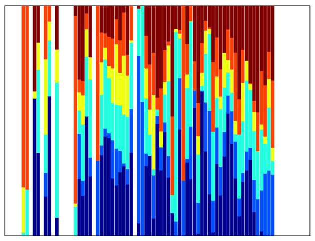
(b) T Model



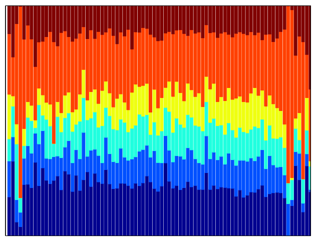
(c) Network T Mod



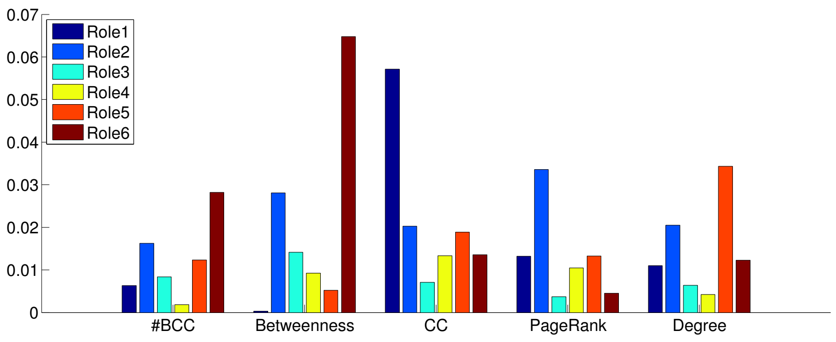
(d) Louise Kitchen



(e) Sara Shackleton



(f) Network



(g) Role Interpretation

Fig. 4.9.: The DRMM transition model provides an effective means of automatically discovering and visualizing nodes with anomalous temporal behavior. (a)-(b) are the transition models for two of the most anomalous nodes in the Enron email network compared to (c) the normal network transition model. (d)-(f) show the corresponding role memberships over time. (g) shows the characteristics of roles.

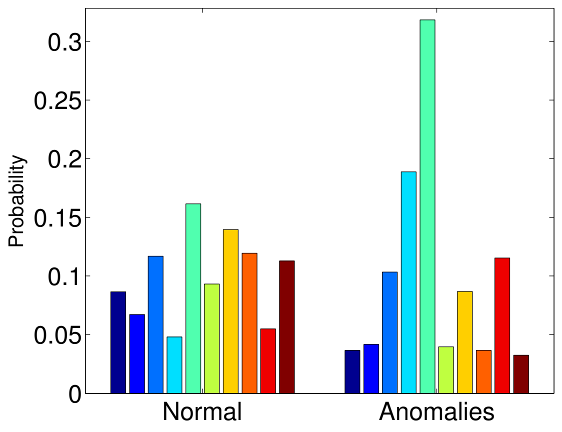
Node Transition Anomalies While there are many ways to define an anomaly detection technique with respect to the DRMM model, we propose an intuitive algorithm shown in Alg. 2 that uses a node's transition model for predicting the network memberships at $t + 1$. The anomaly score is the difference between the predicted network mixed-memberships and the ground-truth mixed-memberships. Therefore, the score represents the divergence of that nodes transitions from the entire network. One simple example is shown in Figure 4.9(a) where we find Louise Kitchen as having unusual behavioral transitions.

Time-varying Node Anomalies For detecting the specific time interval in which a node has unusual behavior we use the previous method with a few subtle distinctions. The global and node models are estimated at each *timestep* (in a sort of streaming fashion) using the stacked representation with a shorter window (for leveraging past training examples). The final result is a ranked list of potential node anomalies for each timestep, shown in Figure 4.11. The justification for such an approach is that nodes may become anomalous or have unusual behavior only for a specified time interval. In the case of IP-communications, it is unlikely for the behavior of an IP-address to remain unusual as IP-addresses are released/expires and users are assigned entirely new IP-addresses. These types of dynamic anomalies are shown in Figure 4.11.

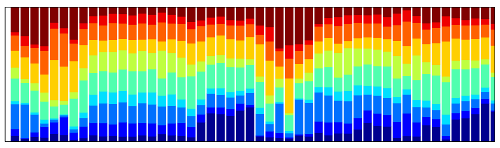
Algorithm 2 Anomalous Structural Transitions

```
1 procedure AnomTransition($\mathbf{Z} = \{\mathbf{Z}_t : t = 1, ..., t_{max}\}$ (evolving mixed-memberships))
2     for $i \leftarrow 1$ to $n$ do
3         $\mathbf{P}^{(i)} \in \mathbb{R}^{r \times r} \leftarrow NMF(\mathbf{Z}^{(i)}_{1:t-1}, \mathbf{Z}^{(i)}_{2:t})$
4         $\hat{\mathbf{Z}}_{t+1} = \mathbf{P}^{(i)} \cdot \mathbf{Z}_t$
5         $\mathbf{x}^{(i)} = \left\| \hat{\mathbf{Z}}_{t+1} - \mathbf{Z}_{t+1} \right\|_F$
```

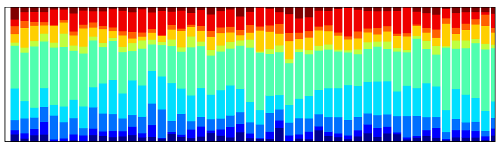
Anomalous Structural Transitions We first interpret the roles and their temporal variation quantitatively as shown in Figure 4.9(g) and then provide some simple examples of nodes that have unusual behavior transitions. Intuitively, the first role represents nodes with high clustering coefficient, the second role represents mainly nodes with high pagerank, while the third and fourth roles represent some type of combination of these properties indicating a more complex structural motif that is not sufficiently represented by the selected node metrics. However, the fifth role represents nodes with high degree and the sixth role represents nodes that are articulation



