DYANE: DYnamic Attributed Node rolEs Generative Model

Giselle Zeno
Purdue University
West Lafayette, IN, USA
gzenotor@purdue.edu

Jennifer Neville
Microsoft Research & Purdue University
West Lafayette, IN, USA
jenneville@microsoft.com

ABSTRACT

Recent work has shown that modeling higher-order structures, such as motifs or graphlets, can capture the complex network structure and dynamics in a variety of graph domains (e.g., social sciences, biology, chemistry). However, many dynamic networks are not only rich in structure, but also in content information. For example, an academic citation network has content such as the title and abstracts of the papers. Currently, there is a lack of generative models for dynamic networks that also generate content. To address this gap, in this work we propose DYnamic Attributed Node rolEs (DYANE)—a generative model 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. We evaluate DYANE on real-world networks against other dynamic graph generative model baselines. DYANE outperforms the baselines in graph structure and node behavior, improving the KS score for graph metrics by 21-31% and node metrics by 17-27% on average, and produces content embeddings similar to the observed network. We also derive 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.

CCS CONCEPTS

• **Theory of computation → Dynamic graph algorithms**; **Network formation**; *Social networks*; • **Computing methodologies → Motif discovery**; *Neural networks*; Topic modeling.

KEYWORDS

Dynamic Attributed Networks; Temporal Attributed Graphs; Graph Neural Networks; Motifs; Network Evolution

ACM Reference Format:
Giselle Zeno and Jennifer Neville. 2023. DYANE: DYnamic Attributed Node rolEs Generative Model. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583780.3614858

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

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3614858

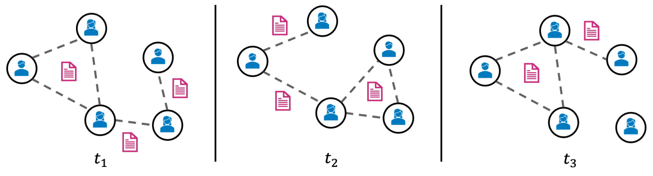


Figure 1: Academic Co-authorship Network Example, with graph structure and attributes changing over time

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.

In this paper, 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 autocorrelation 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 paper is organized as follows: We first go over related work to show where our model fits in the literature in Section 2. In Section 3, we introduce our main definitions and formal problem statement. Section 4 presents our proposed model, DYANE. In Section 5, we present our evaluation methodology, metrics, datasets, and baselines used. Experimental results and discussion is in Section 6. Finally, we present our conclusions in Section 8.

2 RELATED WORK

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 [7, 8, 11, 17, 22]. Although traditionally most graph models have been edge-based, motifs have been established as building blocks for the structure of networks [2, 16, 37]. 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) [19] 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) [34] 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. TagGen [40] 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 [36] captures structural and temporal patterns of dynamic networks using temporal walks and a discriminator.

In the case of attributed graphs, although there are various models such as MAG [9], AGM [18], and CSAG [21], they have focused on static graphs. The dynamic attribute network embedding model (Dane) [33] uses an activeness-aware neighborhood embedding method (GraphSAGE [5]) to extract the higher-order neighborhood information at each given timestamp. Dane is used to predict the network status at the next timestamp (i.e., edges and node categories). Dynamic Graph Normalizing Flows (DGNF) is a graph representation model for dynamic attributed graphs that can be used for link prediction [30]. CTWalk models dynamic attributed networks, capitalizing on temporal random walks and conditional GANs [13]. Specifically, it builds upon TagGen [40], for temporal edge generation, and CTGAN [32], 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, apart from our proposed model.

3 DEFINITIONS

Definition 3.1 (Attributed Graph Snapshot). A graph snapshot is a time-slice of a network at time t, defined as $G_t = (V_t, E_t, X_t, S_t)$, where $V_t \subseteq V$ is the set of active nodes, $E_t \subseteq E$ is the set of edges at time t, X_t is the set of attributes at time t, and $S_t \subseteq S$ are the edge timestamps.

Definition 3.2 (Attributed Dynamic Network). A dynamic network (or graph) $\mathbf{G} = \{G_1, \ldots, G_T\}$ is a sequence of attributed graph time-slices where T is the number of timesteps.

Definition 3.3 (Motif). We define a motif as a 3-node subgraph $\{u, v, w\}$ and its *motif type i* 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 edges).

3.1 Problem Statement

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 1: ***Attributed Dynamic Network Generation***

INPUT: *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: *An attributed dynamic network $\mathbf{G}' = \left\{G'_1, \ldots, G'_{T'}\right\}$, where the distribution of graph structure for $\mathbf{G}'$ matches $\mathbf{G}$, the node behavior of a specific node $v_{i'}$ in $\mathbf{G}'$ 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 $\mathbf{s}_{out} = \left\{s(G'_1), \ldots, s(G'_{T'})\right\}$. 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 $\{\mathbf{s}(v_{j'}|\mathbf{G}')\}_{j'\in\mathbf{G}'}$. Likewise, 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 $\mathbf{X}' = \{X'_1, \ldots, X'_{T'}\}$.

4 DYNAMIC ATTRIBUTED NODE ROLES

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)[1], extends motif-based node roles (Fig. 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
(5) Node interactions are determined by the node roles and latent topics/interests

We first describe DYANE’s generative process and 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 (Fig. 2).

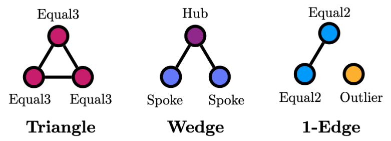


Figure 2: Motif Types and Node Roles

The motivation for our 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.

