DYNAMIC NETWORK MODELING FROM TEMPORAL MOTIFS AND ATTRIBUTED NODE ACTIVITY

by

Giselle M. Zeno Torres

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

Department of Computer Science

West Lafayette, Indiana

August 2023

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Jennifer Neville, Chair

Department of Computer Science

Dr. Clifton Bingham

Department of Computer Science

Dr. Bruno Ribeiro

Department of Computer Science

Dr. Bharat Bhargava

Department of Computer Science

Approved by:

Dr. Kihong Park

Para Abi,

quien siempre me alentó a estudiar y salir adelante—lo logré.

ACKNOWLEDGMENTS

I am thankful to all those who have encouraged and advised me throughout this journey. Words cannot convey the extent of my gratitude to you all, but I shall try nonetheless.

To begin, I thank my advisor Jennifer Neville. She has guided me from my very first day at Purdue, through classes, internships, papers, sucessful (and failed) projects, qualifying exams, and, lastly, the proposal and defense of this dissertation. She has taught me how to do good and thorough research, and how to communicate it—how to write papers and grants, prepare research posters, and make engaging presentations. She has helped and advised me beyond these things, and I will always be grateful to her.

I would also want to thank the other members of my committee, Clifton Bingham, Bruno Ribeiro, and Bharat Bhargava. I'd also like to thank Walid Aref, who was in my preliminary exam committee. I appreciate all their insightful questions and thoughtful comments. I am also very thankful of my internship mentors, William M. Campbell and Timothy La Fond, whose collaborations with me inspired the topics in this dissertation.

I've had many mentors that helped me in the beginning of my journey. My undergraduate advisor, Lilliam Bras, who guided and encouraged me. My friend and professor, Arturo Geigel, who first got me interested in research. My undergraduate professors: Nelliud Torres, who helped me get my first internship, and guided me through applying to graduate school and scholarships. Filiberto Arniella, who challenged and motivated me. And all who taught and helped me along the way, *gracias.*

I thank my labmates from the Network Learning and Discovery lab, for the insightful discussions over the years. I thank Sebastian Moreno, Joseph Pfeiffer, and Nesreen Ahmed, who gave me support and advice when I first joined the lab. Pablo Robles and Timothy LaFond, whose work and collaborations contributed to my own. Hogun Park, Ellen Lai, Iman Alodah, who welcomed me and provided support. Jiasen Yiang, Guilherme Gomes, Mengyue Hang, Mahak Goindani and Jean Lin, for the interesting discussions and providing me valuable feedback.

To my friends and Purdue family: Debbie, *gracias por tu apoyo, las comidas, los cafés, los movie nights, y hacerme parte de tu familia.* Victor, Miguel, Frank, Kristen and Huda,

thank you for all the support, the long study nights, the laughs and commiserating. Marina, Mina and Monique, thank you for your friendship and support through the years. You all have made my time here much better.

To my amazing and loving husband, Rubén—*sin ti no estaría aquí.* Thank you for always supporting me and believing in me; I will never be able to thank you enough. Teddy, whose unconditional love helped me through difficult times. To my chosen family, who have always cheered me on: Margie, *por quererme como una hija y siempre apoyarme.* Vivi and Marga, thank you for all your love and support. To my family, who have been supportive during this journey: *Titi Joyce*, thank you for encouraging me to study and learn since a very young age. *Papi, tu orgullo por mí siempre me inspira.* And to my grandmother, *Abi*, whom I was very close to and will always miss, I dedicate this dissertation to you—*espero hacerte orgullosa siempre.*

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The most important networks from different domains—such as Computing, Organization, Economic, Social, Academic, and Biology—are networks that change over time. For example, in an organization there are email and collaboration networks (e.g., different people or teams working on a document). Apart from the connectivity of the networks changing over time, they can contain attributes such as the topic of an email or message, contents of a document, or the interests of a person in an academic citation or a social network. Analyzing these dynamic networks can be critical in decision-making processes. For instance, in an organization, getting insight into how people from different teams collaborate, provides important information that can be used to optimize workflows.

Network generative models provide a way to study and analyze networks. For example, benchmarking model performance and generalization in tasks like node classification, can be done by evaluating models on synthetic networks generated with varying structure and attribute correlation. In this work, we begin by presenting our systemic study of the impact that graph structure and attribute auto-correlation on the task of node classification using collective inference. This is the first time such an extensive study has been done. We take advantage of a recently developed method that samples attributed networks—although static—with varying network structure jointly with correlated attributes. We find that the graph connectivity that contributes to the network auto-correlation (i.e., the local relationships of nodes) and density have the highest impact on the performance of collective inference methods.

Most of the literature to date has focused on static representations of networks, partially due to the difficulty of finding readily-available datasets of dynamic networks. Dynamic network generative models can bridge this gap by generating synthetic graphs similar to observed real-world networks. Given that motifs have been established as building blocks for the structure of real-world networks, modeling them can help to generate the graph structure seen and capture correlations in node connections and activity. Therefore, we continue with a study of motif evolution in *dynamic* temporal graphs. Our key insight is that motifs rarely change configurations in fast-changing dynamic networks (e.g. wedges into

triangles, and vice-versa), but rather keep reappearing at different times while keeping the same configuration. This finding motivates the generative process of our proposed models, using temporal motifs as building blocks, that generates dynamic graphs with links that appear and disappear over time.

Our first proposed model generates dynamic networks based on motif-activity and the roles that nodes play in a motif. For example, a wedge is sampled based on the likelihood of one node having the role of hub with the two other nodes being the spokes. Our model learns all parameters from observed data, with the goal of producing synthetic graphs with similar graph structure and node behavior. We find that using motifs and node roles helps our model generate the more complex structures and the temporal node behavior seen in real-world dynamic networks.

After observing that using motif node-roles helps to capture the changing local structure and behavior of nodes, we extend our work to also consider the attributes generated by nodes' activities. We propose a second generative model for attributed dynamic networks that (i) captures network structure dynamics through temporal motifs, and (ii) extends the structural roles of nodes in motifs to roles that generate content embeddings. Our new proposed model is the first to generate synthetic dynamic networks and sample content embeddings based on motif node roles. To the best of our knowledge, it is the only attributed dynamic network model that can generate *new* content embeddings—not observed in the input graph, but still similar to that of the input graph. Our results show that modeling the network attributes with higher-order structures (e.g., motifs) improves the quality of the networks generated.

The generative models proposed address the difficulty of finding readily-available datasets of dynamic networks—attributed or not. This work will also allow others to: (i) generate networks that they can share without divulging individual's private data, (ii) benchmark model performance, and (iii) explore model generalization on a broader range of conditions, among other uses. Finally, the evaluation measures proposed will elucidate models, allowing fellow researchers to push forward in these domains.

1. INTRODUCTION

Complex systems and the data they produce can be difficult to understand. Networks, a necessary abstraction for complex systems and relational data, are pervasive in many domains such as social, biological, computing, and communication. This abstraction consists of representing elements of the system as nodes and using links to portray relationships between pairs of nodes. These networks provide insights into the underlying structures and behaviors that shape our interconnected world. For instance, a social network can illustrate friendships between people, a communication network can denote messages sent and received between people or devices, and a biological network can represent protein interactions.

These systems are dynamic and undergo continuous evolution, with nodes and edges constantly being added or removed. For example, in social networks, users establish or remove connections with each other through actions like following, mentioning, and replying. The networks may also contain correlated attributes such as the topic of a message in a communication network or the hobbies and interests of a person in a social network. Moreover, the attributes of users, such as textual features in their generated content, might also change over time. For instance, in the context of academic co-authorship networks, researchers seek collaborators (represented as neighboring nodes) who possess similar or complementary knowledge, and the content generated is the papers they co-author. Their personal research interests may evolve based on new collaborations. These dynamics—social links and user attributes—may influence each other, introducing a complex and valuable area of study.

We can study complex systems using graph generative models to understand the underlying mechanisms that produce the observed data. They can generate new and synthetic data from a few parameters and a specified generative process. Generating synthetic graphs is useful for evaluating other tasks on a wider range of graphs when there might be lack of enough real-world data, or sharing without divulging private data. For example, benchmarking model performance and generalization in tasks like node classification, can be done by evaluating models on synthetic networks generated with varying structure and attribute correlation.

Until recently, existing graph generative models have represented complex systems as being static by using a snapshot of the network structure at one particular point in time. Current temporal graph generative models have either focused on the aggregate structure over time (i.e., when a link is formed it will remain permanently), or can only generate the structure at the next time-window for an existing network. Additionally, existing models for generating attributed graphs have also focused on static graphs. Whereas, in reality, the majority of these networks possess dynamic properties.

Furthermore, modeling motifs as building blocks can help to generate the graph structure seen on real-world networks, and potentially capture correlations in node connections and activity in dynamic networks. However, in the literature, motifs have been studied in the context of growing temporal networks, where links are aggregated over time. In the context of dynamic networks, motifs have been modeled to produce the next time time-window of an observed graph, and cannot produce entirely synthetic networks. The main challenge that motif-based generative models face, when generating synthetic data, is sampling the initial time-window and motif placement.

Finally, when generating synthetic data, we need metrics that measure how well a model can reproduce the real data. Presently, as more synthetic generative models are proposed, there is a need to develop evaluation metrics *tailored* for *dynamic networks*. Such metrics ought to evaluate the changing graph structure, behavior of nodes and attributes in dynamic networks.

1.1 Research Questions

Developing dynamic network generative models that create realistic synthetic graphs and evaluating these models poses some challenges. The first is incorporating higher-order graph structure with temporal dynamics. The second is also taking into account the behavior of individual nodes. The third is including the attributes generated by interactions on networks. The last challenge is designing evaluation metrics for model comparison that take into account all three challenges previously stated.

1. With respect to using synthetic graphs for model evaluation on other tasks: We investigate how graph structure and attributes affect performance of collective inference models for node classification. We generate a wide-range of synthetic graphs with varying structure and attributes. These synthetic graphs are used to determine what structure statistics, along with the level of attribute auto-correlation, are the best predictors of the performance for the collective classification models.

2. With respect to dynamic network generative models:

   (a) On temporal graphs, we investigate the usefulness of temporal motifs for modeling dynamic networks. First, we inspect the evolution of motif configurations over time. Second, we consider the inter-arrival times of observed motifs for modeling the changing structure. Third, we examine the roles of nodes in different motif configurations for motif participation and edge placement. Lastly, we propose how to adequately evaluate dynamic network generative models that produce synthetic graphs.

   (b) On *attributed* temporal graphs, we investigate the usefulness of motif node-roles on attributed activity for modeling dynamic networks with content. First, we consider the use of node role and content embeddings for motif participation and placement. Second, we examine how to extend motif node-roles to roles that generate content. Lastly, we propose an evaluation methodology that includes measures for content topics and embeddings.

1.2 Main Hypothesis and Proposed Research

The goal of this dissertation is to verify the following hypotheses:

1. With regard to the impact of graph structure and attributes on collective classification: As attribute correlation and/or link density increases, the accuracy of collective classification models increases.

2. With regard to dynamic network generative models:

(a) On temporal graphs: (1) If the motif edge placement if known, then modeling inter-arrivals from a Poisson distribution creates the changing graph structure similar to the observed input graph. (2) Using motif node-roles for motif placement, creates underlying graph structure and node behavior similar to the input graph.

(b) On *attributed* temporal graphs: Modeling node roles and their attributes can generate synthetic graphs with dynamic structure similar to the observed input graph.

1.2.1 Proposed Research

This dissertation is divided into three components:

1. The first part is a model to predict the accuracy of collective classification models from graph structure statistics and attribute auto-correlation.

2. The second part is a dynamic network generative model combining motif activity and node roles, along with a model evaluation strategy.

3. The third is an attributed dynamic network generative model, extending node roles to roles that generate content embeddings, and an evaluation methodology for attributed interactions.

The rest of this document is organized as follows: First, we introduce the background of important concepts and methods in Chapter 2. Then, we present a systematic study of how graph structure and attribute auto-correlation impacts the accuracy of collective classification models in Chapter 3. In Chapter 4, we investigate motif evolution on dynamic graphs, propose a synthetic graph generative model from temporal motifs and node roles, and present a model evaluation strategy. Afterwards, in Chapter 5, we outline our proposed generative method for attributed dynamic networks and evaluation measures for generated attributes. Finally, Chapter 6 has a summary highlighting the contributions and implications of the works proposed here and outlining future directions.

2. BACKGROUND

2.1 Data Definitions

Definition 2.1.1 (Static Graph). *A graph G is defined as $G = (V, E)$, where V is the set of nodes and E is the set of edges.*

Definition 2.1.2 (Attributed Static Graph). *An attributed static graph G is defined as $G = (V, E, X)$, where V is the set of nodes, E is the set of edges, and X contains the node (or edge) attributes.*

Definition 2.1.3 (Graph Snapshot). *A graph snapshot is a time-slice of a network at time t, defined as $G_t = (V_t, E_t, S_t)$, where $V_t \subseteq V$ is the set of active nodes, $E_t \subseteq E$ is the set of edges at time t, and $S_t \subseteq S$ are the edge timestamps.*

Definition 2.1.4 (Dynamic Network). *A dynamic network $\boldsymbol{G} = \{G_1, \ldots, G_T\}$ is a sequence of graph time-slices (or snapshots), where T is the number of timesteps.*

Definition 2.1.5 (Aggregate Temporal Graph). *An aggregate graph is the union of all graph snapshots and defined as $G_A = (V,\ E,\ S)$, where $V = V_1 \cup \ldots \cup V_T$ is the set of all nodes, $E = E_1 \cup \ldots \cup E_T$ is the set of all edges, and S contains the timestamps for all edges.*

Definition 2.1.6 (Attributed Graph Snapshot). *An attributed graph snapshot is a time-slice of a network at time t, defined as $G_t = (V, E_t, X_t)$, where V is the set of nodes, E_t is the set of edges at time t, and X_t is the set of attributes at time t.*

Definition 2.1.7 (Attributed Dynamic Network). *An attributed dynamic network $\boldsymbol{G} = \{G_1, \ldots, G_T\}$ is a sequence of graph time-slices (or snapshots), where each time-slice G_t is an attributed graph snapshot and T is the number of timesteps.*

2.2 Task Definitions

2.2.1 Graph Generation

Generative graph models are used to produce synthetic graphs in order to study complex systems from a wide range of domains, such as social, biological, computing and communica-

tion networks. Ideally, they can generate graphs with the properties observed in real-world networks.

Static Graph Generation

The problem of static graph generation is formally defined as:

Problem 1. ***Static Graph Generation***

Input 1. *A graph* $G = (V, E)$.

Output 1. *A graph* $G' = (V, E')$*, where* V *is the set of nodes,* E' *is the set of edges, and the graph structure of* G' *matches* G.

In the Erdős-Rényi (ER) random graph model [1], $G(N, p)$, edges are formed independently with probability p among N nodes. The WattsStrogatz (WS) random graph model [2] produces graphs with short average path lengths and high clustering (small-world properties) as observed in social networks. The Barabási-Albert (BA) model [3] incorporates growth and preferential attachment (i.e., new nodes are more likely to connect with high degree nodes) and generates random scale-free networks (unlike the ER and WS models). The family of Exponential Random Graph Models (ERGMs) $p*$ [4, 5], where p_1 is the ER model, have a probability distribution on all possible networks with N nodes. The Chung-Lu (CL) model [6] is a degree-preserving model developed to create graphs with specified degree distributions similar to real-world networks. Transitive Chung-Lu (TCL) [7] is an extension to the CL model that adds triangle closures to create clustering.

Blockmodels partition the set of nodes V into subsets (blocks) so that the block structure and pattern of edges captures the graph structural properties [8]. Stochastic Blockmodels (SBM) [9–12] extend blockmodeling with a probability distribution that is invariant under permutations of nodes within blocks. The Mixed Membership Stochastic Blockmodel [13] associates each unit of observation with multiple clusters rather than a single cluster, via a membership probability-like vector.

Random Dot Product Graph Model [14–16] assigns to each vertex a vector in $\mathbb{R}^d$ and then any edge is present with probability equal to the dot product of the endpoints. The

Kronecker Product Graph Model (KPGM) [17] is based on a matrix operation (Kronecker product) produces networks with fractal structure. The mixed-Kronecker Product Graph Model (mKPGM) [18, 19] extends KPGM to capture the level of clustering observed in many networks.

More recently, deep graph neural networks have received a lot of attention for their ability to model realistic graphs. Graph auto-encoders aim at representing nodes into low-dimensional vectors. Graph Convolutional Networks (GCNs) use the connectivity of the graph as a filter to execute neighborhood aggregation for node embeddings. The Variational Graph Auto-Encoder (VGAE) model [20] uses GCNs to encode nodes in the graph and a simple decoder to reconstruct the adjacency matrix.

Generative adversarial networks (GANs) [21] can approximate hidden probability distributions using a competitive learning process in which one neural network generates a sample, and a second neural network attempts to discriminate between synthetic samples and examples drawn from the real distribution. GraphGAN [22] model tries to fit, for a given node, its underlying true connectivity distribution over all other vertices and produces "fake" samples to fool the discriminative model. NetGAN [23] poses the problem of graph generation as learning the distribution of biased random walks over the input graph. Graph Recurrent Neural Network (GraphRNN) [24] is a deep auto-regressive model. GraphRNN trains on a set of graphs and breaks down the graph generation process into a sequence of node and edge formations, conditioned on the graph structure generated so far.

Attributed Graph Generation

The problem of attributed graph generation is formally defined as:

Problem 2. ***Attributed Graph Generation***

Input 2. *A graph* $G = (V, E, X)$.

Output 2. *A graph* $G' = (V, E', X')$*, with node set* V*, edge set* E' *and mode attributes* X'*, where the graph structure and attribute auto-correlation of* G' *matches* G.

The Multiplicative Attribute Graphs (MAG) Model [25] generalizes and uses node attributes to better model the structural properties found in real world networks. MAG is a latent space network model, where nodes have some discrete or continuous latent attributes and the probability of two nodes forming an edge depends on their latent attribute values. The Attributed Graph Model (AGM) [26] combines structure and attributes independently in a proposal distribution by first sampling the node attributes and then using a generative network model to sample attributed edges. Constrained Sampling for Attributed Graphs (CSAG) [27] samples from the joint distribution of structure and attributes to generate networks with correlated attributes.

2.2.2 Node Classification

For the task of node classification, we typically have as input a partially labeled graph and wish to infer the missing labels. The problem is formally defined as follows:

Problem 3. ***Node Classification***

Input 3. *An attributed graph $G = (V, E, X)$, with node set V and edge set E, where X contains attributes only for a subset of nodes.*

Output 3. *An attributed graph $G = (V, E, \widehat{X})$, where $\widehat{X} \supset X$ contains the attribute labels for all nodes.*

Collective Inference Approaches

Collective inference have recently been used to significantly improve the predictive accuracy of node classifications in network domains [28]. These models exploit network correlation between the attribute values of linked nodes to improve classification accuracy. The collective classification models in [29] consist of a local model, that classifies individual nodes, and a global model, which propagates the predicted binary labels. Collective classification methods have also been adapted to the multi-label case [30]. Deep Collective Inference (DCI) [31], a more recently proposed model, takes advantage of recent advances in deep learning and achieves reductions in classification error.

Node Representation

The goal of node embedding is to project nodes into a low-dimensional vector that summarizes their graph position and the structure of their local neighborhood. An *encoder* maps each node to a low-dimensional vector (embedding) and a *decoder* deduces structural information about the graph from the learned embeddings.

Random walk approaches learn the embedding from random walk statistics. The motivation is to optimize the embeddings so nodes that tend to co-occur on short random walks have similar embeddings. DeepWalk [32] and node2vec [33] use a shallow embedding (i.e., the encoder is an embedding lookup function) and a decoder based on the inner product. The main difference is that node2vec allows for a flexible definition of random walks, whereas DeepWalk uses simple unbiased random walks on the graph. struct2vec [34] learns representations that capture the structural similarities (or isomorphisms) between nodes, independent of their global graph positions.

Large-scale Information Network Embedding (LINE) [35] is not based on random walks, but is often compared with DeepWalk and node2vec. LINE has two encoder-decoder objectives that optimize "first order" (1-hop neighborhood) and "second order" (2-hop neighborhood) node similarity, instead of combining them in fixed-length random walks.

The Graph Neural Network (GNN) model [36] extends existing neural network methods for relational data in graph domains by mapping graphs and their nodes into an m-dimensional Euclidean space. Structural Deep Network Embedding (SDNE) [37] directly incorporates graph structure into the encoder by compressing information about a node's local neighborhood. SDNE can only generate embeddings for nodes that were present during the training [38]. GraphSAGE [39] aggregate feature information from a node's local neighborhood, such as the degrees or text attributes for example.

3. IMPACT OF GRAPH STRUCTURE AND ATTRIBUTE CORRELATION ON COLLECTIVE INFERENCE METHODS

3.1 Introduction

Relational machine learning and collective inference have recently been used to significantly improve the predictive accuracy of node classifications in network domains [29]. These models exploit network correlation between the attribute values of linked nodes to improve classification accuracy. For example, in social networks a pair of linked friends is more likely to share the same political views than two randomly selected people. Machine learning methods that automatically identify network correlation patterns in observed network data and then use them in a collective (i.e., joint) inference process to propagate predictions throughout the network, have been used successfully across a range of network domains, from social networks to physical and biological networks.

Collective classification methods (e.g., [40]) take advantage of *homophily* (i.e., the principle that links between similar people occurs at a higher rate than among dissimilar people). Many *collective classification* models consist of local model templates that are "rolled out" over the heterogeneous network structure for learning and inference. Thus the local model structure (e.g., attribute correlation across links) may interact with the network structure (e.g., graph connectivity) to impact the accuracy we can obtain using *collective classification*.

The range of empirical evaluation in the field has indicated that network structure and network autocorrelation impact the performance of collective classification models. However, this is based on observations from a limited set of real world networks available for study. There has been some work using synthetic data to systematically study model performance as data characteristics are varied, but the complexity of generating realistic network structures made it difficult to consider characteristics jointly. Specifically, Sen et al. [41] study some of the effects of varying link density and local homophily on classification accuracy. However, in their work one of the metrics is "fixed" while the other is varied. Thus there are still open questions about how both characteristics jointly impact performance. Moreover, there are many other graph and attribute characteristics that could be impacting accuracy of the various methods as well.

The goal of this work is to investigate the following open questions: (1) How do attribute correlation and link density jointly affect the accuracy of *collective classification*? (2) Are there other graph/attribute measures that can help determine the accuracy of *collective classification* methods? (3) In which scenarios is it better to learn a model vs. simple label propagation?

In our work, we exploit a recently developed method for sampling attribute networks with realistic network structure (i.e., parameters learned from real networks) and correlated attributes [27]. Using synthetic data generated from the model, we conduct a systemic study of relational learning and collective inference methods to investigate how graph characteristic interact with attribute correlation to impact classification accuracy. Specifically, we jointly vary the label correlation and link density in order to study these effects simultaneously, something that wasn't possible in prior work.

Our results show that link density and attribute correlation are the characteristics that best describe the changes in accuracy of collective classification. As one of them or both increase, the accuracy of collective classification increases too. Notably, we find that AUC performance can be accurately predicted with a linear function of link density and attribute correlation. Finally, as we have more labeled data available, it is better to learn a model vs. doing simple label propagation.

3.2 Related Work

Macskassy and Provost [29] explore various *collective classification* methods with homogeneous, univariate networks (the class label is the only attribute). They outline the two main components for collective classification, which are the *collective inference* method and the *relational classifier*. In particular, they show results on available datasets and compare how the combination of components from methods in the literature, as well as percentages of labeled data in training, affect the accuracy of the predictions.

Previously there had been no large-scale, systematic experimental study of machine learning methods for within-network classification. Some of the obstacles for such a systematic study are that many learning algorithms can be separated into subcomponents (ideally, the

contributions of such subcomponents should be assessed). For collective inference methods, they study commonly used approximate inference algorithms: the iterative classification algorithm (ICA), relaxation labeling (RL) and gibbs sampling (GS). For relational classifiers, they studied weighted-vote relational neighbor (WVRN), class-distribution relational neighbor (CDRN), network-only naïve Bayes (NBC) and network-only link-based classification (NLB). Although the authors address that angle, they do not explore how the network-characteristics of the datasets used impact the results of collective classification methods.

P. Sen, G. Namata, M. Bilgic et al. [41] study of some of the effects of varying link density and the local homophily. For collective inference methods, they study commonly used approximate inference algorithms: the iterative classification algorithm (ICA) and gibbs sampling (GS). For relational classifiers, they studied naïve Bayes (NB) and Logistic Regression (LR). In their results, they show that when homophily in the graph was low, both content-only (CO) and collective classification (CC) algorithms performed equally well, which was expected result based on similar work. As they increased the amount of homophily (leaving link density fixed), all CC algorithms drastically improved their performance over CO classifiers. Finally, as they increased the link density of the graphs (leaving correlation fixed), accuracies for all CC algorithms went up, possibly because the relational information became more significant and useful. In future work, the authors state that they believe that a better understanding of when these algorithms perform well will lead to more widespread application of these algorithms to more real-world tasks and that this should be a subject of future research.

In this work, we systematically study how network structure and attribute correlation together affect the performance of CC algorithms. Furthermore, since the proportion of labeled vertices in the network has been showed to affect the results, we will include it in the analysis. Our goal is to determine the network structure characteristics that affect CC the most, along with the attribute correlation and the proportion of labeled vertices, and characterize their impact in the accuracy of the classifications made. These results may differ according to the CC subcomponents employed.

3.3 Problem Statement

The research questions we wish to answer with this work: (1) How do attribute correlation and link density jointly affect the accuracy of collective classification?, (2) Are there other graph measures that can help determine the accuracy of collective classification? and (3) When is it better to learn a model versus doing label propagation (with weighted-voting)? More formally, we wish to test the following hypothesis: *As attribute correlation and/or link density increases, the accuracy of collective classification models increases.*

3.4 Algorithms

Collective classification is the simultaneous prediction of the class of several objects given the objects' attributes and their relations.

The experimental setup is similar to that of Macskassy et al. [29], where there is a single attribute X_i of a vertex v_i, representing the class, can take a binary value – for two classes, $X = \{0, 1\}$. Here, c refers to a non-specified class value.

Given a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X})$ where X_i is the (single) attribute of vertex $v_\mathrm{i} \in \mathbf{V}$, and given known values x_i for some subset of vertices $\mathbf{V}^K$, we infer the values x_i of X_i for the remaining unknown vertices, $\mathbf{V}^U = \mathbf{V} - \mathbf{V}^K$, or a probability distribution over those values.

Given $\mathcal{N}_\mathrm{i}$ (1-hop neighborhood of vertex v_i), a relational model can be used to estimate x_i. It is worth noting that just like estimates of the labels of $\mathcal{N}_\mathrm{i}^U$ influence the estimate for x_i, then x_i also influences the estimates of the labels of $v_\mathrm{j} \in \mathcal{N}_\mathrm{i}^U$. In order to simultaneously estimate these interdependent values $\mathbf{x}^U$, we need to apply a collective inference method as well.

3.4.1 Relational Classifiers

The relational model makes use of the relation in the network as well as the values of attributes of related entities, possibly through long chains of relations.

Given $\mathbf{G}^K$ (the graph with known labels), the relational classifier returns a model $\mathcal{M}_R$ that will estimate x_i using v_i and $\mathcal{N}_\mathrm{i}$. Ideally, $\mathcal{M}_R$ will estimate a probability distribution over the possible values for x_i.

Weighted-voting

Also referred to as weighted-vote relational neighbor classifier (WVRN) [29], it is the simplest classifier that estimates class-membership probabilities by assuming the existence of *homophily*.

Given $v_\mathrm{i} \in \mathbf{V}^U$, WVRN estimates $P(x_\mathrm{i} \mid \mathcal{N}_\mathrm{i})$ as the (weighted) mean of the class-membership probabilities in $\mathcal{N}_\mathrm{i}$:

$$P(x_\mathrm{i} = c \mid \mathcal{N}_\mathrm{i}) = \frac{1}{Z} \sum_{v_\mathrm{j} \in \mathcal{N}_\mathrm{i}} w_\mathrm{ij} \cdot P(x_\mathrm{j} = c \mid \mathcal{N}_\mathrm{j}), \tag{3.1}$$

where Z is the usual normalizer. This can be viewed simply as an inference procedure, or as a probability model.