(a) Anomaly-Role Patterns



(b) Network Memberships

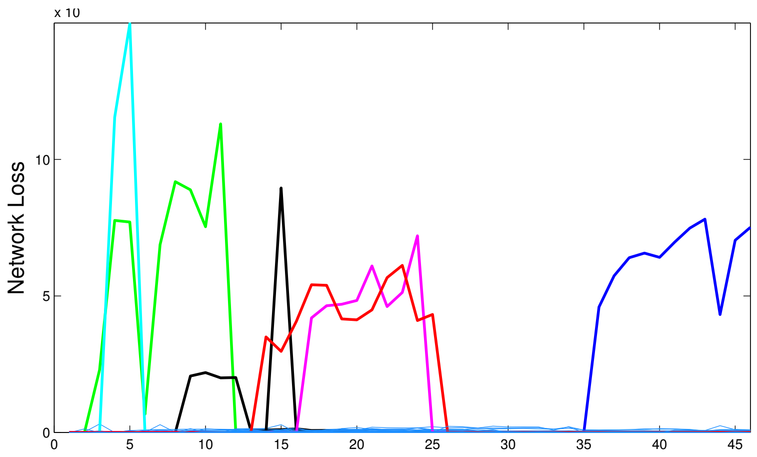


(c) Anomaly Memberships

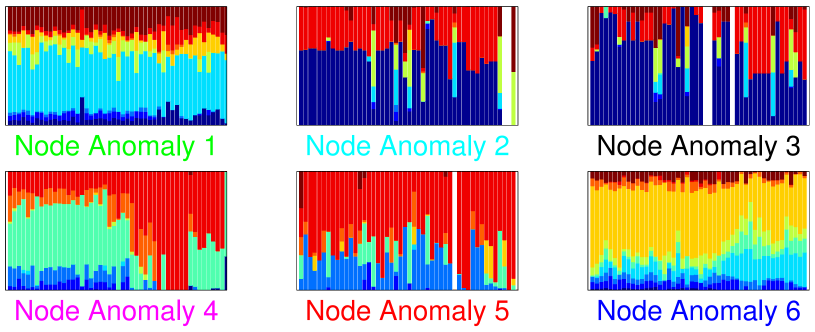
Fig. 4.10.: The DRMM anomaly detector effectively captures differences in both static and dynamic behavior in an email network. (a) shows that normal and anomalous nodes (top-100) differ in their role distribution (i.e., overall static behavior). (b)-(c) show that normal and anomalous nodes also differ in how their behavior changes over time, with anomalies exhibiting more stable behavior over time than normal.

points or that have high betweenness. Additionally, by analyzing the neighbors roles dynamically, we find that nodes with high clustering coefficient primarily are neighbors to nodes with high betweenness or high degree (this plot has been removed for brevity).

In Figure 4.9(e), we find Louise Kitchen, one of the Enron executives who was involved in the Fraud, as having unusual behavioral transitions. Further examination of the network transition model and the average evolution of the behavioral mixed-memberships provide further insights into his abnormal activities. In particular, there are two main role transitions ($r_3 \rightarrow r_7$ and $r_4 \rightarrow r_1$) in Louise Kitchen's transition model that are in contradiction with the network transition model shown in Figure 4.9(c). Furthermore, analyzing the individual changes to the mixed-memberships over time compared with the average behavioral mixed-memberships provides additional insight. For instance, the first two mixed-memberships vectors of Louise



(a) Time-varying Anomalies



(b) Evolving memberships of the time-varying anomalies

Fig. 4.11.: The DRMM model allows us to find nodes that are anomalous for only short periods of time and normal otherwise. Such temporally local anomalies are often impossible to find using static graph analysis because brief abnormal periods are drowned out by mostly normal behavior. (a) shows examples of short lived anomalies in a computer network. (b) shows the corresponding behavior over time for each node in detail.

Kitchen are mainly red and then begin to deviate significantly with seemingly no underlying correlation or pattern between the role transitions. Moreover, there is not any significant correlation between Louise's mixed-memberships and the average memberships at each point in time.

In addition, we also identify interesting patterns of the nodes with unusual behavioral transitions in Figure 4.10. In particular, the DRMM anomaly detector effectively captures differences in both static and dynamic behavior in an email network. Interestingly, in 4.10(b), normal users exhibit a clear cyclical pattern which indicates that normal nodes have a set of roles during the day and another set at night (which agrees with intuition). In contrast, the anomalies 4.10(c) have stable roles over the time that barely fluctuate.

Figure 4.11 also indicates that the DRMM model allows us to find nodes that are anomalous for only short periods of time and normal otherwise. Such temporally local anomalies are often impossible to find using static graph analysis because brief abnormal periods are drowned out by mostly normal behavior. 4.11(a) shows examples of short lived anomalies in a computer network. 4.11(b) shows the corresponding behavior over time for each node in detail.

Synthetic Data. In a separate set of experiments, we further validate our "unusual structural transition" anomaly detector by injecting anomalies into synthetic data (see § 4.5.2). Initially, the dynamics of nodes are predefined to have normal transitions between patterns (e.g., star-center to clique). Then we inject some nodes with anomalous transition behavior by randomly transitioning to an abnormal pattern which we define as star-edge to clique. For 200 repeated simulations, we achieve high accuracy (88.5%) in detecting the anomalous behavior.

4.7 Related Work

There has been an abundance of work in analyzing dynamic networks. However, the majority of this work focuses on dynamic patterns [82, 167, 174, 200, 201], temporal link prediction [78], anomaly detection [202], dynamic communities [203–205], dynamic ranking [44, 206], and many others [153, 170].

In contrast, we propose a scalable temporal behavioral model that captures the node behaviors over time and consequently learns a predictive model for how these behaviors evolve over time. Perhaps the most related work is that of [175] where they develop the dMMSB model to identify roles in the graph and how these memberships change over time. However, this type of mixed-membership model assumes a specific parametric form, which is not scalable (1,000 nodes takes a day to model), and where the groups are defined through linkage to specific nodes (in particular types of groups) rather than more general node behavior or structural properties [176]. This is in contrast to our proposed model, which is based on our intuitive behavioral representation and can be interpreted quantitatively. In addition, our model is not tied to any single notion of behavior and thus is flexible in the roles discovered and generalizable. Moreover, not only do we evaluate our model on detecting unusual behavior, identifying explainable patterns and trends, and for clustering nodes with respect to their transition patterns, but we apply our model on large real-world networks to demonstrate its scalability. To the best of our knowledge, our proposed model is the first scalable dynamic mixed-membership model capable of identifying explainable patterns and trends on large networks.

4.8 Summary

We proposed a dynamic behavioral mixed-membership model for large networks and used it for identifying interesting and explainable patterns and trends. Moreover, we demonstrated its scalability on a variety of real-world temporal networks and provided striking performance results. The experiments have shown the scalability, flexibility, and effectiveness of our model for identifying interesting patterns, detecting unusual structural transitions, and predicting the future structural changes of the network and individual nodes.

Future work will investigate using motifs [207] for role discovery as well as edge role discovery techniques.