4.1 Generative Process

The overall generative process is described in Fig. 3, and the model architecture is illustrated in Fig. 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 (Fig. 3a and eq. 1).

$$\begin{aligned} x_v &\sim \Gamma(\alpha, \beta) \\ g &:= \left[v \in V \mapsto \lfloor |x_v| \rfloor \right] \end{aligned} \tag{1}$$

At each timestep t, we first calculate any new triplets $\mathcal{U}_t$ (Eq. 1) that can be sampled as motifs from the set of active nodes V'_t (Eq. 3), where $g(v)$ is the arrival time of node v (Eq. 1).

$$\mathcal{U}_t = \left\{ \{u, v, w\} \subset V'_t : \{u, v, w\} \notin \mathcal{U}_{t-1} \right\} \tag{2}$$

$$V'_t = \{ v \in V : g(v) \leq t \} \tag{3}$$

[1]Code is available at https://github.com/zeno129/DYANE

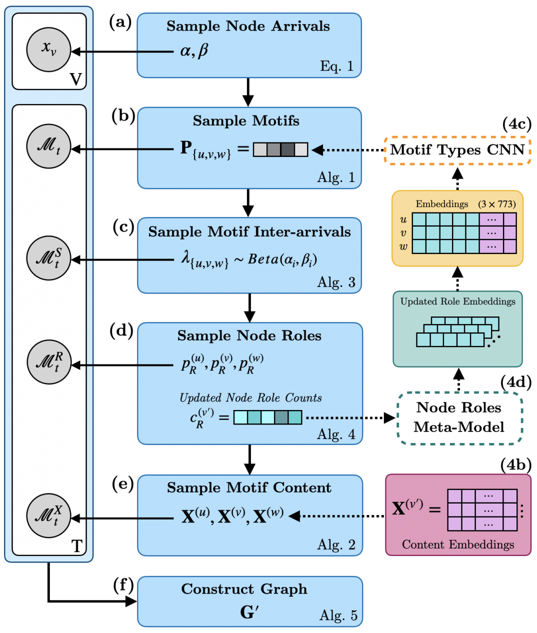


Figure 3: DYANE Generative Process

In fig. 3b, we sample motifs from these new triplets in $\mathcal{U}_t$, using the motif type probabilities obtained from the Motif Types CNN (Alg. 1 and line 3), which are based on the node roles and content embeddings (Fig. 4c). For the motif inter-arrivals (Fig. 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 (Alg. 2). Afterwards, we sample node roles for each motif node and update the counts for node roles assigned (Fig. 3d). The updated counts (Alg. 3 and line 15) are then used to update the node role embeddings with the meta-model (Fig. 4d).

In fig. 3e, we sample content embeddings for each timestep where the motif occurs (Alg. 4). In line 3, we only consider the content embeddings of “influential” nodes (i.e., nodes that form 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 appears, 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 sampled content embeddings $\mathcal{M}^X$ (Alg. 5 and fig. 3f).

4.2 Estimation

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

4.2.1 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:

$$\begin{aligned} \hat{x}_v &= \arg\min_t \mathbb{1}(v \in V_t) \\ X &= (\hat{x}_1, \ldots, \hat{x}_{|V|}) \\ X &\sim \Gamma(\alpha, \beta) \end{aligned}$$

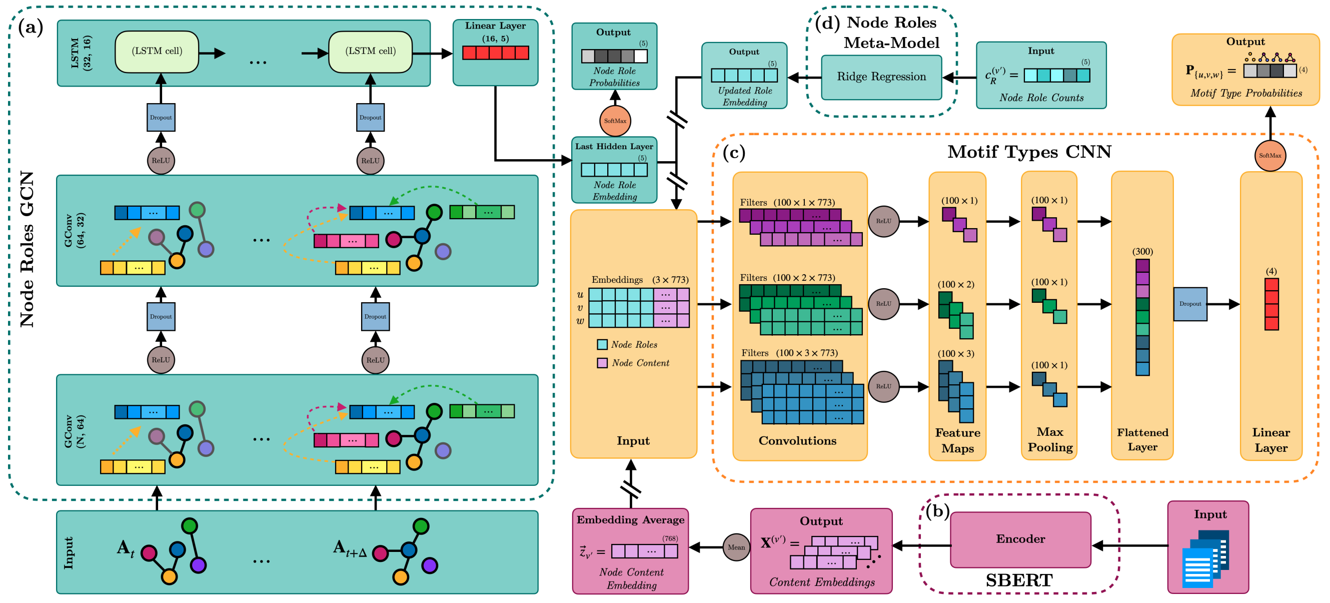


Figure 4: DYANE Model Architecture. (a) Node Roles GCN, (b) Content Model, (c) Motif Types CNN, (d) Node Roles Meta-Model