Relational Naive Bayes

Also referred to as network-only bayes classifier (NBC) [29], it uses multinomial naive Bayesian classification based on the classes of v_i's neighbors.

$$P(x_\mathrm{i} = c \mid \mathcal{N}_\mathrm{i}) = \frac{P(\mathcal{N}_\mathrm{i} \mid c) \cdot P(c)}{P(\mathcal{N}_\mathrm{i})}, \tag{3.2}$$

where

$$P(\mathcal{N}_\mathrm{i} \mid c) = \frac{1}{Z} \prod_{v_\mathrm{j} \in \mathcal{N}_\mathrm{i}} P(x_\mathrm{j} = \tilde{x}_\mathrm{j} \mid x_\mathrm{i} = c)^{w_\mathrm{ij}}$$

where Z is a normalizing constant and $\tilde{x}_\mathrm{j}$ is the class observed at node v_j. As usual, because $P(\mathcal{N}_\mathrm{i})$ is the same for all classes, normalization across the classes allows us to avoid explicitly computing it. Laplace smoothing is applied with $m = |X|$ (i.e., the number of classes).

Relational Logistic Regression

This classifier is based similar to the approach of Lu and Getoor [42] and Macskassy and Provost [29]. The relational part consists of creating a feature vector of aggregated labels for a node's 1-hop neighborhood. In particular, the feature vector $\vec{f_i}$ for a node $v_i \in \mathbf{V}^U$ consists of the unnormalized class label counts and ratio of class labels of the neighbors.

Then, a logistic regression classifier (LR) is used to build a discriminative model based on these feature vectors. The learned model is then applied to estimate $P(x_i = c \mid \mathcal{N}_i)$.

3.4.2 Collective Inference Methods

The collective inference component determines how the unknown values are estimated together, possibly influencing each other.

Given a graph $\mathbf{G}$ possibly with some x_i known, a set of prior estimates for $\mathbf{x}^U$, and a relational model $\mathcal{M}_R$, this module applies collective inferencing to estimate $\mathbf{x}^U$.

The initialization of class probabilities can be done in several ways: (1) using a local classifier (which only uses the attributes of the vertex v_i), (2) randomly initializing with the class prior (as in the case of Gibbs Sampling). The probability is computed using the relational model $\mathcal{M}_R$. Finally, the probability for the vertex v_i is updated at each iteration step according to the collective inference method's update step.

Gibbs Sampling

The update step here is to sample a class label with the probability given by the relational model $\mathcal{M}_R$.

The above outlined iteration is repeated 200 times without keeping any statistics–this is known as the burnin period. Then, it is repeated for a maximum of 2000 iterations. Counting the number of times each X_i is assigned a particular value $c \in X$. Normalizing these counts forms the final class probability estimates. [29]

Relaxation Labeling

The update step here is to estimate the class membership probabilities as follows:

$$\hat{\mathbf{c}}_{\mathrm{i}}^{(t+1)} = \beta^{(t+1)} \cdot \mathcal{M}_R(v_{\mathrm{i}}^{(t)}) + (1 - \beta^{(t+1)}) \cdot \hat{\mathbf{c}}_{\mathrm{i}}^{(t)}, \text{ where}$$
$$\beta^0 = k, \text{ and } \beta^{(t+1)} = \beta^{(t)} \cdot \alpha$$

and where t is the current iteration, k is a constant between 0 and 1, which is set to 1.0, and α is a decay constant, which is set to 0.99, following the experimental setup by Macskassy et al. [29] Lastly, the maximum number of iterations for this method is set to 100.

3.5 Data

Using the recent work of Robles et al. [27] on Sampling of Attributed Networks From Hierarchical Generative Models, we can generate attributed networks with varying network structure and attribute dependence. Their CSAG method, which is implemented using *mixed Kronecker Product Graph Models* (mKPGM), samples networks from a hierarchical generative network model (GNM) and constrains every step of the sampling process to consider the structure of the GNM in order to bias the search to regions of the space with higher likelihood. This allows to generate networks with both correlated attributes and complex structure.

CSAG is a new method to approximate sampling from structure and attributes $P(G, \mathbf{X})$ using mKPGM models in the proposal distribution. CSAG is a 2-stage constrained sampling method from a distribution $Q' \sim P$: it first samples a set of blocks from a region of feasibility and then samples edges from the selected block-space. CSAG is applicable to mKPGM and other models.

In the first dataset of networks generated, we start from the same θ initiatior matrix for mKPGM [18]—used to determine edge probabilities between vertices— in Equation Eq. (3.3), and vary the individual θs in small increments to increase the link density.

$$\theta = \begin{bmatrix} 0.7466 & 0.6629 \\ 0.6629 & 0.1402 \end{bmatrix} \tag{3.3}$$

To vary the level of attribute correlation, we simply vary a parameter passed to the CSAG generative method. We generated 168 networks from the θ in Equation Eq. (3.3).

To corroborate that the results we obtained from using the θ initiatior matrix in Equation Eq. (3.3), we also created a second set of networks from a combination of possible values for each individual θ in the initiator matrix. The resulting θ initiatior matrix is shown in Equation Eq. (3.4). We generated 401 networks from using these θs:

$$\theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix} \tag{3.4}$$

where $\theta_{11} \in \{0.99, 0.95, 0.9, 0.85, 0.8\}$, $\theta_{12} \in \{0.55, 0.45,$
$0.35, 0.25, 0.15\}$ and $\theta_{22} \in \{0.75, 0.65, 0.55, 0.45, 0.35\}$. The values for $\theta_{11}, \theta_{12}, \theta_{22}$ were picked by making θ_{11} to be the largest value to enforce identifiability [43] and making $\theta_{21} = \theta_{12}$ to create undirected graphs [18]. The theta in Equation Eq. (3.3) was also derived in this manner, following these relationships in the values which are also seen in learned parameters from real-world data [18]. All the graphs created have one binary label per vertex for the attributes, with balanced classes across the graph, and the edges are all weighted the same (i.e., $w_{ij} = 1 \; \forall \; e_{ij} \in \mathbf{E}$).

3.6 Methodology

Experiment 1

We wish to study how the results of *collective classification* are affected by: (1) graph characteristics, (2) attribute characteristics, (3) *collective classification* methods, and (4) percentage of available node labels in the "bootstrapping" [41] of the predictions. By exploring these characteristics, we can obtain a better understanding of the conditions needed

to obtain good results using *collective classification*. This understanding will be particularly useful for employing these methods with real data.

For the learning step, the size of the training set in the experiments will be varied among 20%, 50%, and 80% using cross-validation. The relational model $\mathcal{M}_R$ is learned from the folds used for training (i.e. $\mathbf{V}^K$). In the classification step, some existing background knowledge is needed to be able to get good results [29]. Therefore, we will use the vertices from $\mathbf{V}^K$ (the vertices with known labels) as background knowledge.

To assess the contributions for the different subcomponents, we perform the experiments on all combinations of the collective inference methods with the classifiers, as described in Section 3.3.

With the generated networks, we will be able to measure the quality of the predictions made by different *collective classification* methods using the area under the ROC curve (AUC). Another aspect of the performance that we will evaluate, is how the percentage of available node labels in the "bootstrapping" [41] of the predictions can also impact the results we observe.

Likewise, although we will vary parameters such as link density and correlation for the generation of networks, different θ initiator matrices for mKPGM will affect network structure. Therefore, additional networks were generated by varying the θ initiator matrix values, as previously described in Section 3.5.

In regards to the graph and attribute characteristics previously mentioned, some *graph characteristics* that could be affecting the quality of the predictions and we study are: (1) link density, (2) clustering coefficient, (3) average path length, (4) size of the largest connected component, (5) average node degree, (6) s-metric value of the network [44], and (7) eigengap of the two largest eigenvalues. For *attribute characteristics*, we studied the following: (1) 1-hop neighborhood class correlation (i.e., Pearson correlation coefficient), (2) 2-hop neighborhood class correlation, (3) average class entropy, (4) baseline autocorrelation due to random chance.

Experiment 2

To compare the difference of learning a model versus doing label propagation, we plotted the increase in AUC scores of the NBC over WVRN. We show the resulting plots for the results from Experiment 1, where RL is used as the collective inference method.

Experiment 3

To answer our third question—*when is it better to learn a model versus doing label propagation?*—we need to characterize the impact of the (1) proportion of labeled vertices (for bootstrapping), (2) graph density, and (3) attribute correlation, in the resulting AUC scores obtained with collective classification. Thus, we learned a linear regression model on the CC results so that we can predict the AUC score for CC depending on the three previously mentioned characteristics of the input graph. We included these three characteristics as the features for the model and we predict the AUC score. Our input here are the graph characteristics used for Experiment 1, as features, and the resulting AUC score, for the linear regression task. We used 5-fold cross-validation to evaluate the model. We learned three versions of the model, (1) for all networks (that we generated in Section 5) and for those with (2) positive correlation only, and (3) negative correlation only.

3.7 Results

In the Figures for Experiments 1 and 2, the small circles represent a graph generated with the given metric (e.g., correlation) and value on the y-axis, and the given metric (e.g., link density in Figure 3.3 and clustering coefficient in Figure 3.2) and value on the x-axis. The space is filled with color according to the AUC score using the score of the nearest points (i.e., graphs). In particular, this is done by finding the convex hull [45, 46] of all points and then doing a Delaunay triangulation to separate the spaces for coloring according to the AUC score.

Experiment 1

In the experiments for the first set of networks from the θ in Equation Eq. (3.3), it seems that the characteristics that jointly affect the AUC scores in a monotonic manner are the *1-hop attribute correlation* combined with the *link-density* (See Figure 3.3). The results and plots (omitted for space) for the second set of networks from the θ in Equation Eq. (3.4), exhibit the same behavior to those of the first θ in Equation Eq. (3.3). NBC performs well with higher correlation (positive or negative) or density. WVRN performs well with higher correlation (positive) or density. LR performed poorly with the relational features used; the AUC scores were near random and the plots are omitted for space.

From the rest of combinations of *attribute characteristics* and *graph characteristics*, the 1-hop attribute correlation combined with the clustering coefficient seems to also have a relationship behaving similar to the previous results, although it does not seem completely monotonic (See Figure 3.2). Another combination of characteristics that seems to have a relation with the AUC score, is the 1-hop attribute correlation and eigengap of the two largest eigenvalues. In particular, if you take a look at Figures 3.6d and 3.6h, as there are more labeled nodes available for "bootstrapping", the eigengap has more impact on the scores. The rest of the graph characteristics we evaluated, did not have a monotonic relationship with any of the attribute characteristics (See Figures 3.5 and 3.6, for sample plots).

Experiment 2

As observed in Figure 3.4, learning a model (NBC) performs better for negative correlation so this is the area where we see the most gain in AUC scores. There are also some gains when the networks have positive correlation and we have more training data available.

Experiment 3

Table 3.1 lists the mean accuracy obtained on the cross-validation folds (as determined by absolute error) of predicting the AUC scores, using networks generated from θ (Equation 3.3). We can most accurately predict the AUC scores when the collective inference method is

Table 3.1. Mean Accuracy of Predicting Collective Classification AUC Scores

Network Samples	NBC+RL	NBC+GS	WVRN+RL	WVRN+GS
All	36%	18%	15%	14%
Positive Correlation	80%	66%	91%	91%
Negative Correlation	86%	86%	90%	90%

relaxation labeling (RL). WVRN is a simple classifier, and thus the collective inference method does not have an effect on how well we can predict the AUC score. In Figure 3.1, note that the coefficient for the density feature is much larger in part due to the densities being very small proportions compared to the other features (i.e., the features are not normalized).

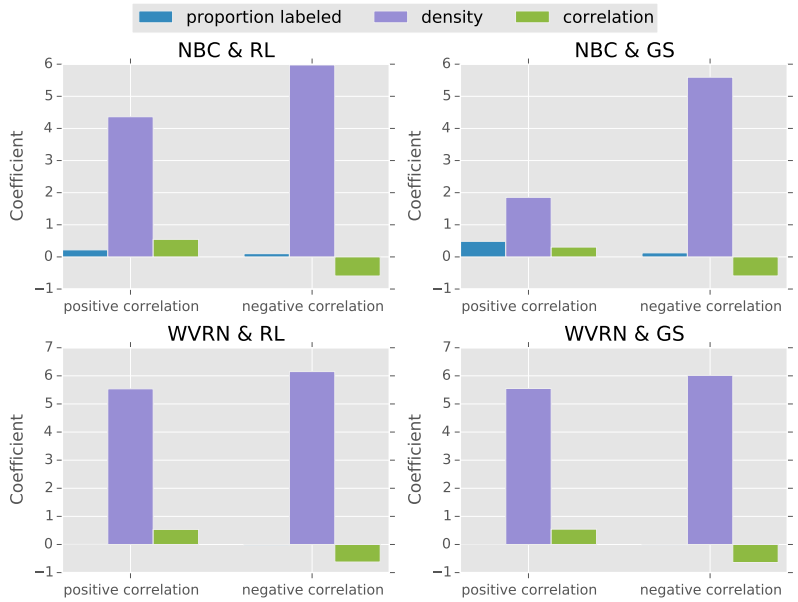


Figure 3.1. Linear Regression Coefficients

3.8 Discussion

Experiment 1. Some observations from the results in Figure 3.3: (1) The weighted-voting classifier does not learn (it assumes homophily) and thus cannot model negative correlation. Therefore, it needs high (positive) correlation and high density to work well. (2)

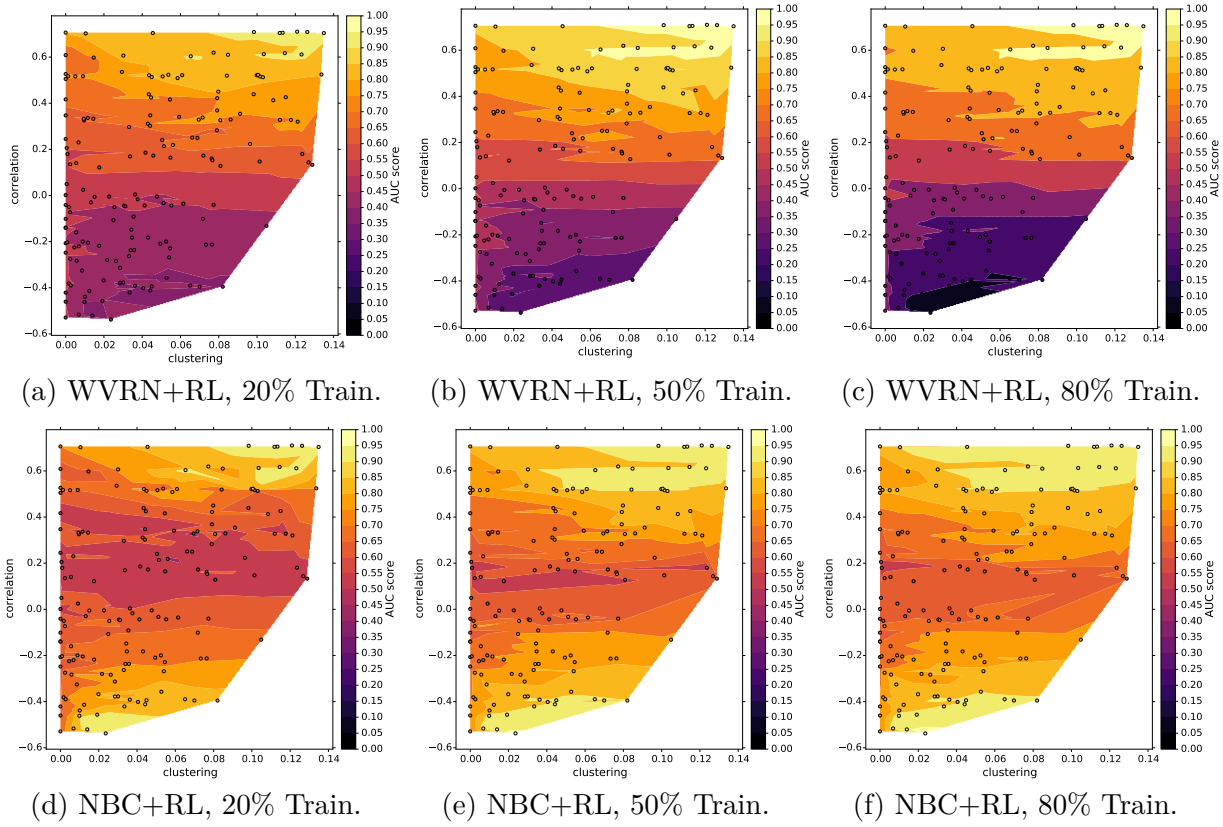
Logistic regression needs more correlation to linearly separate classes; Naive Bayes is significantly more accurate on these networks. (3) Relaxation labeling often performs better than Gibbs sampling. (4) At least 20-30% correlation is needed to improve accuracy, depending on the graph density (5) When there is low density, a higher correlation is needed to learn a better model; As density increases, lower correlation can achieve similar results.

Regarding our hypothesis, the results indicate that: *as attribute correlation and/or link density increases, the accuracy of collective classification models also increases.* The significance of this is that even if we have lower levels of label autocorrelation, if our network is more dense then we can achieve higher accuracy.

Experiment 2. Furthermore, as we have more labeled data (known vertices) it is better to learn a model like NBC versus doing label propagation with WVRN, as observed in Figure 3.4.

Experiment 3. We have also shown, in Figure 3.1 and Table 3.1, that it is possible to predict the AUC score we can obtain with CC methods. The significance of this is that with the learned linear regression coefficients for a CC method on some networks, we can just predict the AUC score and use this to pick the CC method that will likely perform the best for a particular network. In Figure 3.1, it is interesting that (1) when predicting AUC scores for NBC, we have more accuracy in the learned model for negative correlation, than for positive correlation, and (2) for WVRN the proportion of labeled vertices has almost no impact.

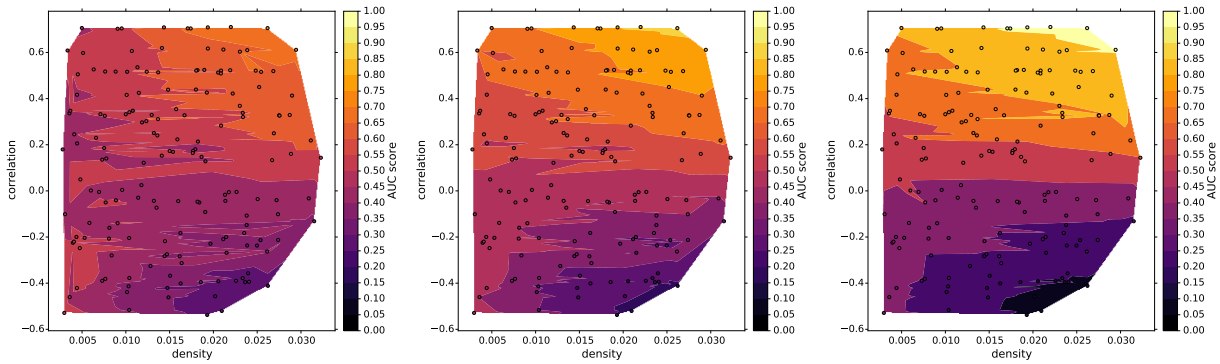
3.9 Additional Figures



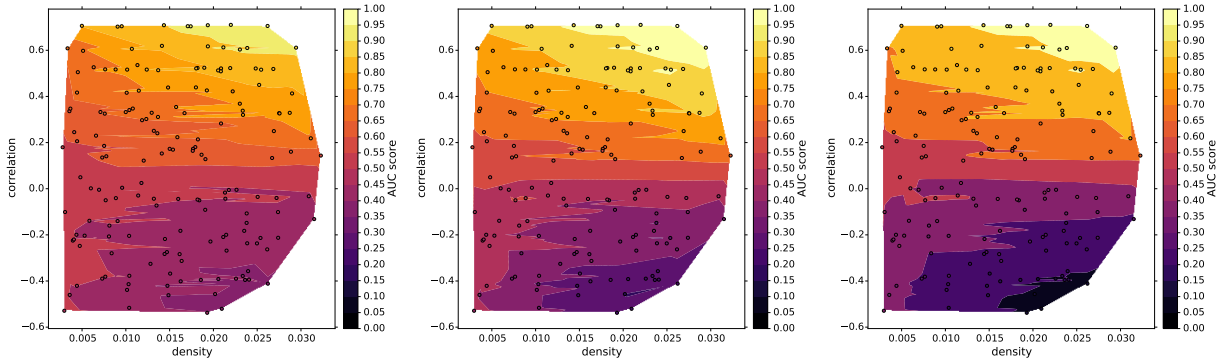
(a) WVRN+RL, 20% Train. (b) WVRN+RL, 50% Train. (c) WVRN+RL, 80% Train.

(d) NBC+RL, 20% Train. (e) NBC+RL, 50% Train. (f) NBC+RL, 80% Train.

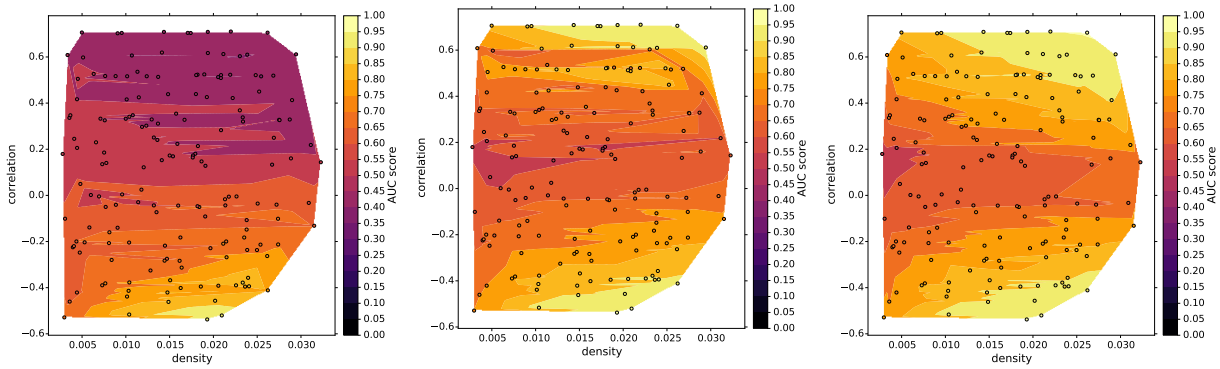
Figure 3.2. Attribute Correlation (1-hop) vs Clustering Coefficient



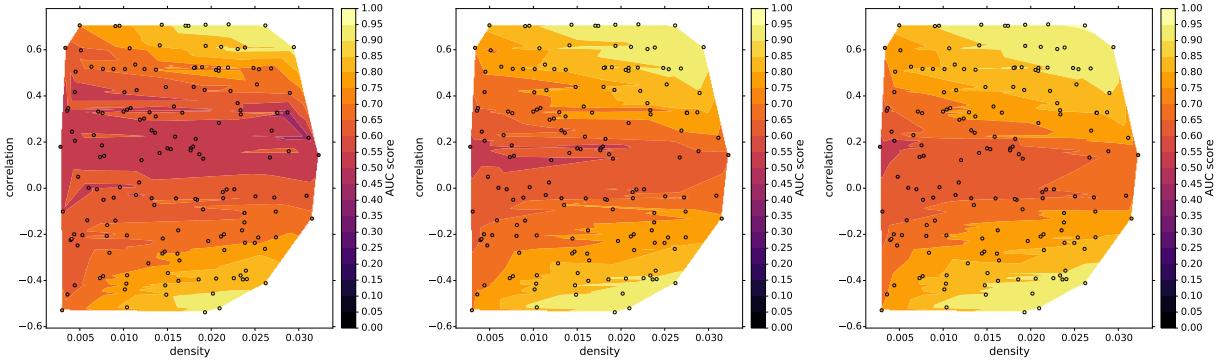
(a) WVRN+GS, 20% Train. (b) WVRN+GS, 50% Train. (c) WVRN+GS, 80% Train.



(d) WVRN+RL, 20% Train. (e) WVRN+RL, 50% Train. (f) WVRN+RL, 80% Train.

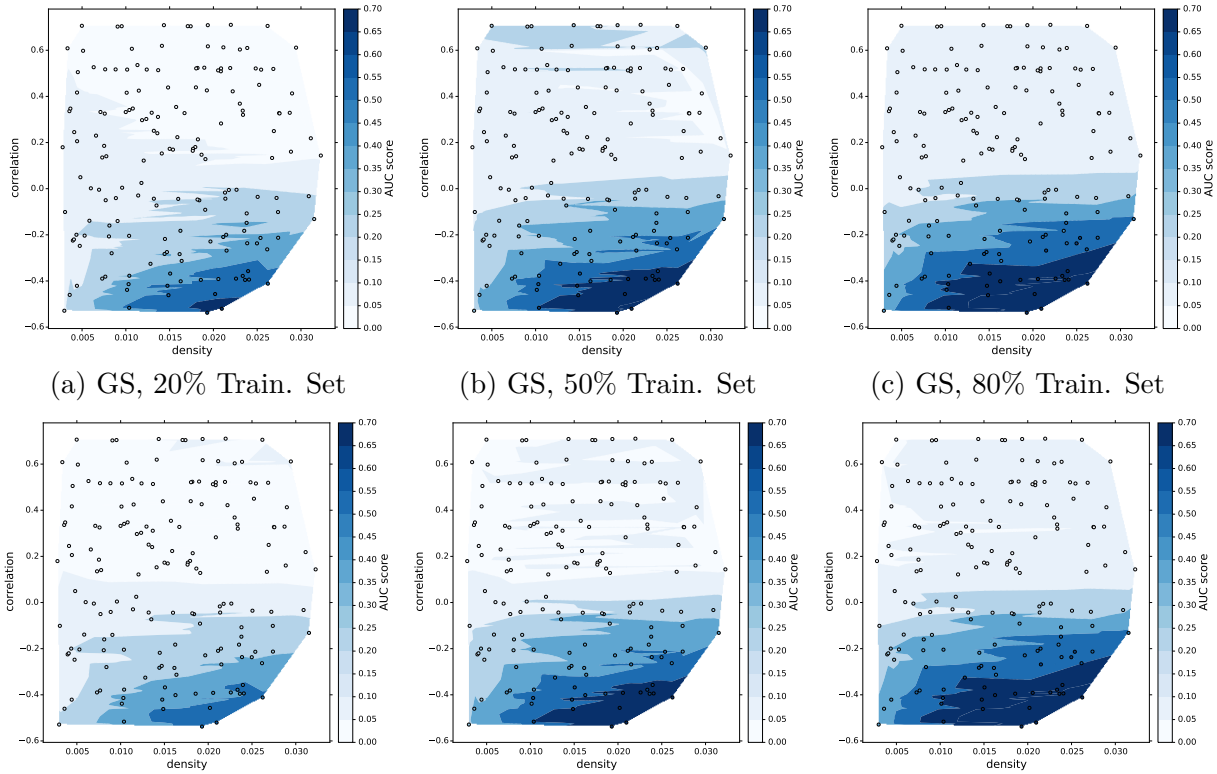


(g) NBC+GS, 20% Train. (h) NBC+GS, 50% Train. (i) NBC+GS, 80% Train.



(j) NBC+RL, 20% Train. (k) NBC+RL, 50% Train. (l) NBC+RL, 80% Train.