5. SUMMARY AND CONCLUSION

This dissertation studied the problem of improving relational learning techniques by leveraging both temporal and relational dependencies. To that end, we introduced a unifying taxonomy that serves as a foundation for studying the main node representation tasks that arise in dynamic attributed network data. This includes the node representation tasks of dynamic node labeling, weighting, and predicting the existence of a node. For each of the three fundamental representation tasks for nodes, we proposed learning techniques designed for modeling relational and temporal dependencies in dynamic attributed networks. Using the dynamic relational representation from the above techniques, we systematically investigate a variety of time-series forecasting tasks on graphs using the proposed methods. To this extent, we demonstrated the utility of the learned representation for learning various relational time series prediction models for the tasks of (i) predicting discrete class labels (classification), and (ii) predicting a future real-valued continuous weight (regression). For each dynamic node representation task, we found that modeling and incorporating the temporal and relational dependencies improved the accuracy (or decreased the error) of the predictive models compared to baseline models that ignore the full-range of temporal dependencies in the dynamic attributed networks. To the best of our knowledge, we are the first to investigate the problem of learning dynamic graph data representation for improving accuracy of predictive models. All techniques are extensively evaluated for real applications such as importance/ranking of web pages, anomaly detection, and pattern mining.

5.1 Future Work and Challenges

A discussion of future challenges and directions are discussed below.

5.1.1 Automatic Kernel Function Learning

In this dissertation, we addressed the problem of learning the parameters automatically given a specific given a kernel function. Future work should investigate the related problem of learning the kernel function automatically. While this dissertation investigated a range of kernel functions and found the exponential to work best in most situations, we expect that for certain relational time-series data, such an approach is likely to result in a significantly better predictive model. Moreover, it also makes it easier for applying the relational time-series learning (for many real-world tasks), without requiring much effort on the part of the user, in terms of knowledge and assumptions about the data. However, techniques proposed in the future must address the challenges associated with the computational cost of such an approach and carefully investigate the benefits (both theoretically and empirically).

Robustness to Noise Another important problem is to investigate the ability of relational time-series learning methods to handle varying levels of noise in both the relational and temporal information? Further, does modeling the temporal dependencies reduce the impact of noise, specifically, when the relational data is noisy (e.g., missing or erroneous links)?

5.1.2 Space and Time Characterization

Future work should also investigate the tradeoff between space and time. Characterizing thee tradeoffs are challenging, for instance, relational time-series learning methods may learn a model using less data by considering only the most recent observations, whereas relational learning approaches, that ignore temporal information, use all available data. Moreover, modeling temporal dependencies may also lead to simpler/more accurate models, and more efficient learning and inference algorithms. However, relational time-series models typically require an appropriate temporal granularity and kernel function, and learning both of these automatically may be costly.

LIST OF REFERENCES

LIST OF REFERENCES

[1] L. Getoor and B. Taskar, Eds., *Introduction to Statistical Relational Learning.* MIT Press, 2007.

[2] R. A. Rossi, L. K. McDowell, D. W. Aha, and J. Neville, "Transforming graph data for statistical relational learning," *Journal of Artificial Intelligence Research*, vol. 45, pp. 363–441, 2012.

[3] S. Macskassy and F. Provost, "A simple relational classifier," in *Proceedings of the SIGKDD 2nd Workshop on Multi-Relational Data Mining*, 2003, pp. 64–76.

[4] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer, "Learning probabilistic relational models," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence.* Springer-Verlag, 1999, pp. 1300–1309.

[5] L. K. McDowell, K. M. Gupta, and D. W. Aha, "Cautious collective classification," *Journal of Machine Learning Research*, vol. 10, pp. 2777–2836, 2009.

[6] L. McDowell, K. Gupta, and D. Aha, "Meta-Prediction for Collective Classification," in *Proceedings of the 23rd International FLAIRS Conference*, 2010.

[7] L. De Raedt and K. Kersting, *Probabilistic Inductive Logic Programming.* Springer-Verlag, 2008.

[8] J. Neville, D. Jensen, and B. Gallagher, "Simple estimators for relational Bayesian classifers," in *Proceedings of the 3rd IEEE International Conference on Data Mining*, 2003, pp. 609–612.

[9] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *Proceedings of the ACM SIGCOMM International Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1999, pp. 251–262.

[10] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," *Computer Networks*, vol. 33, no. 1-6, pp. 309–320, 2000.

[11] R. Albert, H. Jeong, and A. Barabási, "Internet: Diameter of the world-wide web," *Nature*, vol. 401, no. 6749, pp. 130–131, 1999.

[12] A. McGovern, L. Friedland, M. Hay, B. Gallagher, A. Fast, J. Neville, and D. Jensen, "Exploiting relational structure to understand publication patterns in high-energy physics," *SIGKDD Explorations*, vol. 5, no. 2, pp. 165–172, 2003.

[13] M. Newman, "The structure of scientific collaboration networks," *Proceedings of the National Academy of Sciences*, vol. 98, no. 2, pp. 404–409, 2001.

[14] R. Pastor-Satorras and A. Vespignani, "Epidemic spreading in scale-free networks," *Physical Review Letters*, vol. 86, no. 14, pp. 3200–3203, 2001.

[15] C. Moore and M. Newman, "Epidemics and percolation in small-world networks," *Physical Review E*, vol. 61, no. 5, pp. 5678–5682, 2000.

[16] R. May and A. Lloyd, "Infection dynamics on scale-free networks," *Physical Review E*, vol. 64, no. 6, p. 66112, 2001.

[17] A. Kleczkowski and B. Grenfell, "Mean-field-type equations for spread of epidemics: The small world model," *Physica A: Statistical Mechanics and its Applications*, vol. 274, no. 1-2, pp. 355–360, 1999.

[18] R. A. Rossi and J. Neville, "Modeling the evolution of discussion topics and communication to improve relational classification," in *Proceedings of the ACM SIGKDD 1st Workshop on Social Media Analytics*, 2010, pp. 89–97.

[19] H. Jeong, B. Tombor, R. Albert, Z. Oltvai, and A. Barabási, "The large-scale organization of metabolic networks," *Nature*, vol. 407, no. 6804, pp. 651–654, 2000.

[20] A. Wagner and D. Fell, "The small world inside large metabolic networks," *Proceedings of the Royal Society of London. Series B: Biological Sciences*, vol. 268, no. 1478, pp. 1803–1810, 2001.

[21] J. Dunne, R. Williams, and N. Martinez, "Food-web structure and network theory: The role of connectance and size," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 20, p. 12917, 2002.