Algorithm 1: `SampleMotifs`

```
input:  $\mathcal{U}_t$, $q(i)$
output: $\mathcal{M}_t$ // sampled motifs
        $\mathcal{M}_t^T$ // motif types
begin
    $n := [i \in [3,2,1] \mapsto q(i) \cdot |\mathcal{U}_t|]$
    // Get probabilities from CNN (Eq. 10)
    $\mathbf{P} \leftarrow [\{u,v,w\} \in \mathcal{U}_t \mapsto$ MotifTypesCNN$(u,v,w)]$
    for $i \in [3,2,1]$ 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
            $\mathcal{U}'_t \leftarrow$ ReservoirSampling$(\mathcal{U}_t,$ max_size$)$
            $\mathbf{P}' \leftarrow [\{u,v,w\} \in \mathcal{U}'_t \mapsto \mathbf{P}_{\{u,v,w\}}]$
        $\mathcal{M}^{(i)} \sim Mult(\mathbf{P}'^{(i)}, n^{(i)})$ // sample motifs
        $\mathcal{M}_t^T := [m \in \mathcal{M}^{(i)} \mapsto i]$ // save motif types
        $\mathcal{U}_t \leftarrow \mathcal{U}_t - \mathcal{M}^{(i)}$ // triplets left to sample
```

Algorithm 2: `SampleMotifTimesteps`

```
input:  $T$, $t$, $\mathcal{M}_t$, $\mathcal{M}_t^T$
output: $\mathcal{M}_t^S$ // motif timesteps
begin
    for $m \in \mathcal{M}_t$ do
        $\lambda_{\{u,v,w\}} \sim Beta(\alpha_i, \beta_i)$ // sample, $i := \mathcal{M}_t^T(m)$
        $p \leftarrow \frac{\lambda_{\{u,v,w\}} \cdot n + \alpha_i}{n + \alpha_i + \beta_i}$ // adjust for $n := T - t$
        $t' \leftarrow t$ // init
        while $t' \le T$ do
            $k \sim NB(1, p)$ // sample inter-arrival time
            $t' \leftarrow t' + k$ // update next timestep
            if $t' \le T$ then $\mathcal{M}_t^S(m)$.append$(t')$
```

4.2.2 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 $q(i)$ 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):

$$\hat{q}(i) = \begin{cases} \frac{|\{\{u,v,w\} \in \mathcal{M} : \mathcal{M}^T(\{u,v,w\}) = i\}|}{\binom{|V|}{3}}, & \text{if } i \in [1,2,3] \\ 1 - \sum_{i=1}^{3} \hat{q}(i), & \text{otherwise} \end{cases} \tag{4}$$

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 with nodes u, v, w.

4.2.3 Motif Inter-Arrivals. We estimate the inter-arrival rates of each observed motif $\{u, v, w\}$ using weighted edge counts (Eq. 5). Then, our model fits a Beta distribution with parameters α_i, β_i from 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 ($i = 0$).

$$\widehat{\lambda}_{\{u,v,w\}} = \frac{\sum_{t=1}^{T} d_t(\{u,v,w\})}{T} \tag{5}$$

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 $\{u, v, w\}$ (Eq. 6a). The weights $\omega_t^{(i)}$ will depend on the motif type i of $\{u, v, w\}$ and are calculated for each edge of the motif (Eq. 7a).

Algorithm 3: `SampleNodeRoles`

```
input: $\mathcal{M}_t$, $\mathcal{M}_t^S$, $c_R$
output: $\mathcal{M}_t^R$, $c_R$ // sampled node roles & updated counts
1  begin
2      for $m \in \mathcal{M}_t$ do
3          if $i = 3$ then // triangle
4              $\mathcal{M}_t^R(m, v) \leftarrow$ equal3, $\forall v \in m$
5          else if $i = 2$ then // wedge
6              $p_h \leftarrow \left[v \in m \mapsto \frac{P[v,\text{hub}]}{\sum_{v' \in m} P[v',\text{hub}]}\right]$ // normalize
7              $v_h \sim Bin(m, p_h)$ // sample hub node
8              $\mathcal{M}_t^R(m, v_h) \leftarrow$ hub
9              $\mathcal{M}_t^R(m, v) \leftarrow$ spoke, $\forall v \in m, v \neq v_h$
10         else if $i = 1$ then // 1-edge
11             $p_o \leftarrow \left[v \in m \mapsto \frac{P[v,\text{outlier}]}{\sum_{v' \in m} P[v',\text{outlier}]}\right]$ // normalize
12             $v_o \sim Bin(m, p_o)$ // sample outlier node
13             $\mathcal{M}_t^R(m, v_o) \leftarrow$ outlier
14             $\mathcal{M}_t^R(m, v) \leftarrow$ equal2, $\forall v \in m, v \neq v_o$
15         count$(v, \mathcal{M}_t^R(m, v))$ −= $|\mathcal{M}_t^S(m)|$, $\forall v \in m$ // update
```