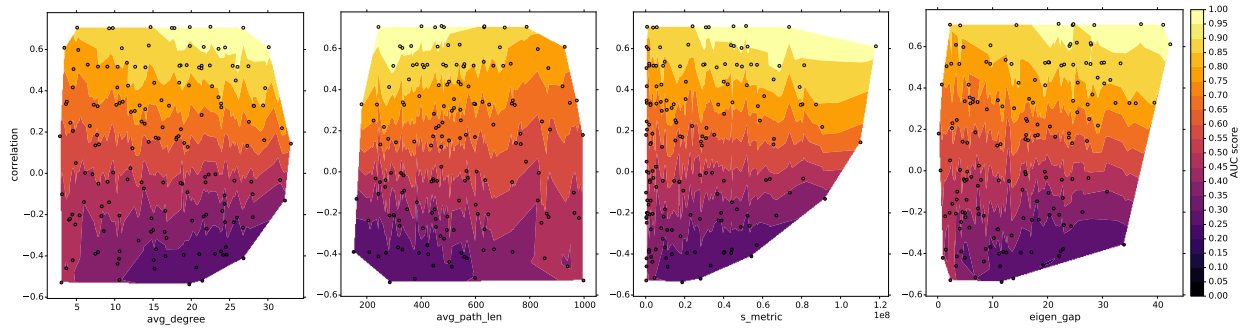
Figure 3.3. Attribute Correlation (1-hop) vs Link-Density



(a) GS, 20% Train. Set (b) GS, 50% Train. Set (c) GS, 80% Train. Set

(d) RL, 20% Train. Set (e) RL, 50% Train. Set (f) RL, 80% Train. Set

Figure 3.4. Difference of learning a model: 1-hop Attribute Correlation vs Density

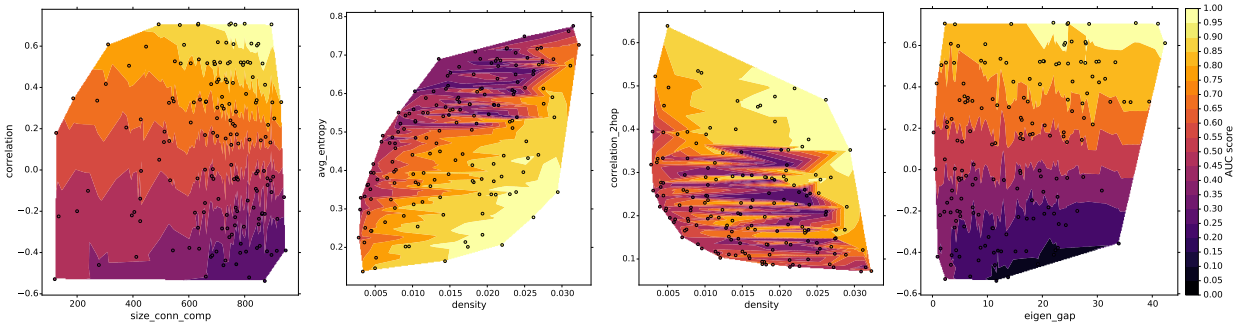


(a) 1-hop Corr. and Avg. Degree

(b) 1-hop Corr. and Avg. Path Length

(c) 1-hop Corr. and S-metric

(d) 1-hop Corr. and Eigengap (50%)



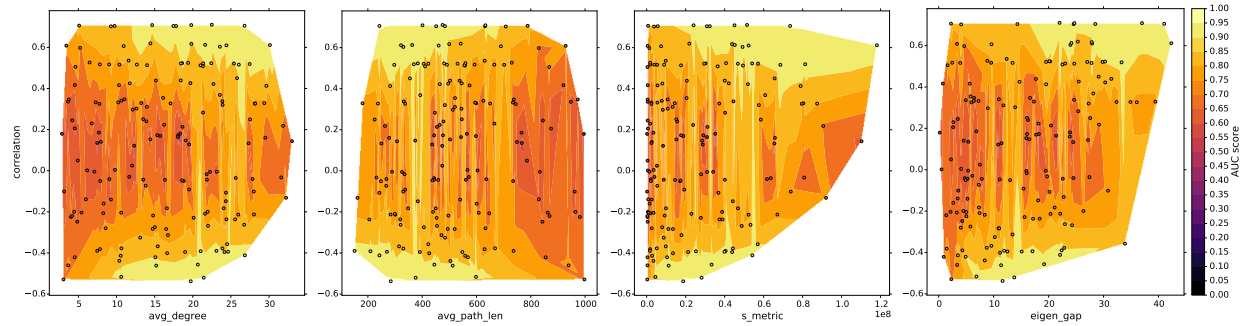
(e) 1-hop Corr. and LCC

(f) Avg. Entropy and Density

(g) 2-hop Corr. and Density

(h) 1-hop Corr. and Eigengap (80%)

Figure 3.5. AUC Scores on various metrics without learning a model (WVRN+RL, 50% Training Set)

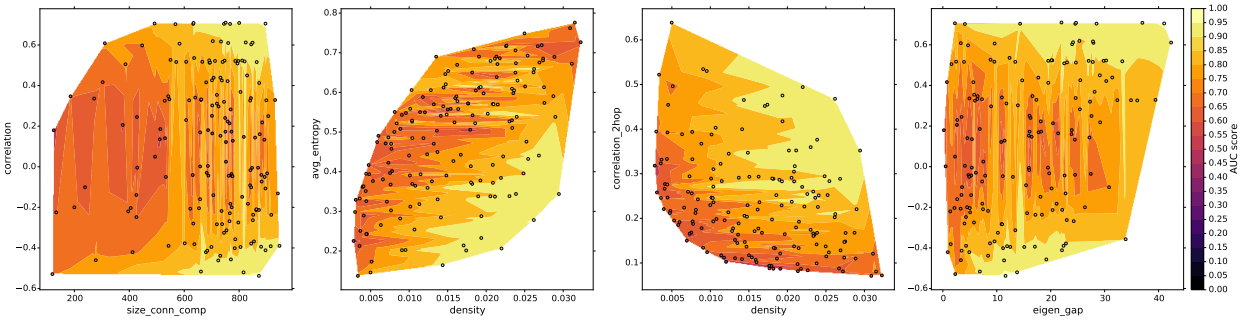


(a) 1-hop Corr. and Avg. Degree

(b) 1-hop Corr. and Avg. Path Length

(c) 1-hop Corr. and S-metric

(d) 1-hop Corr. and Eigengap (50%)



(e) 1-hop Corr. and LCC

(f) Avg. Entropy and Density

(g) 2-hop Corr. and Density

(h) 1-hop Corr. and Eigengap (80%)

Figure 3.6. AUC Scores on various metrics with a learned model (NBC+RL, 50% Training Set)

4. DYNAMIC NETWORK GENERATIVE MODEL FROM TEMPORAL MOTIF-ACTIVITY AND NODE-ROLES

4.1 Introduction

Networks are a way to study complex systems from a wide range of domains, such as social, biological, computing and communication networks. Generating synthetic networks is useful for evaluating other tasks on a wide range of graph structure or sharing without divulging private data, for example. Historically, complex networks with temporal attributes have been studied as static graphs by modeling them as growing networks or aggregating temporal data into one graph. In reality, most of these networks are dynamic in nature and evolve over time, with nodes and edges constantly being added and removed. Many generative models for static graphs have aimed to generate synthetic graphs that can simulate real-world networks. Random graph models were among the first proposed [1]. They were adapted to produce degree distributions similar to those of social networks, but still failed to generate the clustering of real-world networks [4, 6]. Block models were later proposed for creating communities observed in social networks [12, 17, 47]. However, these methods focus on capturing either global or local graph properties, but not both. Initial models for temporal or dynamic networks (where links appear and disappear, such as social-network communication patterns) focused on modeling the edges over time, ignoring higher-order structures [48–50]. Although traditionally most graph models have been edge-based, motifs have been established as building blocks for the structure of networks [51]. Modeling motifs can help to generate the graph structure seen on real-world networks and capture correlations in node connections and activity. Following work that studied the evolution of graphs using higher-order structures (or motifs) [52–56], a generative model using temporal motifs was proposed, which produces networks where links are aggregated over time [57]. However, their approach assumes that edges will not be removed once they are placed.

In this work, we propose a dynamic network model, using temporal motifs as building blocks, that generates dynamic graphs with links that appear and disappear over time. First, we go over related work and discuss where our model fits in Section 4.2. Then in Section 4.3, we present our empirical study of the evolution of motifs in temporal graphs. In

this initial study, we observed that motifs did not change configurations from on timestep to another, rather they kept re-appearing after the first time. Motivated by this finding, we propose DYnamic MOtif-NoDes (DYMOND), a generative model that first assigns a motif configuration (or motif type) and then samples inter-arrival times for the motifs. One challenge that comes with it is sampling the motif placement. To this end, we define motif node roles and use them calculate the probability of each motif type. The motifs and node roles can capture correlations in node connections and activity. We describe our DYMOND generative model in more detail on Section 4.4. Another key challenge is *how to evaluate dynamic graph models?* Previous work has focused on evaluating models using the structure metrics defined for static graphs without incorporating measures that reflect the temporal behavior of the network. In Section 4.5, we describe how to adapt the metrics to consider temporal structure and node behavior. We evaluate our model on five datasets against three other baseline models. We show in our results that our model generates dynamic networks with the closest graph structure to the real data and with similar node behavior. Finally, we discuss the results of our evaluation and present our conclusions.

4.2 Related Work

Most models for temporal or dynamic networks have focused on modeling the edges over time, such as [48–50]. A straightforward approach to generating temporal networks is to generate first a static graph from some model, and for each link generate a sequence of contacts [58]. Holme [49] uses an approach where they draw degrees from a probability distribution and match the nodes in random pairs for placing links. Then, for every link, they generate an active interval duration from a truncated power-law distribution and uniform random starting time within that time frame. Rocha and Blondel [50] use a similar method where the active interval of a node starts when another node's interval ends. Another approach is to start with an empty graph. Then, for every node make it active according to a probability and connect it to m random nodes. Perra et al. [59] use a truncated power-law distribution for each node's probability of being active. Laurent et al. [60] extend this model to include memory driven interactions and cyclic closure. Other extensions include aging

effects [61] and lifetimes of links [62]. Vestergard et al. [63] model nodes and links as being active or inactive using temporal memory effects. Nonetheless, all these node and edge-based models do not consider higher-order structures and fail to create enough clustering in the networks generated.

Motivated by the work that established motifs have as building blocks for the structure of networks [51], the definition of motifs was extended to temporal networks by having all the edges in a given motif occur inside a time period [52, 54, 55]. Zhang et al. [53] study the evolution of motifs in temporal networks by looking at changes in bipartite motifs in subsequent timesteps. Benson et al. [56] study higher-order networks and how 3-node motifs evolve from being empty to becoming a triangle in aggregated temporal graphs. Purohit et al. [57] propose a generative model for synthetic temporal networks where links are aggregated over time (i.e., no link deletions). Zhou et al. [64] propose a dynamic graph neural network model that takes into account higher-order structure by using node-biased temporal random walks to learn the network topology and temporal dependencies.

4.3 Empirical Study

The work of [56] showed the evolution of motifs (e.g., wedges becoming triangles) in growing temporal networks. In this initial study, we investigated if this motif behavior occurs in the evolution of dynamic networks when subsequent time windows are considered (e.g., if the motifs appear, merge, split and/or disappear over time). Specifically, we investigated 3-node motifs and looked for changes from one motif type to another (for example, wedges becoming triangles and vice-versa).

4.3.1 Motif Evolution

The work of [56] showed the evolution of motifs in growing temporal networks (e.g., wedges becoming triangles). In this study, we investigated if this motif behavior occurs in dynamic networks across subsequent time windows (e.g., if the motifs appear, merge, split and/or disappear over time). Specifically, we investigated 3-node motifs and looked

(a) Enron Emails (b) EU Emails (c) Wikipedia Links

Figure 4.1. Motif Transition Probabilities

for changes from one motif type to another (for example, wedges becoming triangles and vice-versa).

4.3.2 Definitions

Definition 4.3.1 (Graph Snapshot)**.** *A graph snapshot is a time-slice of a network at time t, defined as $G_t = (V_t, E_t, S_t)$, where $V_t \subseteq V$ is the set of active nodes, $E_t \subseteq E$ is the set of edges at time t, and $S_t \subseteq S$ are the edge timestamps.*

Definition 4.3.2 (Dynamic Network)**.** *A dynamic network $\mathbf{G} = \{G_1, \ldots, G_T\}$ is a sequence of graph time-slices (or snapshots), where T is the number of timesteps.*

Definition 4.3.3 (Motif)**.** *We define a motif as a 3-node subgraph $\{u, v, w\}$ and its motif type is determined by the number of edges between the nodes (i.e., empty has 0, 1-edge has 1, wedge has 2, triangle has 3).*

The rest of notations and symbols for this chapter are summarized in Table 4.4.

4.3.3 Empirical Study

We tested the hypothesis that motifs changing configurations was driven by a time-homogeneous Markov process where the graph structure at the next timestep $t+1$ depends on the current timestep t. Each timestep corresponds to a time window of the temporal

graph. Then, we consider all 3-node motifs at each timestep to either transform from one motif type to another or remain the same, and isomorphisms are combined into the same configuration.

We studied the effectiveness of this approach on the Enron Emails and EU Emails datasets, described in Subsection 4.5.2. Additionally, we used a Wikipedia Links which shows the evolution of hyperlinks between Wikipedia articles [65, 66]. The nodes represent articles. Edges include timestamps and indicate that a hyperlink was added or removed depending on the edge weight (-1 for removal and +1 for addition). The transition probability matrices for both email datasets (Enron Emails and EU Emails) show that the motifs with edges (i.e., 1-edge, wedge, and triangle) will either keep their current motif type, or become empty with almost equal probability (Figures 4.1a and 4.1b). For each motif type with edges, the count of staying is very close to the count of becoming empty at the next time period. In contrast, for the Wikipedia dataset, the graph keeps growing with more links between articles being added and very few removed. This makes it unlikely to see any motif with edges becoming empty (Figure 4.1c).

In these datasets: (1) we did not observe motifs with edges changing from one motif type to another (e.g., wedges becoming triangles and vice-versa), even when picking different time windows to create the timesteps, and (2) motifs stay with the same type or disappear in the next time window. We also compared modeling edges versus motifs and found that modeling the inter-arrival times of motifs, with an Exponential distribution, retained the graph clustering. This motivates us to propose a motif-based generative model for dynamic networks, which we will outline next.

4.4 DYnamic MOtif-NoDes Model

We formally define the problem of dynamic graph generation as follows:

Problem 4. ***Dynamic Network Generation***

Input 4. *A dynamic network* $\mathbf{G} = \{G_1, \ldots, G_T\}$

Output 4. *A dynamic network* $\mathbf{G}' = \{G'_1, \ldots, G'_{T'}\}$*, where the distribution of graph structure for* $\mathbf{G}'$ *matches* $\mathbf{G}$ *and the node behavior of a specific node* v_{i} *in* $\mathbf{G}'$ *should be similar to a specific node* $v_{\mathrm{i}'}$ *in* $\mathbf{G}$*.*

Specifically, consider an arbitrary graph statistic $s(G)$ (e.g., average path length). Then the distribution of statistic values observed in the input dynamic network $\mathbf{s}_{in} = \{s(G_1), \ldots, s(G_T)\}$ should match the distribution of statistic values observed in the output dynamic network $\mathbf{s}_{out} = \{s(G'_1), \ldots, s(G'_{T'})\}$. Similarly, take any node statistic $\mathbf{s}(v_{\mathrm{i}} \mid \mathbf{G})$ (e.g., node degree). Then, using the temporal distribution of values for a node $\mathbf{s}(v_{\mathrm{i}} \mid \mathbf{G}) = \{s(v_{\mathrm{i}} \mid G_1), \ldots, s(v_{\mathrm{i}} \mid G_T)\}$, the distribution of values for all nodes in the input dynamic network $\{\mathbf{s}(v_{\mathrm{j}} \mid \mathbf{G})\}_{\mathrm{j} \in \mathbf{G}}$ should match the distribution of values for all nodes in the output dynamic network $\{\mathbf{s}(v_{\mathrm{j}'} \mid \mathbf{G}')\}_{\mathrm{j}' \in \mathbf{G}'}$.

To generate dynamic networks as specified above, we propose the DYnamic MOtif-NoDes (DYMOND) generative model. The model makes the following assumptions about the graph generative process:

1. nodes in the graph become active and remain that way,
2. nodes have a probability distribution over role types that they play in motifs,
3. triplets have a single motif type over time,
4. there is a distribution of motif types over the set of graphs,
5. motif occurrences over time are distributed exponentially with varying rate

First we describe the DYMOND generative process below. Then we describe our approach to estimate model parameters from an observed dynamic network. We model the time until nodes become active as Exponential random variables with the same rate λ_V. Since all possible 3-node motifs are considered, there will be edges shared among them. Therefore, to estimate the inter-arrival rate for each motif, we weigh the count of times a motif appeared by the number of edges shared with other motifs in a timestep. For each motif type with edges (i.e., triangle, wedge, and 1-edge), the model fits an Exponential distribution with the motif inter-arrival rates of that type. Motivated by our findings in Subsection 4.3.3, when a motif is first sampled it will keep the same configuration in the future.

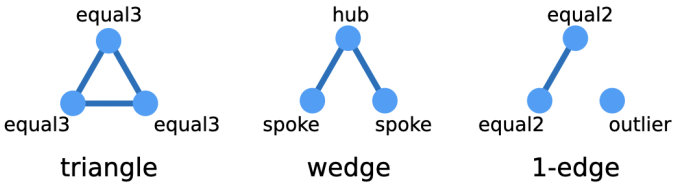


Figure 4.2. Motif Types and Node Roles

In the generation process, the motifs are sampled from a probability distribution based on the probability of the nodes of a triplet participating in a particular motif type, while also ensuring the motif type proportions in the graph are maintained. The motif type probability for a triplet considers the roles each node would play in a motif. For example, in a wedge one node would be a hub and the other two would be spokes (Figure 4.2). The node-roles probabilities are learned from the input graph's structure and the motifs that the node participates in.

The motivation for this modeling approach is based on: (1) by modeling higher-order structures (i.e., motifs), the model will capture the underlying distribution of graph structure, and (2) by using the motif roles that nodes take in the dynamic network, the model will also capture correlations in node connections and activity.

4.4.1 Generative Process

The overall sampling procedure is described in Algorithm 1: In line 2, we first get the nodes that are active at each timestep using the node arrival rate λ_V (see Algorithm 4). Whenever new nodes become active, we calculate the new triplets of active nodes that are now eligible to be sampled as a motif in line 5. In line 6, we proceed to sample the motifs, based on the node role probabilities p_R for each motif types, and also the timesteps they will appear using the motif inter-arrival rates λ_M (see Algorithm 2).

In Algorithm 2: Line 4, the model calculates the expected count of motifs $n^{(\mathrm{i})}$ to be sampled for each motif type i using the motif type proportions p_M. Then in line 5, the expected number of motifs for each type is sampled, given the probability p_T that the nodes

Algorithm 1: `SampleDynamicGraph`

input : T num. of timesteps to generate,
N num. of nodes,
λ_V rate at which nodes become active,
$p_M = (p_M^{(0)}, \ldots, p_M^{(3)})$ proportions motif types,
$\lambda_M = (\lambda_M^{(1)}, \ldots, \lambda_M^{(3)})$ rates of inter-arrival rates,
p_R node roles probabilities,
c_R node roles counts

output: $\mathbf{G}' = \{G_1, \ldots, G_T\}$, where $G_t = (V_t, E_t, S_t)$

1 **begin**
  `// Get active nodes at each timestep` t
2 $V \leftarrow$ `GetActiveNodes`$(T,\ N,\ \lambda_V)$
3 $M \leftarrow \emptyset, M^S \leftarrow \emptyset, M^E \leftarrow \emptyset$
4 **for** $t \in [1, \ldots, T]$ **do**
    `// New active triplets at timestep` t
5    $U_t \leftarrow \{\{u, v, w\} \subseteq V_t \ :\ \{u, v, w\} \notin U_{t-1}\}$
6    $M_t, M_t^T, M_t^S, M_t^R, p_R, c_R \leftarrow$ `SampleMotifs`$(U_t, p_M, \lambda_M, p_R, c_R)$
7    $M \leftarrow M \cup M_t$ `// save new motifs`
8    $M_t^E \leftarrow$ `PlaceMotifEdges`(M_t, M^T, M_t^R)
9    $M^T \leftarrow M^T \cup M_t^T$ `// store their types`
10    $M^E \leftarrow M^E \cup M_t^E$ `// store their edges`
11    $M^S \leftarrow M^S \cup M_t^S$ `// store their timestamps`
12 $\mathbf{G}' \leftarrow$ `ConstructGraph`(M, M^E, M^S)

in the triplet take on the roles needed (Equation 4.6). Each node will have an expected count c_R of times they will show having each role, for the total number of timesteps T to be generated. For this reason, in line 8 we sample the timesteps each motif will appear in, and in line 10 we use the timestep counts for sampling the node roles (see Algorithm 3).

Algorithm 2: `SampleMotifs`

input : U_t active triplets at time t
p_M proportions of each motif type
$\lambda_M = (\lambda_M^{(1)}, \ldots, \lambda_M^{(3)})$ rates of inter-arrival rates,
p_R node roles probabilities,
c_R node roles counts

output: M_t sampled motifs,
M_t^T motif types,
M_t^S motif timestamps,
M_t^R motif node roles
p_R node roles probabilities,
c_R node roles counts

begin
  $L := U_t$ `// triplets to sample from`
  for $\mathrm{i} \in [3, 2, 1]$ **do**
    $n^{(\mathrm{i})} := |U_t| \cdot p_M^{(\mathrm{i})}$ `// num. motifs to sample`
    `// sample motifs`
    $M^{(\mathrm{i})} \sim Mult\left(\left[\{u, v, w\} \in L \mapsto \frac{p_T^{(\mathrm{i})}(\{u,v,w\})}{\sum_{\{u',v',w'\} \in L} p_T^{(\mathrm{i})}(\{u',v',w'\})}\right],\ n^{(\mathrm{i})}\right)$
    $M_t^T := \left[\{u', v', w'\} \in M^{(\mathrm{i})} \mapsto \mathrm{i}\right]$ `// store motif types`
    $L := L - M^{(\mathrm{i})}$ `// remaining triplets to sample from`
    $S^{(\mathrm{i})} \leftarrow$ `SampleMotifTimesteps`$(t, M^{(\mathrm{i})}, \mathrm{i}, \lambda_M)$
    $M^{(\mathrm{i})'} := \left\{\{u', v', w'\} \in M^{(\mathrm{i})} \ : \ |S^{(\mathrm{i})}(\{u', v', w'\})| > 0\right\}$
    $R^{(\mathrm{i})}, p_R, c_R \leftarrow$ `SampleNodeRoles`$(M^{(\mathrm{i})'}, \mathrm{i}, p_R, c_R, S^{(\mathrm{i})})$
    $M_t \leftarrow M_t \cup M^{(\mathrm{i})'}$
    $M_t^S \leftarrow M_t^S \cup S^{(\mathrm{i})}$
    $M_t^R \leftarrow M_t^R \cup R^{(\mathrm{i})}$

Algorithm 3: `SampleNodeRoles`

input : $M^{(\mathrm{i})'}$ motifs type i
p_R node roles probabilities
$S^{(\mathrm{i})}$ motif timesteps
output: $R^{(\mathrm{i})}$ node roles for motifs in $M^{(\mathrm{i})'}$
p_R updated role probabilities
c_R updated role counts

begin
  for $m \in M^{(\mathrm{i})'}$ **do**
    if $\mathrm{i} = 3$ **then** // `triangle`
      $R^{(\mathrm{i})}(m, \texttt{equal3}) \leftarrow m$
      $c_R(v, \texttt{equal3}) \mathrel{-}= \left|S^{(\mathrm{i})}(m)\right|,\ \forall v \in m$
    else if $\mathrm{i} = 2$ **then** // `wedge`
      $p_h = \left[v \in m \mapsto \frac{p_R(v,\ \texttt{hub})}{\sum_{v' \in m} p_R(v',\ \texttt{hub})}\right]$ // `hub node`
      $v_h \sim Bin(m,\ p_h)$
      $R^{(\mathrm{i})}(m, \texttt{hub}) \leftarrow v_h$
      $c_R(v_h, \texttt{hub}) \mathrel{-}= \left|S^{(\mathrm{i})}(m)\right|$
      $R^{(\mathrm{i})}(m, \texttt{spoke}) \leftarrow m \setminus \{v_h\}$ // `spoke nodes`
      $c_R(v, \texttt{spoke}) \mathrel{-}= \left|S^{(\mathrm{i})}(m)\right|,\ \forall \{v \in m : v \neq v_h\}$
    else if $\mathrm{i} = 1$ **then** // `1-edge`
      $p_o = \left[v \in m \mapsto \frac{p_R(v,\ \texttt{outlier})}{\sum_{v' \in m} p_R(v,\ \texttt{outlier})}\right]$ // `outlier node`
      $v_o \sim Bin(m,\ p_o)$
      $R^{(\mathrm{i})}(m, \texttt{outlier}) \leftarrow v_o$
      $c_R(v_o, \texttt{outlier}) \mathrel{-}= \left|S^{(\mathrm{i})}(m)\right|$
      $R^{(\mathrm{i})}(m, \texttt{equal2}) \leftarrow m \setminus \{v_o\}$ // `equal2 nodes`
      $c_R(v, \texttt{equal2}) \mathrel{-}= \left|S^{(\mathrm{i})}(m)\right|,\ \forall \{v \in m : v \neq v_o\}$
    for $v \in m$ **do** // `update role distr` (Equation 4.7)
      $p_R(v, \texttt{role}) = P[v \text{ is } \texttt{role}],\ \forall r \in R$

4.4.2 Learning

Given an observed dynamic graph $\mathbf{G}$, we estimate the input parameters for our generative process.

Node Arrivals

We begin by estimating the node arrival rate $\widehat{\lambda}_V$, which will determine when nodes become active in the dynamic network, by using the first timestep in which each node had its first edge.

$$\widehat{\lambda}_V = \frac{\sum_{v \in V} \left(\arg\min_t \mathbb{1}(v \in V_t) \right)}{|V|} \tag{4.1}$$

Motif Proportions

In Algorithm 8, we find the motifs in G_t in each time window t. For each 3-node motif $\{u, v, w\}$, we find its motif type i at timestep t (line 8). If we have previously seen $\{u, v, w\}$ and the motif type i is of higher order at the current timestep t, then we update the type stored to be i (line 13). For example, if we observe the triplet $\{u, v, w\}$ is a triangle at timestep t and we previously saw it as a wedge, we update $M^T\Big(\{u, v, w\}\Big)$ as a triangle.

Then, we calculate the motif proportions $\widehat{p}_M^{(\mathrm{i})}$ of each type in the graph, where i corresponds to the number of edges in the motif (i.e., i = 1 for a 1-edge, i = 2 for a wedge, and i = 3 for a triangle motif).

$$\begin{aligned} \widehat{p}_M^{(\mathrm{i})} &= \frac{\Big|\Big\{\{u, v, w\} \in M \mid M^T\Big(\{u, v, w\}\Big) = \mathrm{i}\Big\}\Big|}{\binom{N}{3}}, && \text{for i} \in [1, 2, 3] \\ \widehat{p}_M^{(0)} &= 1 - \sum_{\mathrm{i}=1}^{n} \widehat{p}_M^{(\mathrm{i})} \end{aligned} \tag{4.2}$$

where M is the set of motifs, and $\{u, v, w\}$ is a motif consisting of the nodes u, v, w.

Motif Inter-Arrivals

We estimate the inter-arrival rates of each observed motif $\{u, v, w\}$ using weighted edge counts (Equation 4.3a). Their rates are then used to learn a rate of inter-arrival rates $\widehat{\lambda}_M^{(\mathrm{i})}$ from the motifs of each type i (Equation 4.3b). Note that we do not need to estimate rates for the empty motif type ($\mathrm{i} = 0$).

$$\widehat{\lambda}_M\Big(\{u, v, w\}\Big) = \frac{\sum_{t=1}^{T} C_t^M\Big(\{u, v, w\}\Big)}{T} \tag{4.3a}$$

$$\widehat{\lambda}_M^{(\mathrm{i})} = \frac{\sum_{\{u,v,w\} \in M^{(\mathrm{i})}} \Big(\widehat{\lambda}_M\Big(\{u, v, w\}\Big)\Big)}{|M^{(\mathrm{i})}|} \tag{4.3b}$$

where $M^{(\mathrm{i})}$ is the set of all motifs of type i.

Since edges might be shared by more than one motif, we use edge-weighted Poisson counts C_t^M per timestep t, to estimate the inter-arrival rate for each motif $\{u, v, w\}$ (Equation 4.4). The weights $W_t^{(\mathrm{i})}$ will depend on the motif type i of $\{u, v, w\}$ and are calculated for each edge of the motif (Equation 4.5).

$$C_t^M\Big(\{u, v, w\}\Big) = \frac{\sum_{(u',v') \in E_t\left(\{u,v,w\}\right)} W_t^{(\mathrm{i})}\Big((u', v')\Big)}{|E_t(\{u, v, w\})|} \tag{4.4}$$

For a motif $\{u, v, w\}$, we calculate the weight of its edge (u', v') using the count for the edge in the timestep window and considering its motif type i (Equation 4.5a for triangles, Equation 4.5c for wedges, and Equation 4.5e for 1-edge motifs). We give larger edge-weight to motif types with more edges, since they are more likely to produce the observed edges. This also ensures that motifs types with smaller proportion $p_M^{(\mathrm{i})}$ (Equation 4.2) have a high enough inter-arrival rate to show up (i.e., triangles).

$$W_t^{(3)}(u', v') = \frac{c_t(u', v')}{|N^{(3)}(u', v')|} \tag{4.5a}$$

$$r_t^{(2)}(u', v') = \max\Big(0,\ c_t(u', v') - \Big|N^{(3)}(u', v')\Big|\Big) \tag{4.5b}$$

$$W_t^{(2)}(u', v') = \frac{r_t^{(2)}(u', v')}{|N^{(2)}(u', v')|} \tag{4.5c}$$

$$r_t^{(1)}(u', v') = \max\left(0,\ r_t^{(2)}(u', v') - \left|N^{(2)}(u', v')\right|\right) \tag{4.5d}$$

$$W_t^{(1)}(u', v') = \frac{r_t^{(1)}(u', v')}{|N^{(1)}(u', v')|} \tag{4.5e}$$

where $c_t(u', v')$ is the number of times (u', v') appears in E_t and $\left|N^{(\mathrm{i})}(u', v')\right|$ is the number of motifs of type i.