[22] J. Camacho, R. Guimerà, and L. Nunes Amaral, "Robust patterns in food web structure," *Physical Review Letters*, vol. 88, no. 22, pp. 228 102: 1–4, 2002.

[23] S. Maslov and K. Sneppen, "Specificity and stability in topology of protein networks," *Science*, vol. 296, no. 5569, pp. 910–913, 2002.

[24] H. Jeong, S. Mason, A. Barabasi, and Z. Oltvai, "Lethality and centrality in protein networks," *arXiv preprint cond-mat/0105306*, 2001.

[25] J. Neville, O. Simsek, D. Jensen, J. Komoroske, K. Palmer, and H. Goldberg, "Using relational knowledge discovery to prevent securities fraud," in *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2005, pp. 449–458.

[26] V. Krebs, "Mapping networks of terrorist cells," *Connections*, vol. 24, no. 3, pp. 43–52, 2002.

[27] C. M. Bishop *et al.*, *Pattern Recognition and Machine Learning.* Springer, 2006.

[28] J. R. Anderson, R. S. Michalski, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine Learning: An Artificial Intelligence Approach.* Morgan Kaufmann, 1986, vol. 2.

[29] D. Jensen, J. Neville, and B. Gallagher, "Why collective inference improves relational classification," in *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 593–598.

[30] M. McPherson, L. Smith-Lovin, and J. M. Cook, “Birds of a feather: Homophily in social networks,” *Annual Review of Sociology*, vol. 27, no. 1, pp. 415–444, 2001.

[31] D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’networks,” *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[32] M. Newman, A.-L. Barabasi, and D. J. Watts, *The Structure and Dynamics of Networks.* Princeton University Press, 2006.

[33] A. Einstein, “Zur theorie der brownschen bewegung,” *Annalen der Physik*, vol. 324, no. 2, pp. 371–381, 1906.

[34] J. Bock, A. Cooray, S. Hanany, B. Keating, A. Lee, T. Matsumura, M. Milligan, N. Ponthieu, T. Renbarger, and H. Tran, “The experimental probe of inflationary cosmology (EPIC): A mission concept study for NASA’s Einstein inflation probe,” *arXiv:0805.4207*, 2008.

[35] J. Tang, M. Musolesi, C. Mascolo, V. Latora, and V. Nicosia, “Analysing information flows and key mediators through temporal centrality metrics,” in *Proceedings of the 3rd Workshop on Social Network Systems*, 2010, pp. 1–6.

[36] P. Holme and J. Saramäki, “Temporal networks,” *Physics Reports*, 2012.

[37] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Computing Surveys*, vol. 41, no. 3, p. 15, 2009.

[38] T. Ide and H. Kashima, “Eigenspace-based anomaly detection in computer systems,” in *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 440–449.

[39] C. Noble and D. Cook, “Graph-based anomaly detection,” in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 631–636.

[40] H. Tong and C. Lin, “Non-negative residual matrix factorization with application to graph anomaly detection,” in *Proceedings of the 7th SIAM International Conference on Data Mining*, 2011.

[41] R. A. Rossi, B. Gallagher, J. Neville, and K. Henderson, “Modeling dynamic behavior in large evolving graphs,” in *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, 2013, pp. 667–676.

[42] J. O’Madadhain and P. Smyth, “EventRank: A framework for ranking time-varying networks,” in *Proceedings of the LinkKDD Workshop*, 2005, pp. 9–16.

[43] A. Das Sarma, S. Gollapudi, and R. Panigrahy, “Estimating PageRank on graph streams,” in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2008, pp. 69–78.

[44] R. A. Rossi and D. Gleich, “Dynamic PageRank using evolving teleportation,” *Algorithms and Models for the Web Graph*, vol. 7323, pp. 126–137, 2012.

[45] R. A. Rossi, D. F. Gleich, A. H. Gebremedhin, and M. A. Patwary, “A fast parallel maximum clique algorithm for large sparse graphs and temporal strong components,” *arXiv:1302.6256*, pp. 1–9, 2013.

[46] R. A. Rossi, D. Gleich, and A. Gebremedhin, "Triangle core decomposition and maximum cliques," *SIAM Network Science Workshop*, pp. 1–2, 2013.

[47] R. A. Rossi, "Fast triangle core decomposition for mining large graphs," in *Advances in Knowledge Discovery and Data Mining*, 2014, vol. 8443, pp. 310–322.

[48] S. Oyama, K. Hayashi, and H. Kashima, "Cross-temporal link prediction," in *Proceedings of the 11th International Conference on Data Mining*, 2011, pp. 1188–1193.

[49] M. A. Hasan, V. Chaoji, S. Salem, and M. Zaki, "Link prediction using supervised learning," in *Proceedings of the SDM Workshop on Link Analysis, Counterterrorism and Security*, 2006.

[50] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American Society for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[51] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[52] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun, "Temporal recommendation on graphs via long-and short-term preference fusion," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010, pp. 723–732.

[53] A. Menon and C. Elkan, "Link prediction via matrix factorization," in *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2011, pp. 437–452.

[54] J.-L. Lassez, R. Rossi, and K. Jeev, "Ranking links on the web: Search and surf engines," in *Proceedings of the IEA/AIE International Conference*. Springer, 2008, pp. 199–208.

[55] E. Acar, D. Dunlavy, and T. Kolda, "Link prediction on evolving data using matrix and tensor factorizations," in *Proceedings of the 9th IEEE International Conference on Data Mining Workshops*, 2009, pp. 262–269.

[56] M. Al Hasan and M. J. Zaki, "A survey of link prediction in social networks," in *Social Network Data Analytics*. Springer, 2011, pp. 243–275.

[57] D. Schall, "Link prediction in directed social networks," *Social Network Analysis and Mining*, vol. 4, no. 1, pp. 1–14, 2014.

[58] G. E. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2013.

[59] R. S. Pindyck and D. L. Rubinfeld, *Econometric Models and Economic Forecasts*. McGraw-Hill New York, 1981, vol. 2.

[60] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*. Taylor & Francis, 2002, vol. 1.

[61] N. Ahmed, A. Atiya, N. El Gayar, and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," *Econometric Reviews*, vol. 29, no. 5-6, pp. 594–621, 2010.

[62] M. Clements and D. Hendry, *Forecasting Economic Time Series.* Cambridge University Press, 1998.

