MSc thesis in Computing Science

MGTCOM: Community Detection in Temporal Multimodal Graphs

Egor Dmitriev (6100120)
egordmitriev2@gmail.com

June 2022

Supervisors

Dr. M. W. Chekol	m.w.chekol@uu.nl	Utrecht University
Dr. S. Wang	s.wang2@uu.nl	Utrecht University

A thesis submitted to the Utrecht University in partial fulfillment of the requirements for the degree of Master of Science in Computing Science

Utrecht University

Abstract

Community detection is the task of discovering groups of nodes sharing similar patterns within a network. With recent advancements in deep learning, methods utilizing graph representation learning and deep clustering have shown great results in community detection. However, these methods often rely on the topology of networks (i) ignoring important features such as network heterogeneity, temporality, multimodality and other possibly relevant features. Besides, (ii) the number of communities is not known a priori and is often left to model selection. In addition, (iii) in multimodal networks all nodes are assumed to be symmetrical in their features; while true for homogeneous networks, most of the real-world networks are heterogeneous where feature availability varies. In this paper, we propose a novel framework (named MGTCOM) that overcomes the above challenges (i)–(iii). MGTCOM allows to discover dynamic communities through multimodal feature learning by leveraging a new sampling technique for unsupervised learning of temporal embeddings. Importantly, MGTCOM is an end-to-end framework optimizing network embeddings, communities, and the number of communities in tandem. In order to assess its performance, we carried out an extensive evaluation on a number of multimodal networks. We found out that our method is competitive against state-of-the-art and performs well under the inductive setting.

Contents

List of Figures

List of Tables

List of Algorithms

1. Introduction

Various systems can be modelled as complex networks such as social [28], citation [45, 60], biological [24] and transaction [56] networks. The task of identifying patterns of nodes with common properties, in such networks, is referred to as community detection. There is an abundant number of community detection methods in literature that approach this problem through modularity optimization [50, 4, 59], clique identification [19, 40], and spectral optimization [69, 30]. With recent advancements in graph representation learning a new type of methods have emerged which utilize context-based learning techniques (e.g., DeepWalk [55], LINE [63] or Node2Vec [27]) to obtain topology-aware node embeddings. These embeddings are either combined with existing clustering methods [7, 81] or are jointly optimized with found clusters [9, 58, 34] to obtain communities.

In the above studies, the dynamic and multimodal characteristics of real-world networks are overlooked. These characteristics can manifest as meta-topological features (node and relation types) [8], temporal features, and contentual features (e.g., text and image attributes). Introduction of multimodality contrasts *homophily* assumed by previous methods as *heterophily* and can play an essential role in detecting communities in multimodal networks, as connected nodes may belong to different communities when multiple feature types are considered [94]. While it is common for causal links to be present between these features, it cannot be assumed without extensive domain knowledge. Various algorithms have been devised to address the issue of temporality and multimodality [26, 43, 41, 18], though as far as we are aware none of the methods are able to address the lossless setting where all the features are incorporated.

Another challenge is information variance present in heterogeneous real-world networks. Different node or relation types may have different feature subsets and/or dimensionality. Let us consider the Twitter dataset (SDS) we use as a case study in Chapter 7. This network consists of users, tweets, hashtags, and various relations in-between. Here tweets have content as textual features and post dates as temporal features, while users only have biography as textual features, and hashtags have neither. Similarly, users form a directed follower relation link, while multiple relations may be present between tweets such as retweet, mention or quote. The meta-topological information describes important semantics of this network, while varying features and topology can be used to identify individual nodes. If these characteristics are ignored by a model, then the quality of the communities discovered can be affected.

With the emergence of web-scale network datasets (often exceeding billions of nodes), recent advancements have pushed for scalability in graph representation learning [29, 88]. To this end, graph convolution methods have allowed for inductive inference on unseen nodes no longer requiring storing full graph Laplacian or node embeddings in memory. Utilizing this representation function learning helps solve scaling issues faced by many auto-encoder-based and shallow embedding community detection methods [47, 52, 71].

In this paper, we propose a novel community detection framework (MGTCOM) that is able to address the aforementioned challenges. MGTCOM discovers dynamic communities through

multimodal feature learning and unsupervised learning with a new sampling technique. In particular, our key contributions include:

(i) A robust method for unsupervised representation learning on multimodal networks

(ii) A new sampling technique for unsupervised learning of temporal embeddings

(iii) An end-to-end framework optimizing network embeddings, communities, and number of communities in tandem

(iv) Extensive evaluation on the quality of various features in multimodal networks

(v) Implementation of various graph sampling algorithms found in the literature[1] (See repository).

We compare MGTCOM with state-of-the-art methods and demonstrate its robustness on inference tasks.

The rest of the thesis is organized as follows. Related works and relevant material is discussed in Chapter 2 and Chapter 3 respectively. Chapter 4 covers the details of our frameworks. In Chapter 5 we present extensive experimental results including comparison with baseline methods. Chapter 6 provides ablation studies to support our design decisions. Finally, in Chapter 7 we provide a deep dive into results produced on the Social Distancing Students dataset as a case study. Source code for the MGTCOM framework can be found on github[2].

[1] https://github.com/EgorDm/tch-geometric
[2] https://github.com/EgorDm/MGTCOM

2. Related Work

In this section, we provide an in-depth overview of related work and highlight important differences with our work by mainly focusing on graph embedding and community detection methods. A comparison of MGTCOM with the state-of-the-art is given in Table 2.1. As can be seen, MGTCOM is able to generate: (i) node, (ii) meta-topology, (iii) content, and (iv) temporal embeddings as well as (v) is able to infer the number of communities (K). By contrast, state-of-the-art methods (such as GraphSAGE and ComE) are able to produce either two or three of the above. A commonality of all the methods is that they are all utilize topological features. Similarly, we focus on representation based community detection methods in contrast to traditional link-based methods. We explain in detail these methods below. Throughout the paper we will use the terms network and graph interchangeably.

2.1. Graph Embedding

With the growing amount of rich graph data, efficient representation is highly demanded for retrieval and analytical purposes. Graph embedding focuses on the representation of nodes into low-dimensional vectors. The graph representation field stems from computational linguistics, which relies heavily on the notion of *distributional semantics*, stating that words occurring in the same context are semantically similar. By creating a parallel between words and nodes the linguistic approaches can be generalized to work in the context of graphs and vice versa.

Approaches such as DeepWalk [55], LINE [63], SDNE [68] and Node2Vec [27] utilize random walks as a means to generate context and adopt the Skip-gram [48] model to directly learn the node embeddings (*shallow embedding* methods). By defining a trade-off between first- and higher-order proximity they provide a way to fine-tune the learned topological representations for the task at hand. Grover and Leskovec [27] observe in their work that depth-first search sampling strategies (higher-order proximity) encourage network communities while breadth-first search (first-order proximity) encourages structural similarity as the local neighborhood is more thoroughly explored.

On the other hand, matrix factorization-based approaches represent first-order proximity using an adjacency or Laplacian matrix. Consequently, they decompose the matrix in order to obtain node-based representation matrix [6]. As this process is quite expensive $O(n^{2.372})$, graph autoencoders (GAE) [65, 37] and graph convolutional networks (GCN) [38] are used instead.

Newer methods aim to solve various issues with current approaches involving scalability [29, 88], incorporation of node/edge features [29, 79], application to heterogeneous [5, 16, 32], attributed [12, 77] and temporal [51, 14, 78] networks (see Table 2.1 for comparison). In line with this, our proposed method (MGTCOM) is able to address all of the above issues.

Table 2.1.: A comparison of MGTCOM with state-of-the-art on embedding (node, meta-topology, content and temporal information) and ability to infer the number of communities k. (top: graph embedding methods; bottom: community detection methods).

	topology	meta-topology	content	temporal	infers K
GraphSAGE [29]	•		•		
SageDy [78]	•		•	•	
CTDNE [51]	•			•	
HGT [32]	•	•	•		
ComE [9]	•				
GEMSEC [58]	•				
GRACE [82]	•		•		
Fani et al. [18]	•		•	•	
CP-GNN [43]	•	•			
MGTCOM	•	•	•	•	•

2.1.1. Scalability

The authors of GraphSAGE (Hamilton et al. [29]) argue that many graph embedding methods are *transductive* and therefore have to be retrained upon the introduction of new or unseen nodes (nodes that are not part of the training data). Additionally, with the emergence of web-scale graphs containing billions of nodes, it is not possible to keep all node embeddings in memory [88]. Hence, in GraphSAGE, they introduce a local k-hop neighborhood sampling strategy and a GCN architecture that is able to infer node representations based on the sampled subgraph. A caveat of this approach is that the GCN architecture requires the presence of node features. While various workarounds exist to use zero or random vectors for missing features, this limits its application for various graph datasets. In MGTCOM we overcome this issue by introducing auxiliary embeddings for nodes with missing features. By keeping auxiliary embeddings of most important nodes at hand, a primary embedding can be computed for each node within the graph.

2.1.2. Heterogeneous networks

The above methods mainly work on homogeneous networks in which all nodes and edges belong to the same types. Often real-world data cannot be efficiently represented using homogeneous networks. Hence, to accurately represent real-world information heterogeneous networks are used. These networks involve *meta-topological* information that characterizes various relationships between different types of nodes/entities [83]. Since most graph embedding methods are designed for homogeneous networks, extending them to incorporate heterogeneous networks is not trivial.

One way to address meta-topological features is by using meta-path constrained random walks to capture semantic and structural relations between different node/entity types [16, 21]. Meta-path describes a sequence of entity and relation types. For example, an "APA" meta-path would define a path between Author-Paper-Author node types. Derivative works introduce attention-based mechanisms to learn the importance of the meta-types [75].

While highly successful, the technique faces certain limitations, mainly that the construction of meta-paths requires extensive domain knowledge and that in highly heterogeneous networks such as knowledge graphs, the amount of meta-paths becomes unmanageable.

Other methods utilize a representation-based approach [5, 76] to explicitly capture meta-topological features by defining relations as translations between different node types. This approach is further utilized in GCN-based methods [32, 91] to apply them in an inductive setting. Furthermore, in Hu et al. [32] the authors improve the neighborhood sampling algorithm by introducing a type-based budget for unbiased sampling.

2.1.3. Temporal networks

Multimodal networks are dynamic and may evolve over time. Temporal networks are a specialization of multimodal networks as they attach a start and end timestamp to each node and edge. Accordingly, graph representation methods should have the ability to capture this evolution.

Temporal graph embedding approaches are mainly split into two categories. Snapshot-based approaches operate by temporally splitting the graph into multiple snapshots or subgraphs and applying (modifying) existing graph embedding methods by temporally smoothing between the snapshots [93, 25, 44, 53]. The second category are the continuous temporal representation approaches which attempt to capture temporal information within the learned embeddings. Generally, these methods look at the temporal progression of individual nodes rather than utilizing predefined snapshots. The techniques vary; CTDNE introduces biased temporal random walks [51]; SageDy introduces a neighborhood sampling technique to filter for temporal neighborhood [78]; BurstGraph captures node representation changes using a RNN [92]; HyTe [14] modifies representation-based techniques to explicitly learn temporal information.

2.2. Community Detection

A community reveals patterns within its members that are different from those in other communities in a network. There is an abundance of work concerning the finding of community structures by relying mainly on topological features [20, 80]. Despite this, the term *community* does not have a universally accepted definition. In their work Peel et al. [54] argue that community detection does not have a one size fits all solution and that definition and quality highly depend on the task at hand. Similarly, they observe that the task of community detection is analogous to finding clusters in document vectors. Nevertheless a few common characteristics distinct community detection from tasks such as topic analysis and clustering including the involvement of topological information and the fact that the number of communities is not known a priori.

Recent community detection methods focus on exploiting feature-rich information found in multimodal networks [61]. The focus has shifted from link-based methods towards deep learning methods which combine graph embedding methods with clustering algorithms (such as k-means or spectral clustering) [65, 39]. Similar methods are employed to learn find communities that take into account global context by utilizing graph autoencoders [74, 7] or graph affiliation networks [84]. More advanced methods utilize multi-objective optimization

by combining topological accuracy and cluster quality metrics during graph representation learning [9, 58, 70, 90]. Other methods focus on modifying [34] or augmenting [35] graph context sampling algorithms to reinforce communities within learned representations.

2.2.1. Multimodal Methods

Many methods rely on *homophily* which refers to the assumption that "individuals" sharing similar patterns are more likely to be connected [46]. With the emergence of multimodal community detection methods *heterophily* becomes equally important as similarity may not always be correlated with topological features [94]. Some argue that consideration of link or content information alone is sufficient for identifying communities [18]. Sparsity, noise, and irrelevant information may mislead traditional community detection or topic modeling algorithms. However, both types of information may be of interest for analysis and may be valuable in overcoming noise in multimodal networks.

In line with this, various methods [42, 62] modify Latent Dirichlet Allocation algorithm to incorporate attribute, topological and meta-topological information. Cao et al. [7] and Yang et al. [82] utilize autoencoders to jointly optimize graph embeddings on content and topological information. Fani et al. [18] use topic models to construct a user interest histogram over a time axis, which in turn is used to learn temporal content-based node representations. These representations are interpolated with topological representations, the similarity along edges is computed, and fed as edge weight to the existing link-based community detection algorithm (namely Louvain [4]).

2.2.2. Heterogeneous Networks

Meta-topological information is a valuable asset for the analysis of found communities. This information can be used in various ways to assist in community detection, for instance, by the representation of small node-specific ego-networks [33] and learning the importance of network relations [62]. Luo et al. [43] propose CP-GNN which combines a heterogeneous graph transformer architecture with k-means clustering to find communities in content-rich heterogeneous graphs. Moreover, they devise a context-path-based k-hop neighborhood sampler to reinforce the discovery of community structures in topological data.

2.3. Clustering

The task of clustering is to find groups of documents in a d-dimensional vector space based on a predefined similarity metric in an unsupervised manner [49]. Many community detection algorithms rely on existing clustering algorithms such as k-means [43, 7, 65, 39, 86] and Gaussian Mixture Models (GMM) [9, 13]. Others employ end-to-end clustering techniques such as deep embedding clustering [82] and clustering loss based parameter optimization [58, 90].

As it is uncommon to know the number of clusters in community detection tasks [20], determining the optimal cluster count is often left to model selection which might become computationally expensive. While various non-parametric clustering algorithms exist such

as DBSCAN [17], OPTICS [2] and BIRCH [89] they are not straightforward to incorporate into end-to-end applications.

Bayesian non-parametric methods such as Dirichlet Process Mixture (DPM) have had great results in clustering and community detection tasks where the number of communities is unknown [95, 96, 66]. As these models can evaluate the likelihood of a set of cluster parameters being drawn from a prior distribution, the task is transformed into a Markov Chain Monte Carlo (MCMC) sampling problem. Since there are prohibitively many possible parameter states, various hierarchical algorithms are proposed to explore the most promising states efficiently [64, 11]. Because these methods can be estimated using Expectation-Maximization (EM) algorithms, the previously introduced embedding methods can be utilized to learn representations and clusters in an end-to-end manner [9, 57].

3. Preliminaries

We present a brief overview of the key concepts and notations used in community detection and graph representation learning. The notations can be found in Table 3.1.

Definition 3.0.1 (Heterogeneous graph). A heterogeneous graph, denoted as $G = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$ consists of a set of nodes $\mathcal{V}$, a set of edges $\mathcal{E}$, and their associated type mapping functions $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and $\psi : \mathcal{E} \rightarrow \mathcal{R}$. ϕ (resp. ψ) maps a node (resp. edge) to its type. $\mathcal{A}$ and $\mathcal{R}$ denote predefined sets of node and edge types, respectively, where $|\mathcal{A}| \geq 1$ and $|\mathcal{R}| \geq 1$. G is a homogeneous graph if $|\mathcal{A}| = 1$ and $|\mathcal{R}| = 1$.