Motif Types

The probability of a triplet becoming a triangle, wedge, or 1-edge motif is based on the probability that each node takes on the roles needed to form that motif type. The roles for each motif type are shown in Figure 4.2. In detail, a triangle requires all three nodes to have the `equal3` role (Equation 4.6a), a wedge requires one node to be a `hub` and the rest have the `spoke` role (Equation 4.6b), a 1-edge requires two nodes to have the `equal2` role and the remaining one the `outlier` role (Equation 4.6c).

$$p_T^{(3)}\Big(\{u, v, w\}\Big) = P[u \text{ is } \texttt{equal3} \wedge v \text{ is } \texttt{equal3} \wedge w \text{ is } \texttt{equal3}] \tag{4.6a}$$

$$\begin{aligned} p_T^{(2)}\Big(\{u, v, w\}\Big) &= P[u \text{ is } \texttt{hub} \wedge v \text{ is } \texttt{spoke} \wedge w \text{ is } \texttt{spoke}] \\ &\vee P[u \text{ is } \texttt{spoke} \wedge v \text{ is } \texttt{hub} \wedge w \text{ is } \texttt{spoke}] \\ &\vee P[u \text{ is } \texttt{spoke} \wedge v \text{ is } \texttt{spoke} \wedge w \text{ is } \texttt{hub}] \end{aligned} \tag{4.6b}$$

$$\begin{aligned} p_T^{(1)}\Big(\{u, v, w\}\Big) &= P[u \text{ is } \texttt{outlier} \wedge v \text{ is } \texttt{equal2} \wedge w \text{ is } \texttt{equal2}] \\ &\vee P[u \text{ is } \texttt{equal2} \wedge v \text{ is } \texttt{outlier} \wedge w \text{ is } \texttt{equal2}] \\ &\vee P[u \text{ is } \texttt{equal1} \wedge v \text{ is } \texttt{equal2} \wedge w \text{ is } \texttt{outlier}] \end{aligned} \tag{4.6c}$$

$$P[u \text{ is } \texttt{role}] = \frac{count(u, \texttt{role})}{\sum_{r \in R} count(u, r)} \tag{4.7}$$

where $R = \{\texttt{equal3}, \texttt{hub}, \texttt{spoke}, \texttt{equal2}, \texttt{outlier}\}$ is the set of all possible roles, and $count(u, r)$ is the weighted count of times that node u had role r (see Algorithm 9 in Section 4.10). The

weights are used to avoid over-counting the roles for motifs of the same type with a shared edge.

4.5 Methodology

4.5.1 Baselines

The related work using motif-based models for temporal graphs focus on the aggregated temporal graph and not its dynamic changes over time [57]. With that in mind, we picked baselines that aim to model the changes in dynamic graphs. We compare our model with three baselines: a temporal edge-based model (SNLD), a model based on node-activity (ADN), and a graph neural network (GNN) model based on temporal random walks (TagGen).

Static Networks with Link Dynamics Model (SNLD)

We used an approach based on [49], where they begin by generating a static graph and then generate a series of events. Their procedure begins by sampling degrees from a probability distribution. They refer to these degrees as "stubs" and they create links by connecting these "stubs" randomly. Finally, for each link, they assign a time-series from an inter-event distribution.

In our implementation of the SNLD model, we start by sampling the degrees from a Truncated Power-law distribution. Since our starting point is a static graph, we assume all the nodes to be active already. Then, we sample inter-event times for every edge. We found that we could best model the edge inter-event times in the real data using an Exponential distribution. To learn the Truncated Power-law parameters, we aggregated and simplified the real graph.

Activity-Driven Network Model (ADN)

We use the approach in [60], which extends the model in [59] by adding memory effects and triadic closure. The triadic closure takes place when node i connects to node k forming

a triangle with its current neighbor j. The memory effect is added by counting the number of times that the nodes have connected up to the current time t. The procedure starts by creating an empty graph G_t at each timestep. Then, for each node i: delete it with probability p_d or mark it as active with probability a_{i}. If the node is "deleted", then the edges in the current timestep are removed, the counts of connections set to zero, and another degree is sampled to estimate a new a_{i}. If a node i is sampled as active, we connect it to either: (1) a neighbor j, (2) a neighbor of j, or (3) a random node.

In our implementation of the SNLD model, we base the probability to create new edges a_{i} on the degree of node i, which we sample from a Truncated Power-law distribution. We estimate the parameters using the average degree across timesteps for the nodes in the real graph. There is a fixed probability p_d for any node being "deleted" (losing its previous connections memory and sampling a new a_{i}). We estimate this probability using the average ratio of nodes becoming disconnected in the next timestep. To estimate the probability for triadic closure (forming a triangle), we use the average global clustering coefficient across timesteps.

TagGen

TagGen is a deep graph generative model for dynamic networks [64]. In their learning process they treat the data as a temporal interaction network, where the network is represented as a collection of temporal edges and each node is associated with multiple timestamped edges at different timestamps. It trains a bi-level self-attention mechanism with local operations (addition and deletions of temporal edges), to model and generate synthetic temporal random walks for assembling temporal interaction networks. Lastly, a discriminator selects generated temporal random walks that are plausible in the input data, and feeds them into an assembling module. We used the available implementation of TagGen to learn the parameters from the input graph and assemble the dynamic network using the generated temporal walks [1].

[1]↑https://github.com/davidchouzdw/TagGen

4.5.2 Datasets

We use the datasets described below, with more detailed statistics shown in Table 4.1 and Figure 4.5.

Table 4.1. Statistics of the Dynamic Network Datasets

Dataset	$\|V\|$	$\|E\|$	Unique	Timesteps
Enron Emails	785	5,794	2,517	20
EU Emails	784	68,091	6,878	79
DNC Emails	1,579	33,378	3,911	23
Facebook Wall-Posts	2,245	23,507	4,976	43
CollegeMsg	1,083	34,328	5,589	31

Enron Emails

The Enron dataset is a network of emails sent between employees of Enron Corporation [66, 67]. Nodes in the network are individual employees and edges are individual emails. Since it is possible to send an email to oneself, loops were removed.

EU Emails

The EU dataset is an email communication network of a large, undisclosed European institution [66, 68]. Nodes represent individual persons and edges indicate at least one email has been sent from one person to the other. All edges are simple and spam emails have been removed from the dataset.

DNC Emails

The DNC dataset is the network of emails of the Democratic National Committee that were leaked in 2016 [66, 69]. The Democratic National Committee (DNC) is the formal governing body for the United States Democratic Party. Nodes in the network correspond to persons and an edge denotes an email between two people. Since an email can have any number of recipients, a single email is mapped to multiple edges in this dataset.

Facebook Wall-Posts

The Facebook dataset is a network of a small subset of posts to other users' wall on Facebook [66, 70]. The nodes of the network are Facebook users, and each directed edge represents one post, linking the users writing a post to the users whose wall the post is written on. Since users may write multiple posts on a wall, the network allows multiple edges connecting a single node pair. Since users may write on their own wall, loops were removed.

CollegeMsg

The CollegeMsg dataset is comprised of private messages sent on an online social network at the University of California, Irvine [71, 72]. Users could search for other users in the network, based on profile information, and then begin conversation. An edge (j, k, t) means that user j sent a private message to user k at time t.

4.5.3 Evaluation

We estimate initial motif configurations and all parameters of our DYMOND model as described in Subsection 4.4.2 from dataset graphs. Similarly, we estimate all parameters of the SNLD and ADN baselines as described in Subsection 4.5.1. For the TagGen baseline, we use the available implementation from [64], which is generally described in Subsection 4.5.1.

Graph Structure

To compare the graph structure generated by the models against the real data, we use the following graph metrics: density, average local clustering coefficient, global clustering coefficient, average path length of largest connected component (LCC), and s-metric. *Density* measures ratio of edges in the graph versus the number of edges if it were a complete graph. The *local clustering coefficient* quantifies the tendency of the nodes of a graph to cluster together, and the *global clustering coefficient* measures the ratio of closed triplets (triangles) to open and closed triplets (wedges and triangles). The *average (shortest) path length*, for

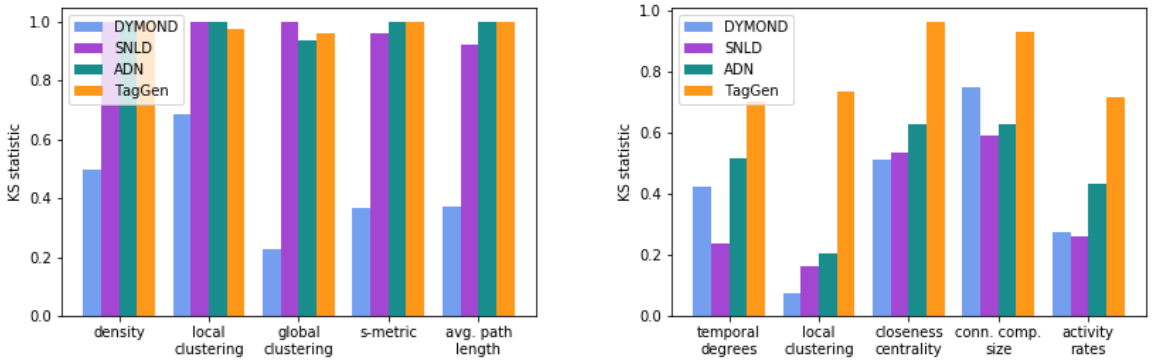
all possible pairs of nodes, measures the efficiency of information transport. The *s-metric* measures the extent to which a graph has hub-like structure [44].

We calculate the graph structure metrics for each generated graph snapshot $\mathbf{G}'_t$ and the input graph. Given that we want to model the distribution of graph structure, and not just generate the same graph sequence, we calculate the Kolmogorov-Smirnov (KS) test statistic for each metric and evaluate $\mathbf{G}'$ against $\mathbf{G}$. We use the KS test because it does not make assumptions about the distribution of the data and it can capture variability or dispersion in the data (which cannot be done by looking at average or median values only).

Node Behavior

To compare the temporal node behavior of the generated graphs against the datasets, we use the following node-aligned metrics: temporal degree distribution, clustering coefficient, closeness centrality, activity rate, and the size of its connected component. The *temporal degree distribution* of a node u is the set of degrees of u over all snapshots G_t. The *local clustering coefficient* of a node measures how close its neighbors are to becoming a clique. The *closeness centrality* of a node u in a (possibly) disconnected graph is the sum of the reciprocal of shortest path distances to u over all other reachable nodes. The *activity rate* of a node measures how often it participates in an edge. The *connected component size* of a node u is the number of nodes in the same connected component.

To evaluate node behavior, we calculate the node-aligned temporal metrics for every node in the dataset $\mathbf{G}$ and in the generated graphs $\mathbf{G}'$. When we refer to node-aligned, we mean that each node will have a distribution of values over time $\mathbf{s}(v_{\mathrm{i}} \mid \mathbf{G}) = \{s(v_{\mathrm{i}} \mid G_1), \ldots, s(v_{\mathrm{i}} \mid G_T)\}$ for each metric s. Since the nodes in $\mathbf{G}$ do not necessarily correspond to those in $\mathbf{G}'$, we consider the inter-quartile range (IQR) of values over time $\{\mathbf{s}(v_{\mathrm{j}} \mid \mathbf{G})\}_{\mathrm{j}\in\mathbf{G}}$. We then perform a 2-dimensional KS test using the Q_1 and Q_3 values of all nodes in $\mathbf{G}$ and $\mathbf{G}'$. In this way, we capture each node's individual behavior and their joint behavior.



(a) Graph Structure Metrics

(b) Node Behavior Metrics

Figure 4.3. EU Emails - KS statistics

4.6 Results

In Figure 4.3, we show the KS statistic (lower is better) for the graph structure and node behavior of the EU Emails dataset, as an example. The full set of results of all the other datasets, for the five graph metrics and the five node metrics, are in . In order to compare the models more easily, we calculated the mean reciprocal rank (MRR) of the KS statistic for the graph structure and the node behavior metrics. To calculate the MRR, we ranked the model results by using the average KS statistic.

In Table 4.2, we can observe that our DYMOND model outperforms the baselines when considering all the graph structure metrics together using the MRR (higher is better).

Table 4.2. Graph Structure Mean Reciprocal Rank

Model	Enron	EU	DNC	Facebook	CollegeMsg
DYMOND	**0.57**	**1.00**	**0.80**	**0.77**	**0.90**
SNLD	0.52	0.61	0.29	0.47	0.45
ADN	0.46	0.67	0.34	0.50	0.43
TagGen	0.53	0.58	0.64	0.30	0.30

In Table 4.3, our model performs best on the node behavior for two of the datasets (Enron Emails and Facebook). SNLD performs better on the EU Emails dataset, with our model

being a close second, and the CollegeMsg dataset. Finally, ADN performs best on the DNC Emails dataset.

Table 4.3. Node Behavior Metrics Mean Reciprocal Rank

Model	Enron	EU	DNC	Facebook	CollegeMsg
DYMOND	**0.93**	0.70	0.50	**0.85**	0.48
SNLD	0.40	**0.73**	0.35	0.65	**0.95**
ADN	0.47	0.37	**0.93**	0.42	0.48
TagGen	0.25	0.25	0.27	0.25	0.25

4.6.1 Discussion

SNLD creates a static graph with a degree distribution learned from the input graph and models the edge inter-event times independently. This fails to create graph structure similar to the datasets due to little clustering. The CollegeMsg dataset has low clustering (local and global) but high s-metric, which indicates large star structure in the graph (i.e., high degree nodes), as seen in Figure 4.5. In this case, SNLD is able to match better the clustering than the other datasets (Figure 4.6h).

ADN models node activation rates using sampled node degrees from a Power-law distribution. However, during sampling a node might be "deleted" and have their rate changed. When evaluating node-aligned metrics over time, these rate modifications will change the behavior of a node. This explains the poor performance of ADN on the node activity rates metric of all the datasets (Figures 4.6b, 4.6d, 4.6f and 4.6h) except the EU Emails dataset (Figure 4.3b). Lastly, this model incorporates a triadic closure mechanism to create clustering in the graph structure, which actually helps it perform better on datasets with high clustering (Figure 4.6d)

Though TagGen doesn't perform well on most of the graph structure metrics, it manages to create graph clustering comparable to our model in three of the datasets: Enron Emails, DNC Emails, and Facebook (Figures 4.6a, 4.6c and 4.6e). These three datasets have various star structures (very high degree nodes) across timesteps and shorter diameter than the other datasets. TagGen performs biased temporal random walks and, according to the authors,

high degree nodes tend to be highly active resulting in a weak dependence between the topology and temporal neighborhoods. This would explain why it performs better in these three datasets than the others.

Using motifs helps our DYMOND model create better clustering in the graph than the other baselines (Figures 4.6a, 4.6e and 4.6g), which also impacts the graph density and s-metric that is generated. The motif inter-arrival times are able to capture the node-aligned behavior in the graph Figures 4.6b, 4.6d and 4.6f). The addition of motif node roles determines the placement of the motifs in the graph, which in turn impacts the node closeness and shortest path lengths produced. These roles also help capture the node-aligned temporal degree distribution even though we do not directly optimize it.

4.7 Scalability Analysis

For conciseness, in this section we use V as the number of nodes, E as the number of edges, M as the number of motifs, and $\mathcal{U}_t$ as the number of new triplets, where the subscript t denotes the timestep.

The time complexity of DYMOND *parameter estimation* is dominated by the number of motifs in the input graph. In the worst case, the number of 3-node motifs is all combinations of 3 nodes. We only consider connected motifs (Figure 4.2), which can be found by iterating over the edges at each graph snapshot t (i.e., $\mathcal{O}(V^2 \cdot T)$). In practice, most real-world graphs are very sparse so the time complexity of finding the motifs is $\mathcal{O}(E \cdot V \cdot T)$.

The time complexity of the DYMOND *generative process* is dependent upon the motif sampling from the new triplets $\mathcal{U}_t$. In the worst case, all nodes become active at the same time and the number of new triplets is $\mathcal{U}_t = \binom{V}{3}$, which would result in $\mathcal{O}(V^3)$ time. In practice, the number of connected 3-node motifs M is proportional to the number of nodes and edges (i.e., $E \cdot V \leq M < V^3$). To analyze the time complexity of sampling motifs, we recorded the number of node arrivals over time and calculated the number of new active triplets $\mathcal{U}_t$ available to sample from (Figure 4.4). The orange line in the plot, shows how $\mathcal{U}_t$ evolves to the theoretical complexity $\binom{V}{3}$. We calculate $\mathcal{U}_t$ at each timestep as the 3-node

triplets we have not considered before (i.e., $\mathcal{U}_t = \binom{V_t}{3} + \binom{V_t}{2} \cdot V_{t-1} + \binom{V_{t-1}}{2} \cdot V_t$), substantially reducing the time complexity.

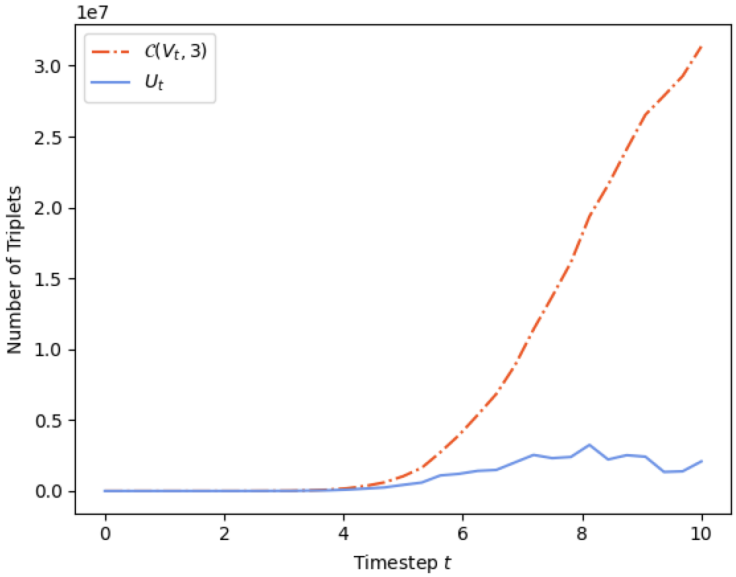


Figure 4.4. Number of triplets to sample from

4.8 Concluding Remarks

Our proposed dynamic-graph generative model, DYnamic MOtif-NoDes (DYMOND), not only considers the dynamic changes in overall graph structure using temporal motif activity, but also considers the roles the nodes play in motifs (e.g., one node plays the hub role in a wedge, while the remaining two act as spokes). In our empirical study of dynamic networks, we demonstrated that motifs with edges: (1) do not change configurations (e.g., wedges becoming triangles and vice-versa); (2) once they appear, they will stay during the next time window or disappear. We also developed a novel methodology for comparing dynamic-graph generative models and measuring how well they capture the underlying graph structure distribution and the node behavior of a real graph. Specifically, using node-aligned metrics over the graph snapshots helps to evaluate the node's topological connectivity and temporal activity. We note that using motifs helps our DYMOND model create better graph structure overall, while the motif node roles can better represent the temporal node behavior. Our quantitative evaluation of graph structure and node behavior is performed

on five different real-world datasets. DYMOND is the first motif-based dynamic network generative model and, through the use of motif node roles, it creates realistic graph structure and node behavior.

4.9 Additional Tables and Figures

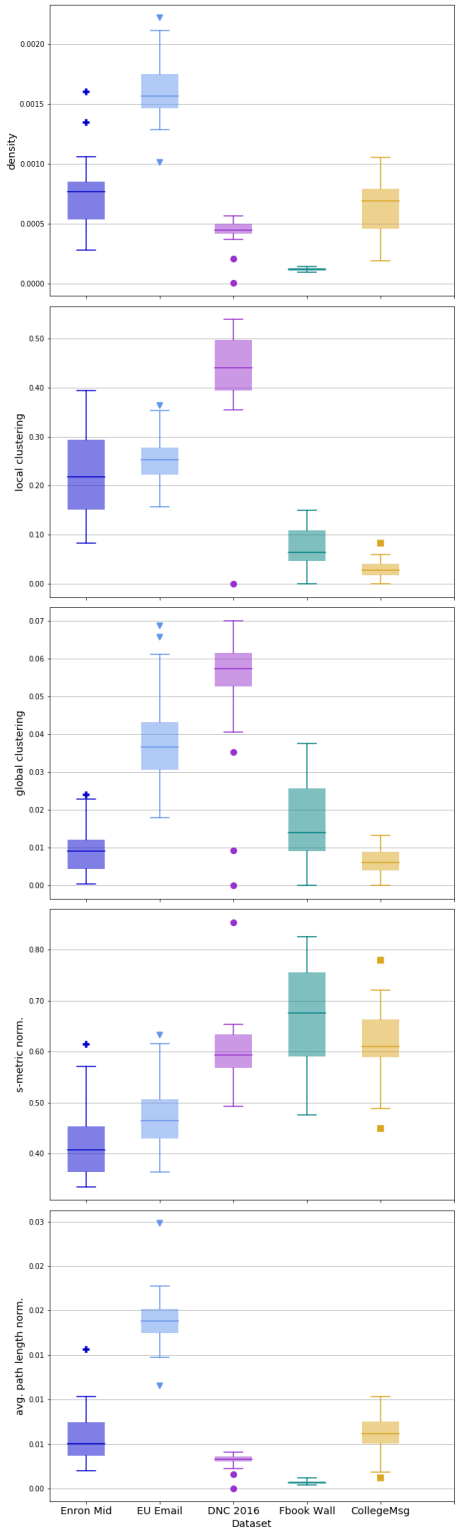
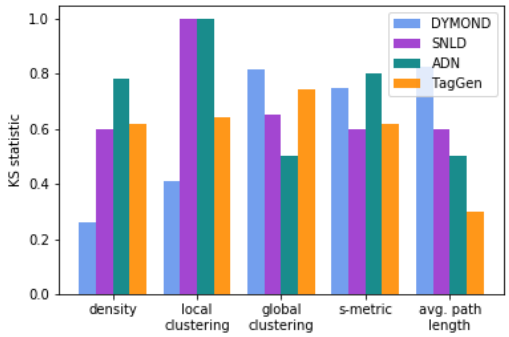
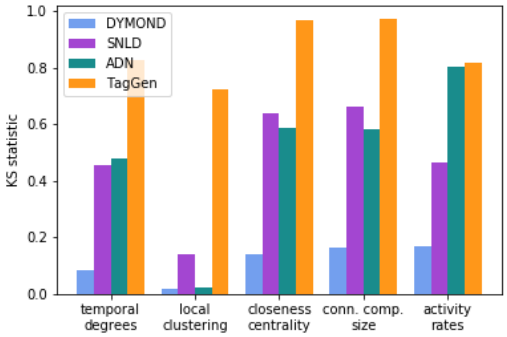


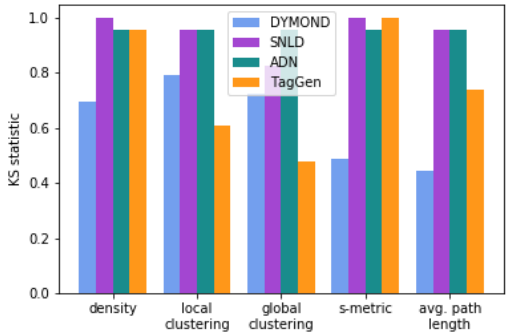
Figure 4.5. Dataset Graph Structure Metrics



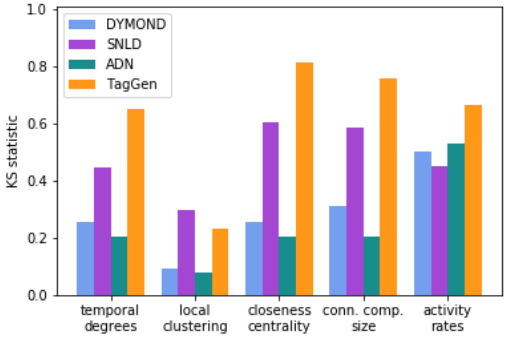
(a) Enron: Graph Structure Metrics



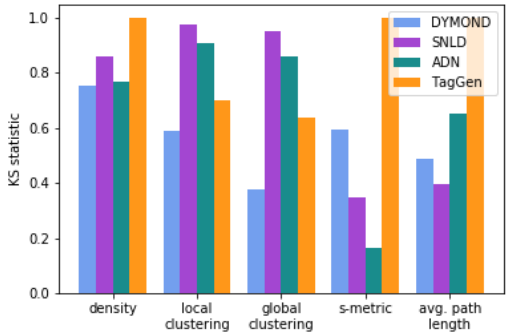
(b) Enron: Node Behavior Metrics



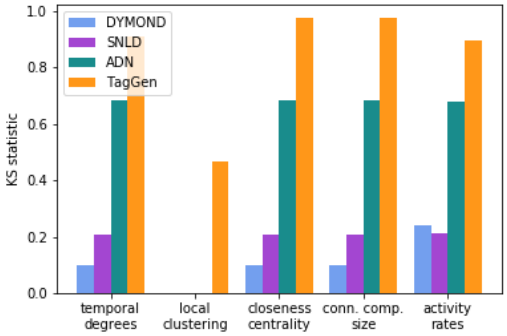
(c) DNC: Graph Structure Metrics



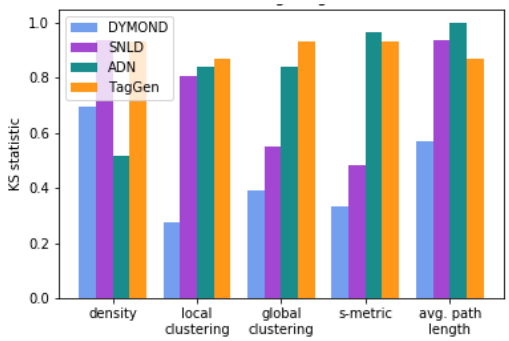
(d) DNC: Node Behavior Metrics



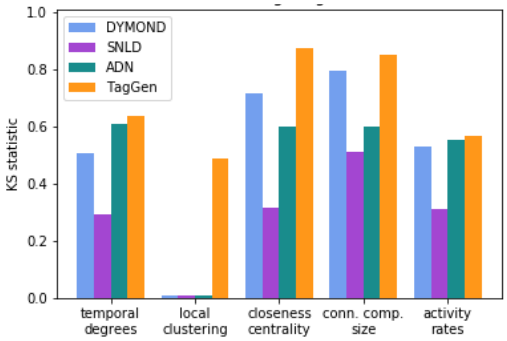
(e) Facebook: Graph Structure Metrics



(f) Facebook: Node Behavior Metrics



(g) CollegeMsg: Graph Structure Metrics



(h) CollegeMsg: Node Behavior Metrics

Figure 4.6. KS statistic of Graph Structure Metrics

Table 4.4. Notations and Symbols

Symbol	Description
$\mathbf{G} = \{G_1, \ldots, G_T\}$	dynamic temporal network
$t \in [1, \ldots, T]$	timestep (time-window)
$G_t = (V_t, E_t, S_t)$	graph snapshot at time t
$V_t \subseteq V$	set of active nodes at timestep t
$E_t \subseteq E$	set of edges at timestep t
$S_t\Big((u, v)\Big)$	list of timestamps for edge (u, v)
M	set of motifs
M^T	motif types
M^E	motif edges
M^S	motif timesteps
$c_t(u', v')$	num. times times $(u', v') \in E_t$
$\Big\lvert N^{(\mathrm{i})}(u', v') \Big\rvert$	num. of motifs type i