Algorithm 4: `SampleMotifContent`

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

Algorithm 5: `ConstructGraph`

```
input: $\mathcal{M}$, $\mathcal{M}^S$, $\mathcal{M}^E$, $\mathcal{M}^X$
output: $\mathbf{G}' = \{G'_1, \ldots, G'_{T'}\}$ // where $G'_t = (V'_t, E'_t, X'_t, S'_t)$
1  begin
2      $\mathcal{M}_t := \{\{u, v, w\} \in \mathcal{M} : t \in \mathcal{M}^S(\{u, v, w\})\}$
3      for $t \in [1, \ldots, T']$ do
4          $V'_t \leftarrow \{v \in V : g(v) \leq t\}$ // nodes & content
5          $\mathbf{X}'_t := [v' \in V'_t \mapsto \{\mathcal{M}^X(m) : v' \in m, m \in \mathcal{M}_t\}]$
           // edges & timestamps
6          $E'_t \leftarrow \{(u', v') \in \mathcal{M}^E(m) : \{u'v'\} \subset m, m \in \mathcal{M}_t\}$
7          $S'_t := [(u'v') \in \mathcal{M}^E(m) \mapsto t : \{u'v'\} \subset m, m \in \mathcal{M}_t]$
```

$$d_t(\{u, v, w\}) = \frac{\sum_{(u',v') \in e_t(\{u,v,w\})} \omega_t^{(i)}(u', v')}{|e_t(\{u, v, w\})|} \tag{6a}$$

$$e_t(\{u, v, w\}) = \{(u', v') \mid \{u', v'\} \subset \{u, v, w\}, (u', v') \in E_t\} \tag{6b}$$

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 (Eq. 7a). 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 $q(i)$ (Eq. 4) have a high enough inter-arrival rate to show up (i.e., triangles).

$$\omega_t^{(i)}(u', v') = \frac{min\left(r_t^{(i)}(u', v'),\ n_t^{(i)}(u', v')\right)}{n_t^{(i)}(u', v')} \tag{7a}$$

$$r_t^{(i)}(u', v') = \begin{cases} c_t(u', v'), & \text{if } i = 3 \\ r_t^{(i+1)}(u', v') - n_t^{(i+1)}(u', v'), & \text{if } i \in [1, 2], \text{ and } r_t^{(i+1)}(u', v') > 0 \\ 0, & \text{otherwise} \end{cases} \tag{7b}$$

$$n_t^{(i)}(u', v') = \begin{cases} |\mathcal{N}_t(u') \cap \mathcal{N}_t(v')|, & \text{if } i = 3 \\ |\mathcal{N}_t(u') \oplus \mathcal{N}_t(v')|, & \text{if } i = 2 \\ |V - (\mathcal{N}_t(u') \cup \mathcal{N}_t(v'))|, & \text{if } i = 1 \end{cases} \tag{7c}$$

where $|n_t^{(i)}(u', v')|$ 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 $r_t^{(i)}(u', v')$ 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.

4.2.4 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_i^{roles} be its node role probabilities, x_i^{roles} its weighted node role counts (elsewhere noted as $c_R^{(v_i)}$ for conciseness), and $n = \sum_r x_{i,r}^{\text{roles}}$ the total number of role counts, such that $x_i^{\text{roles}} \sim Mult(n, p_i^{\text{roles}})$. We then estimate the node role probabilities as follows:

$$\hat{p}_i^{\text{roles}} = \frac{x_i^{\text{roles}}}{n} = \left(\frac{x_{i,1}^{\text{roles}}}{n}, \ldots, \frac{x_{i,k}^{\text{roles}}}{n}\right) \tag{8}$$

where $\mathcal{R} = \{$`equal3`, `hub`, `spoke`, `equal2`, `outlier`$\}$ is the set of possible roles, $k = |\mathcal{R}|$ is the number of roles, and $x_{i,r}^{\text{roles}} = count(v_i, r)$ is the weighted count of times that node v_i had role r (Alg. 6). Edge-weights (Eq. 7a) are used to avoid over-counting the roles for motifs of the same type with a shared edge.

4.2.5 Node Roles GCN (Fig. 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 convolution layers.

$$\mathbf{H}_t^{(l+1)} = f\left(\mathbf{H}_t^{(l)}, \mathbf{A}_t\right) = \text{ReLU}\left(\mathbf{A}_t \mathbf{H}_t^{(l)}\right) \mathbf{W}_t^{(l)}$$
$$\mathbf{H}_t^{(1)} = \mathbf{I}$$

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 LSTM sequence combines the temporal aspect. The temporal input for an LSTM cell at time t is $\mathbf{H}_t^{(L+1)}$.