We define a multimodal information network by combining the notion of heterogeneous, continuous-time, and contentual networks.

Definition 3.0.2 (Multimodal graph). A multimodal graph is defined as $G = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \mathcal{X})$ where $\mathcal{T}$ is a set of timestamps t and $\mathcal{X}$ is a set of type specific feature matrices $\mathcal{X}_{\phi(\cdot)}$. Each node $v \in \mathcal{V}$ (resp. edge $e \in \mathcal{E}$) has a time range $\tau(v) = [t_s, t_e]$ (resp. $\tau(e) = [t_s, t_e]$), indicating the time period on which it is considered valid, where $t_s, t_e \in \mathcal{T}$. In addition, each node v has an attribute vector $\mathbf{x} \in \mathcal{X}_{\phi(v)}$.

Definition 3.0.3 (Incompleteness constraints). Real-world multimodal networks can be noisy, incomplete, and may change over time. In order to represent this information we introduce additional indicator functions to denote whether a node has a time interval $\mathbf{1}_{\mathcal{T}} : \mathcal{V} \rightarrow \{0, 1\}$, has a feature vector $\mathbf{1}_{\mathcal{X}} : \mathcal{V} \rightarrow \{0, 1\}$, or is unseen during training $\mathbf{1}_{\mathcal{V}} : \mathcal{V} \rightarrow \{0, 1\}$. We refer to the noisiness, incompleteness, and temporality as incompleteness constraints.

Definition 3.0.4 (Context window). A context window connects nodes based on some predefined criteria. Two nodes are *context neighbors* if they occur in the same context window. In our work, we use two different kinds of context windows. The first is the *topological context window*. It connects two nodes v_i and v_j if there exists a k-hop path $p_k^{\mathcal{E}}$ in graph G through which they are connected. The second is *temporal context window* $p_\omega^{\mathcal{T}}$. It connects v_i and v_j if they occur within a given time window $\omega = [t_s, t_e]$. Going forward we use $P_k^{\mathcal{E}}$ and $P_\omega^{\mathcal{T}}$ to denote a fixed size sample of all possible context windows.

Definition 3.0.5 (Gaussian Mixture Model). Gaussian mixture models (GMM) is a clustering algorithm that assumes the data points are generated by K d-dimensional multivariate Gaussian distributions Eq. (3.1). Here cluster parameters θ_k for $k \in \{1, ..., K\}$ consist of the mean vector $\mu_k \in \mathbb{R}^d$ and the covariance matrix $\mathbf{\Sigma}_k \in \mathbb{R}^{d \times d}$. A K-dimensional binary variable $\mathbf{z}$ is used to denote membership of a particular point n where $\sum_k z_{nk} = 1$. Mixing coefficients π_k specify a marginal distribution over $\mathbf{z}$, such that $\sum_{n \in N} p(z_{nk} = 1) = \pi_k$ where $\pi_k \in [0, 1]$ and $\sum_{k=1}^{K} \pi_k = 1$. Consequently r_k represents conditional probability of $\mathbf{z}$ given a data point $\mathbf{x}$ Eq. (3.2).

Table 3.1.: Notation used in this paper.

Notation	Description
$\mathcal{V}$	The set of nodes in a graph
$\mathcal{A}$	The set of node types
$\mathcal{E}$	The set of edges in a graph
$\mathcal{R}$	The set of edge types/relations
$\mathcal{T}$	The set of timestamps in a graph
$\mathcal{X}_{\phi(\cdot)}$	Feature matrix for node type $\phi(\cdot)$
$\phi(v) \in \mathcal{A}$	Type of node v
$\psi(e) \in \mathcal{R}$	Type of edge e
G_v	k-hop neighborhood subgraph for node v
$d \in \mathbb{N}$	Size of node embedding vector
$E_v \in \mathbb{R}^d$	Auxiliary embedding vector for node v
$Z_v \in \mathbb{R}^d$	Primary embedding vector for node v
$Z_v^{\mathcal{E}}, Z_v^{\mathcal{T}} \in \mathbb{R}^d$	Task specific embedding vectors for node v
$K \in \mathbb{N}$	Number of communities
$\mathcal{N}(\mu_k, \mathbf{\Sigma}_k), \theta_k$	Parameters for the k'th cluster/community
$\mu_k \in \mathbb{R}^d$	k'th cluster mean vector
$\mathbf{\Sigma}_k \in \mathbb{R}^{d \times d}$	k'th cluster covariance vector
$\mathbf{z} \in \{0, .., K\}^{\lvert\mathcal{V}\rvert}$	Community membership assignment vector
ω	Interval window for temporal context sampling
$P_l \in \mathcal{V}^l$	Sampled context window of length l

$$p(\mathbf{x}; \theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{\mu}_k, \mathbf{\Sigma}_k) \tag{3.1}$$

$$r_k = p(z_k = 1|\mathbf{x}) = \frac{\pi_k \mathcal{N}\left(\mathbf{x} \mid \boldsymbol{\mu}_k, \mathbf{\Sigma}_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\mathbf{x} \mid \boldsymbol{\mu}_j, \mathbf{\Sigma}_j\right)} \tag{3.2}$$

Assuming that the points $\mathbf{X} \in \mathbb{R}^{N \times d}$ are drawn independently from the distribution, the log-likelihood function is given by Eq. (3.3). The value of μ_k, $\mathbf{\Sigma}_k$, π_k can be found by setting derivative of $\ln p(\mathbf{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \mathbf{\Sigma})$ to zero with respect to their values yielding closed form equations Eqs. (3.4) to (3.6). N_k represents the number of points assigned to cluster k. While model parameters can be computed given values of $\mathbf{X}$ and $\mathbf{r}$ are known, it is important to note that $\mathbf{r}$ is dependent on the model parameters Eq. (3.2). Expectation-Maximization (EM) is an elegant iterative technique devised to find such clustering parameters. Given an initial cluster assignment that may be obtained using k-means or a similar technique, *expectation* (E) and *maximization* (M) steps are applied alternatively Fig. 3.1. The E step uses current cluster parameters to evaluate posterior probabilities Eq. (3.2), while the M step uses these probabilities to compute new model parameters using Eqs. (3.4) to (3.6). The model is deemed converged once the change in parameters or assignment falls below a certain threshold.

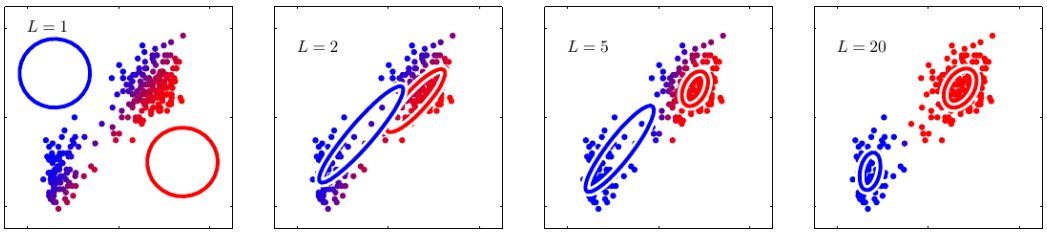


Figure 3.1.: Visualization of Expectation-Maximization algorithm [3]. (a) Clusters are randomly initialized. (b) Cluster centers are updated according to the initial assignment (M step). (c), (d) Expectation and Maximization steps are repeated until convergence.

$$\ln p(\mathbf{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \right\} \tag{3.3}$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n \tag{3.4}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \left(\mathbf{x}_n - \boldsymbol{\mu}_k\right) \left(\mathbf{x}_n - \boldsymbol{\mu}_k\right)^{\mathrm{T}} \tag{3.5}$$

$$\pi_k = \frac{N_k}{N} \tag{3.6}$$

Gaussian Mixture Models suffer from severe overfitting problems in the form of single-point collapse and the fact that the number of clusters needs to be known a priori. Bayesian parametric (BP) and non-parametric (BNP) mixture models aim to solve these issues by introducing prior distributions governing the model parameters $(\pi, \mu, \boldsymbol{\Sigma})$ and using maximum a priori (MAP) instead of maximum likelihood estimation.

Definition 3.0.6 (Dirichlet Process Mixture Model). Dirichlet process mixture model (DPMM) is a part of BNP mixture models which finds a clustering solution when K is unknown. DPMM extends GMM as it is an infinite mixture model Eq. (3.7) with the Dirichlet process as prior distribution on the number of clusters Eq. (3.8). Here hyperparameter α_0 is the concentration parameter referring to the prior amount of observations associated with each component, and Γ refers to the mathematical function "gamma" which in its essence is a generalization of the factorial function that can deal with any real number > 0. The cluster parameters θ are assumed to be i.i.d. and are drawn from a prior distribution. In our case, Normal Wishart Distribution (NW) Eq. (3.9) where hyperparameters κ and ν represent the concentration parameter and degrees of freedom of the Wishart distribution respectively. The data is parameterized by the data mean μ and $\boldsymbol{\Lambda}$ which is the precision matrix (inverse of the covariance matrix $\boldsymbol{\Sigma}$).

$$p(\boldsymbol{x}) = \sum_{i=1}^{\infty} \pi_i \mathcal{N}\left(\boldsymbol{x}, \boldsymbol{\mu}_i, \boldsymbol{\Lambda}^{-1}\right) \tag{3.7}$$

$$p(\boldsymbol{\pi}) = \mathrm{Dir}\left(\boldsymbol{\pi}; \boldsymbol{\alpha}_0\right) = \frac{\Gamma(\alpha_0)}{\prod_{i=1}^{K} \Gamma(\alpha_0)} \prod_{i=1}^{K} \pi_i^{\alpha_0 - 1} \tag{3.8}$$

$$\begin{aligned} p(\boldsymbol{\mu}, \boldsymbol{\Lambda}) &= \mathrm{NW}(\boldsymbol{\mu}, \boldsymbol{\Lambda}; \kappa_0, \mu_0, \nu_0, \boldsymbol{W}_0) \\ &= \prod_{i=1}^{K} \underbrace{\mathcal{N}\left(\boldsymbol{\mu}_i \mid \mu_0, \left(\kappa_0 \boldsymbol{\Lambda}_i\right)^{-1}\right)}_{p(\mu_i \mid \boldsymbol{\Lambda}, \kappa_0, \mu_0)} \underbrace{\mathcal{W}\left(\boldsymbol{\Lambda}_i \mid \mathbf{W}_0, \nu_0\right)}_{p(\boldsymbol{\Lambda} \mid \mathbf{W}_0, \nu_0)} \end{aligned} \tag{3.9}$$

The prior parameters α_0, κ_0, and ν_0 are set to a predetermined values, whereas prior parameters μ_0 and W_0 are calculated on a sample of the full dataset using Eqs. (3.14) and (3.15). Here $\alpha_0, \nu_0, \kappa_0 \in \mathbb{R}^+$ and $\nu_0 > d + 1$.

$$\bar{\mathbf{x}}_i = \frac{1}{N_i} \sum_{n=1}^{N} r_{ni} \mathbf{x}_n \tag{3.10}$$

$$\mathbf{S}_i = \frac{1}{N_i} \sum_{n=1}^{N} r_{ni} \left(\mathbf{x}_n - \bar{\mathbf{x}}_i\right) \left(\mathbf{x}_n - \bar{\mathbf{x}}_i\right)^{\mathrm{T}} \tag{3.11}$$

EM can similarly be used to approximate solutions for DPM models. During the E step Eq. (3.2) is once again used to estimate the assignments. While during the M step Eqs. (3.10) and (3.11) equations analogous to Eqs. (3.4) and (3.5) are used to estimate the data covariance and data mean. Subsequently the following closed form equations are used to compute posterior parameters for the given prior Eqs. (3.12) to (3.16). Given the posterior parameters, the new cluster parameters are inferred using Eqs. (3.14) and (3.17). When α_0, κ_0, and ν_0 are much smaller than N, the posterior distribution will be influenced primarily by the data rather than the prior. We use λ to denote computed posterior parameters.

$$\pi_i = \frac{N_i}{\sum_{j=1}^{K} N_i + \alpha_0} \tag{3.12}$$

$$\kappa_i = \kappa_0 + N_i \tag{3.13}$$

$$\mu_i = \frac{1}{\kappa_i} \left(\kappa_0 \mu_0 + N_i \bar{\mathbf{x}}_i\right) \tag{3.14}$$

$$\mathbf{W}_i^{-1} = \mathbf{W}_0^{-1} + N_i \mathbf{S}_i + \frac{\kappa_0 N_i}{\kappa_0 + N_i} \left(\bar{\mathbf{x}}_i - \mu_0\right) \left(\bar{\mathbf{x}}_i - \mu_0\right)^{\mathrm{T}} \tag{3.15}$$

$$\nu_i = \nu_0 + N_i \tag{3.16}$$

$$\boldsymbol{\Sigma}_i = \frac{\nu \mathbf{W}_i^{-1}}{\nu - d + 1} \tag{3.17}$$

The described implementation solves overfitting and cluster count, though it is an incomplete one since in practice the cluster count has a defined upperbound K (computationally

and storage-wise). While the clusters can get pushed out of existence, no additional clusters can be created. To solve this issue many variants of DPMM have been proposed utilizing the Chinese Restaurant process, Collapsed Weight sampling, etc. We focus on the split/merge sampling algorithm introduced by Chang and Fisher III (DPMMSC) [11]. For an exhaustive discussion, we refer interested readers to [3, 10].

DPMMSC exploits an alternate perspective in which DPMM is defined as a Monte Carlo Markov Chain if all the chosen priors are conjugate (i.e. prior distribution is in the same form as the posterior distribution). The stationary distribution is defined by the probability of cluster parameters given the data observations Eq. (3.19). Intuitively in this approach sampling methods are used to approximate the E step of EM by sampling from the current estimate of posterior distribution $p(\mathbf{z}|\mathbf{X}, \theta^{\text{old}})$ (proposal distribution), where during M step the new state θ is found. A similar methodology is employed to transition between different values of K by proposing θ directly. As proposal space is unmanageably large, a greedy strategy is employed to propose the most promising states.

$$H_s = \frac{\alpha\Gamma(N_{i_1})p(X_{i_1};\lambda_{i_1})\Gamma(N_{i_2})p(X_{i_2};\lambda_{i_2})}{\Gamma(N_i)p(X_i;\lambda_i)} \tag{3.18}$$

$$p(\mu_i, \boldsymbol{\Sigma}_i|\mathbf{X}_i) = \text{NW}(\boldsymbol{\mu}, \boldsymbol{\Lambda}; \kappa_0, \mu_0, \nu_0, \boldsymbol{W}_0) \tag{3.19}$$

$$\begin{aligned} p(\mathbf{X};\lambda) &= \int p\left(\mathbf{X} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right) p\left(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i; \lambda\right) d\left(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right) \\ &= \frac{1}{\pi^{\frac{Nd}{2}}} \frac{\Gamma_d\left(\nu_0/2\right)}{\Gamma_d(\nu_i/2)} \frac{|\nu_0\boldsymbol{\Lambda}_0|^{\nu_0/2}}{|\nu_i\boldsymbol{\Lambda}_i|^{\nu_i/2}} \left(\frac{\kappa_i}{\kappa_0}\right)^{d/2} \end{aligned} \tag{3.20}$$

For each supercluster i, two auxiliary subclusters are defined with parameters θ_{i_1} and θ_{i_2} forming a two-component GMM. Once subclusters are in a converged state, the split proposals are made given the supercluster and its two subcomponents. Similarly, supercluster merges are proposed by picking k nearest candidates for each supercluster.