4.10 Additional Algorithms

Algorithm 4: `GetActiveNodes`

input : T num. of timesteps
N num. of nodes
λ_V node arrival rate
output: V active nodes per timestep

begin
  $V_t \leftarrow \emptyset,\ \forall \mathrm{j} \in [1, \ldots, T]$
  for $v \in [1, \ldots, N]$ **do**
    $a \sim Exp(\lambda_V)$ `// sample arrival time`
    for $t \in [a, \ldots, T]$ **do**
      `// add to active nodes each timestep` t
      $V_t \leftarrow V_t \cup \{v\}$
  $V = V_1 \cup \ldots \cup V_T$

Algorithm 5: `SampleMotifTimesteps`

input : t timestep triplets become active
M_t sampled motifs
M^T motif types
$\lambda_M = (\lambda_M^{(1)}, \ldots, \lambda_M^{(3)})$ rates distribution
output: M_t^S motif timestamps

begin
  for $\{u, v, w\} \in M_t$ **do**
    $\mathrm{i} \leftarrow M^T\big(\{u, v, w\}\big)$ `// motif type`
    `// sample inter-arrival rate`
    $\beta_M\big(\{u, v, w\}\big) \sim Exp\big(\lambda_M^{(\mathrm{i})}\big)$
    $\lambda_M\big(\{u, v, w\}\big) \leftarrow \frac{1}{\beta_M(\{u,v,w\})}$
    $prev \leftarrow t$ `// first time motif can appear`
    `// sample inter-arrival time`
    $next \sim Exp\big(\lambda_M\big(\{u, v, w\}\big)\big)$
    while $prev + next < T$ **do**
      `// save timestamp to list`
      $M_t^S\big(\{u, v, w\}\big)$.`append`$(prev + next)$
      $prev \leftarrow prev + next$
      `// sample next inter-arrival time`
      $next \sim Exp\big(\lambda_M\big(\{u, v, w\}\big)\big)$

Algorithm 6: `PlaceMotifEdges`

input : M_t sampled motifs
M_t^T motif types
M_t^R motif node roles
output: M_t^E edges for motifs in M_t

begin
  for $m \in M_t$ **do**
    if $M_t^T(m) = 3$ **then** `// triangle`
      $M_t^E(m) \leftarrow \binom{m}{2}$
    else if $M_t^T(m) = 2$ **then** `// wedge`
      $h \leftarrow M_t^R(m, \texttt{hub})$
      $s_1, s_2 \leftarrow M_t^R(m, \texttt{spoke})$
      $M_t^E(m) \leftarrow \{(h, s_1), (h, s_2)\}$
    else if $M_t^T(m) = 1$ **then** `// 1-edge`
      $\mathrm{e}_1, \mathrm{e}_2 \leftarrow M_t^R(m, \texttt{equal2})$
      $M_t^E(m) \leftarrow \{(\mathrm{e}_1, \mathrm{e}_2)\}$

Algorithm 7: `ConstructGraph`

input : M motifs
M^S motif timestamps
M^E motif edges
output: $\mathbf{G}' = \{G'_1, \ldots, G'_T\}$, where $G'_t = (V_t, E_t, S_t)$

begin
  $E_t \leftarrow \emptyset,\ \forall t \in [1, \ldots, T]$
  $M_t := \Big\{ m \in M : t \in M^S(m) \Big\}$
  for $\{u, v, w\} \in M_t$ **do**
    `// Place edges`
    $E_t \leftarrow E_t \cup M^E\Big(\{u, v, w\}\Big)$
    `// Add timestamps to edges`
    $S_t := \Big[(u', v') \in M^E\left(\{u, v, w\}\right) \mapsto t \Big]$

Algorithm 8: `GetMotifsGraph`

input : $\mathbf{G} = \{G_1, \ldots, G_T\}$ graph to learn from
T num. of timesteps
output: M motifs in G
M^T motif types

begin
  $M \leftarrow \emptyset$; $M^T \leftarrow \emptyset$
  for $t \in [1, \ldots, T]$ **do**
    for $(u, v) \in E_t$ **do**
      if $u \neq v$ **then**
        for $w \in V_t$ **do**
          if $w \neq u$ **and** $w \neq v$ **then**
            `// number unique edges is motif type i`
            $\mathrm{i} := \Big| M^E\Big(\{u, v, w\}\Big) \Big|$
            if $\{u, v, w\} \notin M$ **then**
              $M \leftarrow M \cup \Big\{ \{u, v, w\} \Big\}$
              $M^T\Big(\{u, v, w\}\Big) \leftarrow \mathrm{i}$
            else if $\mathrm{i} > M^T\Big(\{u, v, w\}\Big)$ **then** `// prefer higher-order motif types`
              $M^T\Big(\{u, v, w\}\Big) \leftarrow \mathrm{i}$

Algorithm 9: `GetNodeRoleCounts`

input : M motifs in $\mathbf{G}$
M^T motif types
M^E motif edges
C^M count times motifs appear
T num. of timesteps

output: $count(v, r)$ node role counts

begin
  $count(v, r) = 0,\ \forall v \in V,\ \forall r \in R$ `// init. role counts`
  for $t \in [1, \ldots, T]$ **do**
    for $(u, v) \in E_t$ **do**
      if $|N^{(3)}(u, v)| > 0$ **then** `// triangles`
        $\omega^{(3)} := \frac{min(c_t(u,v), |N^{(3)}(u,v)|)}{|N^{(3)}(u,v)|}$ `// role weight triangle`
        for $w \in \left(\mathcal{N}_t(u) \cap \mathcal{N}_t(v)\right)$ **do**
          for $n \in [u, v, w]$ **do**
            $count(n, \texttt{equal3})$ += $\frac{\omega^{(3)}}{3}$
      if $|N^{(2)}(u, v)| > 0$ ***and*** $r_t^{(2)}(u, v) > 0$ **then** `// wedges`
        $\omega^{(2)} := \frac{min(r_t^{(2)}(u,v), |N^{(2)}(u,v)|)}{|N^{(2)}(u,v)|}$ `// role weight wedge`
        for $w \in \left(\mathcal{N}_t(u) \oplus \mathcal{N}_t(v)\right)$ **do**
          for $n \in [u, v, w]$ **do**
            $r \leftarrow$ `GetRoleTimestep(`$E_t, \{u, v, w\}, n$`)`
            if $r =$ `hub` **then**
              $count(n, \texttt{hub})$ += $\frac{\omega^{(2)}}{2}$
            else
              $count(n, \texttt{spoke})$ += $\omega^{(2)}$
      if $|N^{(1)}(u, v)| > 0$ ***and*** $r_t^{(1)}(u, v) > 0$ **then** `// 1-edge`
        $\omega^{(1)} := \frac{min(r_t^{(1)}(u,v), |N^{(1)}(u,v)|)}{|N^{(1)}(u,v)|}$ `// role weight 1-edge`
        for $w \in \left(V - \left(\mathcal{N}_t(u) \cup \mathcal{N}_t(v)\right)\right)$ **do**
          for $n \in [u, v, w]$ **do**
            $r \leftarrow$ `GetRoleTimestep(`$E_t, \{u, v, w\}, n$`)`
            if $r =$ `equal2` **then**
              $count(n, \texttt{equal2})$ += $min(r_t^{(1)}(u, v), |N^{(1)}(u, v)|)$
            else
              $count(n, \texttt{outlier})$ += $\omega^{(1)}$

5. ATTRIBUTED DYNAMIC NETWORK GENERATIVE MODEL FROM ATTRIBUTED NODE-ACTIVITY AND ROLES

5.1 Introduction

Graphs are pervasive in our world, serving as valuable tools for studying complex systems across diverse domains such as social, biological, computing, and communication networks. These networks provide insights into the underlying structures and behaviors that shape our interconnected world. Creating synthetic graphs proves beneficial for assessing systems across diverse structures and information sharing without compromising private data. Previously, complex networks with temporal characteristics were examined as static graphs, either by modeling them as growing networks or by consolidating temporal data into a single graph. In reality, the majority of these networks possess dynamic properties and undergo continuous evolution, with nodes and edges constantly being added or removed. For instance, in social networks, users establish or remove connections with each other through actions like following, mentioning, and replying. Moreover, the attributes of users, such as textual features in their generated content, also change. These two dynamics—social links and user attributes—may influence each other. In the context of academic co-authorship networks, researchers seek collaborators (represented as neighboring nodes) who possess similar or complementary knowledge, and the content generated is the papers they co-author. Additionally, their personal research interests may evolve based on new collaborations. The interplay between the evolving graph structure and the changing attributes of its nodes introduces a complex and valuable area of study.

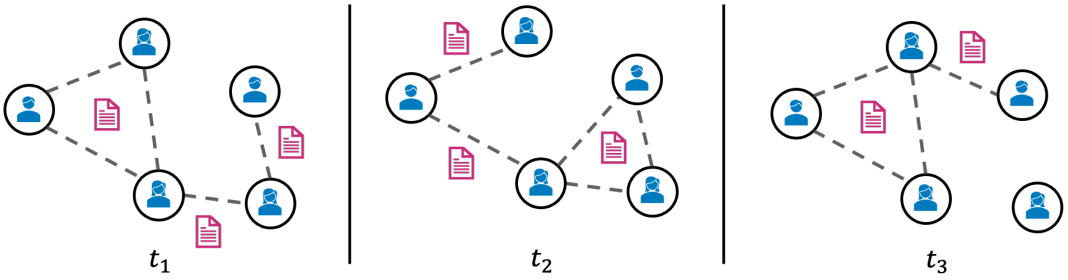


Figure 5.1. Academic Co-authorship Network Example, with graph structure and attributes/content changing over time

In this work, we propose DYnamic Attributed Node rolEs (DYANE), a generative model for dynamic networks with content. Since lengthy content generation is influenced by more than just network interactions, we focus on modeling content *embeddings* as node attributes that evolve over time, influenced by network interactions. We employ temporal motifs as building blocks of network structure and extend motif node roles for content embedding generation. The use of motifs and node roles can capture correlations in node connections and activity. Modeling the network content embedding attributes with higher-order structures (e.g., motifs) can further improve the quality of the networks generated by exploiting any overlaps of nodes' latent interests. To this end, we design a *Node Roles* Graph Convolutional Network (GCN) and a *Motif Types* Convolutional Neural Network (CNN) for sampling motifs (and their configuration) based on nodes' roles and their content embedding attributes.

Graph structure evaluation metrics, such as density and clustering coefficient for example, have been designed for static graphs. With recent work in dynamic network generative models, there is a need of metrics that consider the temporal dimension. Evaluating generative models for attributed dynamic networks poses another challenge. On static graphs, we can measure the attribute auto-correlation of nodes and their neighbors. In dynamic networks, we need to also consider the temporal dimension. We tailor graph metrics to consider temporal structure and node behavior. To address the second challenge, we evaluate the content embeddings generated by nodes using the distribution of attributes from graph snapshots, with metrics based on embedding distances. We evaluate our proposed model, using these three sets of metrics (for temporal structure, node content embeddings and behavior), against three recent models on four real-world datasets. The results show that DYANE generates networks with similar node topic and behavior to the observed networks and better graph structure overall, compared to the other models.

To summarize, we make the following contributions: (1) we developed a generative model for dynamic networks with content, that combines a GCN and CNN for generating synthetic graphs with new structure similar to the observed input network, and is also able to sample *new* content embeddings (previously unseen), and (2) we outline a methodology to evaluate nodes' latent interests over time, based on their content embeddings and keywords extracted from the content (as topic representations).

The rest of the chapter is organized as follows: In Subsection 5.1.1, we formally define the problem of attributed dynamic network generation. We then go over related work to show where our model fits in the literature in Section 5.2. Section 5.3 presents our proposed model, DYANE. In Section 5.4, we present our evaluation methodology, metrics, datasets, and baselines used. Experimental results and discussion is in Section 5.5. Finally, we present our conclusions in Section 5.8.

5.1.1 Problem Definition

The problem of generating dynamic networks with content embeddings is a specific example of generating temporal networks with changing structure and attributes. The goal of attributed dynamic network generation is to generate a new synthetic dynamic network similar to an observed network. We formally define the problem as follows:

Problem 5. ***Attributed Dynamic Network Generation***

Input 5. *An attributed dynamic network* $\mathbf{G} = \{G_1, \ldots, G_T\}$*, where* $G_t = (V, E_t, X_t)$ *is a graph snapshot,* V *is the set of nodes,* E_t *is the set of edges at time* t*, and* X_t *is the set of attributes at time* t*.*

Output 5. *An attributed dynamic network* $\mathbf{G}' = \{G'_1, \ldots, G'_{T'}\}$*, where the distribution of graph structure for* $\mathbf{G}'$ *matches* $\mathbf{G}$*, the node behavior of a specific node* $v_{i'}$ *in* $\boldsymbol{G'}$ *should be similar to a specific node* v_i *in* $\mathbf{G}$*, and the distribution of attributes for* $\mathbf{G}'$ *matches* $\mathbf{G}$*.*

Concretely, consider an arbitrary graph statistic $s(G)$ (e.g., density). Then the distribution of statistic values observed in the input attributed dynamic network $\mathbf{s}_{in} = \{s(G_1), \ldots, s(G_T)\}$ should match the distribution of statistic values observed in the output attributed dynamic network $\mathbf{s}_{out} = \{s(G'_1), \ldots, s(G'_{T'})\}$. Likewise, take any node statistic $\mathbf{s}(v_i|\mathbf{G})$ (e.g., node degree). Then, using the temporal distribution of values for a node $\mathbf{s}(v_i|\mathbf{G}) = \{s(v_i|G_1), \ldots, s(v_i|G_T)\}$, the distribution of values for all nodes in the input dynamic network $\{\mathbf{s}(v_j|\mathbf{G})\}_{j \in \mathbf{G}}$ should match the distribution of values for all nodes in the output dynamic network $\{\mathbf{s}(v_{j'}|\mathbf{G}')\}_{j' \in \mathbf{G}'}$. Similarly, the distribution of attributes in the *input* attributed dynamic network $\mathbf{X} =$

$\{X_1, \ldots, X_T\}$ should match the attributes observed in the *output* attributed dynamic network $\mathbf{X}' = \{X'_1, \ldots, X'_{T'}\}$.

5.2 Related Work

We describe related work on generative network models and in Subsection 5.2.4 we compare them and discuss open areas to explore.

5.2.1 Static Graph Models

Many generative models for static graphs have aimed to generate synthetic graphs that can simulate real-world networks (see Subsection 2.2.1). Statistical models where model parameters can be estimated from observed data, such as Exponential Random Graph Models (ERGMs) [4, 6, 73] and the Kronecker Product Graph Model (KPGMs) [17, 19, 47], allow generation of graphs by sampling from the estimated distribution. However, these methods focus on capturing either global or local graph properties, but not both.

More recently, deep graph neural networks have received a lot of attention for their ability to model realistic graphs. GraphGAN [22] and NetGAN [23] are both based on generative adversarial networks. GraphGAN learns from each node's connectivity distribution, while NetGAN learns from a distribution of biased random walks. GraphRNN [24] learns from a representative set of graphs and generates the adjacency of a graph by generating the adjacency vector of each node.

5.2.2 Temporal Graph Models

Initial models for temporal or dynamic networks (where links appear and disappear, such as social-network communication patterns) focused on modeling the edges over time, ignoring higher-order structures [48–50]. Static Networks with Link Dynamics (SNLD) [49] first creates a static graph from a degree distribution, and for each link it generates a sequence of contacts. Activity-Driven Network (ADN) [59, 60] starts with an empty snapshot, marks nodes as active according to a probability, and (if active) connects it to m random nodes.

Although traditionally most graph models have been edge-based, motifs have been established as building blocks for the structure of networks [51, 53, 56]. Thus, modeling motifs can help to generate the graph structure seen on real-world networks and capture correlations in node connections and activity. The Structural Temporal Model (STM) [57] finds structural motifs in an input graph and estimates their arrival rates. Afterwards, using a variation of preferential attachment, STM produces aggregated temporal networks (i.e., edges will not be removed once they are placed). DYnamic MOtif-NoDes (DYMOND) [74] is a probabilistic dynamic-graph generative model that samples graphs with realistic structure and temporal node behavior using motifs. DYMOND considers both the dynamic changes in overall graph structure using temporal motif activity and the roles nodes play in motifs. DynamicTriad [75] learns node embeddings by modeling the triadic closure process. DynGraph2Vec [76] learns the structure of evolution in dynamic graphs and can predict unseen links with higher precision. TagGen [64] is a dynamic graph neural network model that takes into account higher-order structure by using node-biased temporal random walks to learn the network topology and temporal dependencies. TG-GAN [77] captures structural and temporal patterns of dynamic networks using temporal walks and a discriminator.

5.2.3 Attributed Graph Models

Although there are various models for generating attributed graphs, such as MAG [25], AGM [26], and CSAG [27], they have focused on static graphs (See Subsection 2.2.1). The dynamic attribute network embedding model (Dane) [78] uses an activeness-aware neighborhood embedding method (GraphSAGE [39]) to extract the higher-order neighborhood information at each given timestamp. The activeness-aware mechanism emphasizes more on nodes that are active in social activities. Dane is used to predict the network status at the next timestamp (i.e., the link connections and node categories at $t + 1$). Dynamic Graph Normalizing Flows (DGNF) is a graph representation model for dynamic attributed graphs that can be used for link prediction [79]. CTWalk models dynamic attributed networks, capitalizing on temporal random walks and conditional GANs [80]. Specifically, it builds upon TagGen [64], for temporal edge generation, and CTGAN [81], for generating

node and edge attributes seen in the input graph. To the best of our knowledge, CTWalk is the only generative model for attributed dynamic networks—that can generate synthetic graphs—apart from our proposed model.

5.2.4 Discussion

Table 5.1 shows a comparison of the models presented in the related work with our proposed model DYANE for attributed dynamic networks, presented in Section 5.3. Few models have been proposed for generating synthetic dynamic graphs. Although motifs have been shown to be building blocks of real-world networks [51], DYMOND (Section 4.4) is the only dynamic network generative model that considers motifs. Apart from DYMOND, only TagGen [64] and TG-GAN [77] consider higher-order graph structure, but can only generate the same number of timesteps as the input graph.

For attributed dynamic networks, Dane [78] and CTWalk [80]. However, Dane cannot generate entirely synthetic graphs. Our proposed DYANE model will generate synthetic attributed dynamic graphs. CTWalk generates network attributes based on edges or individual nodes, ignoring the higher-order structures. Additionally, CTWalk only generates discrete attributes that were observed in the input graph (i.e., cannot generate new attributes based on input graph).

5.3 DYnamic Attributed Node rolEs Model

We propose a novel method for generating attributed dynamic networks. We assume that node interactions are determined by the users' latent interests (static or slowly changing) and the way of expressing those interests. Our proposed model, DYnamic Attributed Node rolEs (DYANE), extends motif-based node roles (Figure 5.2) to roles that generate content embeddings. The model makes the following assumptions about the graph generative process:

1. All nodes remain active
2. Nodes have a probability distribution over role types that they play in motifs
3. Node attributes are aggregated from edges (interactions)
4. Node latent topics/interests vary over time

Table 5.1. Comparison of Graph Generative Models

		Temporal Edges		Higher-Order			
Model	**Synthetic**	**Aggregated**	**Dynamic**	**Structure**	**Motifs**	**Node Roles**	**Attributes**
ERGMs	✓						
KPGMs	✓			✓			
GraphGAN	✓						
NetGAN	✓			✓			
GraphRNN	✓			✓			
SNLD	✓	✓	✓				
ADN	✓	✓	✓				
TagGen	✓	✓	✓	✓			
TG-GAN	✓	✓	✓	✓			
dyngraph2vec		✓	✓				
STM		✓		✓	✓		
DynamicTriad		✓		✓	✓		
DYMOND	✓	✓	✓	✓	✓	✓	
AGM	✓			✓			✓
CSAG	✓			✓			✓
Dane		✓					✓
CTWalk	✓	✓	✓				✓
DYANE	✓	✓	✓	✓	✓	✓	✓

5. Node interactions are determined by the node roles and latent topics/interests

First we describe DYANE's generative process below. Then we outline our approach to estimate model parameters from an observed dynamic network. In the generation process, the motifs are sampled from a probability distribution based on the latent interests of the nodes and the roles they would have to play in a particular motif type, while also ensuring the motif type proportions in the graph are maintained. For example, in a wedge one node would be a hub and the other two would be spokes (Figure 5.2).

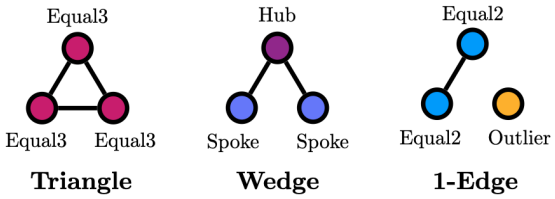


Figure 5.2. Motif Types and Node Roles

The motivation for our modeling approach is based on the following conjectures: (1) modeling higher-order structures (i.e., motifs), will capture the underlying distribution of graph structure, and (2) considering the nodes' latent interests and motif roles will also capture correlations in node content embeddings, connections, and activity.

5.3.1 Generative Process

The overall generative process is described in Figure 5.3, and the model architecture is illustrated in Figure 5.4. We model the time until nodes become active as Gamma random variables with the same rate, and the motif inter-arrivals as a Bernoulli Process, whose parameters depend on the motif type. Therefore, we use Geometric random variables as the inter-arrival times and a Beta prior on the distribution of inter-arrival rates for each motif type.

We first sample the arrival times for all nodes from a Gamma distribution (Figure 5.3a and equation 5.1).

$$x_v \sim \Gamma(\alpha, \beta)$$

$$g := \left[v \in V \mapsto \lfloor |x_v| \rfloor \right] \tag{5.1}$$

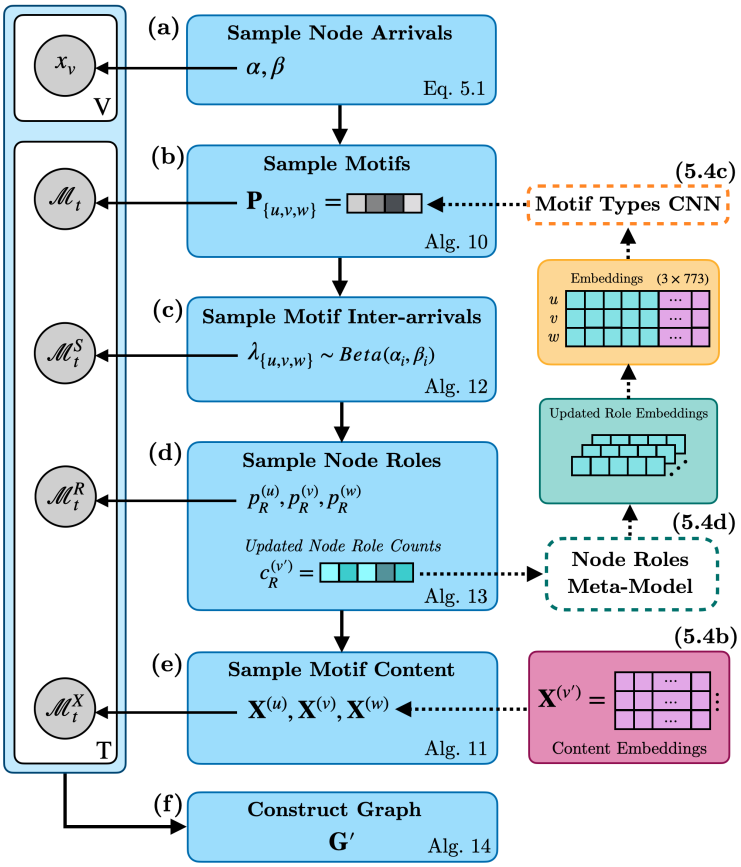


Figure 5.3. DYANE Generative Process

At each timestep t, we first calculate any new triplets $\mathcal{U}_t$ (Equation 5.2) that can be sampled as motifs from the set of active nodes V_t' (Equation 5.3), where $g(v)$ is the arrival time of node v (Equation 5.1).

$$\mathcal{U}_t = \left\{ \{u, v, w\} \subset V_t' : \{u, v, w\} \notin \mathcal{U}_{t-1} \right\} \tag{5.2}$$

$$V_t' = \{ v \in V : g(v) \leq t \} \tag{5.3}$$

In figure 5.3b, we sample motifs from these new triplets in $\mathcal{U}_t$, using the motif type probabilities obtained from the Motif Types CNN (Algorithm 10 and line 3), which are based on the node roles and content embeddings (Figure 5.4c). For the motif inter-arrivals

(Figure 5.3c), we first sample an inter-arrival rate $\lambda_{\{u,v,w\}}$ from a Beta distribution with parameters α_i, β_i, which depend on the motif type i sampled. We sample motif inter-arrival times using that rate (Algorithm 12). Afterwards, we sample node roles for each motif node and update the counts for node roles assigned (Figure 5.3d). The updated counts (Algorithm 13 and line 15) are then used to update the node role embeddings with the meta-model (Figure 5.4d).

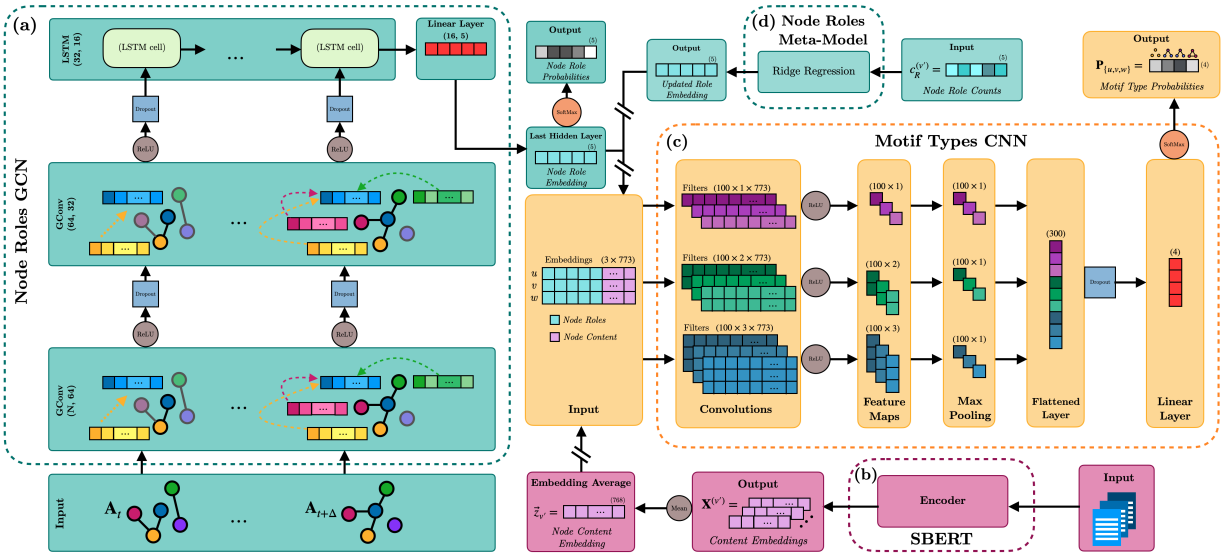


Figure 5.4. DYANE Model Architecture. (a) Node Roles GCN, (b) Content Model, (c) Motif Types CNN, (d) Node Roles Meta-Model

In figure 5.3e, we sample content embeddings for each timestep the motif will be in (Algorithm 11). In line 3, we only consider the content embeddings of "influential" nodes (i.e., nodes that are part of the motif's edges). We "seed" the embedding with the influential nodes' average content embeddings $\vec{z}_{v'}$ (line 6). For each time the motif will appear, we create a new content embedding by adding Gaussian noise (line 10), using influential nodes' content embeddings variance $\vec{y}_{v'}$. Lastly, we assemble $\mathbf{G}'$ using the motifs $\mathcal{M}$, inter-arrival times $\mathcal{M}^S$, node roles $\mathcal{M}^R$, and content embeddings sampled $\mathcal{M}^X$ (Algorithm 14 and figure 5.3f).