$$\mathbf{o}_i = \text{LSTM}\left(\mathbf{H}_{1:T}^{(L+1)}(v_i)\right)$$

Algorithm 6: GetNodeRoleCounts

```
input: $V, E, T$
output: $c_R := [v \in V, r \in \mathcal{R} \mapsto count(v,r)]$ // 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 $w \in (\mathcal{N}_t(u) \cap \mathcal{N}_t(v))$ do
                    $count(v', \text{equal3})$ += $\frac{\omega^{(3)}}{3}$, $\forall v' \in \{u,v,w\}$
            if $n_t^{(2)} > 0$ and $r_t^{(2)}(u,v) > 0$ then // wedges
                for $w \in (\mathcal{N}_t(u) \oplus \mathcal{N}_t(v))$ do
                    for $v' \in \{u,v,w\}$ do
                        $r \leftarrow$ GetRoleTimestep($v', E_t, \{u,v,w\}$)
                        if $r = \text{hub}$ then
                            $count(v', \text{hub})$ += $\frac{\omega^{(2)}}{2}$ // Eq. 7a
                        else
                            $count(v', \text{spoke})$ += $\omega^{(2)}$ // Eq. 7a
            if $n_t^{(1)} > 0$ and $r_t^{(1)}(u,v) > 0$ then // 1-edges
                $count(u, \text{equal2})$ += $r_t^{(1)}(u',v')$ // Eq. 7b
                $count(v, \text{equal2})$ += $r_t^{(1)}(u',v')$ // Eq. 7b
                for $w \in (V - (\mathcal{N}_t(u) \cup \mathcal{N}_t(v)))$ do
                    $count(w, \text{outlier})$ += $\omega^{(1)}$ // Eq. 7a
```

$$\mathbf{o}_i^{\text{roles}} = \mathbf{W}_{\text{final}}\mathbf{o}_i + \mathbf{b}_{\text{final}} \tag{9}$$
$$\hat{\mathbf{y}}_i^{\text{roles}} = \text{softmax}\left(\mathbf{o}_i^{\text{roles}}\right)$$

Then, the output of the LSTM sequence $\mathbf{o}_i$, for node v_i, is passed through a Linear layer and a Softmax function to obtain the node role probabilities $\hat{\mathbf{y}}_i$ 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 (Eq. 8) as the ground truth $\mathbf{y}_i$ for node v_i.

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

The last hidden layer of the Node Roles GCN is used as the initial node role embedding for a node (Eq. 9).

4.2.6 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 sampled node role counts c_R (Alg. 3 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,

$$R(w) = \sum_{i=j}^{d} w_i^2$$
$$\frac{1}{n}||Y_{\text{roles}} - X_{\text{roles}}w||_2^2 + \lambda\sum_{j=1^d}|w_j|^2 \rightarrow \underset{w \in \mathbb{R}^d}{\arg\min}$$

and the closed-form solution for w is:

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

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

$$\hat{\mathbf{y}}_i^{\text{roles}} = \sum_j w_j x_j^{\text{roles}\top} x_i^{\text{roles}}$$

Note that in this case $\hat{\mathbf{y}}_i^{\text{roles}}$ is the predicted node role embedding of node v_i, and the ground truth $\mathbf{y}_i^{\text{roles}}$ is the Node Roles GCN last hidden layer output $\mathbf{o}_i^{\text{roles}}$ (Eq. 9).

4.2.7 Content Model (Fig. 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 [20] encoder. Then, we take the average of those embeddings as the content portion of the node embedding (i.e., $\vec{z}_{v'} = \text{avg}(\mathbf{X}^{(v')})$). We use available pre-trained models. Specifically, we use DistilRoBERTa [23] and SPECTER [3, 24] for scientific/academic content.

4.2.8 Motif Types CNN (Fig. 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:

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

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:

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

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

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:

$$\mathbf{o} = \left(o_k^1, \ldots, o_k^L, \ldots, o_K^1, \ldots, o_K^L\right)$$
$$\mathbf{o}' = \mathbf{W}_{\text{final}}\mathbf{o} + \mathbf{b}_{\text{final}}$$

To get the motif type probability, we apply a Softmax function to obtain $\mathbf{P}_{\{u,v,w\}}$. Let $\mathbf{P}_{\{u,v,w\}}^{(i)}$ 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:

$$\mathbf{P}_{\{u,v,w\}} = \text{softmax}\left(\mathbf{o}'\right) \tag{10}$$
$$\hat{y}_{\{u,v,w\}} = \arg\max_i \left(\mathbf{P}_{\{u,v,w\}}^{(i)}\right)$$

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 **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:

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

5 METHODOLOGY

We first describe the baseline models (Subsection 5.1) and datasets (Subsection 5.2) used in our evaluation. Then, we introduce the metrics for evaluating graph structure, node behavior and content embeddings (Subsection 5.3). We estimate model parameters on the observed input graph for our model and all baselines.

5.1 Baselines

5.1.1 TagGen is a deep graph generative model for dynamic networks [40]. 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[2].

5.1.2 TG-GAN learns distributions of temporal walks to capture topological and temporal patterns of temporal graphs [36]. It consists of two parts, a temporal walk generator and a discriminator. We used the available implementation of TG-GAN[3]. 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.

5.1.3 CTWalk models temporal graphs with attributes, capitalizing on temporal random walks and conditional GANs [13]. Specifically, it builds upon TagGen [40], for temporal edge generation, and CTGAN [32], 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 [4].

5.2 Datasets

We use the datasets described below, with more detailed statistics shown in Table 1.