The proposed candidates are either accepted or rejected by the Metropolis-Hastings (MH) algorithm moving the model to the next state. As the split acceptance ratio H_s is defined by the probability of data being sampled from the split state in contrast to the current state Eq. (3.18). Analogously, the merge ratio is its inverse, namely $\frac{1}{H_s}$. Eqs. (3.3), (3.9), (3.12) and (3.19) are used to derive the marginal probability of data being generated by parameter set λ given prior parameters Eq. (3.20) (note that π refers to the mathematical constant, and Γ_d refers to mathematical function digamma). The proposals are considered once the supercluster model has converged. If no proposal is accepted, then DPMMSC is considered as converged.

Definition 3.0.7 (Graph Convolutional Neural Networks). Graph Convolutional Neural Networks (GCN) [38, 29, 36] generate node embeddings given a spatial filter which is applied as a convolution given each node's graph neighborhood. The convolution operation enables GCNs to propagate structural information of graphs throughout the network (referred to as message-passing). By layering this process, the receptive field of each node expands to its k-hop neighborhood.

Suppose H_t^l is the representation of node t at layer l, a forward step of the message-passing procedure is defined as Eq. (3.21) where $N(t)$ is t's neighboring node set and $\mathcal{E}(s,t)$ is the set

of edges between nodes t and its neighbor s. Here the operator **Message**$(\cdot)$ extracts useful information from the neighboring source nodes s, while the **Aggregate**$(\cdot)$ operator gathers the neighborhood information via some aggregation operator such as *mean*, *sum* or *max* to get contextualized representation of t.

$$H_t^{(l)} = \underset{\forall s \in N(t), \forall e \in \mathcal{E}(s,t)}{\textbf{Aggregate}} \left[\textbf{Message} \left(H_s^{(l-1)}, e, H_t^{(l-1)} \right) \right] \tag{3.21}$$

The time complexity to run a forward step over the entire training set is $O(|\mathcal{V}| \cdot deg \cdot d^2)$ where deg refers to the average node degree. While $deg \ll |\mathcal{V}|$ is true for most graphs, a vital optimization step is to sample a fixed size N_v ensuring that deg is bounded by a constant.

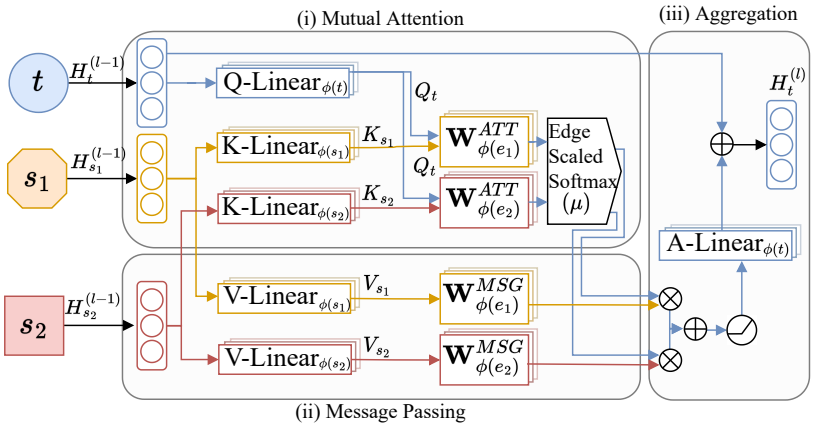


Figure 3.2.: Visualization of a Heterogeneous Graph Transformer Layer. Given target node t and neighboring source nodes s_1 and s_2 by edges e_1 and e_2, mutual attention and messages are computed. Within aggregation step the messages are attended and combined with previous target node embedding $H_t^{(l-1)}$ resulting in the new embedding vector $H_t^{(l)}$

Definition 3.0.8 (Heterogeneous Graph Transformer). Classical GCNs focus mainly on homogeneous graphs. A fair amount of works describe ways to adapt existing algorithms by introducing a **Message** step parameterized by meta-topological types. Based on the observation that the value of different connections varies given a node type, attention-based mechanisms are introduced into the aggregation process. Inspired by success in NLP Heterogeneous Graph Transformer (HGT) [32] adopts the transformer architecture [67] by calculating mutual attention based on representation and meta-types of source, target and relation information.

$$H_t^l = \underset{\forall s \in N(t), \forall e \in \mathcal{E}(s,t)}{\textbf{Aggregate}} \left[\textbf{Attention}\,(s,e,t) \cdot \textbf{Message}\,(e,t) \right] \tag{3.22}$$

HGT consists mainly of three components, (i) mutual attention possession performance of each source node, (ii) message passing extracts information from source nodes, and (iii) target-specific aggregation which combines the neighborhood messages. A general form for a forward pass is defined as Eq. (3.22) and is visualized in Fig. 3.2.

The attention vector is calculated by mapping source node s into Key K and target node t into a Query Q vectors Eqs. (3.25) and (3.26). A single head attention vector is calculated as inner product similarity vector between Key K and Query Q vectors (See Eq. (3.24)) given a relation specific interaction matrix, where prior tensor $\mu \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{R}|\times|\mathcal{A}|}$ denotes significance of each relation triplet. $K(s)$ and $Q(t)$ are computed as projections of source s and target t nodes respectively Eqs. (3.25) and (3.26). The final attention vector results from a concatenation of h attention heads per source node Eq. (3.23).

$$\textbf{Attention}_{HGT}(s,e,t) = \underset{\forall s\in N(t)}{\text{Softmax}}\left(\underset{i\in[1,h]}{\Big\|} \textit{ATT-Head}^i(s,e,t)\right) \tag{3.23}$$

$$\textit{ATT-Head}^i(s,e,t) = \left(K^i(s)W^{ATT}_{\psi(e)}Q^i(t)^T\right)\cdot\frac{\mu_{\langle\phi(s),\psi(e),\phi(t)\rangle}}{\sqrt{d}} \tag{3.24}$$

$$K^i(s) = \text{K-Linear}^i_{\phi(s)}\left(H^{(l-1)}_s\right) \tag{3.25}$$

$$Q^i(t) = \text{Q-Linear}^i_{\phi(t)}\left(H^{(l-1)}_t\right) \tag{3.26}$$

Similarly, the multi-head message is computed by applying type-dependent projection (V-Linear) to the input source node representation and transforming it using the edge type matrix $W^{MSG}_{\psi(e)} \in \mathbb{R}^{\frac{d}{h}\times\frac{d}{h}}$ to incorporate the relation dependency into the result Eq. (3.28). In both operations, edge interaction matrices and the head-specific type projection matrices are shared to minimize the number of used parameters.

$$\textbf{Message}_{HGT}(s,e,t) = \underset{i\in[1\ldots h]}{\Big\|} \textit{MSG-Head}^i(s,e,t) \tag{3.27}$$

$$\textit{MSG-Head}^i(s,e,t) = \text{V-Linear}^{\,i}_{\phi(s)}\left(H^{(l-1)}_s\right)W^{MSG}_{\psi(e)} \tag{3.28}$$

Finally, during the aggregation step, the calculated attention is applied to neighborhood messages and summed into the neighborhood representation vector Eq. (3.29). The final node representation vector $H^{(l)}_t$ results from the summation of the projected neighborhood vector into the target node space and the previous representation of the target vector Eq. (3.30).

$$\widetilde{H}^{(l)}_t = \underset{\forall s\in N(t)}{\bigoplus}\left(\textbf{Attention}_{HGT}(s,e,t)\cdot\textbf{Message}_{HGT}(s,e,t)\right) \tag{3.29}$$

$$H^{(l)}_t = \text{A-Linear}_{\phi(t)}\left[\sigma\left(\widetilde{H}^{(l)}_t\right)\right] + H^{(l-1)}_t \tag{3.30}$$

See Fig. 3.2 for a visualization of a forward pass of single layer HGT.

Problem formulation. Given a multimodal graph G, our goal is to learn a node embedding function $\zeta : G_v \rightarrow \mathbb{R}^d$ which given a k-hop neighborhood subgraph G_v of node v produces a d-dimensional embedding vector Z_v. The objective is to minimize the distance between embedding Z_v to other node embeddings, given that they are topological and/or temporal context neighbors of node v. Taking into account incompleteness constraints (Definition 3.0.3), ζ should work under any valuation of $(\mathbf{1}_{\mathcal{X}(v)}, \mathbf{1}_{\mathcal{T}(v)}, \mathbf{1}_{\mathcal{V}(v)})$. We also aim to find community parameters $\grave{} = \{\mathcal{N}(\mu_1, \Sigma_1), ..., \mathcal{N}(\mu_K, \Sigma_K)\}$ and node-to-community assignment $\mathbf{z} \in \{0, .., K\}^{|\mathcal{V}|}$ such that their members have a low inter-proximity in contrast to other nodes. Finally, the found community count K should approximate the ground truth number of communities.

4. The Proposed Approach

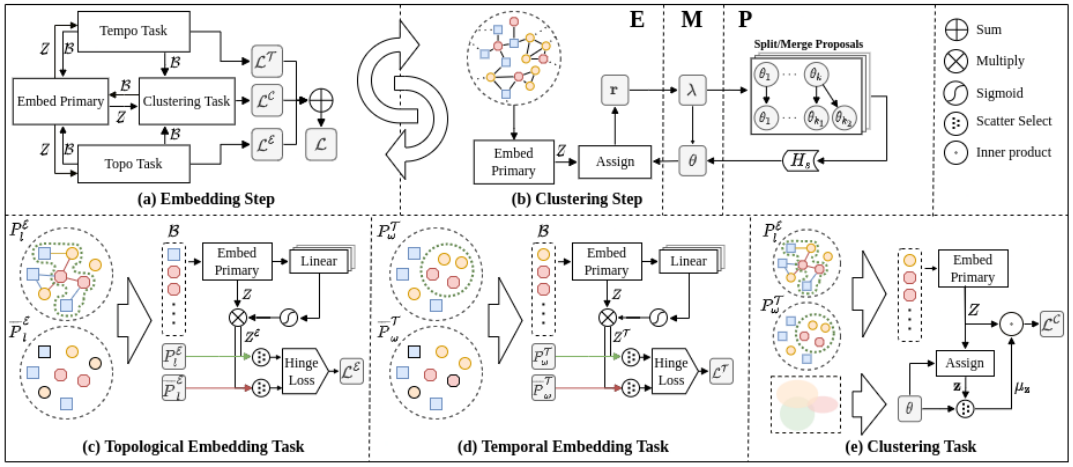


Figure 4.1.: Overview of the MGTCOM framework. (a) In the embedding step primary embeddings are used in auxiliary tasks to construct the multi-objective loss. (b) Clustering step updates clustering by alternating between Expectation (**E**), Maximization (**M**), and Proposal (**P**) steps. (c) In the topological (topo) embedding task, random walk sampling and feature-wise attention minimize inter-node proximity. (d) In the temporal (tempo) embedding task, ballroom walk sampling and feature-wise attention minimize proximity between temporally related nodes. (e) Clustering task adds community awareness to the embeddings by minimizing proximity between nodes within the same cluster.

We present our framework for Community Detection in Temporal Multimodal Graphs (MGTCOM) that learns multimodal representation vectors for graph nodes and detects communities in tandem. We achieve this by leveraging heterogeneous graph transformers [32] to learn a primary node embedding function ζ. In order to handle the incompleteness constraints, we introduce an auxiliary embedding vector E for known (or seen) nodes with missing features. Next, we learn task-specific node representation for topological and temporal information by combining primary embeddings with task-specific transformation/attention and context sampling. As we utilize random walks for topological context sampling, we introduce its analogue as an unbiased temporal window sampling algorithm for temporal context collection. Finally, we adopt DPMM for community detection and close the loop by introducing cluster-based loss to ensure the graph embeddings are *community-aware*. MGTCOM consists of three major components (as can be seen in Fig. 4.1): primary embedding module, (a) task-specific learning, and (b) community detection/clustering module. MGTCOM also has a graph sampling component. In the following, we describe the components in detail.

Algorithm 4.1: Batchwise primary node embedding

1 **Procedure** `EmbedPrimary()`
  Input: multimodal graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \mathcal{X})$, mini-batch $\mathcal{B} \subseteq \mathcal{V}$, auxiliary node embedding $\mathbf{E} \in \mathbb{R}^{N_{\bar{\mathcal{X}}} \times d}$ where $N_{\bar{\mathcal{X}}} = |\{v | v \in \mathcal{V}, \mathbf{1}_{\mathcal{X}}(v) = 0\}|$, number of convolutional layers L
  Output: The primary embedding $Z_{\mathcal{B}}$ for nodes in batch $\mathcal{B}$
2 $G_{\mathcal{B}}(\mathcal{V}_{\mathcal{B}}, \mathcal{E}_{\mathcal{B}}, \mathcal{A}_{\mathcal{B}}, \mathcal{R}_{\mathcal{B}}, \mathcal{X}_{\mathcal{B}}) \leftarrow \text{HeteroSample}(G, \mathcal{B}, L)$;
3 **for** $s \in \mathcal{V}_{\mathcal{B}}$ **do**
4 $$H_s^{(0)} = \begin{cases} \text{Linear}(\mathbf{x_s}) & \mathbf{1}_{\mathcal{X}}(s) = 1 \\ \text{Dropout}(\mathbf{E}_s) & \mathbf{1}_{\mathcal{V}}(s) = 1 \\ 0_d & \text{otherwise} \end{cases};$$
5 **for** $l = 1$ **to** L **do**
6 $H^{(l)} = \text{GeLU}(\text{HeteroConv}(G_{\mathcal{B}}, H^{(l-1)}))$;
7 $Z_{\mathcal{B}} = \{H_t^{(L)} | t \in \mathcal{B}\}$;
8 **return** $Z_{\mathcal{B}}$

4.1. Primary embedding module

The central component of our framework is responsible for inferring the primary representation vector Z_v given a node $v \in \mathcal{V}$ in a graph G. Motivated by the success of inductive GCN-based methods [29, 88, 32], we build our architecture by combining L graph convolution layers (*HeteroConv* or HGCN) and a graph subsampler (*HeteroSample*).

Specifically, we use the budget-based subgraph sampling algorithm and the heterogeneous graph transformer (HGT) proposed by Hu et al. [33]. HGT captures topological, meta-topological, and content-based aspects by combining off-the-shelf graph convolution with node type-specific projection and edge type-based attention.

Algorithm 4.1 provides a full overview of the primary embedding algorithm. The basic idea is to infer node representation from its k-hop heterogeneous neighborhood subgraph G_v while handling edge cases introduced by the incompleteness constraints (Definition 3.0.3) in order to handle web-scale multimodal graphs. The inference starts by sampling a subgraph $G_{\mathcal{B}}$ given a batch of central nodes using the *budget sampling* algorithm on line 2. The *budget sampling* algorithm works by restricting sampled subgraph at each layer given a per node type limit. For our use case, we define this limit as multiple $|\mathcal{B}|$ to avoid re-tuning its value for each dataset.

Once the graph is sampled we split the task of initial feature inference into three cases to handle the incompleteness constraints. (i) If a feature vector is present, then it is simply projected into the representation space. (ii) If the node is in the training set while no feature vector is present, then its representation is drawn from the *auxiliary embedding* matrix $\mathbf{E}$. To avoid overreliance on the embeddings in preference for feature vectors we apply dropout on the resulting representation. (iii) Finally, if an unseen node without a feature vector is encountered, the zero vector (denoted as 0_d) is used, indicating that its feature vector has zero weight during the aggregation step of the graph convolution. Note that for large datasets, it may not be feasible to keep a full auxiliary embedding matrix in memory. In Section 6.3 we explore a setting where auxiliary embeddings are limited to a subset of important nodes.