Algorithm 10: `SampleMotifs`

input: $\mathcal{U}_t,\ q(\mathrm{i})$
output: $\mathcal{M}_t$ `// sampled motifs`
$\mathcal{M}_t^T$ `// motif types`

begin
  $n := \left[\mathrm{i} \in [3,2,1] \mapsto q(\mathrm{i}) \cdot |\mathcal{U}_t|\right]$
  `// Get probabilities from CNN`
  $\mathbf{P} \leftarrow \left[\{u,v,w\} \in \mathcal{U}_t \mapsto \texttt{MotifTypesCNN}(u,v,w)\right]$
  for $\mathrm{i} \in [3,2,1]$ **do**
    $\mathcal{U}_t' := \mathcal{U}_t,\ \mathbf{P}' := \mathbf{P}$ `// initialize`
    if $|\mathcal{U}_t| >$ `max_size` **then**
      `// Use a reservoir if too large`
      $\mathcal{U}_t' \leftarrow \texttt{ReservoirSampling}(\mathcal{U}_t, \texttt{max\_size})$
      $\mathbf{P}' \leftarrow \left[\{u,v,w\} \in \mathcal{U}_t' \mapsto \mathbf{P}_{\{u,v,w\}}\right]$
    $\mathcal{M}^{(\mathrm{i})} \sim \mathit{Mult}(\mathbf{P}'^{(\mathrm{i})},\ n^{(\mathrm{i})})$ `// sample motifs`
    $\mathcal{M}_t^T := \left[m \in \mathcal{M}^{(\mathrm{i})} \mapsto \mathrm{i}\right]$ `// save motif types`
    $\mathcal{U}_t \leftarrow \mathcal{U}_t - \mathcal{M}^{(\mathrm{i})}$ `// triplets left to sample`

Algorithm 11: `SampleMotifContent`

input: $\mathbf{X},\ \mathcal{M}_t,\ \mathcal{M}_t^S,\ \mathcal{M}_t^R$
output: $\mathcal{M}_t^X$ `// sampled motif content`

begin
  for $m \in \mathcal{M}_t$ **do** `// ea. motif sampled`
    `// get influential nodes from motif roles`
    $m' := \left\{v' \in m : \mathcal{M}_t^R(m,v') \neq \texttt{outlier}\right\}$
    `// get embedding mean and variance ea. node`
    $\mathbf{Y} \leftarrow \left[\vec{y}_{v'} = \texttt{var}(\mathbf{X}^{(v')}) \mid v' \in m'\right]$
    $\mathbf{Z} \leftarrow \left[\vec{z}_{v'} = \texttt{avg}(\mathbf{X}^{(v')}) \mid v' \in m'\right]$
    $\vec{b} \leftarrow \texttt{avg}(\mathbf{Z})$ `// seed embedding`
    $\vec{\sigma}^2 \leftarrow \texttt{avg}(\mathbf{Y}),\ \vec{\mu} \leftarrow [0,\ldots,0]$
    for $t \in \mathcal{M}_t^S(m)$ **do** `// ea. timestep of motif`
      $\vec{n} \sim N(\vec{\mu}, \vec{\sigma}^2)$ `// sample Gaussian noise`
      $\mathcal{M}_t^X(m) \leftarrow \vec{b} + \vec{n}$ `// new embedding`

5.3.2 Learning

Given an observed dynamic graph $\mathbf{G}$, we estimate the input parameters for our generative process.

Node Arrivals

We begin by fitting a Gamma distribution to the node arrival times, estimating the shape and rate parameters (α and β respectively). We estimate the arrival time x_v of a node v, with the timestep in which v had its first edge:

$$\hat{x}_v = \arg\min_t \mathbb{1}(v \in V_t) \tag{5.4}$$

$$X = (\hat{x}_1, \ldots, \hat{x}_{|V|}) \tag{5.5}$$

$$X \sim \Gamma(\alpha, \beta) \tag{5.6}$$

Motif Proportions

We iterate over the graph snapshots G_t to find the 3-node motifs in each timestep t. To determine the motif type i of a motif $\{u, v, w\}$, we keep track the edge configurations we find and select the higher-order motif type. For example, if we observe the triplet $\{u, v, w\}$ is a triangle at timestep t and we previously saw it as a wedge, we update its type to a triangle.

Then, we use the maximum likelihood estimate (MLE) of the Binomial distribution to calculate the motif proportions $q(\mathrm{i})$ of each type in the graph, where i corresponds to the number of edges in the motif (i.e., i = 1 for a 1-edge, i = 2 for a wedge, and i = 3 for a triangle motif):

$$\hat{q}(\mathrm{i}) = \begin{cases} \frac{\left|\left\{\{u,v,w\} \in \mathcal{M} : \mathcal{M}^T(\{u,v,w\}) = \mathrm{i}\right\}\right|}{\binom{|V|}{3}}, & \text{if i} \in [1,2,3] \\ 1 - \sum_{\mathrm{i}=1}^{3} \hat{q}(\mathrm{i}), & \text{otherwise} \end{cases} \tag{5.7}$$

where $\mathcal{M}$ is the set of observed motifs, $\binom{|V|}{3}$ is all possible 3-node combinations, and $\{u, v, w\}$ is a motif consisting of the nodes u, v, w.

Motif Inter-Arrivals

We assume that the motif inter-arrivals come from a Bernoulli process. The inter-arrival times of each observed motif $\{u, v, w\}$ follow a Geometric distribution with parameters $\lambda_{\{u,v,w\}}$, and the number of times the motif appears follows a Binomial distribution with parameters $n, \lambda_{\{u,v,w\}}$, where $n = T$ is the number of trials (timesteps). When n is known, we can estimate the parameter $\lambda_{\{u,v,w\}}$ with the MLE:

$$\widehat{\lambda}_{\{u,v,w\}} = \frac{\sum_{t=1}^{T} d_t\Big(\{u, v, w\}\Big)}{T} \tag{5.8}$$

Given that we consider all possible 3-node motifs, there will be common edges among them. Thus, to count the number of times a motif appeared, we use edge-weights based on how many motifs share that edge. We use these edge-weighted counts d_t, per timestep t, to estimate the inter-arrival rate for each motif $\{u, v, w\}$ (Equation 5.9a). The weights $\omega_t^{(\mathrm{i})}$ will depend on the motif type i of $\{u, v, w\}$ and are calculated for each edge of the motif (Equation 5.10a).

$$d_t\Big(\{u, v, w\}\Big) = \frac{\sum_{(u',v') \in \mathrm{e}_t(\{u,v,w\})} \omega_t^{(\mathrm{i})}(u', v')}{|\mathrm{e}_t(\{u, v, w\})|} \tag{5.9a}$$

$$\mathrm{e}_t(\{u, v, w\}) = \Big\{\{u', v'\} \subset \{u, v, w\} : (u', v') \in E_t\Big\} \tag{5.9b}$$

For a motif $\{u, v, w\}$, we calculate the weight of its edge (u', v') using the count for the edge in the timestep window and considering its motif type i (Equation 5.10a). We give larger edge-weight to motif types with more edges, since they are more likely to produce the observed edges. This also ensures that motif types with smaller proportion $q(\mathrm{i})$ (Equation 5.7) have a high enough inter-arrival rate to show up (i.e., triangles).

$$\omega_t^{(\mathrm{i})}(u', v') = \frac{min\Big(r_t^{(\mathrm{i})}(u', v'),\ n_t^{(\mathrm{i})}(u', v')\Big)}{n_t^{(\mathrm{i})}(u', v')} \tag{5.10a}$$

$$r_t^{(\mathrm{i})}(u', v') = \begin{cases} c_t(u', v'), & \text{if i} = 3 \\ r_t^{(\mathrm{i}+1)}(u', v') - n_t^{(\mathrm{i}+1)}(u', v'), & \text{if i} \in [1, 2],\ r_t^{(\mathrm{i}+1)}(u', v') > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.10b}$$

$$n_t^{(\mathrm{i})}(u', v') = \begin{cases} |\mathcal{N}_t(u) \cap \mathcal{N}_t(v)|, & \text{if i} = 3 \\ |\mathcal{N}_t(u) \oplus \mathcal{N}_t(v)|, & \text{if i} = 2 \\ |V - \Big(\mathcal{N}_t(u) \cup \mathcal{N}_t(v)\Big)|, & \text{if i} = 1 \end{cases} \tag{5.10c}$$

where $|n_t^{(\mathrm{i})}(u', v')|$ is the number of motifs of type i that share edge (u', v'), the number of times (u', v') appears in E_t is $c_t(u', v')$, the remaining edge count is $r_t^{(\mathrm{i})}(u', v')$ for motif type i, and $\mathcal{N}_t(u),\ \mathcal{N}_t(v)$ are the neighbor's at time t of nodes u and v, respectively.

We use Beta distributions with parameters $\alpha_{\mathrm{i}},\ \beta_{\mathrm{i}}$ as a prior on the inter-arrival rates of motifs with the same type, for each motif type i. Note that we do not need to estimate rates for the empty motif type (i = 0). During the generative process, we sample inter-arrivals for new motifs of type i from the corresponding prior distribution.

Node Role Probabilities

For every node v_{i}, we estimate it's node role probabilities from the counts that v_{i} had each role $r \in \mathcal{R}$, with the MLE of the Multinomial distribution. For a node v_{i}, let $p_{\mathrm{i}}^{\text{roles}}$ be its node role probabilities, $x_{\mathrm{i}}^{\text{roles}}$ its weighted node role counts (elsewhere noted as $c_R^{(v_{\mathrm{i}})}$ for conciseness), and $n = \sum_r x_{\mathrm{i},r}^{\text{roles}}$ the total number of role counts, such that $x_{\mathrm{i}}^{\text{roles}} \sim Mult(n, p_{\mathrm{i}}^{\text{roles}})$. We then estimate the node role probabilities as follows:

$$\hat{p}_{\mathrm{i}}^{\text{roles}} = \frac{x_{\mathrm{i}}^{\text{roles}}}{n} = \left(\frac{x_{\mathrm{i},1}^{\text{roles}}}{n}, \ldots, \frac{x_{\mathrm{i},k}^{\text{roles}}}{n}\right) \tag{5.11}$$

where $\mathcal{R} = \{\texttt{equal3}, \texttt{hub}, \texttt{spoke}, \texttt{equal2}, \texttt{outlier}\}$ is the set of possible roles, $k = |\mathcal{R}|$ is the number of roles, and $x^{\text{roles}}_{\text{i},r} = \mathit{count}(v_\text{i}, r)$ is the weighted count of times that node v_i had role r (Algorithm 15). Edge-weights (Equation 5.10a) are used to avoid over-counting the roles for motifs of the same type with a shared edge.

Node Roles GCN

The Node Roles graph convolutional network (GCN) (Figure 5.4a) is used to obtain the initial node role embeddings that will be fed into the Motif Types CNN for training. The Node Roles GCN takes as input the graph adjacency $\mathbf{A}_t$ (with the identity matrix $\mathbf{I}$ as node features) at each timestep t, and each is passed individually through the GCN Convolution layers.

$$\begin{aligned}\mathbf{H}_t^{(l+1)} &= f\left(\mathbf{H}_t^{(l)}, \mathbf{A}_t\right) \qquad (5.12)\\ &= \text{ReLU}\left(\mathbf{A}_t\mathbf{H}_t^{(l)}\right)\mathbf{W}_t^{(l)}\\ \mathbf{H}_t^{(1)} &= \mathbf{I} \qquad (5.13)\end{aligned}$$

Each GCN layer has it's own $\mathbf{W}_t^{(l)}$ at each timestep, where $l \in [1, \ldots, L]$. Each final output from the GCN layers will be an input for an LSTM cell, where the sequence of LSTM cells combines the temporal aspect. The temporal input for an LSTM cell at time t is $\mathbf{H}_t^{(L+1)}$.

$$\begin{aligned}\mathbf{o}_\text{i} &= \text{LSTM}\left(\mathbf{H}_{1:T}^{(L+1)}(v_\text{i})\right) \qquad (5.14)\\ \mathbf{o}_\text{i}^{\text{roles}} &= \mathbf{W}_{\text{final}}\mathbf{o}_\text{i} + \mathbf{b}_{\text{final}} \qquad (5.15)\\ \hat{\mathbf{y}}_\text{i}^{\text{roles}} &= \text{softmax}\left(\mathbf{o}_\text{i}^{\text{roles}}\right) \qquad (5.16)\end{aligned}$$

Then, the output of the LSTM sequence $\mathbf{o}_\text{i}$, for node v_i, is passed through a Linear layer and a Softmax function to obtain the node role probabilities $\hat{\mathbf{y}}_\text{i}$ of node v_i. We train the Node Roles GCN for 200 epochs and use categorical cross-entropy as a loss function at the final

output layer. We use the estimated node role probabilities (Equation 5.11) as the ground truth $\mathbf{y}_{\text{i}}$ for node v_{i}.

$$\mathcal{L}(\hat{\mathbf{y}}_{\text{i}}^{\text{roles}}, \mathbf{y}_{\text{i}}^{\text{roles}}) = -\sum_{\text{i} \in V} \sum_{\text{j}}^{|\mathcal{R}|} y_{\text{i,j}}^{\text{roles}} \; log(\hat{y}_{\text{i,j}}^{\text{roles}}) \tag{5.17}$$

The last hidden layer of the Node Roles GCN is used as the initial node role embedding for a node.

Node Roles Meta-Model

The Node Roles Meta-Model is used to update the node role embeddings. As motifs and node roles are sampled at each timestep t in the generative process, the node role probability distributions must be updated. Similarly, the node role embeddings will be updated using the most recent node role counts c_R (Algorithm 13 and line 15).

We fit a Ridge Regression model (i.e., linear least squares with ℓ_2-norm regularization) using the node role counts as the input X_{roles} and the node role embeddings as the labels Y_{roles}. We use cross-validation during training and perform a grid search over the parameters. The loss function R is the squared ℓ_2 norm,

$$R(w) = \sum_{\text{i=j}}^{d} w_{\text{i}}^2 \tag{5.18}$$

$$\frac{1}{n}||Y_{\text{roles}} - X_{\text{roles}} w||_2^2 + \lambda \sum_{\text{j}=1^d} |w_{\text{j}}|^2 \rightarrow \underset{w \in \mathbb{R}^d}{\arg\min} \tag{5.19}$$

and the closed-form solution for w is:

$$w = (X_{\text{roles}}^{\top} X_{\text{roles}} + \lambda I)^{-1} X_{\text{roles}}^{\top} Y_{\text{roles}} \tag{5.20}$$

To update the node role embedding for a node v_{i}, we then pass the updated node role counts as input $x_{\text{i}}^{\text{roles}} = c_R^{(v_{\text{i}})}$ and we have:

$$\hat{\mathbf{y}}_{\text{i}}^{\text{roles}} = \sum_{\text{j}} w_{\text{j}} x_{\text{j}}^{\text{roles}\top} x_{\text{i}}^{\text{roles}} \tag{5.21}$$

Note that in this case $\hat{\mathbf{y}}_{\mathrm{i}}^{\mathrm{roles}}$ is the predicted node role embedding of node v_{i}, and the ground truth $\mathbf{y}_{\mathrm{i}}^{\mathrm{roles}}$ is the Node Roles GCN last hidden layer output $\mathbf{o}_{\mathrm{i}}^{\mathrm{roles}}$ (Equation 5.15).

Content Model

The Content Model (Figure 5.4b) is used to obtain the node content embeddings that will be fed into the Motif Types CNN as part of the input. We obtain the content embedding for a node, by passing all content authored by the node through a Sentence-Transformer [82] encoder. Then, we take the average of those embeddings as the content portion of the node embedding (i.e., $\vec{z}_{v'} = \texttt{avg}(\mathbf{X}^{(v')})$). We use available pre-trained models to embed the content of the observed graph $\mathbf{G}$. Specifically, we use Distil-RoBERTa [83] and SPECTER [84, 85] for scientific/academic content.

Motif Types CNN

The Motif Types convolutional neural network (CNN) (Figure 5.4c) takes as input the node embeddings for a triplet $\{u, v, w\}$ (i.e., the node roles and content embeddings concatenated for each node) and will output the probabilities for each motif type i (i.e., probability of being a triangle, wedge, 1-edge, or empty). More concretely, let $\mathbf{x} \in \mathbb{R}^{N \times d}$ be the matrix of node embeddings for the triple $\{u, v, w\}$, defined as:

$$\mathbf{x} = (\vec{r}_u \oplus \vec{z}_u, \vec{r}_v \oplus \vec{z}_v, \vec{r}_w \oplus \vec{z}_w) \tag{5.22}$$

where $d = |\vec{r}_{v'} \oplus \vec{z}_{v'}|$ is the node embedding dimension, and $N = 3$ is the number of nodes in the triple.

To perform a convolution operation for input $\mathbf{x}$, windows slide from top to bottom through multiple convolution kernels of size $k \times d$ ($k \in [1, 2, 3]$), and the number of filters for each kernel is $L = 100$. In a window of k node embeddings $\mathbf{x}_{n:n+k-1}$, a filter F^l ($1 < l < L$) returns the feature map c_n^l, defined as:

$$c_n^l = \mathrm{ReLU}\left(\mathbf{W}^l \circ \mathbf{x}_{n:n+k-1} + \mathbf{b}^l\right) \tag{5.23}$$

where $\circ$ is the convolutional operator, $\mathbf{W}^l \in \mathbb{R}^{k \times d}$ and $\mathbf{b}^l$ denote the weight matrix and bias, and k is the length of the filter. As the filter F^l traverses from $\mathbf{x}_{1:k-1}$ to $\mathbf{x}_{N+k-1:N}$, we get the output feature maps $\mathbf{c}^l = (c_1^l, \ldots, c_{N-k+1}^l)$.

Afterwards, we perform max pooling on the features maps we obtained for filter F^l (with length k) and we get $o_k^l = \max(\mathbf{c}^l)$. Then, in the flattened layer we get $\mathbf{o}$ and pass that through a linear layer to get $\mathbf{o}'$, as follows:

$$\mathbf{o} = \left(o_k^1, \ldots, o_k^L, \ldots, o_K^1, \ldots, o_K^L\right) \tag{5.24}$$

$$\mathbf{o}' = \mathbf{W}_{\text{final}}\mathbf{o} + \mathbf{b}_{\text{final}} \tag{5.25}$$

To get the motif type probability, we apply a Softmax function to obtain $\mathbf{P}_{\{u,v,w\}}$. Let $\mathbf{P}^{(\text{i})}_{\{u,v,w\}}$ denote the probability of motif $\{u, v, w\}$ having motif type i. Then the predicted motif type is $\hat{y}_{\{u,v,w\}}$ is calculated as:

$$\mathbf{P}_{\{u,v,w\}} = \text{softmax}\left(\mathbf{o}'\right) \tag{5.26}$$

$$\hat{y}_{\{u,v,w\}} = \arg\max_{\text{i}} \left(\mathbf{P}^{(\text{i})}_{\{u,v,w\}}\right) \tag{5.27}$$

We train the Motif Types CNN for 300 epochs and use cross-entropy as the loss function. We use the observed motif type in the input network $\mathbf{G}$ as the ground truth $y_{\{u,v,w\}}$ and perform stratified sampling to select motifs for the training set $\mathcal{M}_{\text{train}}$ and test set $\mathcal{M}_{\text{test}}$. Let j be the index of motif $\{u, v, w\}$ in the training set, then the loss function is defined as:

$$\mathcal{L}\left(\hat{y}_{\text{j}}^{\text{motif}}, y_{\text{j}}^{\text{motif}}\right) = -\sum_{\text{j} \in \mathcal{M}_{\text{train}}} y_{\text{j}}^{\text{motif}}\ log(\hat{y}_{\text{j}}^{\text{motif}}) \tag{5.28}$$

5.4 Methodology

We first describe the baseline models (Subsection 5.4.1) and datasets (Subsection 5.4.2) used in our evaluation. Then, we introduce the metrics for evaluating graph structure, node behavior and content embeddings (Subsection 5.4.3). We learn model parameters on the observed input graph for our model and all baselines.

5.4.1 Baselines

TagGen

The TagGen baseline is a deep graph generative model for dynamic networks [64]. It models a deep generative process of k-length temporal walks using local operations (addition and deletions of temporal edges), to generate synthetic temporal random walks. We used the available implementation of TagGen[1].

TG-GAN

The TG-GAN baseline learns distributions of temporal walks to capture topological and temporal patterns of temporal graphs [77]. It consists of two parts, a temporal walk generator and a discriminator. We used the available implementation of TG-GAN[2]. Given that TG-GAN relies on continuous timesteps, we transform the datasets edges to use the edge timestamps directly. After generating a new network, we transform it to match the graph snapshots and time-windows of the observed network.

CTWalk

The CTWalk baseline models temporal graphs with attributes, capitalizing on temporal random walks and conditional GANs [80]. Specifically, it builds upon TagGen [64], for temporal edge generation, and CTGAN [81], for generating node and edge attributes. We used the implementation kindly provided by the authors. Given that CTWalk relies on discrete attributes, we created topic labels from the content text and embeddings with BERTopic [86].

5.4.2 Datasets

We use the datasets described below, with more detailed statistics shown in Table 5.2.

[1]↑https://github.com/davidchouzdw/TagGen
[2]↑https://github.com/tongjiyiming/TGGAN

Table 5.2. Attributed Dynamic Networks Statistics

Dataset	$\|V\|$	$\|E\|$	Unique	$\|T\|$	Length	Time Span
Aminer	602	4,124	1,041	10	1 yr.	2010–2019
Congress	1,283	425,297	116,097	762	1 day	12/19–12/21
R-Posts	1,009	48,780	15,166	41	1 mo.	12/13–04/17
R-Replies	1,894	58,836	7,330	41	1 mo.	12/13–04/17

Arnetminer

The Arnetminer dataset (Aminer) is an academic co-authorship network, where nodes represents authors, edges represent co-authorship on a paper, and the content is the title and abstract of a paper [87, 88].

Congress Tweets

The Congress Tweets dataset (Congress) is a social network, where nodes represent Twitter accounts of US Congress, edges represent mentions or re-tweets, and the content is the text of the tweet [89, 90].

Reddit Cross-posts

The Reddit Cross-posts dataset (R-Posts) is a social network, where nodes represent subreddits, edges represent cross-links between subreddits, and the content is the text of the post linking to another subreddit [91–93].

Reddit Replies

The Reddit Replies dataset (R-Replies) is a social network, where nodes represent Reddit users, edges represent replies to posts or comments, and the content is the text of reply [92, 94, 95].

5.4.3 Evaluation

We use two sets of metrics in our evaluation for graph structure and node behavior. The majority of graph structure metrics we selected are widely used to characterize graphs. With these first set of metrics we aim to measure if the overall graph structure of the generated graph $\mathbf{G}'$ is similar to the dataset graph $\mathbf{G}$. For the second set, we propose to use node-aligned metrics to capture node behavior. For the evaluation of node content embeddings, we want to measure how close or similar the observed attributes $\mathbf{X}$ are to $\mathbf{X}'$. We use different distance and similarity metrics for the content embeddings and estimated topics, which are discrete attributes.

Graph Structure

To evaluate the graph structure generated by the models against that of the datasets, we use the following metrics: density, average local clustering coefficient, global clustering coefficient, average path length of largest connected component (LCC), and s-metric. *Density* measures ratio of edges in the graph to the number of edges in a complete graph. The *local clustering coefficient* quantifies the tendency of the nodes of a graph to cluster together, and the *global clustering coefficient* measures the ratio of closed triplets to open and closed triplets (wedges and triangles, respectively). The *average (shortest) path length* measures the efficiency of information transport, for all possible pairs of nodes. The *s-metric* measures the extent to which a graph has hub-like structure [44]. Together with local and global clustering, these metrics provide insight into graph structure, such as tightly knit-groups and large star structures.

We calculate the graph structure metrics for each time-window of the generated graph $\mathbf{G}'$ and the input graph $\mathbf{G}$. Specifically, for each graph structure metric s, we calculate the distribution of values $\mathbf{s}_{gen}$ of the generated graph and $\mathbf{s}_{in}$ of the input graph (where $\mathbf{s}_{in} = \{s(G_1), \ldots, s(G_T)\}$, and $G_t \in \mathbf{G}$). Given that we aim to model the distribution of graph structure, and not just generate the same graph sequence, we calculate the Kolmogorov-Smirnov (KS) test statistic on $\mathbf{s}_{gen}$ and $\mathbf{s}_{in}$ to evaluate $\mathbf{G}'$ against $\mathbf{G}$.

Node Behavior

To compare the temporal node behavior of the generated graphs against the datasets, we use the following *node-aligned, temporal* metrics: activity rate, temporal degree distribution, clustering coefficient, closeness centrality, and the size of its connected component. The *activity rate* of a node reflects how often it participates in an edge. The *temporal degree distribution* of a node u is the set of degrees of u over all snapshots G_t (i.e., it shows how many nodes it interacts with). The *local clustering coefficient* of a node measures how close its neighbors are to becoming a clique. The *closeness centrality* of a node u in a (possibly) disconnected graph is the sum of the reciprocal of shortest path distances to u over all other reachable nodes. Together, the node's closeness centrality and size of its connected component indicate the location of the node relative to others.

We calculate the *node-aligned, temporal* metrics for every node in the input graph $\mathbf{G}$ and in the generated graph $\mathbf{G}'$. *Node-alignment* refers to assumption that node ids are aligned over graph snapshots, within a graph sequence (e.g., $\{G_1, \ldots, G_T\}$). Based on this, we measure the distribution of values a node has over time $\mathbf{s}(v_\mathrm{i}|\mathbf{G}) = \{s(v_\mathrm{i}|G_1), \ldots, s(v_\mathrm{i}|G_T)\}$ for each metric s. Since the nodes in $\mathbf{G}$ do not necessarily correspond to those in $\mathbf{G}'$, we consider the inter-quartile range (IQR) of values over time $\{\mathbf{s}(v_\mathrm{j}|\mathbf{G})\}_{\mathrm{j}\in\mathbf{G}}$. We then perform a 2-dimensional KS test using the Q_1 and Q_3 values of all nodes in $\mathbf{G}$ and $\mathbf{G}'$. With this approach, we can capture each node's individual behavior and their joint behavior. Unlike using the mean and median of the s values, this approach can capture characteristics of the distribution of values and can be misleading. For example, a synthetic graph $\mathbf{G}'$ could have mean and median values of a metric s very close to those of an observed graph $\mathbf{G}$, but have a much larger dispersion of s values than observed in $\mathbf{G}$. We use the KS test on the inter-quartile range (i.e., Q_1 and Q_3) because it does not make assumptions about the distribution of values and can capture variability or dispersion.

Node Content

To evaluate the generated content embeddings, we use the following metrics on the content embeddings: Distance correlation with Central Kernel Alignment, and Fréchet Bert Dis-

tance. Central Kernel Alignment (CKA) generalizes squared cosine similarity and Pearson correlation to the multivariate case (e.g., sets of embeddings) [96, 97]. Distance correlation measures both linear and non-linear relationships between two random vectors, even when they have different dimensions [98]. Fréchet Bert Distance (FBD) measures distance between BERT-based embedding sets—Specifically, it measures the distance between the distribution of generated data and the observed [99]. We use the available implementations of CKA[3], distance correlation[4] [100], and FBD[5].

To evaluate the similarity of the node's topics, we fit a BERTopic model [86] on the observed content and embeddings to get the embedding topics. For the CTWalk baseline, we use directly the discrete attributes generated, which correspond to the topics. For our model, DYMOND, we calculate the topics of the generated embeddings with the previously fitted model. We compare both models with the following topic metrics: Szymkiewicz-Simpson Coefficient (SSC), Jaccard Similarity Index, (Jaccard) and SørensenDice Similarity Coefficient (DSC). SSC measures the overlap of common elements in two sets. In contrast to SSC, Jaccard always penalizes differences between the sets. DSC gauges the similarity of two samples, similar to Jaccard, but gives more weight to set commonalities than differences.

We calculate these metrics for every node on their content in the input graph $\mathbf{G}$ and generated graph $\mathbf{G}'$ (i.e., $\mathbf{X}$ against $\mathbf{X}'$). Specifically, we measure the distribution of values on all nodes $\mathbf{s}(\mathbf{X}, \mathbf{X}') = \{s(v_k|\mathbf{X}, \mathbf{X}')\}_{k \in (\mathbf{G} \wedge \mathbf{G}')}$ for every content metric s and report the average. We also investigate the quality of the generated embeddings by visually inspecting the topics extracted for a small sample of nodes.