5.2.1 Arnetminer (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 [25, 28].

5.2.2 Congress Tweets (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 [14, 29].

5.2.3 Reddit Cross-posts (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 [1, 10, 27].

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

5.2.4 Reddit Replies (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 [1, 6, 15].

Table 1: Attributed Dynamic Networks Statistics

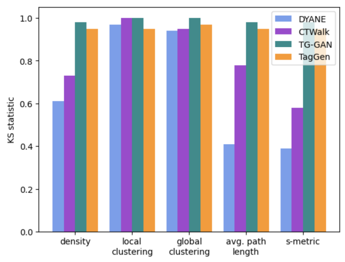
Dataset	$\lvert V\rvert$	$\lvert E\rvert$	Unique	$\lvert T\rvert$	Length	Time Span
Aminer	602	4,124	1,041	10	1 yr.	2010–2019
Congress	1,283	425,297	116,097	762	1 day	12/19–12/21
R-Posts	1,009	48,780	15,166	41	1 mo.	12/13–04/17
R-Replies	1,894	58,836	7,330	41	1 mo.	12/13–04/17

5.3 Evaluation

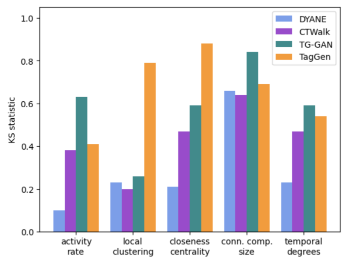
We use two sets of metrics in our evaluation for graph structure and node behavior. The majority of graph structure metrics we selected are widely used to characterize graphs. With these first set of metrics we aim to measure if the overall graph structure of the generated graph $\mathbf{G}'$ is similar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

5.3.1 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 [12]. We calculate the graph structure metrics for each time-window of the generated graph $\mathbf{G}'$ and the input graph **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 **G**.

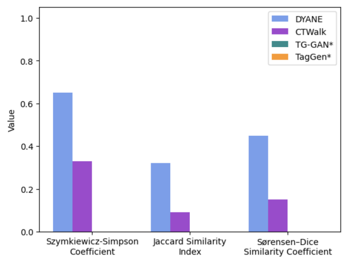
5.3.2 Node Behavior. To compare the temporal node behavior of the generated graphs against the datasets, we use the following *node-aligned, temporal* metrics: activity rate, temporal degree distribution [34], clustering coefficient, closeness centrality, and the size of its connected component. We calculate the *node-aligned, temporal* metrics for every node in the input graph **G** and 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 $\mathbf{s}(v_i|\mathbf{G}) = \{s(v_i|G_1), \ldots, s(v_i|G_T)\}$ for each metric s. Since the nodes in **G** do not necessarily correspond to those in $\mathbf{G}'$, we consider the inter-quartile range (IQR) of values over time $\{\mathbf{s}(v_j|\mathbf{G})\}_{j\in\mathbf{G}}$. We then perform a 2-dimensional KS test using the Q_1 and Q_3 values of all nodes in **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 **G**, but have a much larger dispersion of s values



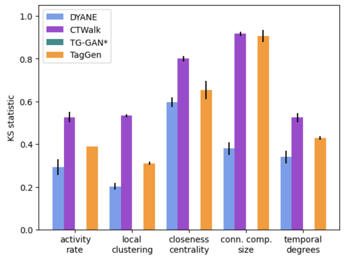
(a) Graph Structure (R-Posts)



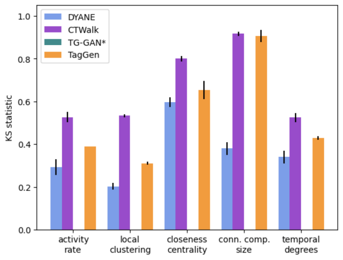
(b) Node Behavior (R-Posts)



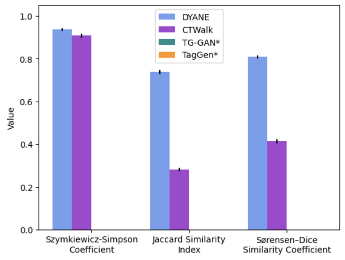
(c) Node Topics (R-Posts)



(d) Graph Structure (Aminer)



(e) Node Behavior (Aminer)



(f) Node Topics (Aminer)

Figure 5: Results for Reddit Cross-Posts (top) and Arnetminer (bottom). (a,b,d,e) Lower value is better, (c,f) higher value is better. (d,e,f) Average statistic and $+1/-1$ Standard Deviation is shown over 10 trials

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.

5.3.3 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 Distance. Central Kernel Alignment (CKA) generalizes squared cosine similarity and Pearson correlation to the multivariate case (e.g., sets of embeddings) [38, 39]. Distance correlation measures both linear and non-linear relationships between two random vectors, even when they have different dimensions [26]. 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 [31].

To evaluate the similarity of the node's topics, we fit a BERTopic model [4] 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, DYANE, 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 Sørensen–Dice 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.

6 RESULTS

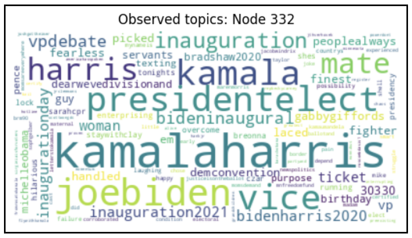
6.1 Evaluation of Generated Graphs

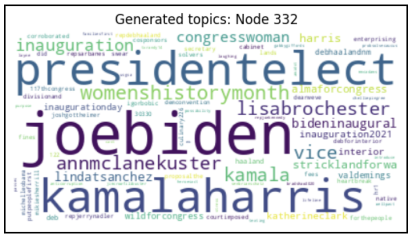
We show the KS statistic (lower is better) for the graph structure (Figures 5a and 5d) and node behavior (Figures 5b and 5e) of the Reddit Cross-Posts and Arnetminer datasets. On average, DYANE improves the KS score of graph structure metrics by 21-31% and the node-aligned metrics by 17-27%. Additional results for Reddit Replies and Congress are reported in [35].

Table 2: Metrics MRR

	Model	Aminer	Congress	R-Posts	R-Replies
Graph Structure	*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
Node Behavior	*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

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 Table 2, 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.