On line 6, given the subgraph $G_{\mathcal{B}}$ and the initial representation vector $H^{(0)}$, L layers of HGT graph convolutions are applied. Each layer uses the representation vector of the previous layer and feeds its output through a GeLU [31] activation function (See Section 6.5 for performance comparison). Finally, the output vectors at the L^{th} layer are used as primary representation vectors and output for each query node in the batch on line 7.

4.2. Multi-task representation learning

During task-specific learning we focus on two main tasks capturing the intricacies of multimodal networks. The topological task identified by $\mathcal{E}$ focuses on minimizing the representation distance between nodes that are proximate within the network. Analogously, the temporal task $\mathcal{T}$ focuses on minimizing the distance between nodes that co-occur at the same timeframes. While fundamentally different since the tasks are trained in parallel, they benefit from weight sharing and from node sharing during primary embedding as the subgraph batches are centered around the same nodes.

4.2.1. Task-based attention

An important observation is that while temporal and topological communities are both important during analysis, they are not always correlated. In fact, in most of the benchmarking datasets such as Cora and DBLP temporal features and graph structure show low correlation. While it is very rare that contentual features are completely independent of topology and temporality, we describe a general implementation that can be applied to such a case.

Given the above observation, we admit that it may not be possible to train a model that excels at both tasks. To work around this issue while still capturing both tasks in a single embedding vector we introduce *task-based attention*. The basic idea is that while primary embedding extracts suitable features from the multimodal network, task-specific attention selects the most relevant of these features for the task at hand. Inspired by transformers [67] we define multi-head attention to capture various feature patterns Eq. (4.2). The task-based transformation function is defined as Eq. (4.1) where the primary representation vector is attended to using a simple matrix multiplication operation. We specialize this function for topological task as $f^{\mathcal{E}}(\mathbf{Z})$ producing $\mathbf{Z}^{\mathcal{E}}$ and temporal task $f^{\mathcal{T}}(\mathbf{Z})$ producing $\mathbf{Z}^{\mathcal{T}}$.

$$f^{task}(\mathbf{Z}) = \mathbf{Z} \cdot ATT^{task}(\mathbf{Z}) \tag{4.1}$$

$$ATT^{task}(\mathbf{Z}) = \Big\|_{i \in [1..h]} \sigma\left[\text{Linear}_i^{task}(\mathbf{Z})\right] \tag{4.2}$$

4.2.2. Objective function

In order to learn model parameters in an unsupervised way, we define contrastive loss. Task-specific positive context sample P and a negative context sample $\bar{P}$, both sharing a central query node q are used to construct positive (q, p) and negative (q, n) node pairs respectively. We define a max-margin-based loss function (Eq. (4.3)) which aims to maximize the inner product similarity between the query and positive examples. On the other hand, the inner

product of query and negative samples is minimized to be smaller than that of the positive samples by some predefined value of Δ (see Section 6.5 hyperparameter experiments). In our tests, we found that averaging similarity over negative samples within the max loop helps to smoothen out the noise caused by context sampling Eq. (4.4).

$$\text{MM-Loss}(\mathbf{Z}, P, \bar{P}, q) = \max_{n \in \bar{P}} \left\{ 0, \mathbf{Z}_q \mathbf{Z}_n - \widetilde{\mathbf{Z}_q \mathbf{Z}_p} + \Delta \right\} \tag{4.3}$$

$$\widetilde{\mathbf{Z}_q \mathbf{Z}_p} = \frac{1}{|P|} \sum_{p \in P} (\mathbf{Z}_q \mathbf{Z}_p) \tag{4.4}$$

4.2.3. Temporal context sampling

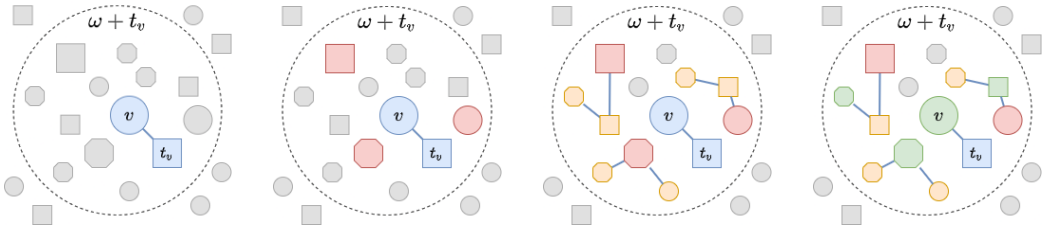


Figure 4.2.: Visual overview of Ballroom Walk temporal sampling algorithm. (a) The sampling timestamp t_v for query node v is inferred given the nearest neighbor if the node is static (blue). The relative time window is determined as $\omega + t_v$. (b) The root context nodes are sampled from the relative time window (red). (c) Context is extended with temporal random walks from the root nodes (yellow). (d) The context path is sampled from the collected context (green).

Temporal features are often not correlated with network topology. We propose a separate context sampling function that, given a query node and an interval window ω, returns other nodes occurring within the same time window. The interval window ω is determined by using the dataset statistics as a fraction of the complete time range $\mathcal{T}$. By picking a small enough interval window, a fine-grained continuous-time representation vector can be learned. This is because additional granularity is achieved by centering the sample around the query node.

Edge cases arising from the incompleteness constraints need to be handled where the nodes are missing timestamps $\mathbf{1}_{\mathcal{T}} = 1$. While the usual semantic approach is to consider these nodes omnipresent (static), the naive window sampling methods quickly get congested with static to static context pairs. Our aim is to alleviate this issue using biased sampling in favor of non-static pairs.

We start by introducing the temporal random walk procedure shown in Algorithm 4.2 which enforces standard random walks over the network to stay within a predetermined temporal window ω^*. Here random walk of size l is constructed by picking a randomly connected node to the current head node (line 5) within a time window. If no such node is present, then

Algorithm 4.2: Temporal Random Walk

```
1  Procedure TemporalRW()
       Input: center node v, temporal window ω*
       Output: Temporal random walk P_l
2      Initialize P_l = [];
3      (u, t_u) = (v, ∅);
4      for i = 1 to l do
5          N(u) = {w | w ∈ V, (u, w) ∈ E, τ(w) ∩ ω* ≠ ∅};
6          if N(u) = ∅ then /* Restart on dead end                    */
7              (u, t_u) = (v, t_v);
8              go to 5;
9          w ~ N(u);
10         t_u = max{t_u, min τ(w)};
11         u = w;
12         P_l.append((u, u_t));
13     return P_l
```

the random walk is restarted from any already picked node line 7. The walk is extended with a new head node until it reaches the desired length.

Utilizing the idea of temporal random walks we propose our own sampling method "Ballroom Walk" whose outline is shown in Algorithm 4.3. It starts by inferring the sampling timestamp t_v by picking a random timestamp the query node v occurs in. If the node is static, the timestamp of its nearest neighbor reachable through temporal random walk is selected (line 3). To reliably sample the temporal neighborhood, n root nodes (w) are picked occurring in the time-window relative to the sampling timestamp on line 4. Temporal context C is constructed by collecting temporal random walks starting from root nodes w given a relative time window $\omega + t_v$ on line 7. Finally, l long context paths are created as random subsets of C. Note that because a sampled context is valid for all member nodes, random walk-like throughput optimization can be used by setting a larger window length than context size [55].

Due to timestamp inference, the first- and second-order proximity static to static pairs are ignored. By only passively sampling omnipresent nodes we mitigate the over-saturation issue while still being fair. Most importantly the neighborhood of central nodes is being sampled independently of their topology. By sampling within a temporal window, we avoid not relying on the correlation of temporality with topology.

4.2.4. Graph sampling

The objective of task-specific learning is mainly defined by the context sampling method. As our method allows for inference of primary and task-specific representations for unseen nodes, we assume that topological, meta-topological and contentual features contain enough information / are correlated with the objective of the tasks. To gather the topological context $P^{\mathcal{E}}$, Node2Vec biased random-walk algorithm is utilized [27]. By choosing a low value for its control parameter q we discourage structural/topological equivalence representation in favor of larger neighborhood exploration (depth-first strategy) which is useful for community representation. Similarly, we use ballroom sampling to collect temporal context $P^{\mathcal{T}}$ of

Algorithm 4.3: Ballroom walk sampling

Algorithm `BallroomWalk()`
  Input: multimodal graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \mathcal{X})$, relative temporal window ω, walks per node n, walk length l, center node v,
  Output: Temporal n random walks P_l
  Initialize C;
  $t_v = \begin{cases} t_v \sim \tau(v) & \mathbf{1}_{\mathcal{T}}(v) = 1 \\ \texttt{TemporalRW}(v, (-\infty, \infty)).first() & \text{otherwise} \end{cases}$;
  $N(v) = \{w | w \in \mathcal{V}, \tau(w) \cap \omega + t_v \neq \emptyset\}$;
  for $i = 1$ **to** n **do**
    $w \sim \mathcal{N}$;
    $C = C \cup \texttt{TemporalRW}(w, \omega + v_t)$;
  `RandomPermute`(C);
  for $i = 1$ **to** n **do**
    $P_l = \{C_j | i \cdot l \leq j < i \cdot l + l\}$;
    return P_l;

size l as introduced in the previous section. The negative nodes are collected by sampling random nodes from the graph. In our framework, the query nodes and negative contexts are shared across both tasks.

4.3. Community detection

For community detection, we adopt the DPMM split/merge algorithm proposed by Chang and Fisher III [11] as discussed in Definition 3.0.6. In our implementation, we use Normal Wishart (NW) as a conjugate prior and use variational lower bound in our convergence criteria. Specifically, we monitor the log sum of the variational lower bound Eq. (4.5) for the supercluster and subcluster models. The variational lower bound is computed as the product of variational distribution $q(\mathbf{z})$, the normalizing constant of the Dirichlet distribution $B(\alpha_0)$, and the normalizing constant of the Normal Wishart distribution $C(W, \nu)$. Once its monitored value starts oscillating, then the model has converged and is moved into the proposal state. If the model parameters remain unchanged during the proposal stage (no split or merge is accepted), then the clustering is complete.

$$\text{Lower-Bound}(r) = \underbrace{\left[\prod_{n=1}^{N}\prod_{k=1}^{K} r_{nk} e^{r_{nk}}\right]}_{q(\mathbf{z})} \underbrace{\frac{\prod_{i=1}^{K}\Gamma(\alpha_0)}{\Gamma\left(\sum_{k=1}^{K}\alpha_0\right)}}_{B(\alpha_0)} \underbrace{2^{\frac{\nu d}{2}} |W|^{\frac{\nu}{2}} \Gamma_d\left(\frac{\nu}{2}\right)}_{C(W,\nu)} \tag{4.5}$$

The only parameters relevant for our clustering method are the prior hyperparameters (See Definition 3.0.6). Most of the parameters (i.e. α, κ, and ν) are not very relevant if they are much smaller than the sample count. We use the Σ_{scale} parameter to scale the dataset covariance for more effective control over the strength of the data-bound prior parameters

μ_0, Σ_0. In Section 6.5 we provide a more detailed analysis of result sensitivity to prior parameters.

To fit the clustering model we use primary embedding to calculate assignment and posterior parameters as it contains features relevant for both temporal and topological tasks. While not explored in this thesis, it is worth noting that it is not necessary to have all the embeddings in memory as exact posterior parameters depend on data μ and Σ which can be calculated over multiple batches.

4.4. End-to-end approach

Given a graph embedding, it is straightforward to find communities by performing the embedding and clustering tasks sequentially. This approach lacks a unified objective, thus, the node embeddings may not be optimized for community detection. We extend the objective with cluster-based loss calculated as the distance between node embedding and its assigned cluster z_v Eq. (4.6). This introduces a feedback loop that encourages the model to reinforce community structures while optimizing the topological and temporal objectives Eq. (4.9). The influence of three objectives can be controlled using hyperparameters $\beta^{\mathcal{E}}$, $\beta^{\mathcal{T}}$, and $\beta^{\mathcal{C}}$.

$$\mathcal{L}^{C} = \|Z_v - \mu_{z_v}\|^2_{\ell 2} \tag{4.6}$$

$$\mathcal{L}^{\mathcal{E}} = \text{MM-Loss}(\mathbf{Z}, P^{\mathcal{E}}, \bar{P}, v) \tag{4.7}$$

$$\mathcal{L}^{\mathcal{T}} = \text{MM-Loss}(\mathbf{Z}, P^{\mathcal{T}}, \bar{P}, v) \tag{4.8}$$

$$\mathcal{L} = \beta^{\mathcal{E}}\mathcal{L}^{\mathcal{E}} + \beta^{\mathcal{T}}\mathcal{L}^{\mathcal{T}} + \beta^{\mathcal{C}}\mathcal{L}^{\mathcal{C}} \tag{4.9}$$

With this closed feedback loop, the training procedure consists of two alternating stages (See Fig. 4.1 and Algorithm 4.4). The *embedding optimization* stage (line 2), is responsible for optimizing the graph embedding function parameters while keeping cluster parameters θ fixed. Once the graph embeddings are updated we run I_c clustering/EM steps to optimize cluster parameters θ while keeping node representations fixed line 10. Note that the *representation optimization* stage is run until convergence as part of pretraining beforehand to ensure the clusters are initialized properly.

Algorithm 4.4: MGTCOM learning pipeline

for *subiter* $= 1$ **to** I **do**
  for $v \in \mathcal{V}$ **do**
    `/* Gather context samples */`
    $P^{\mathcal{E}} = \texttt{Node2VecRandomWalk}(G, l, v)$;
    $P^{\mathcal{T}} = \texttt{BallroomWalk}(G, \omega, l, v)$;
    $\overline{P} \overset{l}{\sim} \mathcal{V}$ Negative sampling;
    $\mathcal{B} = P_l^{\mathcal{E}} \cup P_l^{\mathcal{T}} \cup \bar{P}_l$;
    $\mathbf{Z} = \texttt{EmbedPrimary}(G, \mathcal{B})$;
    Compute task embeddings $Z^{\mathcal{E}}$, $Z^{\mathcal{T}}$ using Eq. (4.1);
    Compute loss $\mathcal{L}^{\mathcal{E}}$, $\mathcal{L}^{\mathcal{T}}$, $\mathcal{L}^{\mathcal{C}}$, $\mathcal{L}$ using Eqs. (4.6) to (4.9) given respective context $P^{\mathcal{E}}$, $P^{\mathcal{T}}$;
  for *iter* $= 1$ **to** I_c **do**
    if $i = 1$ **then**
      Initialize θ using K-means
    Update θ using EM given Z

5. Experiments

In this section, we investigate the effectiveness of the proposed framework *MGTCOM* (in Chapter 4) by evaluating its performance on auxiliary tasks related to multimodal networks. We start by describing our experimental setup, whereafter we compare the performance of our model against baseline methods.

5.1. Evaluation metrics

There are no measures that can assess the quality of communities in multimodal networks. Therefore, we evaluate our model component-wise by defining related auxiliary tasks. On a high level, these tasks evaluate the efficiency of topological and temporal node embeddings and found communities. The found communities shall capture important patterns in the data which are useful for further analysis. In order to measure predictive performance over distinct aspects of our data, we first define the following labels for calculating performance metrics, then describe the auxiliary tasks.