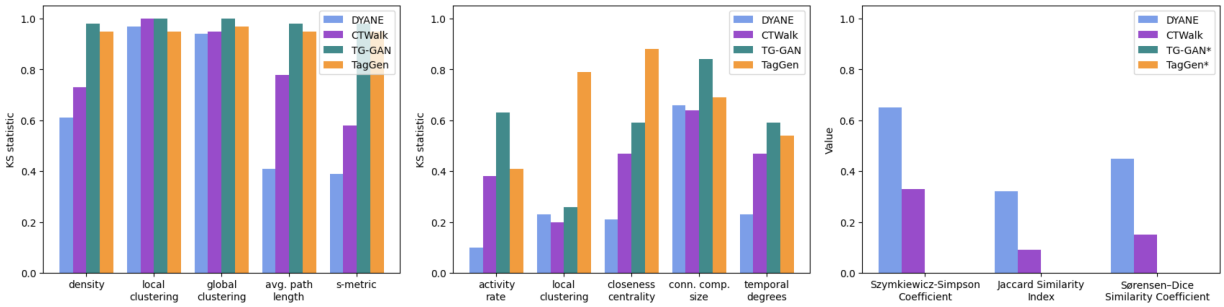
5.5 Results

5.5.1 Evaluation of Generated Graphs

We show the KS statistic (lower is better) for the graph structure (Figures 5.5a and 5.6a[6]) and node behavior (Figures 5.5b and 5.6b) of the Reddit Cross-Posts and Arnetminer

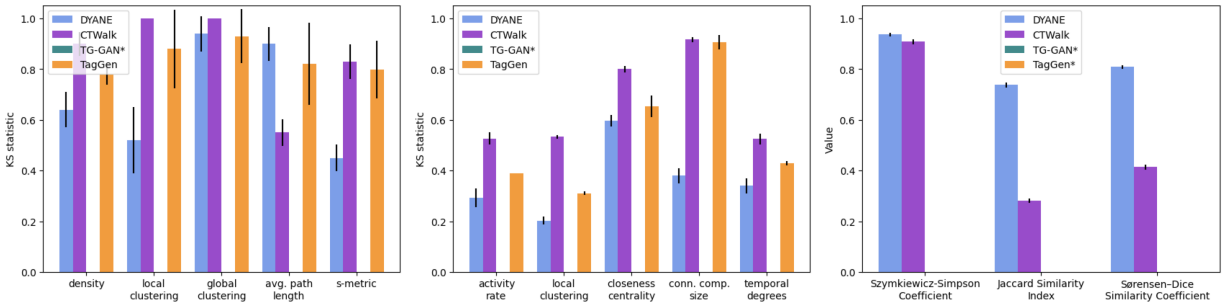
[3]↑https://github.com/babylonhealth/corrsim
[4]↑https://github.com/vnmabus/dcor
[5]↑https://github.com/gretelai/public_research
[6]↑See Section 5.6 for variability analysis of Figure 5.6

datasets. The remaining results for the Reddit Replies and Congress Tweets datasets are in Figures 5.8 and 5.9.



(a) Graph Structure Metrics (b) Node Behavior Metrics (c) Node Topic Metrics

Figure 5.5. Results for Reddit Cross-Posts (R-Posts) dataset: in subfigures (a,b) lower value is better, (c) higher value is better



(a) Graph Structure Metrics (b) Node Behavior Metrics (c) Node Topic Metrics

Figure 5.6. Results for Arnetminer (Aminer) dataset: Average statistic and $+1/-1$ Standard Deviation is shown over 10 trials

In order to compare the models more easily, we calculated the mean reciprocal rank (MRR) of the KS statistics for the graph structure and the node behavior metrics. To calculate the MRR, we ranked all models' results by using the KS statistics. In Tables 5.3 and 5.4, we can observe that our model (DYANE) outperforms the baselines when considering each set of metrics together using the MRR (higher is better), for both graph structure and node behavior. The omitted baseline results are due to two reasons: (1) TG-GAN models continuous time and expects a smaller time granularity than available for the Arnetminer dataset, (2) TagGen was not able to run on the Congress Tweets dataset due to its size.

Table 5.3. Graph Structure Metrics (MRR)

Model	Aminer	Congress	R-Posts	R-Replies
DYANE	**0.87**	**1.00**	**0.90**	**0.87**
CTWalk	0.50	0.60	0.47	0.33
TG-GAN	—	0.37	0.27	0.50
TagGen	0.60	—	0.47	0.53

Table 5.4. Node Behavior Metrics (MRR)

Model	Aminer	Congress	R-Posts	R-Replies
DYANE	**1.00**	**1.00**	**0.80**	**0.90**
CTWalk	0.37	0.43	0.70	0.40
TG-GAN	—	0.53	0.28	0.80
TagGen	0.50	—	0.30	0.28

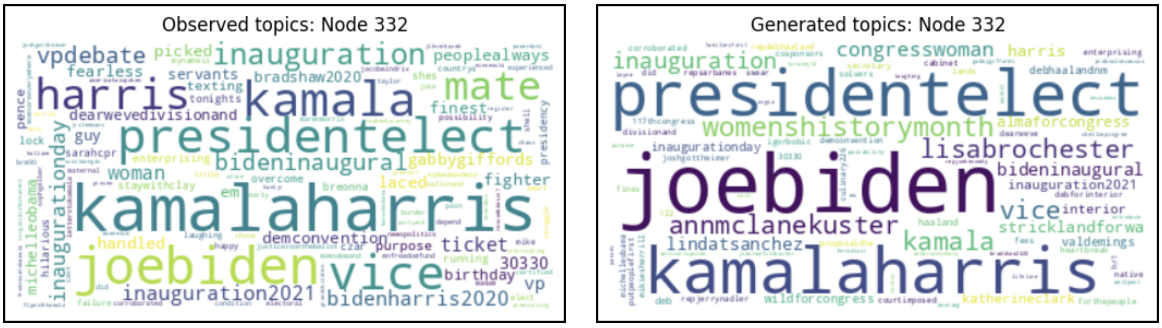
5.5.2 Evaluation of Generated Content

We also show the similarity metrics (higher is better) for the node topics (Figures 5.5c and 5.6c) of the Reddit Cross-Posts and Arnetminer datasets. The evaluation results of the node topics for other datasets are in Figures 5.8 and 5.9. TagGen and TG-GAN do not model any attributes, so their results are not included in the figures. Our model (DYANE) consistently outperforms the CTWalk baseline in the node topics evaluation metrics.

We also evaluated the generated content embeddings against the observed with the content embedding metrics, and show the results in Table 5.5. Our model achieved high CKA distance correlation (d-corr) in the majority of the datasets (higher is better). We also observe that the lowest Fréchet Bert Distance (FBD) is in the Arnetminer dataset (lower is better). This is also consistent with the topic evaluation results.

Table 5.5. Content Embedding Metrics (DYANE)

Metric	Aminer	Congress	R-Posts	R-Replies
d-corr	0.94	0.93	0.73	0.61
FBD	0.24	0.78	1.11	1.37



(a) Topics in Observed Embeddings (b) Topics in Generated Embeddings

Figure 5.7. Embedding Topics in Congress Tweets dataset

To inspect the quality of the sampled content embeddings, we used the topics extracted with BERTopic and created topic word clouds, weighed using their frequency. We show an example, to compare the generated embeddings against the observed content embeddings, for the Congress Tweets dataset (Figure 5.7). Qualitative results for the other datasets are omitted due to space.

5.5.3 Discussion

TagGen, compared to itself, performs better in the Arnetminer dataset than the other datasets. Arnetminer has star structures (high degree nodes) over time and shorter diameter than the other datasets. TagGen benefits from very active and high degree nodes due to its biased temporal random walks. TG-GAN has comparable results to TagGen in graph structure, but performs better in the node-aligned metrics for the Reddit datasets. In both datasets, the majority of the nodes are of high-degree, possibly hindering TagGen's performance in comparison. CTWalk is the only baseline that also models attributes and also extends TagGen. CTWalk performs comparable or better than TagGen in both of the Reddit datasets; demonstrating that modeling attributes can improve the quality of the networks generated. Our model, DYANE, consistently outperforms the baselines while considering the graph structure and node-aligned metrics. DYANE also outperforms CTWalk on the node topic metrics. CTWalk generates attributes based on edges or individual nodes, whereas DYANE considers higher-order structures (motifs). Additionally, CTWalk only generates

discrete attributes that were observed in the input graph. In contrast, DYANE generates new content based on the pooled interests of the nodes in the motif and their roles (using their content embeddings). Generating content using higher-order structures and embeddings can exploit any possible overlaps of latent interest.

5.6 Variability Analysis

The order in which motifs and node roles are sampled introduces variability in the generated networks. We generated 10 synthetic networks with parameters learned on the Arnetminer dataset. In Figure 5.6, we show the average statistic and $+1/-1$ standard deviation for each metric. In Figure 5.6a, the mean KS statistic of the CTWalk baseline for both the local and global clustering coefficient was 1.0 on all 10 experiments. When considering the global clustering coefficient, the standard deviation error bar overlaps the bars for the CTWalk and TagGen baselines. In Figure 5.6b, the mean KS statistic of our model DYANE is lower than the baselines for all node behavior metrics. In Figure 5.6c, the mean metric of our model DYANE is lower than the CTWalk baseline for all topic metrics.

We performed a Welch's t-test for the graph structure and node behavior KS statistics, with $\alpha = 0.05$. For the lower one-tailed test, the hypotheses are:

$$H_0 : \mu_{\text{DYANE}} = \mu_{\text{baseline}} \tag{5.29}$$

$$H_a : \mu_{\text{DYANE}} < \mu_{\text{baseline}} \tag{5.30}$$

In Table 5.6, 4 out of 5 null hypotheses are rejected for the CTWalk baseline ($H_0 : \mu_{\text{DYANE}} = \mu_{\text{CTWalk}}$), and 3 out of 5 null hypotheses are rejected for the TagGen baseline ($H_0 : \mu_{\text{DYANE}} = \mu_{\text{TagGen}}$). The rejected null hypotheses had p-values of 0.01 or significantly lower.

Table 5.6. Graph Structure Variability Results (H_0 rejected)

Baseline	Density	Local CC	Global CC	Avg. Path. Len.	S-metric
CTWalk	✓	✓	✓		✓
TagGen	✓	✓			✓

In Table 5.7, the null hypotheses for both the CTWalk and TagGen baselines (H_0 : $\mu_{\text{DYANE}} = \mu_{\text{baseline}}$) were rejected for all metrics. The rejected null hypotheses had p-values of 0.0001 or significantly lower.

Table 5.7. Node Behavior Variability Results (H_0 rejected)

Baseline	Activity Rate	Local CC	Closeness	Conn. Comp. Size	Temporal Degrees
CTWalk	✓	✓	✓	✓	✓
TagGen	✓	✓	✓	✓	✓

We also performed a Welch's t-test for the topic metrics, with $\alpha = 0.05$. For the greater one-tailed test, the hypotheses are:

$$H_0 : \mu_{\text{DYANE}} = \mu_{\text{baseline}} \tag{5.31}$$

$$H_a : \mu_{\text{DYANE}} > \mu_{\text{baseline}} \tag{5.32}$$

In Table 5.8, the null hypotheses ($\mu_{\text{DYANE}} = \mu_{\text{CTWalk}}$) were rejected for all topic metrics. The rejected null hypotheses had p-values of 0.000001 or significantly lower.

Table 5.8. Topic Variability Results (H_0 rejected)

Baseline	Szymkiewicz-Simpson Coeff.	Jaccard Similarity	Sørensen-Dice Similarity
CTWalk	✓	✓	✓

Overall, the results in Figure 5.6c and Tables 5.6 to 5.8 are consistent with the mean reciprocal rank (MRR) results in Tables 5.3 and 5.4.

5.7 Scalability Analysis

For conciseness, in this section we use V as the number of nodes, E as the number of edges, M as the number of motifs, and $\mathcal{U}_t$ as the number of new triplets, where the subscript t denotes the timestep.

Similar to our previous model DYMOND (Section 4.7), the time complexity, in practice, of the DYANE *parameter estimation* depends on finding the connected motifs in the observed

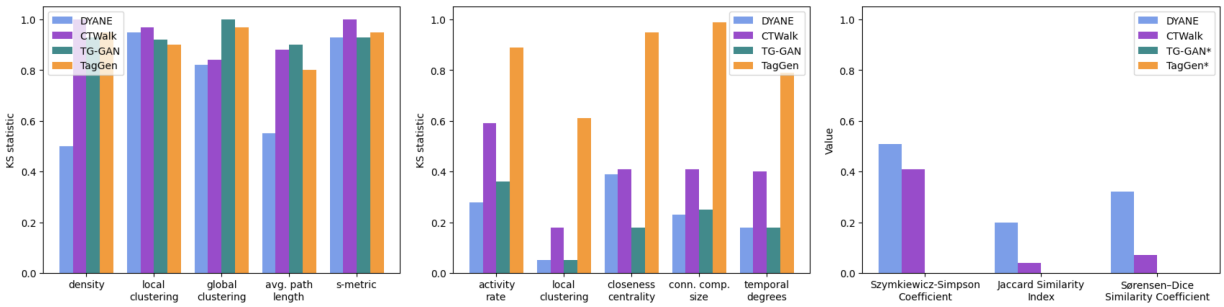
graph, resulting in $\mathcal{O}(E \cdot V \cdot T)$ time. The training set for the Motifs Types CNN includes disconnected motifs (i.e., the empty motif type). Enumerating the disconnected motifs would push the theoretical time complexity to $\mathcal{O}\left(V^3\right)$, given that the number of disconnected motifs is $N = \binom{V}{3} - M$. Instead, we sample J indices for adding to the training set, where each index j is in the interval $\left[1, \binom{V}{3}\right]$. For each sampled index, we calculate the j^{th} combination and add it to the training set, discarding those that correspond to connected motifs in $\mathcal{M}$. We note that J is a constant and substantially smaller than N, thus the time complexity of the DYANE *parameter estimation* remains $\mathcal{O}(E \cdot V \cdot T)$.

Like before (Section 4.7), the DYANE *generative process* depends on the number of new triplets $\mathcal{U}_t$ available to sample motifs from, so the time complexity is $\mathcal{O}\left(\sum_{t=1}^{T} \mathcal{U}_t\right)$, where $\mathcal{U}_t = \binom{V_t}{3} + \binom{V_t}{2} \cdot V_{t-1} + \binom{V_{t-1}}{2} \cdot V_t$.

5.8 Concluding Remarks

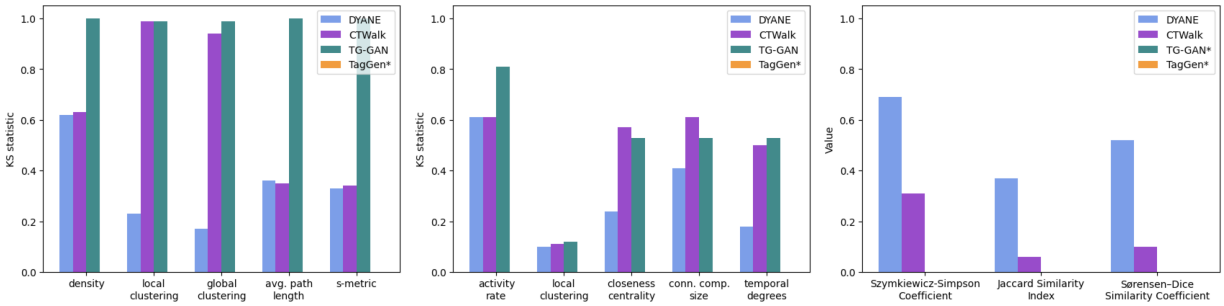
Our proposed model, DYnamic Attributed Node rolEs (DYANE), is the first to generate synthetic dynamic networks and sample content embeddings based on motif node roles. To the best of our knowledge, it is the only attributed dynamic network model that can generate *new* content embeddings–not observed in the input graph, but still similar to that of the input graph. Our results show that modeling the network attributes with higher-order structures (e.g., motifs) improves the quality of the networks generated. The use of the Kolmogorov-Smirnov (KS) test adapts graph structure metrics designed for static graphs to the dynamic graph setting, by considering the distribution of graph statistics. Similarly, in our proposed content evaluation, we take the distribution of attributes over time to evaluate the content embeddings generated by nodes, employing metrics based on embeddings and topic similarity. In conclusion, when jointly considering all three sets of metrics—for temporal graph structure, node behavior, and content—DYANE outshines other models in our evaluation on four real-world datasets.

5.9 Additional Figures and Algorithms



(a) Graph Structure Metrics (b) Node Behavior Metrics (c) Node Topic Metrics

Figure 5.8. Results for Reddit Replies (R-Replies)



(a) Graph Structure Metrics (b) Node Behavior Metrics (c) Node Topic Metrics

Figure 5.9. Results for Congress Tweets (Congress)

Algorithm 12: `SampleMotifTimesteps`

input: T, t, $\mathcal{M}_t$, $\mathcal{M}_t^T$
output: $\mathcal{M}_t^S$ `// motif timesteps`

begin
  $n := T - t$ `// num. timesteps left`
  for $m \in \mathcal{M}_t$ **do**
    $\mathrm{i} := \mathcal{M}_t^T(m)$ `// get motif type`
    $\lambda_{\{u,v,w\}} \sim Beta(\alpha_\mathrm{i}, \beta_\mathrm{i})$ `// sample distr. type` i
    $p \leftarrow \frac{\lambda_{\{u,v,w\}} \cdot n \; + \; \alpha_\mathrm{i}}{n \; + \; \alpha_\mathrm{i} \; + \; \beta_\mathrm{i}}$ `// adjust for` n
    $t' \leftarrow t$ `// init`
    while $t' \leq T$ **do**
      $k \sim NB(1, p)$ `// sample inter-arrival time`
      $t' \leftarrow t' + k$ `// update next timestep`
      if $t' \leq T$ **then**
        $\mathcal{M}^S(m)$.append(t')

Algorithm 13: `SampleNodeRoles`

input: $\mathcal{M}_t$, $\mathcal{M}_t^S$, c_R
output: $\mathcal{M}_t^R$ `// sampled node roles`
c_R `// updated role counts`

begin
  for $m \in \mathcal{M}_t$ **do**
    if $\mathrm{i} = 3$ **then** `// triangle`
      $\mathcal{M}^R(m, v) \leftarrow$ `equal3`, $\forall v \in m$
    else if $\mathrm{i} = 2$ **then** `// wedge`
      $p_h \leftarrow \left[v \in m \mapsto \frac{P[v,\texttt{hub}]}{\sum_{v' \in m} P[v',\texttt{hub}]} \right]$ `// normalize`
      $v_h \sim Bin(m, \; p_h)$ `// sample hub node`
      $\mathcal{M}^R(m, v_h) \leftarrow$ `hub`
      $\mathcal{M}^R(m, v') \leftarrow$ `spoke`, $\forall v \in m, v \neq v_h$
    else if $\mathrm{i} = 1$ **then** `// 1-edge`
      $p_o \leftarrow \left[v \in m \mapsto \frac{P[v,\texttt{outlier}]}{\sum_{v' \in m} P[v',\texttt{outlier}]} \right]$ `// normalize`
      $v_o \sim Bin(m, \; p_o)$ `// sample outlier node`
      $\mathcal{M}^R(m, v_o) \leftarrow$ `outlier`
      $\mathcal{M}^R(m, v) \leftarrow$ `equal2`, $\forall v \in m, v \neq v_o$
    `count`$\Big(v, \; \mathcal{M}^R(m, v)\Big) \mathrel{-}= |\mathcal{M}^S(m)|, \; \forall v \in m$ `// update`

Algorithm 14: `ConstructGraph`

input: $\mathcal{M}$, $\mathcal{M}^S$, $\mathcal{M}^E$, $\mathcal{M}^X$
output: $\mathbf{G}' = \{G'_1, \ldots, G'_{T'}\}$ `//` where $G'_t = (V'_t, E'_t, X'_t, S'_t)$

begin
  $\mathcal{M}_t := \left\{\{u,v,w\} \in \mathcal{M} : t \in \mathcal{M}^S(\{u,v,w\})\right\}$
  for $t \in [1, \ldots, T']$ **do**
    $V'_t \leftarrow \left\{v \in V : \ g(v) \leq t\right\}$ `//` nodes & content
    $\mathbf{X}'_t := \left[v' \in V'_t \mapsto \{\mathcal{M}^X(\{u,v,w\}) : v' \in \{u,v,w\},\ \{u,v,w\} \in \mathcal{M}_t\}\right]$
    `//` edges & timestamps
    $E'_t \leftarrow \left\{(u',v') \in \mathcal{M}^E(\{u,v,w\}) : \{u'v'\} \subset \{u,v,w\},\ \{u,v,w\} \in \mathcal{M}_t\right\}$
    $S'_t := \left[(u'v') \in \mathcal{M}^E(\{u,v,w\}) \mapsto t : \{u'v'\} \subset \{u,v,w\},\ \{u,v,w\} \in \mathcal{M}_t\right]$

Algorithm 15: `GetNodeRoleCounts`

input: V, E, T
output: $c_R := \left[v \in V, r \in \mathcal{R} \mapsto count(v,r)\right]$ `//` role counts

begin
  $count(v,r) = 0,\ \forall v \in V,\ \forall r \in \mathcal{R}$ `//` init. counts
  for $t \in [1, \ldots, T]$ **do**
    for $(u,v) \in E_t$ **do**
      if $n_t^{(3)} > 0$ **then** `//` triangles
        for $w \in \left(\mathcal{N}_t(u) \cap \mathcal{N}_t(v)\right)$ **do**
          $count(v', \texttt{equal3}) \mathrel{+}= \frac{\omega^{(3)}}{3},\ \forall v' \in \{u,v,w\}$ `//` Equation 5.10a
      if $n_t^{(2)} > 0$ ***and*** $r_t^{(2)}(u,v) > 0$ **then** `//` wedges
        for $w \in \left(\mathcal{N}_t(u) \oplus \mathcal{N}_t(v)\right)$ **do**
          for $v' \in \{u,v,w\}$ **do**
            $r \leftarrow \texttt{GetRoleTimestep}(v', E_t, \{u,v,w\})$
            if $r = \texttt{hub}$ **then**
              $count(v', \texttt{hub}) \mathrel{+}= \frac{\omega^{(2)}}{2}$ `//` Equation 5.10a
            else
              $count(v', \texttt{spoke}) \mathrel{+}= \omega^{(2)}$ `//` Equation 5.10a
      if $n_t^{(1)} > 0$ ***and*** $r_t^{(1)}(u,v) > 0$ **then** `//` 1-edges
        $count(u, \texttt{equal2}) \mathrel{+}= r_t^{(1)}(u',v')$ `//` Equation 5.10b
        $count(v, \texttt{equal2}) \mathrel{+}= r_t^{(1)}(u',v')$ `//` Equation 5.10b
        for $w \in \left(V - \left(\mathcal{N}_t(u) \cup \mathcal{N}_t(v)\right)\right)$ **do**
          $count(w, \texttt{outlier}) \mathrel{+}= \omega^{(1)}$ `//` Equation 5.10a

6. SUMMARY

In this chapter, we summarize the contributions of this dissertation, and outline avenues for future research:

In Chapter 3, we presented a study of the impact of graph structure and attribute auto-correlation on the performance of collective classification/inference methods. By using a recent attributed graph generative model that can create synthetic graphs with varying structure and attribute correlation, our study is the first to evaluate these methods on a wide-range of graphs. We considered different graph structure measures and found that the link-density along with the attribute auto-correlation had the most impact on classification accuracy. We also showed that it is possible to predict the area-under-the-curve (AUC) score obtainable with collective classification (CC) methods, using a regression model. The significance of this is that with the learned linear regression coefficients for a CC method on some networks, we can predict the AUC score and use it to pick the method or model that will likely perform the best for a particular network.

In Chapter 4, we first conducted an empirical study of motif evolution on real-world dynamic networks and found that motifs do not change configurations from one time-slice to a subsequent one, but rather kept re-appearing over time. Motivated by this finding, we proposed DYMOND—a generative model that considers (i) the dynamic changes in overall graph structure using temporal motif activity and (ii) the roles nodes play in motifs (e.g., one node plays the hub role in a wedge, while the remaining two act as spokes). Our model first assigns a motif configuration (or motif type) and then samples inter-arrival times for the motifs. One challenge that comes with it is sampling the motif placement. To this end, we defined motif node roles and use them calculate the probability of each motif type. Although motifs have been shown to be the building blocks of networks, we are the first to propose a motif-based generative model for dynamic networks that can produce entirely synthetic graphs. We find that using motifs and node roles in our model helps to outperform other available dynamic network generative models at generating graph structure and node behavior similar to the observed network..

In Chapter 5, we developed a generative model for dynamic networks with content, that (i) captures network structure dynamics through temporal motifs, and (ii) extends the structural roles of nodes in motifs (e.g., a node acting as a hub in a wedge) to roles that generate content embeddings. DYANE combines a node-roles GCN and a motif-type CNN, with a content embedding model, for generating synthetic graphs with new structure and content embeddings similar to the observed input network. Our model is the first to generate synthetic dynamic networks and sample content embeddings based on motif node roles. To the best of our knowledge, it is the only attributed dynamic network model that can generate *new* content embeddings–not observed in the input graph, but still similar to that of the input graph.

We also proposed a novel methodology for evaluating dynamic network generative models. In Chapter 4, we adapt graph structure metrics to take into account the temporal aspect of the network. Our use of the Kolmogorov-Smirnov (KS) test adapts graph structure metrics designed for static graphs to the dynamic graph setting, by considering the distribution of graph statistics. To consider node behavior, we use node-aligned metrics over the graph snapshots to evaluate the node's topological connectivity and temporal activity. In order to compare models more easily, we proposed using the mean reciprocal rank (MRR) of the KS statistics for the graph structure and the node behavior metrics. In Chapter 5, we also derived a methodology to evaluate the content embeddings generated by nodes, taking into account keywords extracted from the content (as topic representations), and using distance metrics. Similar to our node behavior evaluation, we take the distribution of attributes over time to evaluate the content embeddings generated by nodes, employing metrics based on embeddings and topic similarity.

6.1 Contributions

- Frameworks
  - Development of a framework to systematically assess the performance of collective inference methods (Chapter 3).

  - Development of a dynamic network generative framework for synthetic data generation, where the models act as components that allow to choose between generating only structure or also generating content embeddings (Chapters 4 and 5).

- Models
  - Development of DYMOND — a generative model for dynamic networks, leveraging motifs and node roles to create synthetic graphs with structure and node behavior similar to an input graph (Chapter 4).
  - Development of DYANE — a generative model for attributed dynamic networks that extends motif node-roles to roles that generate content embeddings, to create synthetic graphs with structure, node behavior, and node content embeddings similar to an input graph (Chapter 5).
  - Development of a Node Roles GCN and meta-model to create and update structural node embeddings based on motif node roles (Chapter 5).
  - Development of a Motif Types CNN to calculate motif type probabilities, combining the node role embeddings and content embeddings, to sample motifs (Chapter 5).

- Algorithmic Contributions
  - Efficient algorithms to estimate motifs in the input network, and sample motifs in the generated network (Chapters 4 and 5).
  - Efficient algorithm to create training samples from *all* motifs in the input network, for training the Motif Types CNN (Chapter 5).

- Methodological Contributions
  - Graph structure evaluation: Adapted graph structure metrics to the dynamic graph setting using the Kolmogorov-Smirnov (KS) test over the distribution of values (Chapter 4).

- Node behavior evaluation: Proposed use of node-aligned metrics over the graph snapshots and taking the inter-quartile range of their distributions, instead of the mean and median values, to better capture topological connectivity and temporal activity (Chapter 4).
- Dynamic generative models evaluation: Proposed the use of the mean reciprocal rank of the KS statistics to compare models using each set of graph metrics metrics together (Chapter 4).
- Content evaluation: Proposed the use of embedding distance metrics designed for sets of embeddings to evaluate nodes' generated content embeddings against the observed. To evaluate nodes' latent topics, proposed the extraction of keywords from content embeddings and the use of set similarity metrics (Chapter 5).

6.2 Implications and Future Directions

6.2.1 Dynamic network data

The generative models proposed will address the difficulty of finding readily-available datasets of dynamic networks—attributed or not—and provide a way to study and analyze dynamic networks. This work will also allow others to: (i) generate dynamic networks that they can share without divulging individual's private data, (ii) benchmark model performance, and (iii) explore model generalization on a broader range of conditions, among other uses. For example, in the task of node classification, we can now benchmark performance and generalization in dynamic networks with varying structure, attribute auto-correlation, and temporal behavior. Additionally, the evaluation measures proposed will elucidate models, allowing fellow researchers to push forward in these domains.

6.2.2 On motif-based generative models

Incorporating motifs into *walk-based* generative models improves the structure of generated graphs [101]. In graph convolutional networks (GCNs) for the task of node classification, using a motif-based attention with weighted motif-adjacency matrices per motif type

has been shown to improve the accuracy [102]. With this in mind, an alternative to directly sampling motifs from node triples would be to use a random-walk-based model with a discriminator based on motifs, node roles and content. Some ideas to do this would be: (1) adding a motif-counts attention to a GCN, with weights based on shared edges per motif type, or (2) using the node role embeddings from our proposed model DYANE, and (3) using node content embeddings as features for the discriminator.