(a) Topics in Observed Embeddings


(b) Topics in Generated Embeddings

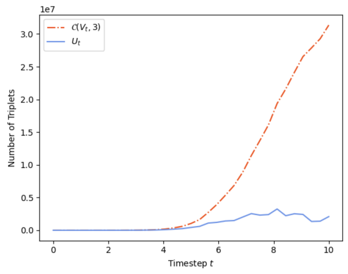

(c) Number of new active triplets

Figure 6: Embedding Topics in Congress Tweets (a,b) and Model Complexity (c)

6.2 Evaluation of Generated Content

We also show the similarity metrics (higher is better) for the node topics (Figures 5c and 5f) of the Reddit Cross-Posts and Arnetminer datasets. Additional results of the node topics for other datasets are reported in [35]. TagGen and TG-GAN do not model any attributes, so their results are omitted. 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 3. 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 3: 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

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 (Figures 6a and 6b). Qualitative results for the other datasets are omitted due to space.

6.3 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, extending 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.

7 SCALABILITY ANALYSIS

The time complexity of DYANE *parameter estimation* is dominated by the number of motifs in the input graph. In the worst case, the number of motifs 3-node combinations. We only consider connected motifs (Fig. 2), which can be found by iterating over the edges at each graph snapshot (i.e., $O(V^2 \cdot T)$). In practice, most real-world graphs are very sparse so the finding the motifs is $O(E \cdot V \cdot T)$. The time complexity of the DYANE *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 $\mathcal{U}_t = \binom{V}{3}$, resulting in $O(V^3)$ 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 (Fig. 6c). 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 node triples 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.

8 CONCLUSION

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 (graph structure metrics by 21-31% and node-aligned metrics by 17-27% on average). 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 on four real-world datasets.

REFERENCES

[1] Jason Baumgartner. [n. d.]. Reddit Statistics. https://pushshift.io/.

[2] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon Kleinberg. 2018. Simplicial Closure and Higher-Order Link Prediction. *Proceedings of the National Academy of Sciences* 115, 48 (Nov. 2018), E11221–E11230. https://doi.org/10.1073/pnas.1800683115 arXiv:1802.06916

[3] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. 2020. SPECTER: Document-level Representation Learning Using Citation-informed Transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 2270–2282. https://doi.org/10.18653/v1/2020.acl-main.207

[4] Maarten Grootendorst. 2022. BERTopic: Neural Topic Modeling with a Class-Based TF-IDF Procedure. https://doi.org/10.48550/arXiv.2203.05794 arXiv:2203.05794 [cs]

[5] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[6] Jack Hessel, Chenhao Tan, and Lillian Lee. 2016. Science, AskScience, and BadScience: On the Coexistence of Highly Related Communities. *Proceedings of the International AAAI Conference on Web and Social Media* 10, 1 (2016), 171–180. https://doi.org/10.1609/icwsm.v10i1.14739

[7] Petter Holme. 2013. Epidemiologically Optimal Static Networks from Temporal Network Data. *PLoS Computational Biology* 9, 7 (2013). https://doi.org/10.1371/journal.pcbi.1003142

[8] Brian Karrer and M. E. J. Newman. 2009. Random Graph Models for Directed Acyclic Networks. *Physical Review E* (2009). https://doi.org/10.1103/PhysRevE.80.046110 arXiv:0907.4346

[9] Myunghwan Kim and Jure Leskovec. 2010. 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)* 6516 LNCS, March (2010), 62–73. https://doi.org/10.1007/978-3-642-18009-5_7 arXiv:1009.3499

[10] Srijan Kumar, William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2018. Community Interaction and Conflict on the Web. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 933–943. https://doi.org/10.1145/3178876.3186141

[11] Guillaume Laurent, Jari Saramäki, and Márton Karsai. 2015. From Calls to Communities: A Model for Time Varying Social Networks. *The European Physical Journal B* 88, 11 (Nov. 2015), 301. https://doi.org/10.1140/epjb/e2015-60481-x arXiv:1506.00393

[12] Lun Li, David Alderson, John C. Doyle, and Walter Willinger. 2005. Towards a Theory of Scale-Free Graphs: Definition, Properties, and Implications. *Internet Mathematics* 2, 4 (2005), 431–523.

[13] Stratis Limnios, Andrew Elliott, Mihai Cucuringu, and Gesine Reinert. 2022. Random Walk Based Conditional Generative Model for Temporal Networks with Attributes. In *NeurIPS 2022 Workshop on Synthetic Data for Empowering ML Research*.

[14] Alex Litel. 2023. *Tweets of Congress*. https://github.com/alexlitel/congresstweets

[15] Paul Liu, Austin R. Benson, and Moses Charikar. 2019. Sampling Methods for Counting Temporal Motifs. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19)*. Association for Computing Machinery, New York, NY, USA, 294–302. https://doi.org/10.1145/3289600.3290988

[16] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. 2002. Network Motifs: Simple Building Blocks of Complex Networks. *Science* 298, 5594 (2002), 824–827. https://doi.org/10.1126/science.298.5594.824

[17] N. Perra, B. Gonçalves, R. Pastor-Satorras, and A. Vespignani. 2012. Activity Driven Modeling of Time Varying Networks. *Scientific Reports* 2 (2012), 1–7. https://doi.org/10.1038/srep00469 arXiv:1203.5351

[18] Joseph J. Pfeiffer, Sebastian Moreno, Timothy La Fond, Jennifer Neville, and Brian Gallagher. 2014. Attributed Graph Models: Modeling Network Structure with Correlated Attributes. *Proceedings of the 23rd International Conference on World Wide Web* (2014), 831–841. https://doi.org/10.1145/2566486.2567993

[19] Sumit Purohit, Lawrence B Holder, and George Chin. 2018. Temporal Graph Generation Based on a Distribution of Temporal Motifs. In *Proceedings of the 14th International Workshop on Mining and Learning with Graphs*. 7.