- **Ground truth labels** L_y. Various datasets include manually selected ground truth labels which capture valuable higher-order relations within data. By measuring prediction performance on this label we gauge the quality of found communities.
- **Node timestamps** $L_{\mathcal{T}}$. We split the nodes evenly into snapshot labels given the timestamp of their first occurrences. This allows measuring the quality of node embeddings on temporal prediction.
- **Link-based communities** L_G. While other measures such as modularity and link prediction are well-suited for measuring the quality of node embeddings in capturing the structure of a given network, they either require community assignment or measure low-proximity similarity. In order to overcome this, we first identify community labels using the Louvain method [4]. Then we use those labels to assess the quality of individual node embeddings for community detection. As the Louvain method greedily approximates optimal communities, we don't use this label for formal comparison.

5.1.1. Classification (CF)

In the classification experiment, we evaluate predictive performance given task-related labels. To elaborate, given a set of node embeddings and their respective ground truth labels, we train a logistic regression model to predict node labels. For the predicted node labels, we calculate the average accuracy classification measure.

5.1.2. Link prediction (LP)

In this set of experiments, we evaluate link prediction performance. Given a set of positive and negative node pairs, binary classification is used to predict whether an edge exists within the graph. We use a held-out positive and randomly sampled negative sets of edges to train a logistic regression model. The inner-product similarity between a pair of node embeddings is used as input for the model. By repeating this process three times, the average accuracy is calculated.

5.1.3. Cluster quality

Given node embeddings and their respective labeling, we calculate the silhouette coefficient and Davies-Bouldin index which are helpful to estimate how coherent a clustering is. In this case, a coherent clustering indicates how well represented the correlated patterns are within the embeddings.

Definition 5.1.1 (**Davies-Bouldin Index**). Davies-Bouldin Index (DBI) is the ratio of the sum of the average distance to the distance between the centers of mass of the two clusters. In other words, it is defined as a ratio of within-cluster, to the between cluster separation. This measure is defined as an average over all the found clusters and is therefore also a good measure to decide how many clusters should be used (See Eqs. (5.2) and (5.3)). The s_i refers to the average distance between each point in cluster i to its cluster center μ_i, and $dist(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)$ refers to the distance between cluster centers μ_i and μ_j. Since we use inner-product for node similarity, we define inner-product distance as Eq. (5.1).

$$dist(\mathbf{Z}_i, \mathbf{Z}_j) = -\sum_{m=0}^{d} Z_{im} Z_{jm} \tag{5.1}$$

$$R_{ij} = \frac{s_i + s_j}{dist(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)} \tag{5.2}$$

$$DBI = \frac{1}{k} \sum_{i=1}^{K} \max_{i \neq j} R_{ij} \tag{5.3}$$

$$\tag{5.4}$$

5.1.4. Link-based Community quality

In this experiment, we measure link-based community quality. Girvan and Newman [23] defined community structure as a group of nodes where inter-community connectivity is higher than intra-connectivity. Following this definition, they introduce a modularity measure to evaluate the quality of found communities in a given network. We make use of this measure in our empirical evaluation. Note that we use modularity (Definition 5.1.2) to measure the quality of topological communities.

Definition 5.1.2 (**Modularity**). Modularity directly measures the density of links inside a graph and is therefore computed on communities (sets of nodes) individually by weighing edges using community similarity (or exact matching). Calculation of modularity is done by aggregating per community r for each pair of nodes v and w the difference between the expected connectivity $\frac{k_v k_w}{2m}$ (expected amount of edges between the nodes) and the actual connectivity A_{vw} (existence of an edge) given their degrees (k_v and k_w). The final result represents the connectivity difference between the current and a random graph, as expected connectivity is determined by random rewirings. Because intracommunity pairs are weighted less than intercommunity pairs, the score can vary. See Eq. (5.5), where S_{vr} denotes membership of node v to community r (Eq. (5.6)), and m represents the total edge count.

$$Q = \frac{1}{2m} \sum_{vw} \sum_{r} \left[\overbrace{A_{vw}}^{\text{Connectivity}} - \underbrace{\frac{k_v k_w}{2m}}_{\text{Expected Connectivity}} \right] \overbrace{S_{vr} S_{wr}}^{\text{Community Similarity}} \tag{5.5}$$

$$S_{vr} = \begin{cases} 1 & \mathbf{z}_v = r \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

5.1.5. Ground-truth community quality (COM)

Similarly, to measure the quality of detected communities for specific tasks, we measure the Normalized Mutual Information Score (NMI) score given a task-based label (Definition 5.1.3).

Definition 5.1.3 (**Normalized Mutual Information Score (NMI)**). Normalized Mutual Information is a popular measure used to evaluate network partitioning. It is a variant of a common measure in information theory called Mutual Information defined by $I(X;Y) = H(X) - H(X|Y)$ and represents a reduction in entropy $H(X)$ of variable X by observing the random variable Y or vice versa. In the context of ground-truth community evaluation setting this measure is used to quantify the overlap between two sets of partitions. The Mutual Information score for two sets of partitions X and Y is computed using Eq. (5.7), where $|X|$ is the size of set X, X_i refers to i'th partition of set X, and N is the total number of data points. Finally, the NMI score is computed by normalizing the MI score using the arithmetic mean of entropy of respective partitions Eq. (5.8).

$$MI(X;Y) = \sum_{i=1}^{|X|} \sum_{j=1}^{|Y|} \frac{|X_i \cap Y_j|}{N} \log \frac{N |X_i \cap Y_j|}{|X_i| |Y_j|} \tag{5.7}$$

$$NMI(X;Y) = \frac{MI(X;Y)}{(H(X) + H(Y))/2} \tag{5.8}$$

5.2. Experimental setup

As shown in Table 2.1 of Chapter 2 there are no directly comparable methods to ours in terms of features. For a fair and coherent comparison, we define three variants of the MGTCOM model for evaluation. In addition to the complete end-to-end model $MGTCOM$, we split our framework into a temporal model $MGTCOM^{\mathcal{T}}$ and topological model $MGTCOM^{\mathcal{E}}$, by removing $\mathcal{L}_{\mathcal{T}}$ and $\mathcal{L}_{\mathcal{E}}$ from the objective respectively.

For evaluation, we split the network edges into disjoint training (80%), validation (10%), and testing (10%) sets. During link prediction, we exclusively use links in the respective set as positive pairs. Negative pairs are sampled given the full set of edges. Similarly, the clustering is computed on the training embeddings while cluster-based metrics are calculated using test and validation sets. During the calculation of predictive metrics such as link prediction and classification, we run logistic regression three times and use the average to get an accurate measurement.

5.2.1. Hyperparameters

The hyperparameters for $MGTCOM$ model can be attributed to either network architecture, topological random walk, temporal random walk or clustering. In Section 6.5 we explore the sensitivity of our model to these hyperparameters. In Appendix A we display a complete overview of all the hyperparameter values used for evaluation. The most important hyperparameters are specified below.

For primary embedding, we use two HGT layers with neighborhood sampling sizes of 8 and 4. All the hidden dimensions are equal to the representation dimension, which is 64 ($d = 64$). For temporal and topological context sampling we use walk length $l = 10$ with 10 walks per node. Node2Vec is configured to use $q = 0.5$ to favor neighborhood exploration. The temporal sampling window ω for ballroom walk is determined for each dataset by splitting $\mathcal{T}$ into 20 even partitions. For the clustering module we define prior parameters as $\nu = d + 1, \kappa = 1, \alpha = 10$ and $\Sigma_{scale} = 0.05$. We set trade-off parameters as $\beta^{\mathcal{E}} = 1, \beta^{\mathcal{T}} = 1, \beta^{C} = 0.01$. For max-margin loss we set Δ to 0.1.

5.2.2. Baselines

We use various graph embeddings and community detection algorithms as baselines, covering state-of-art developments in related fields. For the baselines, we use the hyperparameters reported in their respective papers. To keep the results comparable, we use representation dimension $d = 64$ throughout.

- **ComE [9]** uses Gaussian mixture model to learn homogeneous graph embeddings and cluster parameters jointly while utilizing random walk based context sampling.
- **GEMSEC [58]** uses random walks to learn community structure and embeddings simultaneously on homogeneous graphs.
- **CP-GNN [43]** learns node embeddings from a heterogeneous graph by utilizing transformers and k-hop context sampling.

- **CTDNE [51]** utilizes time-based biased random walks to learn spatio-temporal node representations from dynamic networks.
- **GraphSAGE [29]** uses k-hop neighborhood sampling to learn node embeddings from homogeneous graphs. Its unsupervised variant combines contrastive link sampling with hinge loss.
- **Node2Vec [27]** adopts biased random walk and Skip-Gram to learn node embeddings from homogeneous graphs.

5.2.3. Datasets

Table 5.1.: Dataset statistics. *Temporal* indicates if a dataset is temporal and *labelled* refers to the availability of ground truth labels.

Dataset	Node type	# Nodes	Edge type	# Edges	Temporal	Labelled
DBLP	Author (A) Paper (P) Venue (V)	5,162 5,511 14	A - Authored - P P - Published In - V	11,022 5,511	•	•
IMDB	Person (P) Movie (M) Genre (G)	8,491 5,043 26	P - Directed - M P - Acted In - M M - Tagged - G	4,939 15,086 14,504	•	
SDS	User (U) Hashtag (H) Tweet (T)	34,919 2,341 56,173	U - Tweeted - T T - Reply To - U T - Reply To - T T - Quote - T T - Mention - U T - Mention - H U - Follows - U	56,173 21,769 4,296 882 70,367 12,313 5,649,098	•	
ICEWS	Entity (E)	10,463	123 different types	915,028	•	
Cora	Paper (P)	2,708	P - Cites - P	10556		•

We use four widely used real-world (temporal) datasets for evaluation. These graphs are of different types and contain information on different modalities. We applied additional preprocessing on the IMDB, DBLP-HCN, and ICEWS datasets to include the multimodal features present in the datasets but often not included in the graph due to sparsity of temporal or content-based features. See Table 5.1 for a detailed comparison of node features.

- **DBLP** [85] is a citation network consisting of Authors, Papers and Venues. Aside from being heterogeneous, the dataset also contains timestamps representing paper publication dates and abstracts. There are thirteen ground-truth communities representing publication venues. The network contains 10687 nodes and 33066 edges. This dataset includes ground truth labels.
- **ICEWS** [22] is a temporal knowledge graph in which nodes represent entities and timestamped edges the relationship between them. We model this data as a highly heterogeneous network consisting of different types of nodes (10463 in total) connected by 915028 timestamped edges. Edges are labeled with relations.

- **IMDB5000** [1] network consists of Actor, Director, Movie, and Genre nodes where each Movie node type has a timestamp denoting the release date. Additionally, each actor node has a set of attributes characterizing information unique to the actor such as age and popularity, while movies have box-office data and keywords encoded as feature vectors. This network has 13560 nodes and 69058 edges.
- **SocialDistancingStudents (SDS)** [72] represents a small part of the Twitter network around a set of hashtags related to the COVID pandemic. This heterogeneous network models connections between Users, Tweets, and Hashtags where parallel edges are possible due to relations such as tweeted, retweeted, quoted, etc. The tweet nodes contain publication date timestamps and content encoded as feature vectors. 93433 nodes and 7420366 edges are included.
- **Cora** [87] is a homogeneous citation network. Nodes represent published papers and contain feature vectors representing specific term occurrences in the abstract. Each node is associated with one of the seven ground-truth labels.

5.3. Performance comparison

In this experiment, we evaluate the performance of learned node embeddings and detected clusters. In particular, we evaluate the predictive quality of embeddings using classification and link prediction, i.e., link prediction accuracy (LP_{ACC}), temporal $L_{\mathcal{T}}$ and ground truth L_y label classification accuracy CF_{ACC}. We evaluate the quality of detected clusters by calculating their NMI score based on predefined ground-truth communities L_y, $L_{\mathcal{T}}$, L_G. This tells us whether detected clusters approximate user-defined communities L_y, temporal partitioning $L_{\mathcal{T}}$ or the topology L_G. Additionally, we calculate cluster and community quality scores for the learned community assignments, specifically Davies Bouldin score and modularity.

The embeddings obtained from non-community detection methods were clustered using k-means clustering with $K = 20$. Similarly, we use $K = 20$ for community detection methods (ComE, GEMSEC, CP-GNN) that assume a predefined cluster count. The results are reported in Table 5.2. It can be seen that while MGTCOM is competitive on task-specific measures such as link prediction and timestamp prediction, the community detection methods still have an edge on link-based modularity measures. A possible explanation for this would be the fact that the DPMM process is more prone to getting stuck in local minima as the clusters split and merge. Another possibility is that node features do not contain enough information to model very specific network features such as modularity. In Section 6.3 we further explore this issue by varying the auxiliary embedding ratio.

While CTDNE performs comparatively well in capturing the temporal aspect of the network, we see that it still yields inferior results on datasets where temporal features are weakly correlated with topology.

It is interesting to note that algorithms that rely on pairwise loss measures such as GraphSAGE and CP-GNN perform relatively well on classification-based measures while performing very poorly on cluster quality measures such as DBI and modularity. A possible explanation for such observation is that the combination of neighborhood sampling and pairwise loss reinforces structural similarity despite having a large receptive field. Our method successfully overcomes this issue by modifying Hinge loss to work in a context path setting (See Section 4.2.2).

Table 5.2.: Comparison of performance of baselines on multimodal graph learning tasks. ("-" means no data available, for example for temporal methods on static datasets such as Cora). The calculated metrics are the link prediction accuracy (LP_{ACC}), predictive accuracy on ground truth communities CF_{ACC} L_y, timestamp predictive accuracy CF_{ACC} $L_{\mathcal{T}}$, NMI score of detected communities (COM_{NMI}) given predefined communities (L_y, $L_{\mathcal{T}}$, L_G), Davies-Bouldin Index (DBI) and Modularity.