[63] M. Marcellino, J. H. Stock, and M. W. Watson, “A comparison of direct and iterated multistep ar methods for forecasting macroeconomic time series,” *Journal of Econometrics*, vol. 135, no. 1, pp. 499–526, 2006.

[64] D. Croushore and T. Stark, “A real-time data set for macroeconomists,” *Journal of Econometrics*, vol. 105, no. 1, pp. 111–130, 2001.

[65] S. Chakrabarti, B. Dom, and P. Indyk, “Enhanced hypertext categorization using hyperlinks,” in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1998, pp. 307–318.

[66] P. Domingos and M. Richardson, “Mining the network value of customers,” in *Proceeding of the 7th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2001, pp. 57–66.

[67] S. Amarel, “On representations of problems of reasoning about actions,” *Machine Intelligence*, vol. 3, pp. 131–171, 1968.

[68] M. Minsky, “A framework for representing knowledge,” Massachusetts Institute of Technology, Tech. Rep., 1974.

[69] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2009.

[70] R. Xiang, J. Neville, and M. Rogati, “Modeling relationship strength in online social networks,” in *Proceedings of the 19th International World Wide Web Conference*, 2010, pp. 981–990.

[71] Q. Lu and L. Getoor, “Link-based classification,” in *Proceedings of the 20th International Conference on Machine Learning*, 2003, pp. 496–503.

[72] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery, “Learning to extract symbolic knowledge from the World Wide Web,” in *Proceedings of the 15th AAAI Conference on Artificial Intelligence*, 1998, pp. 509–516.

[73] Y. Koren, “Collaborative filtering with temporal dynamics,” *Communications of the ACM*, vol. 53, no. 4, pp. 89–97, 2010.

[74] H. Ma, H. Yang, M. R. Lyu, and I. King, “SoRec: Social recommendation using probabilistic matrix factorization,” in *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, 2008, pp. 931–940.

[75] C. Chen, H. Yin, J. Yao, and B. Cui, “TeRec: A temporal recommender system over tweet stream,” *Proceedings of the VLDB Endowment*, vol. 6, no. 12, pp. 1254–1257, 2013.

[76] L. Li, L. Zheng, F. Yang, and T. Li, “Modeling and broadening temporal user interest in personalized news recommendation,” *Expert Systems with Applications*, vol. 41, no. 7, pp. 3168–3177, 2014.

[77] N. N. Liu, L. He, and M. Zhao, "Social temporal collaborative ranking for context aware movie recommendation," *ACM Transactions on Intelligent Systems and Technology*, vol. 4, no. 1, p. 15, 2013.

[78] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *Transactions on Knowledge Discovery from Data*, vol. 5, no. 2, pp. 10:1–10:27, February 2011.

[79] M. Lahiri and T. Y. Berger-Wolf, "Structure prediction in temporal networks using frequent subgraphs," in *IEEE Symposium on Computational Intelligence and Data Mining*, 2007, pp. 35–42.

[80] L. Kovanen, M. Karsai, K. Kaski, J. Kertész, and J. Saramäki, "Temporal motifs in time-dependent networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2011, no. 11, p. P11005, 2011.

[81] J. Tang, M. Musolesi, C. Mascolo, and V. Latora, "Temporal distance metrics for social network analysis," in *Proceedings of the 2nd ACM Workshop on Online Social Networks*, 2009, pp. 31–36.

[82] J. Leskovec, L. Adamic, and B. Huberman, "The dynamics of viral marketing," *Transactions on the Web*, vol. 1, no. 1, pp. 1–39, 2007.

[83] B. Xuan, A. Ferreira, and A. Jarry, "Computing shortest, fastest, and foremost journeys in dynamic networks," *International Journal of Foundations of Computer Science*, vol. 14, no. 02, pp. 267–285, 2003.

[84] M. Lahiri and T. Berger-Wolf, "Mining periodic behavior in dynamic social networks," in *Proceedings of the 8th IEEE International Conference on Data Mining*, 2008, pp. 373–382.

[85] U. Redmond, M. Harrigan, and P. Cunningham, "Identifying time-respecting subgraphs in temporal networks," in *Proceedings of the 3rd International Workshop on Mining Ubiquitous and Social Environments*, 2012, pp. 51–63.

[86] R. A. Rossi, D. F. Gleich, A. H. Gebremedhin, and M. A. Patwary, "What if clique were fast? maximum cliques in information networks and strong components in temporal networks," *arXiv:1210.5802*, pp. 1–11, 2012.

[87] S. Bhadra and A. Ferreira, "Complexity of connected components in evolving graphs and the computation of multicast trees in dynamic networks," in *Ad-Hoc, Mobile, and Wireless Networks*, 2003, pp. 259–270.

[88] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, 2007.

[89] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[90] M. A. Hearst, S. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[91] S. Ben Taieb, G. Bontempi, A. F. Atiya, and A. Sorjamaa, “A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition,” *Expert Systems with Applications*, vol. 39, no. 8, pp. 7067–7083, 2012.

[92] N. Agami, A. Atiya, M. Saleh, and H. El-Shishiny, “A neural network based dynamic forecasting model for trend impact analysis,” *Technological Forecasting and Social Change*, vol. 76, no. 7, pp. 952–962, 2009.

[93] P. Esling and C. Agon, “Time-series data mining,” *ACM Computing Surveys*, vol. 45, no. 1, p. 12, 2012.

[94] U. Sharan and J. Neville, “Temporal-relational classifiers for prediction in evolving domains,” in *Proceedings of the 8th IEEE International Conference on Data Mining*, 2008, pp. 540–549.

[95] İ. Güneş, Z. Çataltepe, and Ş. G. Öğüdücü, “GA-TVRC: A novel relational time varying classifier to extract temporal information using genetic algorithms,” in *Machine Learning and Data Mining in Pattern Recognition*. Springer, 2011, pp. 568–583.

[96] A. McGovern, N. Collier, I. Matthew Gagne, D. Brown, and A. Rodger, “Spatiotemporal relational probability trees: An introduction,” in *Proceedings of the 8th IEEE International Conference on Data Mining*, 2008, pp. 935–940.

[97] C. Preisach and L. Schmidt-Thieme, “Relational ensemble classification,” in *Proceedings of the 6th International Conference on Data Mining*, 2006, pp. 499–509.

[98] ——, “Ensembles of relational classifiers,” *Knowledge and Information Systems*, vol. 14, no. 3, pp. 249–272, 2008.

[99] H. Eldardiry and J. Neville, “Across-model collective ensemble classification,” *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pp. 343–349, 2011.