[20] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks. https://doi.org/10.48550/arXiv.1908.10084 arXiv:arXiv:1908.10084

[21] Pablo Robles, Sebastian Moreno, and Jennifer Neville. 2016. Sampling of Attributed Networks from Hierarchical Generative Models. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16* (2016), 1155–1164. https://doi.org/10.1145/2939672.2939808 arXiv:1602.05561v1

[22] Luis E. C. Rocha and Vincent D. Blondel. 2013. Bursts of Vertex Activation and Epidemics in Evolving Networks. *PLoS Computational Biology* 9, 3 (March 2013), e1002974. https://doi.org/10.1371/journal.pcbi.1002974

[23] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. https://doi.org/10.48550/arXiv.1910.01108 arXiv:arXiv:1910.01108

[24] Amanpreet Singh, Mike D'Arcy, Arman Cohan, Doug Downey, and Sergey Feldman. 2022. SciRepEval: A Multi-Format Benchmark for Scientific Document Representations. https://doi.org/10.48550/arXiv.2211.13308 arXiv:arXiv:2211.13308

[25] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15 Companion)*. Association for Computing Machinery, New York, NY, USA, 243–246. https://doi.org/10.1145/2740908.2742839

[26] Gábor J. Székely, Maria L. Rizzo, and Nail K. Bakirov. 2007. Measuring and Testing Dependence by Correlation of Distances. *The Annals of Statistics* 35, 6 (Dec. 2007), 2769–2794. https://doi.org/10.1214/009053607000000505

[27] Chenhao Tan and Lillian Lee. 2015. All Who Wander: On the Prevalence and Characteristics of Multi-community Engagement. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1056–1066. https://doi.org/10.1145/2736277.2741661

[28] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: Extraction and Mining of Academic Social Networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*. Association for Computing Machinery, New York, NY, USA, 990–998. https://doi.org/10.1145/1401890.1402008

[29] Twitter, Inc. [n. d.]. *Twitter API for Academic Research*. https://developer.twitter.com/en/products/twitter-api/academic-research

[30] Daheng Wang, Tong Zhao, Nitesh V. Chawla, and Meng Jiang. 2021. Dynamic Attributed Graph Prediction with Conditional Normalizing Flows. In *2021 IEEE International Conference on Data Mining (ICDM)*. 1385–1390. https://doi.org/10.1109/ICDM51629.2021.00176

[31] Jiannan Xiang, Yahui Liu, Deng Cai, Huayang Li, Defu Lian, and Lemao Liu. 2021. Assessing Dialogue Systems with Distribution Distances. https://doi.org/10.48550/arXiv.2105.02573 arXiv:arXiv:2105.02573

[32] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling Tabular Data Using Conditional GAN. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc.

[33] Zenan Xu, Zijing Ou, Qinliang Su, Jianxing Yu, Xiaojun Quan, and Zhenkun Lin. 2020. Embedding Dynamic Attributed Networks by Modeling the Evolution Processes. In *The 28th International Conference on Computational Linguistics*. arXiv:2010.14047

[34] Giselle Zeno, Timothy La Fond, and Jennifer Neville. 2021. DYMOND: DYnamic MOtif-NoDes Network Generative Model. In *Proceedings of the Web Conference 2021 (WWW '21)*. ACM, New York, NY, USA, Ljubljana, Slovenia, 12. https://doi.org/10.1145/3442381.3450102

[35] Giselle M. Zeno Torres. 2023. *Dynamic Network Modeling from Temporal Motifs and Attributed Node Activity*. Ph. D. Dissertation. West Lafayette, IN, USA. https://doi.org/10.25394/PGS.23782239.v1

[36] Liming Zhang, Liang Zhao, Shan Qin, Dieter Pfoser, and Chen Ling. 2021. TG-GAN: Continuous-time Temporal Graph Deep Generative Models with Time-Validity Constraints. In *Proceedings of the Web Conference 2021 (WWW '21)*. Association for Computing Machinery, New York, NY, USA, 2104–2116. https://doi.org/10.1145/3442381.3449818

[37] Xin Zhang, Shuai Shao, H. Eugene Stanley, and Shlomo Havlin. 2014. Dynamic Motifs in Socio-Economic Networks. *Europhysics Letters* 108, 5 (2014). https://doi.org/10.1209/0295-5075/108/58001

[38] Vitalii Zhelezniak, Aleksandar Savkov, April Shen, and Nils Hammerla. 2019. 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)*. Association for Computational Linguistics, Minneapolis, Minnesota, 951–962. https://doi.org/10.18653/v1/N19-1100

[39] Vitalii Zhelezniak, April Shen, Daniel Busbridge, Aleksandar Savkov, and Nils Hammerla. 2019. 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)*. Association for Computational Linguistics, Hong Kong, China, 77–87. https://doi.org/10.18653/v1/D19-1008

[40] Dawei Zhou, Lecheng Zheng, Jiawei Han, and Jingrui He. 2020. A Data-Driven Graph Generative Model for Temporal Interaction Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. Association for Computing Machinery, New York, NY, USA, 401–411. https://doi.org/10.1145/3394486.3403082