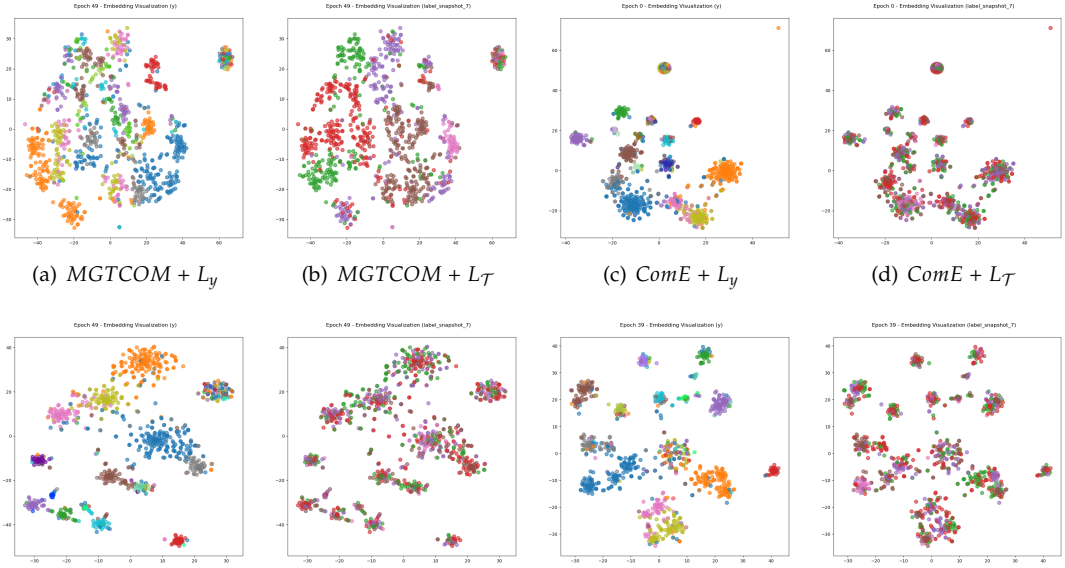
Dataset		GraphSAGE	Node2Vec	ComE	GEMSEC	CTDNE	CP-GNN	*MGTCOM*	$MGTCOM^{\mathcal{T}}$	$MGTCOM^{\mathcal{E}}$
DBLP	LP_{ACC}	0.624	0.710	0.735	0.544	0.701	0.522	0.743	0.634	**0.794**
	CF_{ACC} L_y	0.315	0.832	0.842	0.831	0.809	0.506	**0.896**	0.330	0.884
	CF_{ACC} $L_{\mathcal{T}}$	0.309	0.308	0.328	0.324	0.488	0.313	**0.758**	0.508	0.320
	COM_{NMI} L_y	0.051	**0.549**	0.463	0.385	0.537	0.209	0.465	0.059	0.492
	COM_{NMI} $L_{\mathcal{T}}$	0.006	0.033	0.025	0.022	0.059	0.022	**0.209**	0.168	0.026
	COM_{NMI} L_G	0.040	0.425	**0.470**	0.314	0.401	0.107	0.336	0.039	0.371
	DBI	0.472	2.305	2.205	4.056	1.206	4.780	2.039	4.205	**5.188**
	Modularity	0.028	**0.662**	0.636	0.492	0.642	-0.035	0.427	0.137	0.514
ICEWS	LP_{ACC}	0.525	0.936	0.880	0.768	0.921	0.709	0.903	0.896	**0.945**
	CF_{ACC} $L_{\mathcal{T}}$	0.294	0.301	0.264	0.310	0.285	0.273	0.316	**0.318**	0.313
	COM_{NMI} $L_{\mathcal{T}}$	0.018	0.040	0.015	0.022	0.022	0.013	**0.057**	0.002	0.011
	COM_{NMI} L_G	0.227	0.354	**0.548**	0.309	0.347	0.204	0.119	0.001	0.447
	DBI	1.027	1.697	2.559	3.867	1.533	**4.737**	3.883	3.598	3.182
	Modularity	0.218	0.215	**0.483**	0.311	0.239	0.199	0.007	0.001	0.390
IMDB	LP_{ACC}	0.714	0.757	0.666	0.637	0.728	0.598	0.721	0.724	**0.773**
	CF_{ACC} $L_{\mathcal{T}}$	0.346	0.373	0.394	0.380	0.488	0.316	**0.659**	0.556	0.377
	COM_{NMI} $L_{\mathcal{T}}$	0.022	0.025	0.031	0.013	0.065	0.004	**0.239**	0.231	0.026
	COM_{NMI} L_G	0.039	0.181	**0.197**	0.094	0.160	0.033	0.107	0.031	0.158
	DBI	0.301	1.803	3.840	**4.951**	1.749	4.806	2.257	1.285	4.013
	Modularity	-0.172	0.190	**0.395**	0.073	0.196	0.053	0.119	0.114	0.286
SDS	LP_{ACC}	0.922	0.953	0.758	0.878	0.955	-	0.934	0.616	**0.956**
	CF_{ACC} $L_{\mathcal{T}}$	0.521	0.445	0.386	0.384	0.447	-	0.523	**0.887**	0.492
	COM_{NMI} $L_{\mathcal{T}}$	0.250	0.149	0.117	0.015	0.161	-	0.204	**0.536**	0.044
	COM_{NMI} L_G	0.186	0.277	0.346	0.117	0.233	-	0.120	0.043	**0.389**
	DBI	1.108	2.355	**3.986**	3.410	2.890	-	2.474	1.519	2.559
	Modularity	0.088	0.163	0.301	0.037	0.016	-	0.015	0.005	**0.374**
Cora	LP_{ACC}	0.505	0.939	**0.962**	0.923	-	0.829	-	-	0.958
	CF_{ACC} L_y	0.659	0.798	**0.864**	0.845	-	0.780	-	-	0.854
	COM_{NMI} L_y	0.376	0.345	0.434	0.437	-	0.370	-	-	**0.439**
	COM_{NMI} L_G	0.507	0.543	0.635	0.632	-	0.501	-	-	**0.643**
	DBI	1.526	1.250	2.021	1.500	-	2.634	-	-	**2.647**
	Modularity	0.636	0.691	**0.785**	0.780	-	0.677	-	-	0.754

We also observe that the MGTCOM model performs well on both topology and temporal prediction tasks in comparison to its task-specific counterparts.

5.4. Qualitative results

We further compare *MGTCOM* and the baseline models on the DBLP-HCN network. We apply the T-SNE dimensionality reduction technique to visualize the trained node embeddings in 2D space colored by the ground truth label and the node timestamp (See Fig. 5.1).

Since in the DBLP-HCN dataset the timestamps are weakly correlated with its topology, we can see that topology-focused embedding (and community detection) methods such as ComE and Node2Vec do not capture temporal relations of nodes. On contrary, we observe distinct patterns emerge when looking at *MGTCOM* generated embeddings for both of the labels. Similar to that of *ComE* the community structures are visible in the node embeddings though they are not as distinct.



(a) *MGTCOM* + L_y (b) *MGTCOM* + $L_{\mathcal{T}}$ (c) *ComE* + L_y (d) *ComE* + $L_{\mathcal{T}}$

(e) *Node2Vec* + L_y (f) *Node2Vec* + $L_{\mathcal{T}}$ (g) *CTDNE* + L_y (h) *CTDNE* + $L_{\mathcal{T}}$

Figure 5.1.: Visualization of trained embedding against ground truth labels (L_y, left) and timestamp labels ($L_{\mathcal{T}}$, right) for DBLP-HCN dataset. (Note: The embeddings are calculated on the training dataset. Each of the plots contains a blob of nodes that have no edges in the training set due to the validation split. None of the methods is equipped to handle disconnected nodes.)

5.5. Inference results

Because the *MGTCOM* model operates on sampled neighborhood subgraphs, in contrast to other methods it can operate in an inductive setting. Meaning that it is not necessary to retrain the model to infer representation vectors for previously unseen nodes.

We evaluate the performance of *MGTCOM* and its task-specific variants in inductive settings by controlling the ratio of nodes in the training set to the validation set. The test set remains constant throughout the experiment to accurately assess performance on inferred nodes. The relevant quality measures are computed exclusively on the test set and can be found in Table 5.3.

In Fig. 5.2 we see the same measures plotted with the training ratio on the x-axis. From Fig. 5.2 (a) we observe that varying training set size does not affect link-prediction tasks as much as node classification tasks (b, c, d). Throughout the measures, we can see that using only 75% of the data does not substantially affect the results. Interestingly, we observe that the variance on the temporal prediction task increases when more data is provided.

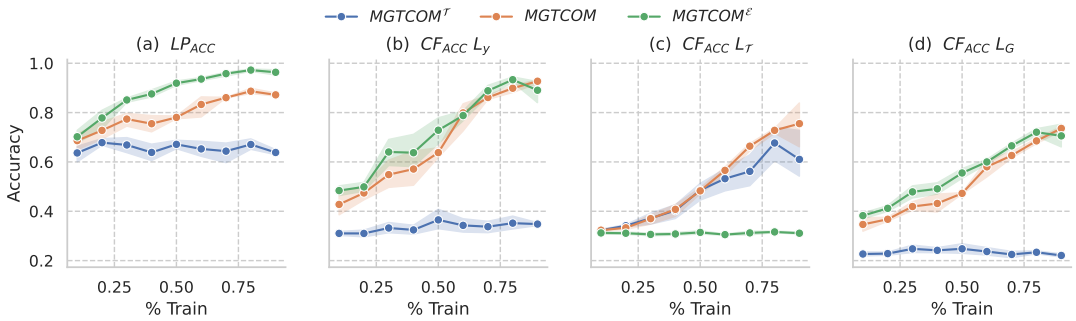


Figure 5.2.: Visual comparison of different model variants in the inference-based setting. Graph nodes are split into three disjointed sets (train, validation, and test). The metrics are measured while the training to validation ratio is varied. The test set is set to 10% of the nodes and is kept constant. The average metrics per data value are plotted along with their standard deviation.

Table 5.3.: Comparison of different model variants in the inference-based setting. Graph nodes are split into three disjointed sets (train, validation, and test). The metrics are measured while the training to validation ratio is varied. The test set is set to 10% of the nodes and is kept constant.

Model	**% Train**	10%	20%	30%	40%	50%	60%	70%	80%	90%
MGTCOM	LP_{ACC}	0.686	0.728	0.774	0.755	0.780	0.833	0.861	0.887	0.872
	CF_{ACC} L_y	0.428	0.474	0.548	0.571	0.638	0.799	0.861	0.899	0.927
	CF_{ACC} $L_{\mathcal{T}}$	0.323	0.333	0.370	0.408	0.483	0.566	0.664	0.728	0.755
	CF_{ACC} L_G	0.347	0.368	0.419	0.432	0.473	0.580	0.626	0.685	0.736
MGTCOM$^{\mathcal{E}}$	LP_{ACC}	0.702	0.778	0.851	0.876	0.919	0.936	0.958	0.972	0.963
	CF_{ACC} L_y	0.484	0.499	0.640	0.637	0.729	0.788	0.888	0.933	0.891
	CF_{ACC} $L_{\mathcal{T}}$	0.312	0.311	0.306	0.308	0.314	0.305	0.312	0.316	0.311
	CF_{ACC} L_G	0.382	0.412	0.479	0.491	0.555	0.600	0.665	0.721	0.706
MGTCOM$^{\mathcal{T}}$	LP_{ACC}	0.636	0.678	0.669	0.639	0.671	0.653	0.644	0.671	0.638
	CF_{ACC} L_y	0.310	0.310	0.332	0.324	0.365	0.343	0.337	0.352	0.348
	CF_{ACC} $L_{\mathcal{T}}$	0.323	0.341	0.372	0.403	0.485	0.532	0.562	0.677	0.610
	CF_{ACC} L_G	0.227	0.228	0.248	0.242	0.248	0.237	0.225	0.234	0.221

5.6. Learnable parameter reduction

An important goal of our work is to prove that inductive-based community detection is feasible. We address the structural similarity bias found in many unsupervised inductive algorithms by introducing a custom loss and sampling methodology in Section 4.2.2. While our model still utilizes embeddings to address the incompleteness constraints, we show in Section 6.3 that importance-based pruning is an effective optimization to keep the model

scalable.

As result, our model takes advantage of the scalability of inductive representation learning methods. In Table 5.4 we compare the parameter count of the *MGTCOM* model to the node2vec model which directly learns node embeddings. Overall *MGTCOM* has fewer parameters since the model size is bound by meta-topology. In highly heterogeneous graphs such as the ICEWS dataset, the number of parameters may become larger than expected. Specifically, the number of parameters is proportional to $|\mathcal{A}| + |\mathcal{R}|$. For exact analysis on the number of learnable parameters the model uses, we refer the reader to Appendix A.2.

Table 5.4.: Parameter count comparison between node2vec and the *MGTCOM* model.

Dataset	node2vec	*MGTCOM*
DBLP	683,968	173,910
ICEWS	669,632	1,072,302
IMDB	867,840	170,846
SDS	5,979,712	231,282
Cora	173,312	136,390

6. Ablation study

In this section, we investigate the sensitivity of our model to the described design choices and hyperparameter values. Throughout the experiments, we keep the same base parameters as described in the experimental setup. Similarly, the DBLP dataset is used throughout as it provides a wide range of features suitable for the evaluation of all supported tasks.

6.1. Auxiliary Embedding Ratio

To address the incompleteness constraints, *MGTCOM* introduces auxiliary embeddings for nodes without features. Zero-vector features are used for nodes that are unseen during training and don't have their own feature vector to encourage its inference from neighboring nodes. While doing this introduces performance benefits, for large datasets it may not be possible to store the auxiliary embeddings in memory.

We define a procedure to work around this scaling issue by noting that embeddings only need to be constructed for a fraction of the most important nodes. This is due to scaling laws applicable to most real-world networks. Specifically, in this experiment, we sort all the nodes without features by their degree and use a fraction of the highest degree nodes for auxiliary embeddings. Other nodes are given a zero-vector upon inference.

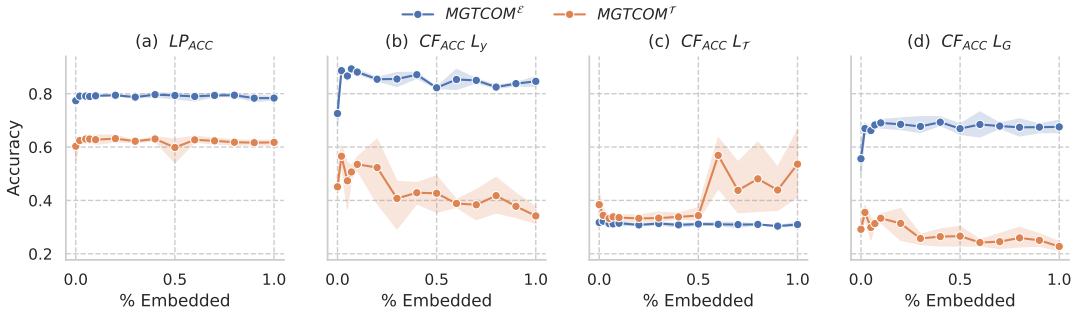


Figure 6.1.: Performance results for topological $MGTCOM^{\mathcal{E}}$ and temporal $MGTCOM^{\mathcal{T}}$ models on various tasks where the ratio of auxiliary embedded nodes varies.

In Fig. 6.1 we see the results of the tasks specific models when the auxiliary ratio is varied. From figure (a) we can observe that while auxiliary embeddings don't have a large influence during link prediction, they are in fact necessary on prediction tasks as figures (b), (c), and (d) indicate. It can be rightfully deduced that embeddings are necessary for temporal tasks (figure (c)) since topology and content-based features are weakly correlated with temporal features.

6.2. Meta-topological features

Meta-topological features are an important part of multimodal graphs. In this experiment, we aim to determine the importance of meta-topology in our evaluation setting. We evaluate the performance measures on heterogeneous and homogeneous variants of the DBLP dataset. By varying convolutional layers between Heterogeneous Graph Transformer and GraphSAGE [15] (each edge type has a separate set of weights), we additionally aim to determine the importance of meta-topology-based attention used during the aggregation step.

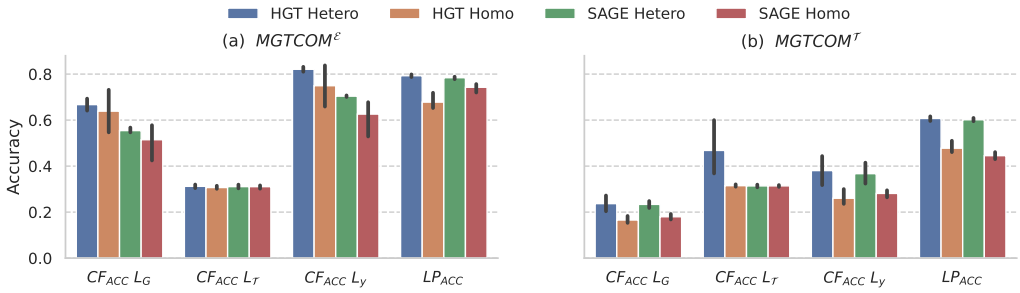


Figure 6.2.: Performance results for (a) topological $MGTCOM^{\mathcal{E}}$ and (b) temporal $MGTCOM^{\mathcal{T}}$ models, on prediction tasks for heterogeneous and homogeneous variants of the DBLP dataset. To determine the importance of meta-topological attention we vary the convolutional layers HGT and GraphSAGE (which is adopted for heterogeneous graphs).

Overall in Fig. 6.2 we see that the addition of meta-topological features has a positive effect on the classification performance of both topological as well as temporal models. This effect is especially pronounced on link prediction and topology-based classification tasks for the temporal model. The cause for this may be that while topological features are not provided during training, meta-topology still conveys enough information about the topology.