[100] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, “New ensemble methods for evolving data streams,” in *Proceeding of the 15th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2009, pp. 139–148.

[101] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[102] Y. Marc’Aurelio Ranzato, L. Boureau, and Y. LeCun, “Sparse feature learning for deep belief networks,” *Advances in Neural Information Processing Systems*, vol. 20, pp. 1185–1192, 2007.

[103] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce, “Learning mid-level features for recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2559–2566.

[104] L. Deng and X. Li, “Machine learning paradigms for speech recognition: An overview,” *Transactions on Audio, Speech and Language Processing*, vol. 21, no. 5, pp. 1060–1089, 2013.

[105] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[106] Y. Bengio, "Learning deep architectures for ai," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[107] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th International Conference on Machine Learning*, 2009, pp. 609–616.

[108] C. Couprie, C. Farabet, and Y. LeCun, "Causal graph-based video segmentation," *arXiv:1301.1671*, 2013.

[109] R. Salakhutdinov and G. E. Hinton, "Deep Boltzmann machines," in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 448–455.

[110] J. Lezama, K. Alahari, J. Sivic, and I. Laptev, "Track to the future: Spatio-temporal video segmentation with long-range motion cues," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[111] R. A. Rossi and J. Neville, "Time-evolving relational classification and ensemble methods," in *Advances in Knowledge Discovery and Data Mining.* Springer, 2012, vol. 7301, pp. 1–13.

[112] R. A. Rossi, B. Gallagher, J. Neville, and K. Henderson, "Role-Dynamics: Fast mining of large dynamic networks," in *Proceedings of the 21st International Conference Companion on World Wide Web*, 2012, pp. 997–1006.

[113] D. F. Gleich and R. A. Rossi, "A dynamical system for pagerank with time-dependent teleportation," *Internet Mathematics*, pp. 188–217, 2014.

[114] R. A. Rossi, S. Fahmy, and N. Talukder, "A multi-level approach for evaluating internet topology generators," in *Proceedings of IFIP Networking*, 2013, pp. 1–9.

[115] R. A. Rossi and N. K. Ahmed, "Role discovery in networks," *Transactions on Knowledge and Data Engineering*, vol. 27, no. 4, pp. 1112–1131, April 2015.

[116] ——, "Interactive data repositories: From data sharing to interactive data exploration & visualization," in *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, 2015, pp. 1–5. [Online]. Available: http://networkrepository.com

[117] ——, "The network data repository with interactive graph analytics and visualization," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015, pp. 4292–4293. [Online]. Available: http://networkrepository.com

[118] ——, "NetworkRepository: An interactive data repository with multi-scale visual analytics," in *arXiv preprint (arXiv:1410.3560)*, 2014.

[119] N. K. Ahmed and R. A. Rossi, "Interactive visual graph analytics on the web," in *Proceedings of the 9th International AAAI Conference on Web and Social Media*, 2015, pp. 566–569.

[120] C. Cortes, D. Pregibon, and C. Volinsky, "Communities of interest," in *Proceedings of the 4th International Symposium of Intelligent Data Analysis*, 2001, pp. 105–114.

[121] J. Neville, D. Jensen, L. Friedland, and M. Hay, "Learning relational probability trees," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 625–630.

[122] P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine Learning*, vol. 29, pp. 103–130, 1997.

[123] T. Dietterich, "Ensemble methods in machine learning," *Multiple Classifier Systems*, pp. 1–15, 2000.

[124] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning*, vol. 3, pp. 993–1022, 2003.

[125] S. Arora, R. Ge, and A. Moitra, "Learning topic models–going beyond svd," in *Proceedings of the IEEE 53rd Annual Symposium on Foundations of Computer Science*, 2012, pp. 1–10.

[126] S. Arora, R. Ge, Y. Halpern, D. Mimno, A. Moitra, D. Sontag, Y. Wu, and M. Zhu, "A practical algorithm for topic modeling with provable guarantees," *arXiv:1212.4777*, 2012.

[127] X. Wang and A. McCallum, "Topics over time: A non-Markov continuous-time model of topical trends," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006, pp. 424–433.

[128] D. Blei and J. Lafferty, "Dynamic topic models," in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 113–120.

[129] L. Yao, D. Mimno, and A. McCallum, "Efficient methods for topic model inference on streaming document collections," in *Proceeding of the 15th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2009, pp. 937–946.

[130] C. X. Lin, Q. Mei, J. Han, Y. Jiang, and M. Danilevsky, "The joint inference of topic diffusion and evolution in social communities," in *Proceedings of the 11th International Conference on Data Mining*, 2011, pp. 378–387.

[131] C. Wang, D. Blei, and D. Heckerman, "Continuous time dynamic topic models," *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, 2008.

[132] R. A. Rossi and J. Neville, "Representations and ensemble methods for dynamic relational classification," *CoRR*, vol. abs/1111.5312, 2011.

[133] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," *Stanford Technical Report*, 1998.

[134] A. N. Langville and C. D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings.* Princeton University Press, 2006.

[135] P. Boldi, R. Posenato, M. Santini, and S. Vigna, "Traps and pitfalls of topic-biased PageRank," in *Proceedings of the 4th International Workshop on Algorithms and Models for the Web Graph*, ser. LNCS. Springer, 2007, pp. 107–116.

[136] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using PageRank vectors," in *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, 2006.

[137] H. Tong, C. Faloutsos, and J.-Y. Pan, "Fast random walk with restart and its applications," in *Proceedings of the 6th International Conference on Data Mining*, 2006, pp. 613–622.

[138] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, "Combating web spam with TrustRank," in *Proceedings of the 30th International Very Large Database Conference*, 2004, pp. 576–587.

[139] R. Singh, J. Xu, and B. Berger, "Pairwise global alignment of protein interaction networks by matching neighborhood topology," in *Proceedings of the 11th Annual International Conference on Research in Computational Molecular Biology*, ser. Lecture Notes in Computer Science, vol. 4453. Springer, 2007, pp. 16–31.

[140] R. A. Horn and S. Serra-Capizzano, "A general setting for the parametric Google matrix," *Internet Mathematics*, vol. 3, no. 4, pp. 385–411, March 2007.

[141] P. G. Constantine and D. F. Gleich, "Random alpha PageRank," *Internet Mathematics*, vol. 6, no. 2, pp. 189–236, September 2010.

[142] P. Grindrod, M. Parsons, D. Higham, and E. Estrada, "Communicability across evolving networks," *Physical Review E*, vol. 83, no. 4, p. 046120, 2011.