6.2.3 LLMs for content generation in dynamic networks

The work for content embedding sampling (Chapter 5) was done before the rise in popularity of generative large language models (LLMs)—which would enable the generation of *new text* for the content sampling component of our model. We make use of pre-trained language models (PLMs), specifically SBERT models [82–85], for embedding the content of the input graph $\mathbf{G}$. The sentence-transformer models we used consist of an encoder, which creates the embeddings, and lack a decoder. They are designed for information retrieval, clustering or sentence similarity tasks. An encoder-decoder model, such as T5, would allow to use the decoder to generate content for the output graph $\mathbf{G}'$. However, it is unclear how to combine the content embeddings of the motif nodes so the decoder generates relevant text (i.e., highly related to the input).

LLMs are scaled up pre-trained PLMs (e.g., model size or data size) and display surprising emergent abilities that may not be observed in previous smaller PLMs [103]. GPT-3 and similar LLMs fall into the category of decoder models, trained with auto-regressive objectives, and are more suitable for generative tasks than encoder-only models [104, 105]. Multimodal perception allows LLMs to acquire knowledge beyond text descriptions, such as structured data (e.g., graphs and tables) [104]. Current work on multimodal large language models (MLLMs) looks into cross-modal transferability, which allows a model to learn from one modality (e.g., text, image, audio) and transfer the knowledge to the other modalities [106]. To integrate this, one possibility would be to use a pre-trained LLM as a decoder for text generation, and train an encoder that incorporates the motif's node embeddings into the LLM prompt.

REFERENCES

[1] P. Erdös and a. Rényi, "On random graphs," *Publicationes Mathematicae*, vol. 6, pp. 290–297, 1959, ISSN: 00029947. DOI: 10.2307/1999405. arXiv: 1205.2923.

[2] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks.," *Nature*, vol. 393, no. 6684, pp. 440–2, 1998, ISSN: 0028-0836. DOI: 10.1038/30918. arXiv: 0803.0939v1.

[3] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, no. 1, pp. 47–97, Jan. 2002. DOI: 10.1103/RevModPhys.74.47.

[4] S. Wasserman and P. Pattison, "Logit models and logistic regressions for social networks: I. An introduction to markov graphs and p," *Psychometrika*, vol. 61, no. 3, pp. 401–425, 1996, ISSN: 00333123. DOI: 10.1007/BF02294547.

[5] G. Robins, P. Pattison, Y. Kalish, and D. Lusher, "An introduction to exponential random graph (p*) models for social networks," *Social Networks*, vol. 29, no. 2, pp. 173–191, 2007, ISSN: 03788733. DOI: 10.1016/j.socnet.2006.08.002.

[6] W. Aiello, F. Chung, and L. Lu, "A random graph model for power law graphs," *Experimental Mathematics*, vol. 10, no. 1, pp. 53–66, 2001.

[7] J. J. Pfeiffer, T. L. Fond, S. Moreno, and J. Neville, "Fast Generation of Large Scale Social Networks While Incorporating Transitive Closures," in *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, Sep. 2012, pp. 154–165. DOI: 10.1109/SocialCom-PASSAT.2012.130.

[8] F. Lorrain and H. C. White, "Structural equivalence of individuals in social networks," *The Journal of Mathematical Sociology*, vol. 1, no. 1, pp. 49–80, Jan. 1971, ISSN: 0022-250X. DOI: 10.1080/0022250X.1971.9989788.

[9] S. E. Fienberg and S. S. Wasserman, "Categorical Data Analysis of Single Sociometric Relations," *Sociological Methodology*, vol. 12, pp. 156–192, 1981, ISSN: 0081-1750. DOI: 10.2307/270741.

[10] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social Networks*, vol. 5, no. 2, pp. 109–137, Jun. 1983, ISSN: 0378-8733. DOI: 10.1016/0378-8733(83)90021-7. (visited on 11/08/2020).

[11] T. A. Snijders and K. Nowicki, "Estimation and prediction for stochastic blockmodels for graphs with latent block structure," *Journal of classification*, vol. 14, no. 1, pp. 75–100, 1997.

[12] K. Nowicki and T. a. B. Snijders, "Estimation and Prediction for Stochastic Blockstructures," *Journal of the American Statistical Association*, vol. 96, no. 455, pp. 1077–1087, 2001, ISSN: 0162-1459. DOI: 10.1198/016214501753208735.

[13] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," *Advances in Neural Information Processing Systems*, vol. 9, no. 2008, pp. 33–40, 2009, ISSN: 1532-4435. DOI: 10.1016/j.bbi.2008.05.010. arXiv: 0705.4485.

[14] S. Young and E. Scheinerman, "Random dot product graph models for social networks," *Algorithms and models for the web-graph*, pp. 138–149, 2007, ISSN: 0302-9743.

[15] C. L. M. Nickel, "Random dot product graphs a model for social networks," PhD Thesis, Johns Hopkins University, 2008.

[16] S. J. Young and E. Scheinerman, "Directed Random Dot Product Graphs," *Internet Mathematics*, vol. 5, no. 1-2, pp. 91–111, Jan. 2008, ISSN: 1542-7951. DOI: 10.1080/15427951.2008.10129301.

[17] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *Journal of Machine Learning Research*, vol. 11, pp. 985–1042, 2010, ISSN: 0012365X. DOI: 10.1145/1756006.1756039. arXiv: 0812.4905.

[18] S. I. Moreno, J. Neville, and S. Kirshner, "Learning mixed kronecker product graph models with simulated method of moments," *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13*, p. 1052, 2013, ISSN: 9781450321747. DOI: 10.1145/2487575.2487675.

[19] S. Moreno, J. Neville, and S. Kirshner, "Tied kronecker product graph models to capture variance in network populations," *ACM Transactions on Knowledge Discovery from Data*, vol. 12, no. 3, 2018, ISSN: 1556472X. DOI: 10.1145/3161885.

[20] T. N. Kipf and M. Welling, "Variational Graph Auto-Encoders," *arXiv:1611.07308 [cs, stat]*, Nov. 2016. arXiv: 1611.07308 [cs, stat].

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2014, pp. 2672–2680.

[22] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "GraphGAN: Graph Representation Learning with Generative Adversarial Nets," *arXiv:1711.08267 [cs, stat]*, Nov. 2017. arXiv: 1711.08267 [cs, stat].

[23] S. Tavakoli, A. Hajibagheri, and G. Sukthankar, "Learning Social Graph Topologies using Generative Adversarial Neural Networks," Jul. 2017. DOI: 10.13140/RG.2.2.16772.94082.

[24] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models," in *International Conference on Machine Learning*, 2018, pp. 5708–5717.

[25] M. Kim and J. Leskovec, "Multiplicative attribute graph model of real-world networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6516 LNCS, no. March, pp. 62–73, 2010, ISSN: 03029743. DOI: 10.1007/978-3-642-18009-5_7. arXiv: 1009.3499.

[26] J. J. Pfeiffer, S. Moreno, T. L. Fond, J. Neville, and B. Gallagher, "Attributed Graph Models: Modeling Network Structure with Correlated Attributes," *Proceedings of the 23rd International Conference on World Wide Web*, pp. 831–841, 2014. DOI: 10.1145/2566486.2567993.

[27] P. Robles, S. Moreno, and J. Neville, "Sampling of Attributed Networks From Hierarchical Generative Models," *KDD*, pp. 421–434, 2016, ISSN: 0146-4833.

[28] D. Jensen, J. Neville, and B. Gallagher, "Why collective inference improves relational classification," *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04*, p. 593, 2004. DOI: 10.1145/1014052.1014125.

[29] S. a Macskassy and F. Provost, "Classification in Networked Data : A Toolkit and a Univariate Case Study," *Journal of Machine Learning Research*, vol. 8, no. December 2004, pp. 935–983, 2007, ISSN: 15324435. DOI: 10.1021/jf901618z.

[30] N. Ghamrawi and A. Mccallum, "Collective Multi-Label Classification," pp. 195–200, 2005.

[31] J. Moore and J. Neville, "Deep collective inference," *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, no. 1, pp. 2364–2372, 2017.

[32] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14, New York, NY, USA: Association for Computing Machinery, Aug. 2014, pp. 701–710, ISBN: 978-1-4503-2956-9. DOI: 10.1145/2623330.2623732.

[33] A. Grover and J. Leskovec, "Node2vec: Scalable Feature Learning for Networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 855–864, ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939754.

[34] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "Struc2vec: Learning Node Representations from Structural Identity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17, New York, NY, USA: Association for Computing Machinery, Aug. 2017, pp. 385–394, ISBN: 978-1-4503-4887-4. DOI: 10.1145/3097983.3098061.

[35] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale Information Network Embedding," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15, Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, May 2015, pp. 1067–1077, ISBN: 978-1-4503-3469-3. DOI: 10.1145/2736277.2741093.

[36] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, Jan. 2009, ISSN: 1941-0093. DOI: 10.1109/TNN.2008.2005605.

[37] D. Wang, P. Cui, and W. Zhu, "Structural Deep Network Embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 1225–1234, ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939753.

[38] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation Learning on Graphs: Methods and Applications," *arXiv:1709.05584 [cs]*, Apr. 2018. arXiv: 1709.05584 [cs].

[39] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[40] J. Neville and D. Jensen, "Iterative classification in relational data," *Learning Statistical Models from Relational Data*, pp. 42–49, 2000. DOI: 10.1.1.23.2875.

[41] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-rad, "Collective Classification in Network Data," *AI magazine*, pp. 93–106, 2008, ISSN: 0738-4602.

[42] Q. Lu and L. Getoor, "Link-based Classification using Labeled and Unlabeled Data," *ICML workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, 2003.

[43] D. F. Gleich and A. B. Owen, "Moment-based estimation of stochastic Kronecker graph parameters," *Internet Mathematics*, no. August 2012, pp. 37–41, 2012.

[44] L. Li, D. Alderson, J. C. Doyle, and W. Willinger, "Towards a theory of scale-free graphs: Definition, properties, and implications," *Internet Mathematics*, vol. 2, no. 4, pp. 431–523, 2005.

[45] Virtanen, Pauli, Gommers, Ralf, Oliphant, Travis E., Haberland, Matt, Reddy, Tyler, Cournapeau, David, Burovski, Evgeni, Peterson, Pearu, Weckesser, Warren, Bright, Jonathan, van der Walt, Stéfan J., Brett, Matthew, Wilson, Joshua, Millman, K. Jarrod, Mayorov, Nikolay, Nelson, Andrew R. J., Jones, Eric, Kern, Robert, Larson, Eric, Carey, C J, Polat, lhan, Feng, Yu, Moore, Eric W., VanderPlas, Jake, Laxalde, Denis, Perktold, Josef, Cimrman, Robert, Henriksen, Ian, Quintero, E. A., Harris, Charles R., Archibald, Anne M., Ribeiro, Antônio H., Pedregosa, Fabian, van Mulbregt, Paul, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[46] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software (TOMS)*, vol. 22, no. 4, pp. 469–483, 1996.

[47] S. Moreno and J. Neville, "An Investigation of the Distributional Characteristics of Generative Graph Models," *Win*, 2009.

[48] B. Karrer and M. E. J. Newman, "Random graph models for directed acyclic networks," *Physical Review E*, 2009, ISSN: 1539-3755. DOI: 10.1103/PhysRevE.80.046110. arXiv: 0907.4346.

[49] P. Holme, "Epidemiologically Optimal Static Networks from Temporal Network Data," *PLoS Computational Biology*, vol. 9, no. 7, 2013, ISSN: 1553734X. DOI: 10.1371/journal.pcbi.1003142.

[50] L. E. Rocha and V. D. Blondel, "Bursts of Vertex Activation and Epidemics in Evolving Networks," *PLoS Computational Biology*, vol. 9, no. 3, 2013, ISSN: 1553734X. DOI: 10.1371/journal.pcbi.1002974.

[51] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: Simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002, ISSN: 00368075. DOI: 10.1126/science.298.5594.824.

[52] Q. Zhao, Y. Tian, Q. He, N. Oliver, R. Jin, and W. C. Lee, "Communication motifs: A tool to characterize social communications," *International Conference on Information and Knowledge Management, Proceedings*, pp. 1645–1648, 2010. DOI: 10.1145/1871437.1871694.

[53] X. Zhang, S. Shao, H. E. Stanley, and S. Havlin, "Dynamic motifs in socio-economic networks," *Europhysics Letters*, vol. 108, no. 5, 2014, ISSN: 12864854. DOI: 10.1209/0295-5075/108/58001.

[54] Y. Hulovatyy, H. Chen, and T. Milenkovi, "Exploring the structure and function of temporal networks with dynamic graphlets," *Bioinformatics*, 2015, ISSN: 14602059. DOI: 10.1093/bioinformatics/btv227. arXiv: 1412.3885.

[55] A. Paranjape, A. R. Benson, and J. Leskovec, "Motifs in Temporal Networks," *WSDM*, 2017. DOI: 10.1145/3018661.3018731. arXiv: 1612.09259.

[56] A. R. Benson, R. Abebe, M. T. Schaub, A. Jadbabaie, and J. Kleinberg, "Simplicial closure and higher-order link prediction," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 115, no. 48, E11221–E11230, 2018, ISSN: 10916490. DOI: 10.1073/pnas.1800683115. arXiv: 1802.06916.

[57] S. Purohit, L. B. Holder, and G. Chin, "Temporal Graph Generation Based on a Distribution of Temporal Motifs," in *Proceedings of the 14th International Workshop on Mining and Learning with Graphs*, 2018, p. 7.

[58] P. Holme, "Modern temporal network theory: A colloquium," *European Physical Journal B*, vol. 88, no. 9, 2015, ISSN: 14346036. DOI: 10.1140/epjb/e2015-60657-4. arXiv: 1508.01303.

[59] N. Perra, B. Gonçalves, R. Pastor-Satorras, and A. Vespignani, "Activity driven modeling of time varying networks," *Scientific Reports*, vol. 2, pp. 1–7, 2012, ISSN: 20452322. DOI: 10.1038/srep00469. arXiv: 1203.5351.

[60] G. Laurent, J. Saramäki, and M. Karsai, "From calls to communities: A model for time-varying social networks," *European Physical Journal B*, vol. 88, no. 11, pp. 1–10, 2015, ISSN: 14346036. DOI: 10.1140/epjb/e2015-60481-x. arXiv: 1506.00393.

[61] A. Moinet, M. Starnini, and R. Pastor-Satorras, "Burstiness and Aging in Social Temporal Networks," *Physical Review Letters*, vol. 114, no. 10, 2015, ISSN: 10797114. DOI: 10.1103/PhysRevLett.114.108701. arXiv: 1412.0587.

[62] A. Sunny, B. Kotnis, and J. Kuri, "Dynamics of history-dependent epidemics in temporal networks," *Physical Review E*, vol. 92, no. 2, p. 022 811, Aug. 2015. DOI: 10.1103/PhysRevE.92.022811.

[63] C. L. Vestergaard, M. Génois, and A. Barrat, "How memory generates heterogeneous dynamics in temporal networks," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 90, no. 4, pp. 1–19, 2014, ISSN: 15502376. DOI: 10.1103/PhysRevE.90.042805. arXiv: 1409.1805.

[64] D. Zhou, L. Zheng, J. Han, and J. He, "A Data-Driven Graph Generative Model for Temporal Interaction Networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20, New York, NY, USA: Association for Computing Machinery, Aug. 2020, pp. 401–411, ISBN: 978-1-4503-7998-4. DOI: 10.1145/3394486.3403082.

[65] J. Preusse, J. Kunegis, M. Thimm, S. Staab, and T. Gottron, "Structural Dynamics of Knowledge Networks.," *ICWSM*, vol. 17, p. 18, 2013.

[66] J. Kunegis, "KONECT - The koblenz network collection," *WWW 2013 Companion - Proceedings of the 22nd International Conference on World Wide Web*, pp. 1343–1350, 2013.

[67] B. Klimt and Y. Yang, "The enron corpus: A new dataset for email classification research," in *European Conference on Machine Learning*, Springer, 2004, pp. 217–226.

[68] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, 2–es, 2007.

[69] R. Pickhardt, *Extracting 2 social network graphs from the Democratic National Committee Email Corpus on Wikileaks*, https://www.rene-pickhardt.de/extracting-2-social-network-graphs-from-the-democratic-national-committee-email-corpus-on-wikileaks/, May 2018.

[70] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in facebook," in *Proceedings of the 2nd ACM Workshop on Online Social Networks*, 2009, pp. 37–42.

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

[72] J. Leskovec and A. Krevl, *SNAP Datasets: Stanford large network dataset collection*, 2014.

[73] W. Aiello, F. Chung, and L. Lu, "A random graph model for massive graphs," in *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, ser. STOC '00, New York, NY, USA: Association for Computing Machinery, May 2000, pp. 171–180, ISBN: 978-1-58113-184-0. DOI: 10.1145/335305.335326.

[74] G. Zeno, T. L. Fond, and J. Neville, "DYMOND: DYnamic MOtif-NoDes Network Generative Model," in *Proceedings of the Web Conference 2021 (WWW '21)*, Ljubljana, Slovenia: ACM, New York, NY, USA, 2021, p. 12. DOI: 10.1145/3442381.3450102.

[75] L.-k. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang, "Dynamic Network Embedding by Modeling Triadic Closure Process," in *AAAI*, 2018, pp. 571–578.

[76] P. Goyal, S. R. Chhetri, and A. Canedo, "Dyngraph2vec: Capturing network dynamics using dynamic graph representation learning," en, *Knowledge-Based Systems*, vol. 187, p. 104 816, Jan. 2020, ISSN: 0950-7051. DOI: 10.1016/j.knosys.2019.06.024.

[77] L. Zhang, L. Zhao, S. Qin, D. Pfoser, and C. Ling, "TG-GAN: Continuous-time Temporal Graph Deep Generative Models with Time-Validity Constraints," in *Proceedings of the Web Conference 2021*, ser. WWW '21, New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 2104–2116, ISBN: 978-1-4503-8312-7. DOI: 10.1145/3442381.3449818. (visited on 02/17/2023).

[78] Z. Xu, Z. Ou, Q. Su, J. Yu, X. Quan, and Z. Lin, "Embedding Dynamic Attributed Networks by Modeling the Evolution Processes," in *The 28th International Conference on Computational Linguistics*, 2020. arXiv: 2010.14047.

[79] D. Wang, T. Zhao, N. V. Chawla, and M. Jiang, "Dynamic Attributed Graph Prediction with Conditional Normalizing Flows," in *2021 IEEE International Conference on Data Mining (ICDM)*, Dec. 2021, pp. 1385–1390. DOI: 10.1109/ICDM51629.2021.00176.

[80] S. Limnios, A. Elliott, M. Cucuringu, and G. Reinert, "Random Walk based Conditional Generative Model for Temporal Networks with Attributes," in *NeurIPS 2022 Workshop on Synthetic Data for Empowering ML Research*, Nov. 2022. (visited on 02/17/2023).

[81] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling Tabular data using Conditional GAN," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019. (visited on 03/16/2023).

[82] N. Reimers and I. Gurevych, *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*, Aug. 2019. DOI: 10.48550/arXiv.1908.10084. arXiv: arXiv:1908.10084. (visited on 04/04/2023).

[83] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, *DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter*, Feb. 2020. DOI: 10.48550/arXiv.1910.01108. arXiv: arXiv:1910.01108. (visited on 04/04/2023).

[84] A. Cohan, S. Feldman, I. Beltagy, D. Downey, and D. Weld, "SPECTER: Document-level Representation Learning using Citation-informed Transformers," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, 2020, pp. 2270–2282. DOI: 10.18653/v1/2020.acl-main.207. (visited on 04/04/2023).

[85] A. Singh, M. D'Arcy, A. Cohan, D. Downey, and S. Feldman, *SciRepEval: A Multi-Format Benchmark for Scientific Document Representations*, Nov. 2022. DOI: 10.48550/arXiv.2211.13308. arXiv: arXiv:2211.13308. (visited on 04/04/2023).

[86] M. Grootendorst, *BERTopic: Neural topic modeling with a class-based TF-IDF procedure*, Mar. 2022. DOI: 10.48550/arXiv.2203.05794. arXiv: 2203.05794 [cs]. (visited on 04/04/2023).

[87] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "ArnetMiner: Extraction and mining of academic social networks," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08, New York, NY, USA: Association for Computing Machinery, Aug. 2008, pp. 990–998, ISBN: 978-1-60558-193-4. DOI: 10.1145/1401890.1402008. (visited on 03/21/2023).

[88] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. (Hsu, and K. Wang, "An Overview of Microsoft Academic Service (MAS) and Applications," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15 Companion, New York, NY, USA: Association for Computing Machinery, May 2015, pp. 243–246, ISBN: 978-1-4503-3473-0. DOI: 10.1145/2740908.2742839. (visited on 03/21/2023).

[89] A. Litel, *Tweets of congress*, Feb. 2023. [Online]. Available: https://github.com/alexlitel/congresstweets (visited on 03/21/2023).

[90] Twitter, Inc., *Twitter api for academic research*, version v2. [Online]. Available: https://developer.twitter.com/en/products/twitter-api/academic-research (visited on 04/03/2023).

[91] S. Kumar, W. L. Hamilton, J. Leskovec, and D. Jurafsky, "Community Interaction and Conflict on the Web," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18, Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, Apr. 2018, pp. 933–943, ISBN: 978-1-4503-5639-8. DOI: 10.1145/3178876.3186141. (visited on 02/17/2023).

[92] J. Baumgartner, *Reddit Statistics*, https://pushshift.io/. (visited on 03/21/2023).

[93] C. Tan and L. Lee, "All Who Wander: On the Prevalence and Characteristics of Multi-community Engagement," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15, Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, May 2015, pp. 1056–1066, ISBN: 978-1-4503-3469-3. DOI: 10.1145/2736277.2741661. (visited on 03/21/2023).

[94] P. Liu, A. R. Benson, and M. Charikar, "Sampling Methods for Counting Temporal Motifs," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, ser. WSDM '19, New York, NY, USA: Association for Computing Machinery, Jan. 2019, pp. 294–302, ISBN: 978-1-4503-5940-5. DOI: 10.1145/3289600.3290988. (visited on 03/21/2023).

[95] J. Hessel, C. Tan, and L. Lee, "Science, AskScience, and BadScience: On the Coexistence of Highly Related Communities," *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 10, no. 1, pp. 171–180, 2016, ISSN: 2334-0770. DOI: 10.1609/icwsm.v10i1.14739. (visited on 03/21/2023).

[96] V. Zhelezniak, A. Shen, D. Busbridge, A. Savkov, and N. Hammerla, "Correlations between Word Vector Sets," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 77–87. DOI: 10.18653/v1/D19-1008. (visited on 04/13/2022).

[97] V. Zhelezniak, A. Savkov, A. Shen, and N. Hammerla, "Correlation Coefficients and Semantic Textual Similarity," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 951–962. DOI: 10.18653/v1/N19-1100. (visited on 04/13/2022).

[98] G. J. Székely, M. L. Rizzo, and N. K. Bakirov, "Measuring and testing dependence by correlation of distances," *The Annals of Statistics*, vol. 35, no. 6, pp. 2769–2794, Dec. 2007, ISSN: 0090-5364, 2168-8966. DOI: 10.1214/009053607000000505. (visited on 05/17/2023).

[99] J. Xiang, Y. Liu, D. Cai, H. Li, D. Lian, and L. Liu, *Assessing Dialogue Systems with Distribution Distances*, May 2021. DOI: 10.48550/arXiv.2105.02573. arXiv: arXiv:2105.02573. (visited on 03/29/2023).

[100] C. Ramos-Carreño and J. L. Torrecilla, "Dcor: Distance correlation and energy statistics in Python," *SoftwareX*, vol. 22, p. 101 326, May 2023, ISSN: 2352-7110. DOI: 10.1016/j.softx.2023.101326. (visited on 05/17/2023).

[101] C. Ling, C. Yang, and L. Zhao, "Motif-guided heterogeneous graph deep generation," *Knowledge and Information Systems*, Mar. 2023, ISSN: 0219-3116. DOI: 10.1007/s10115-023-01863-0. (visited on 05/17/2023).

[102] J. B. Lee, R. A. Rossi, X. Kong, S. Kim, E. Koh, and A. Rao, "Graph Convolutional Networks with Motif-based Attention," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, ser. CIKM '19, New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 499–508, ISBN: 978-1-4503-6976-3. DOI: 10.1145/3357384.3357880. (visited on 05/27/2021).

[103] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen, "A Survey of Large Language Models," 2023. DOI: 10.48550/ARXIV.2303.18223. (visited on 06/20/2023).

[104] J. Li, T. Tang, W. X. Zhao, and J.-R. Wen, "Pretrained Language Model for Text Generation: A Survey," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, Montreal, Canada: International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 4492–4499, ISBN: 978-0-9992411-9-6. DOI: 10.24963/ijcai.2021/612. (visited on 06/20/2023).

[105] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, and L. Sun, "A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT," 2023. DOI: 10.48550/ARXIV.2303.04226. (visited on 06/20/2023).

[106] S. Huang, L. Dong, W. Wang, Y. Hao, S. Singhal, S. Ma, T. Lv, L. Cui, O. K. Mohammed, B. Patra, Q. Liu, K. Aggarwal, Z. Chi, J. Bjorck, V. Chaudhary, S. Som, X. Song, and F. Wei, "Language Is Not All You Need: Aligning Perception with Language Models," 2023. DOI: 10.48550/ARXIV.2302.14045. (visited on 06/20/2023).

[107] G. Zeno, T. L. Fond, and J. Neville, "Dynamic Network Modeling from Motif-Activity," in *Companion Proceedings of the Web Conference 2020 (WWW '20 Companion)*, 2020, ISBN: 978-1-4503-7024-0.

[108] G. Zeno and J. Neville, "Investigating the impact of graph structure and attribute correlation on collective classification performance," in *Proceedings of the 12th International Workshop on Mining and Learning with Graphs (MLG)*, 2016.

[109] R. P. Lippmann, W. M. Campbell, D. J. Weller-Fahy, A. C. Mensch, G. M. Zeno, and J. P. Campbell, "Finding Malicious Cyber Discussions in Social Media," *LINCOLN LABORATORY JOURNAL*, vol. 22, no. 1, 2016.

VITA

Giselle M. Zeno Torres obtained her Bachelor of Science in Computer Science from the University of Puerto Rico, Bayamón, in 2010, graduating with Magna Cum Laude honors. In 2009, she was recognized with the Model Student award by the Department of Computer Science, and she received the Honors Program scholarship throughout her undergraduate studies from 2006 to 2010.

She embarked on her academic journey at Purdue University, where she joined the Department of Computer Science to pursue a PhD. In the Network Learning & Discovery lab, she worked under the guidance of Professor Jennifer Neville. To support her doctoral studies, she was awarded the GEM Fellowship, sponsored by Intel, and the Frederick N. Andrews Fellowship from Purdue University.

In 2021, Giselle successfully completed her Master of Science in Computer Science from Purdue University. Throughout her doctoral studies, she focused on characterizing collective inference methods and the development of generative models for analyzing temporal graph-structured data, including information, communication, and social networks. Her research findings have been shared through multiple publications at esteemed data mining and machine learning conferences and workshops.

In addition to her academic achievements, Giselle boasts six years of industry and national lab experience. She gained valuable insights and practical skills through internships at renowned institutions such as Intel Corp, MIT Lincoln Laboratory, and Lawrence Livermore National Laboratory. This diverse experience has enriched her understanding of real-world applications and challenges in the field.

PUBLICATIONS

- G. Zeno, T. L. Fond, and J. Neville, "DYMOND: DYnamic MOtif-NoDes Network Generative Model," in *Proceedings of the Web Conference 2021 (WWW '21)*, Ljubljana, Slovenia: ACM, New York, NY, USA, 2021, p. 12. DOI: 10.1145/3442381.3450102
- G. Zeno, T. L. Fond, and J. Neville, "Dynamic Network Modeling from Motif-Activity," in *Companion Proceedings of the Web Conference 2020 (WWW '20 Companion)*, 2020, ISBN: 978-1-4503-7024-0
- G. Zeno and J. Neville, "Investigating the impact of graph structure and attribute correlation on collective classification performance," in *Proceedings of the 12th International Workshop on Mining and Learning with Graphs (MLG)*, 2016
- R. P. Lippmann, W. M. Campbell, D. J. Weller-Fahy, A. C. Mensch, G. M. Zeno, and J. P. Campbell, "Finding Malicious Cyber Discussions in Social Media," *LINCOLN LABORATORY JOURNAL*, vol. 22, no. 1, 2016