From the results, we see that meta-topology-based attention yields benefits in classification performance in contrast to naive aggregation techniques (improvement by 10%).

6.3. Trade-off Parameter

During analysis the trade-off parameters ($\beta^{\mathcal{E}}$, $\beta^{\mathcal{T}}$, $\beta^{\mathcal{C}}$) are used to guide the trained embeddings to favor specific tasks. In this experiment, we explore the trade-off between temporal and topological tasks by varying value of $\beta^{\mathcal{E}}$, $\beta^{\mathcal{T}}$ while setting the constraint $1 = \beta^{\mathcal{E}} + \beta^{\mathcal{T}}$. The clustering weight parameter $\beta^{\mathcal{C}}$ remains constant throughout as described in Section 5.2.

In Fig. 6.3 (a) we can see an almost linear correlation between link prediction accuracy and the topological weight parameter $\beta^{\mathcal{E}}$. On the other hand, in figures (b) and (d) we see a more logarithmic curve for topology correlated classification measures. The most interesting takeaway is that while variance is quite high on the temporal classification task, its curve peaks at a value of 0.5. In further work, it may be worth exploring this phenomenon in more

detail. The most probable assumption would be that the temporal model still benefits from the fact that temporal features are weakly correlated with the topology.

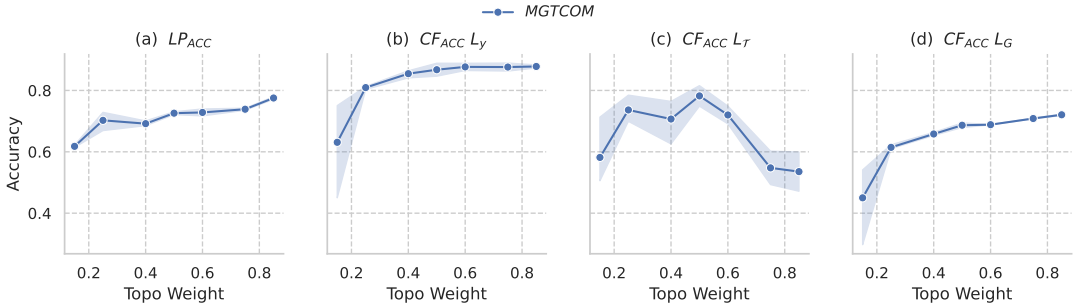
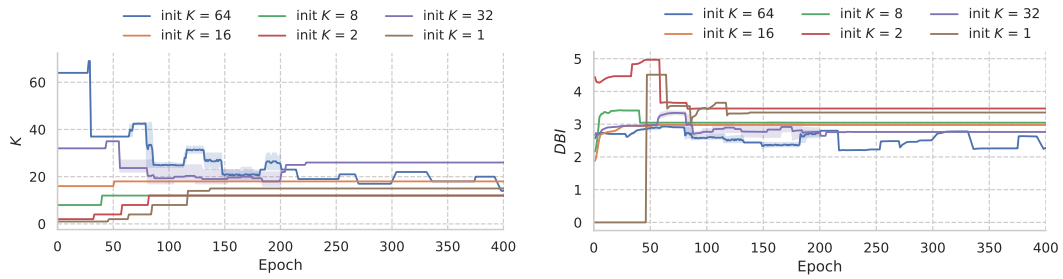


Figure 6.3.: Performance of *MGTCOM* model while varying topological loss weight parameter $\beta^{\mathcal{T}}$ under $1 = \beta^{\mathcal{E}} + \beta^{\mathcal{T}}$ contraint.

6.4. Initial K sensitivity

In this section evaluate the sensitivity of the clustering results to the initial K value selection. While our method does not require setting the cluster to count K, it can still be set to find more accurate initial clustering, and help DPMM avoid local minima. Specifically, we have varied the initial cluster count (init K) used for k-means initialization while keeping all other parameters fixed.



(a) (b)

Figure 6.4.: (a) and (b): Cluster count progression during DPMM clustering given an initial cluster count (init K). The clustering is done on pre-trained *MGTCOM* embeddings for the DBLP dataset.

In Fig. 6.4 (a) we see that despite varying starting values, all the runs converge at 12-18 cluster range. Having a value that strongly deviates from the "optimal" cluster count causes a slower convergence since more split/merge operations are required. We can see a similar pattern in the measured Davies-Bouldin index in Fig. 6.2 (b).

6.5. Hyperparameter sensitivity

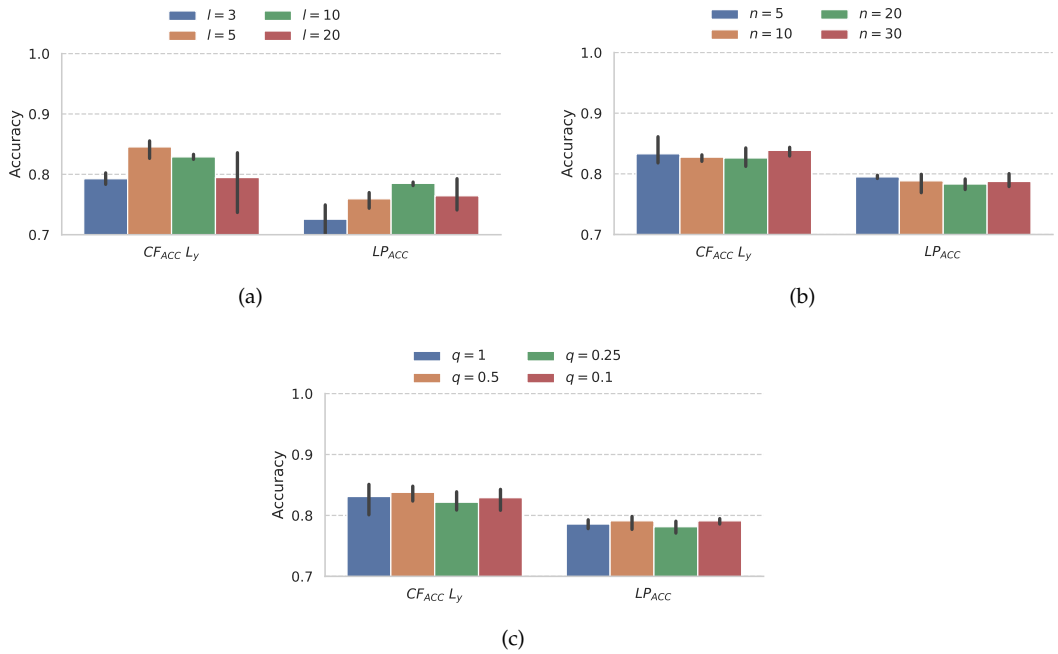
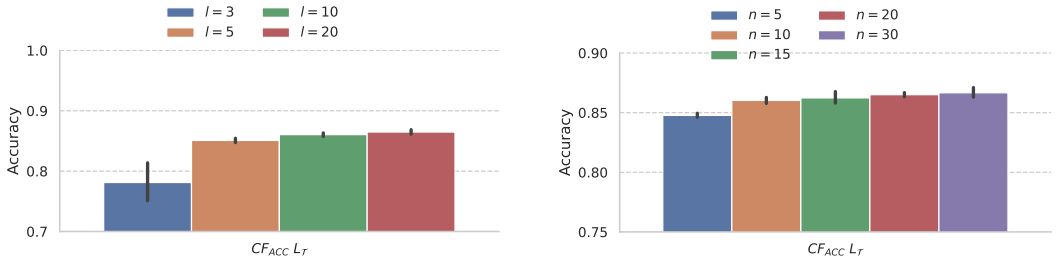


Figure 6.5.: Performance of $MGTCOM^{\mathcal{E}}$ model with varying (a) random walks length l, (b) number of random walks per node n, and (c) the exploration trade-off parameter q.

In this part, the sensitivity of other hyperparameters on the model performance is discussed. The node2vec random walk algorithm used for the topological task relies on parameters such as walk length l, the number of random walks n for each node, and the exploration trade-off parameter q. In Fig. 6.5 we see that while the choice of random walk length has a significant impact on link-prediction and classification performance (a), the model is not as sensitive to the other parameters. A surprising observation is that the trade-off parameter does not significantly affect the productivity accuracy of ground communities (CF_{ACC} L_y). A possible explanation for this may be the fact that we use both random walk and neighborhood sampling algorithms making the trade-off ineffective.

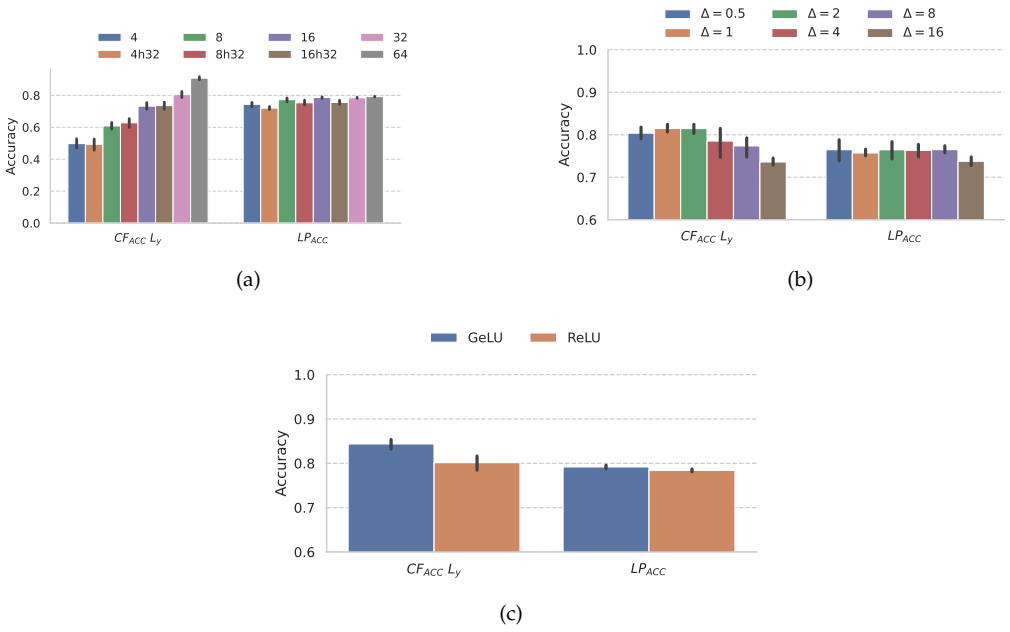
The ballroom walk algorithm introduced in Section 4.2.3 similarly relies on the hyperparameters walk length l and the number of random walks started for each node n, though they serve a different purpose. Increasing either the l or the n parameter only marginally increases the models performance at timestamp prediction (See Fig. 6.6). For both parameters, there is a positive correlation between performance and an increase in the receptive field.

The most sensitive/important parameter for our model is the representation dimension size d. In Fig. 6.7 (a) we plot the predictive performance of the topological model while varying the model representation dimension d and the hidden representation dimension h used in in-between layers of graph convolution. The classification performance seems to benefit the most from a larger d, while link-prediction only sees a marginal improvement. Moreover



(a) (b)

Figure 6.6.: Performance of $MGTCOM^{\mathcal{T}}$ model with varying (a) random walks length l and (b) number of random walks per node n.



(a) (b)

(c)

Figure 6.7.: Performance of $MGTCOM^{\mathcal{E}}$ model with varying (a) representation dimension d, (b) the margin (Δ) parameter for the hinge loss, and (c) activation function.

having hidden dimension size deviate from the representation dimension only seems to degrade the model performance.

In Fig. 6.7 (b) we vary the margin parameter of hinge loss. It is conventional to use $\Delta = 1$ if the similarity is bounded (as is in our case), therefore we can see the model performance degrade as the margin exceeds this threshold. Increasing loss beyond 1 amplifies the relative relevance of small loss samples, which in turn makes the model more prone to noise.

While constructing the network we found that the choice of activation function noticeably

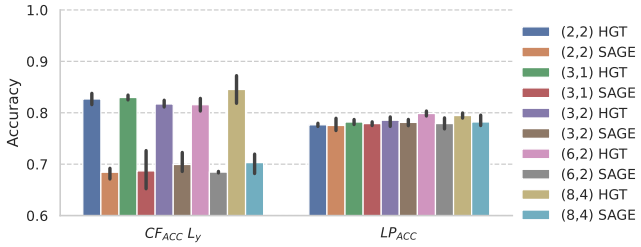


Figure 6.8.: Performance of $MGTCOM^{\mathcal{E}}$ model with varying convolution layers. We vary the architecture by switching between Heterogeneous Graph Transformer (HGT) and Heterogenous GraphSAGE convolutional layers. Similarly, we also modify the neighborhood size of each node within the two-layer convolution setup. Format (x, y) represents the number of neighbors per node in the first (x) and second (y) layer respectively.

affects the model performance. Choosing GeLU over ReLU activation speeds up model convergence and gains a noticeable edge in classification tasks (See Fig. 6.7 (c)).

In Fig. 6.8 we vary the convolution architecture of the topological model and measure the resulting test performance. As observed earlier HGT convolutional layers perform better since they introduce meta-topology-based attention. Varying the layer neighborhood size does not seem to affect the performance substantially, except for the fact that computed performance measures during training are a lot smoother throughout.

Finally, in Fig. 6.9 we analyze the sensitivity of the resulting cluster count to the chosen prior parameters for the DPMM algorithm. We notice that the σ scale and ν parameters have a linear correlation with the resulting number of clusters. The σ scale parameter influences the data-bound W hyperparameter and is to be expected to have a great impact. A larger prior covariance corresponds to a larger probability that any of the clusters are drawn from it and therefore results in a larger number of clusters. On the contrary, a larger degree of freedom requires a larger amount of samples per cluster, therefore reducing the probability of smaller clusters. We see a more noisy pattern from α and κ concentration hyperparameters as they are not data-bound and are less effective when their value is much smaller than the total number of data points N.

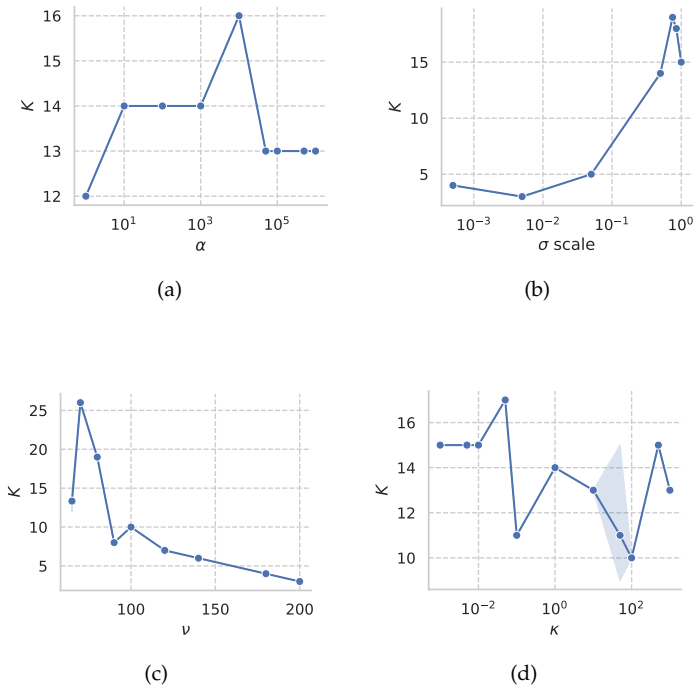


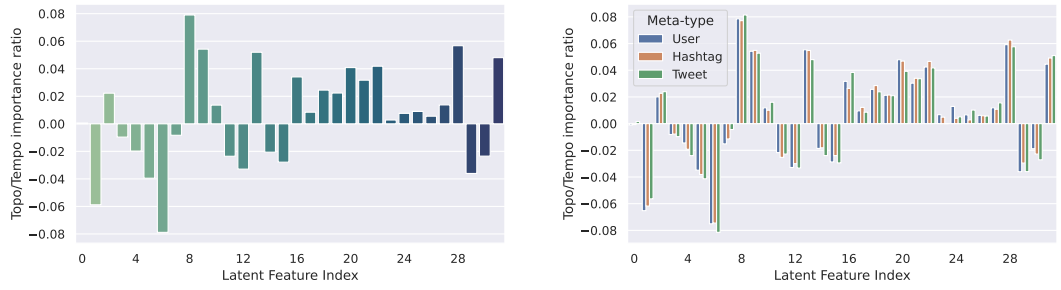
Figure 6.9.: Sensitivity of the resulting cluster count K on the DPMM prior parameters (a) α the cluster concentration parameter, (b) the σ scale parameter influencing the covariance of prior, (c) ν the degrees of freedom parameter for the Wishart prior distribution, and (d) κ the concentration parameter of the Wishart prior distribution.