[143] A. Berman, M. Neumann, and R. J. Stern, *Nonnegative Matrices in Dynamic Systems*. Wiley, 1989.

[144] L. Becchetti, C. Castillo, D. Donato, R. Baeza-Yates, and S. Leonardi, "Link analysis for web spam detection," *ACM Transactions on the Web*, vol. 2, no. 1, pp. 1–42, February 2008.

[145] D. Gleich, P. Constantine, A. Flaxman, and A. Gunawardana, "Tracking the random surfer: Empirically measured teleportation parameters in PageRank," in *Proceedings of the 19th International World Wide Web Conference*, 2010, pp. 381–390.

[146] C. Runge, "Über die numerische auflösung von differentialgleichungen," *Mathematische Annalen*, vol. 46, no. 2, pp. 167–178, 1895.

[147] W. Kutta, "Beitrag zur näherungweisen integration totaler differentialgleichungen," *Zeitschrift für Mathematik und Physik*, pp. 435–453, 1901.

[148] L. F. Shampine and M. W. Reichelt, "The MATLAB ODE suite," *SIAM Journal on Scientific Computing*, vol. 18, no. 1, pp. 1–22, 1997.

[149] R. S. Wills and I. C. F. Ipsen, "Ordinal ranking for Google's PageRank," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, pp. 1677–1696, January 2009.

[150] Various, “Wikipedia database dump,” 2009, version from 2009-03-06. [Online]. Available: http://en.wikipedia.org/wiki/Wikipedia:Database_download

[151] ——, “Wikipedia pageviews,” 2011, accessed in 2011. [Online]. Available: http://dumps.wikimedia.org/other/pagecounts-raw/

[152] P. Boldi, “TotalRank: Ranking without damping,” in *Proceedings of the 14th International World Wide Web Conference*, 2005, pp. 898–899.

[153] J. Yang and J. Leskovec, “Patterns of temporal variation in online media,” in *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, 2011, pp. 177–186.

[154] C. Granger, “Investigating causal relations by econometric models and cross-spectral methods,” *Econometrica: Journal of the Econometric Society*, pp. 424–438, 1969.

[155] J. LeSage, “Applied econometrics using MATLAB,” *Manuscript, Dept. of Economics, University of Toronto*, 1999.

[156] M. Embree and R. B. Lehoucq, “Dynamical systems and non-hermitian iterative eigensolvers,” *SIAM Journal on Numerical Analysis*, vol. 47, no. 2, pp. 1445–1473, 2009.

[157] P. Tsaparas, “Using non-linear dynamical systems for web searching and ranking,” in *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2004, pp. 59–70.

[158] B. Bahmani, R. Kumar, M. Mahdian, and E. Upfal, “PageRank on an evolving graph,” in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012, pp. 24–32.

[159] P. Boldi, M. Santini, and S. Vigna, “Paradoxical effects in PageRank incremental computations,” *Internet Mathematics*, vol. 2, no. 2, pp. 387–404, 2005.

[160] M. Bianchini, M. Gori, and F. Scarselli, “Inside PageRank,” *ACM Transactions on Internet Technologies*, vol. 5, no. 1, pp. 92–128, 2005.

[161] S. Abiteboul, M. Preda, and G. Cobena, “Adaptive on-line page importance computation,” in *Proceedings of the 12th International World Wide Web Conference*, 2003, pp. 280–290.

[162] J. Ratkiewicz, S. Fortunato, A. Flammini, F. Menczer, and A. Vespignani, “Characterizing and modeling the dynamics of online popularity,” *Physical Review Letters*, vol. 105, no. 15, p. 158701, 2010.

[163] J. Bagrow, D. Wang, and A. Barabási, “Collective response of human populations to large-scale emergencies,” *PloS one*, vol. 6, no. 3, p. e17680, 2011.

[164] S. Chien, C. Dwork, R. Kumar, D. Simon, and D. Sivakumar, “Link evolution: Analysis and algorithms,” *Internet Mathematics*, vol. 1, no. 3, pp. 277–304, 2004.

[165] A. N. Langville and C. D. Meyer, “Updating PageRank with iterative aggregation,” in *Proceedings of the 13th International World Wide Web Conference*, 2004, pp. 392–393.

[166] J. Sun, D. Tao, and C. Faloutsos, "Beyond streams and graphs: Dynamic tensor analysis," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006, pp. 374–383.

[167] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: Densification laws, shrinking diameters and possible explanations," in *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2005.

[168] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, "Group formation in large social networks: Membership, growth, and evolution," in *Proceeding of the 12th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2006, pp. 44–54.

[169] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering," in *Proceeding of the 12th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2006.

[170] H. Habiba, Y. Yu, T. Berger-Wolf, and J. Saia, "Finding spread blockers in dynamic networks," in *Advances in Social Network Mining and Analysis*, 2008, pp. 55–76.

[171] M. Cha, A. Mislove, and K. P. Gummadi, "A Measurement-driven Analysis of Information Propagation in the Flickr Social Network," in *Proceedings of the 18th International World Wide Web Conference*, 2009.

[172] R. Pan and J. Saramaki, "Path lengths, correlations, and centrality in temporal networks," *arXiv:1101.5913*, 2011.

[173] S. Asur, S. Parthasarathy, and D. Ucar, "An event-based framework for characterizing the evolutionary behavior of interaction graphs," in *Proceeding of the 13th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2007.

[174] J. Sun, C. Faloutsos, S. Papadimitriou, and P. Yu, "Graphscope: Parameter-free mining of large time-evolving graphs," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[175] W. Fu, L. Song, and E. P. Xing, "Dynamic mixed membership blockmodel for evolving networks," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 329–336.

[176] E. Xing, W. Fu, and L. Song, "A state-space mixed membership blockmodel for dynamic network tomography," *Annals of Applied Statistics*, vol. 4, no. 2, pp. 535–566, 2010.

[177] T. Parsons, "Illness and the role of the physician: A sociological perspective," *American Journal of Orthopsychiatry*, vol. 21, no. 3, pp. 452–460, 1951.

[178] R. Merton, *Social Theory and Social Structure*. Simon and Schuster, 1968.

[179] P. HollandKathryn Blackmond and S. Leinhardt, "Stochastic blockmodels: First steps," *Social Networks*, vol. 5, no. 2, pp. 109–137, 1983.

[180] P. Arabie, S. Boorman, and P. Levitt, "Constructing blockmodels: How and why," *Journal of Mathematical Psychology*, vol. 17, no. 1, pp. 21–63, 1978.

[181] C. Anderson, S. Wasserman, and K. Faust, "Building stochastic blockmodels," *Social Networks*, vol. 14, no. 1, pp. 137–161, 1992.

[182] V. Batagelj, A. Mrvar, A. Ferligoj, and P. Doreian, "Generalized blockmodeling with pajek," *Metodoloski zvezki*, vol. 1, pp. 455–467, 2004.

[183] P. Doreian, V. Batagelj, and A. Ferligoj, *Generalized Blockmodeling.* Cambridge University Press, 2005, vol. 25.

[184] K. Nowicki and T. Snijders, "Estimation and prediction for stochastic blockstructures," *Journal of the American Statistical Association*, vol. 96, no. 455, pp. 1077–1087, 2001.

[185] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[186] S. Smyth and S. White, "A spectral clustering approach to finding communities in graphs," in *Proceedings of the 5th SIAM International Conference on Data Mining*, 2005, pp. 76–84.

[187] M. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E*, vol. 69, no. 6, p. 066133, 2004.

[188] F. Lorrain and H. White, "Structural equivalence of individuals in social networks," *Journal of Mathematical Sociology*, vol. 1, no. 1, pp. 49–80, 1971.

[189] P. Holland and S. Leinhardt, "An exponential family of probability distributions for directed graphs," *Journal of the American Statistical Association*, pp. 33–50, 1981.

[190] L. Sailer, "Structural equivalence: Meaning and definition, computation and application," *Social Networks*, vol. 1, no. 1, pp. 73–90, 1979.

[191] D. White and K. Reitz, "Graph and semigroup homomorphisms on networks of relations," *Social Networks*, vol. 5, no. 2, pp. 193–234, 1983.

[192] K. Nowicki and T. Snijders, "Estimation and prediction for stochastic blockstructures," *Journal of the American Statistical Association*, vol. 96, pp. 1077–1087, 2001.

[193] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos, "It's who you know: Graph mining using recursive structural features," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 1–10.

[194] R. A. Rossi, B. Gallagher, J. Neville, and K. Henderson, "Modeling temporal behavior in large networks: A dynamic mixed-membership model," in *Lawrence Livermore National Laboratory (LLNL) Technical Report, 514271*, 2011, pp. 1–10.

[195] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li, "RolX: Structural role extraction & mining in large graphs," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012, pp. 1231–1239.

[196] D. Hand and R. Till, "A simple generalisation of the area under the roc curve for multiple class classification problems," *Machine Learning*, vol. 45, no. 2, pp. 171–186, 2001.

[197] P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, A. Vahdat *et al.*, "The Internet AS-level topology: Three data sources and one definitive metric," *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, vol. 36, no. 1, pp. 17–26, 2006.

[198] X. Wang, X. Liu, and D. Loguinov, "Modeling the Evolution of Degree Correlation in Scale-Free Topology Generators," in *Proceedings of the IEEE International Conference on Computer Communications*, 2007, pp. 1094–1102.

[199] A. Dhamdhere and C. Dovrolis, "The Internet is flat: Modeling the transition from a transit hierarchy to a peering mesh," in *Proceedings of the 10th International Conference on Emerging Networking EXperiments and Technologies*, 2010, p. 21.

[200] M. Gotz, J. Leskovec, M. McGlohon, and C. Faloutsos, "Modeling blog dynamics," in *Proceedings of the 9th International AAAI Conference on Web and Social Media*, 2009.

[201] S. Papadimitriou, J. Sun, and C. Faloutsos, "Streaming pattern discovery in multiple time-series," in *Proceedings of the 31st International Conference on Very Large Data Bases*, 2005, pp. 697–708.

[202] J. Abello, T. Eliassi-Rad, and N. Devanur, "Detecting novel discrepancies in communication networks," in *Proceedings of the 10th IEEE International Conference on Data Mining*, 2010.

[203] Y. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. Tseng, "Analyzing communities and their evolutions in dynamic social networks," *Transactions on Knowledge Discovery from Data*, vol. 3, no. 2, p. 8, 2009.

[204] L. Tang, H. Liu, J. Zhang, and Z. Nazeri, "Community evolution in dynamic multi-mode networks," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, pp. 677–685.

[205] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks," in *Advances in Social Networks Analysis and Mining*, 2010, pp. 176–183.

[206] J. O'Madadhain, J. Hutchins, and P. Smyth, "Prediction and ranking algorithms for event-based network data," *SIGKDD Explorations*, vol. 7, no. 2, p. 30, 2005.

[207] N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield, "Fast parallel graphlet counting for large networks," *arXiv:1506.04322*, 2015.

APPENDICES

A. DATA

In an effort to support future research in this new area, we have released the data used throughout this dissertation. See [116, 117] for more details.

B. RESEARCH CODES AND LIBRARIES

A number of packages and libraries have resulted from this research, and used by industrial and government agencies on a number of high-impact applications.

B.1 Dynamic PageRank Library

A library for the dynamic PageRank generalization proposed in Section 3.

http://www.ryanrossi.com/dynamic_pagerank

B.2 Parallel Maximum Clique (PMC) Package

A parallel high performance library for solving the maximum clique problem for dense graphs as well as large sparse networks. The parameterized clique library also solves the temporal strong component problem for large dynamic networks. It can be accessed online at: http://www.maxcliques.com

VITA

VITA

Ryan A. Rossi received his Ph.D. in 2015 from the Department of Computer Science at Purdue University. His research focuses on designing scalable machine learning and data mining techniques for large complex data and applying them for real-world applications. He has authored numerous papers at top conferences and journals as well as a number of patents. Ryan is a recipient of the National Science Foundation Graduate Research Fellowship (NSF GRFP), National Defense Science and Engineering Graduate Fellowship (NDSEG), Purdue Frederick N. Andrews Fellowship, as well as the Bilsland Dissertation Fellowship Awarded to Outstanding Ph.D. candidates. He has worked at a number of industrial, government, and academic research labs including Palo Alto Research Center (PARC), Lawrence Livermore National Laboratory (LLNL), Naval Research Laboratory (NRL), NASA Jet Propulsion Laboratory (JPL)/California Institute of Technology, University of Massachusetts Amherst, among others. Ryan has also authored the network repository project (`http://networkrepository.com`), the first data repository with interactive visual graph analytics to help researchers find, explore, and understand graph data in real-time over the web.