7. Case study: Social Distancing Students dataset

The Social Distancing Students dataset explores dutch public sentiment on governmental COVID-19 measures on Twitter based on data between February to September 2020 [72]. Public sentiment is an important measure to consider before implementing certain measures or policies as it may influence compliance or cause protests.

The dataset consists of a tweet network encapsulating tweet, retweet, quote, and mention relations and a follower network modeling dynamics of the Twitter social network platform. The network is constructed by gathering tweets matching a predefined set of keywords related to the COVID-19 crisis. The follower network is constructed central to the users related to the tweets. Following the work in the original paper, the dataset contains sentiment analysis labels indicating whether a tweet is in support or rejection of contemporary social distancing policies.

For this case study we have trained the *MGTCOM* model on the dataset using the same parameters as described in the evaluation (Section 5.2). In the following steps, we use the dataset as well as training results to explore the patterns found in the data. Additionally, we attempt to explain the learned features and communities based on the patterns seen in the data they capture.



(a) (b)

Figure 7.1.: Given a latent feature vector ($d = 32$), for each feature, we plot attention averaged over the whole dataset. A positive value means the feature is more important for topological tasks, while a negative value means it is more important for temporal tasks. In plot (b) we similarly visualize the attention averaged over data points while grouped by node type.

In the first step of our analysis, we focus on the relative importance of various embeddings features given their task-specific attention weight. In Fig. 7.1 we plot the attention ratio,

which is computed as the mean difference between the topological and temporal feature-wise attention ($ATT^{\mathcal{E}} - ATT^{\mathcal{T}}$). Since, the attention is computed per node, in figure (b) we plot the attention ratio grouped by node type. While on the feature level a clear distinction is seen where features are more important for either task, we can see that attention between different types doesn't deviate much from the mean. This is not unexpected as the attention is not parameterized by meta-topology, as it is rather implicitly encoded in the primary embedding vector.

In Fig. 7.2 we visualize a temporal histogram for top nodes given individual features that have a higher average temporal attention. Because our distance measure is based on inner-product, the selection of top nodes works by simply sorting node embeddings by the selected feature in descending order. In almost all the subplots we see pronounced peaks at certain timestamps indicating that the latent features capture certain temporal patterns within the data. Since many features capture the peaks during mid-October it is fair to assume that those tweets have certain distinct underlying properties. As a baseline, we plot the general tweet distribution over time in Fig. 7.3 to compare the peaks in feature histograms against.

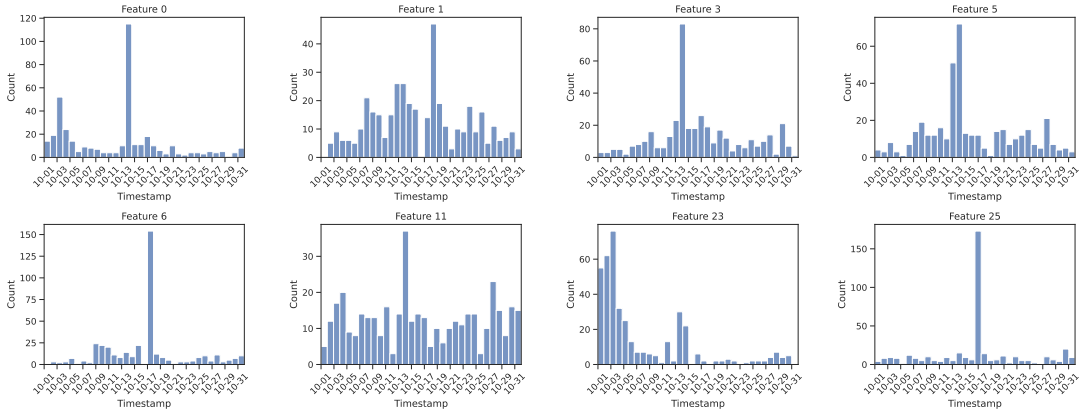


Figure 7.2.: Distribution of top 200 tweets over time ranked by latent embedding feature which is correlated with high temporal attention. The dates on the x-axis are formatted as month-date excluding the year 2021.

To further study the effectiveness of attended features we plot most common words occurring in top tweets given the attended features in Fig. 7.4. In these word clouds, we see more explanations for the patterns seen in the temporal histograms. For latent feature 11 we see the main keywords consisting of "nk", "amsterdam", and "voetbal" referring to contemporary dutch soccer national championship games on 3, 16, 19, 24, and 30th of October. All the games were played by "Ajax" club based in the city Amsterdam and the dates correspond to the peaks in the histogram. Similarly, feature 5 contains keywords "dierendag" and "bioindustrie" corresponding to national animal day on 3rd of October and related tweets raising awareness to the bioindustry. Features 0 and 5 seem to contain more general words referring to trending hashtags ("#scholenveilig" translated school safe) in anticipation to the press conference from the Dutch parliament on 18th of October.

While topological information is gathered from neighboring nodes, it can still tell us about trends on Twitter regarding hashtags and mentions. In Fig. 7.4 (b) we similarly plot key-

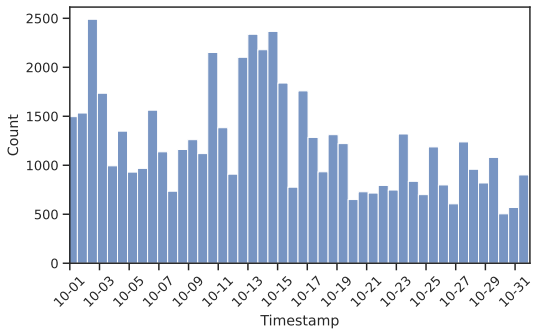


Figure 7.3.: Tweet distribution over time in Social Distancing Students dataset.

(b)

Figure 7.4.: Most common words occurring in the top 200 tweets given (a) temporally correlated latent features, (b) topologically correlated latent features.

words for the most valuable/attended topological features. There we see in features 19 and 13 keywords regarding measures for restaurants and grocery stores respectively. Feature 8 mainly contains travel keywords such as ”ov” and ”ns” referring to travel providers, while feature 9 is more focussed on foreign affairs citing city ”Brussels” and country (”land”).

Next, we are interested in the predictive capability of the embeddings on the sentiment labels provided in the dataset. In Fig. 7.5 we see a plot of the two most correlated latent embedding features colored by the sentiment label. As the predictive accuracy on the label is 81% and does not greatly exceed 79% most frequent label baseline, it is fair to assume that

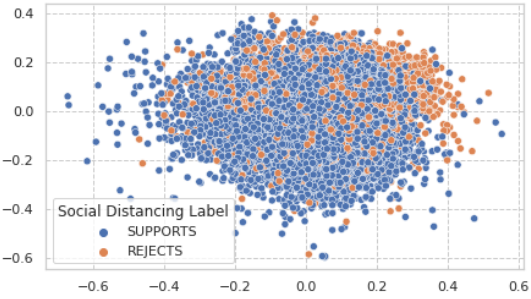


Figure 7.5.: Visualization of two latent embedding features most correlated with social distancing sentiment. Each of the data points is colored corresponding to the sentiment label indicating that the content of the tweet is either for or against a certain measure.

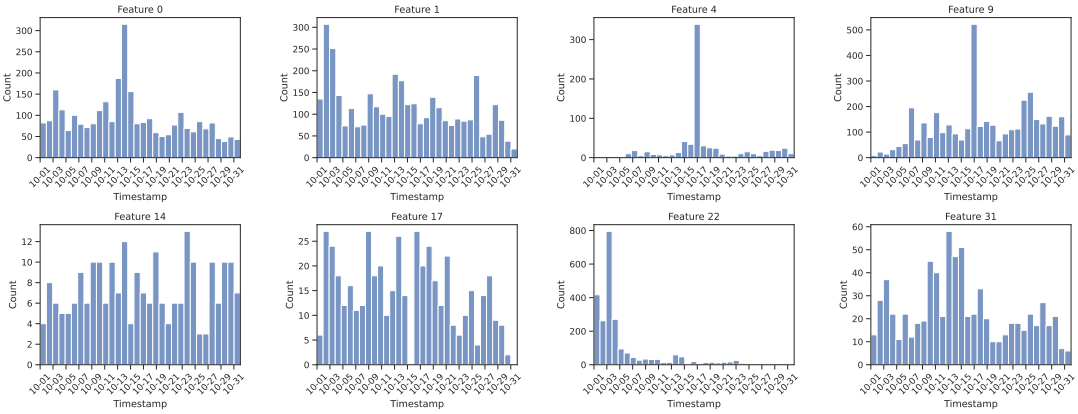


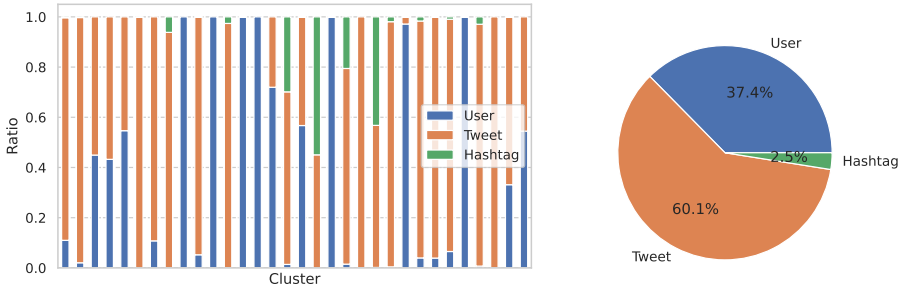
Figure 7.6.: Distribution of tweets given their respective communities over time.

the embeddings do not really capture the sentiment of the tweet contents.

Finally, we analyze the patterns captured by the found communities. In Fig. 7.6 we visualize temporal distribution of Tweets given their corresponding communities. Contrary to temporal patterns, we see that the communities are mixed in their temporal correlation. For example, cluster 14 has a distinct timestamp while cluster 4 is clearly centered around the 17th of October.

To get more insight into the nature of found communities we plot node type distributions for the found communities in Fig. 7.7 (a) alongside the node type distribution of the whole dataset. Aside from mixed-type communities we also see pure communities emerge capturing only follower network data or the tweet data. Surprisingly, while hashtags make up a small portion of the dataset, we see multiple communities containing a non-trivial fraction of hashtags.

Similarly, various patterns the communities capture can be analyzed using the word clouds



(a) (b)

Figure 7.7.: Distribution of node types ("User", "Tweet", "Hashtag") over the (a) found communities, (b) the full dataset.

in Fig. 7.8. There we see more pronounced patterns concerning grocery store policies (feature 1), the overfull hospitals (feature 5) and disease transmission (feature 6).

We conclude this section by noting that the found communities capture patterns by combining similar nodes in terms of content, time events and connections together which is very useful for explorative analysis. This approach is unsuitable for predicting complex patterns such as support for specific policies as it is not encapsulated in the objective function. A more suitable way to capture such feature would use representation vectors for all sentence tokens, instead of the full text average we use currently. We leave exploration of ways to incorporate such feature extraction into graph embedding pipeline as future work.

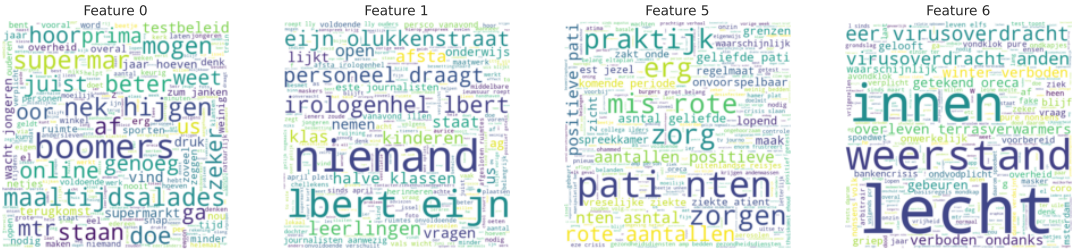


Figure 7.8.: Most common words occurring in various communities.

8. Future Work

While we have explored a multitude of topics, there is still a lot of room for further improvements and exploration. Below we list a various of possible further research directions.

Our experiments have shown that temporal representation learning benefits greatly from auxiliary embeddings as node features may often be too weakly correlated with temporality. In contrast to topological tasks, auxiliary embeddings have been shown to be effective only for the most important nodes. In future work, it may be valuable to explore more flexible settings where representations are augmented with embeddings only for temporal tasks, therefore reducing parameters and inference latency.

The scale of our model is meta-topology bound, meaning that the amount of learnable parameters increases if there are more node or edge types. This reduces the effectiveness of our framework on highly heterogeneous networks such as knowledge graphs. Future works may explore improvements to our embedding method by utilizing techniques used in the knowledge graph embedding field.

While detected communities excel in topological and temporal predictive capabilities, they detected communities still under-perform on the modularity measure. Further work may explore swapping node2vec random walk algorithm by motif-sampling [34] to encourage strong link-based proximity.

The presented framework uses DPMMSC algorithm as introduced in the original paper [11]. Meanwhile, a multitude of works has been published that extend the algorithm to a deep learning setting [57] or that address local minima issues faced by the algorithm. Hierarchical DPMM algorithms have been studied [64, 10] and may be invaluable for community detection in analytical settings. Our clustering implementation can be further improved by exploring the effectiveness of different priors and introducing new split/merge proposal methods. Finally, we note that the detected communities are mainly dictated by the structure of node embeddings. Introducing a control parameter to bias communities towards temporal and topological communities would improve ergonomics of community detection when reusing the learned embeddings.

9. Conclusion

In this paper, we introduce the MGTCOM framework for community detection in multimodal graphs. It utilizes meta-topological, topological, content features, and temporal information to detect communities. Moreover, we address common issues in multimodal graphs such as information incompleteness, and inference on unseen data by adopting a graph convolutional network architecture that combines k-hop neighborhood sampling and random walk context sampling. We devise a unified objective and an efficient temporal sampling method to learn multimodal community-aware node embeddings in an unsupervised manner. Consequently, we leverage a split/merge-based Dirichlet process mixture model for community detection where the number of communities are not known a priori. Our empirical evaluation shows that MGTCOM is quite competitive with the state-of-the-art.

A. Supplemental Material

A.1. Dataset construction

The datasets IMDB, DBLP and ICEWS were preprocessed into multimodel networks for evaluation of our model. In the following sections, we discuss the construction of these multimodal networks.

A.1.1. IMDB

IMDB dataset is originally made up of rows detailing movies and their information. To construct the multimodal we normalize this dataset by splitting listed actors, directors and genres per movie as a separate entity. Actors and directors are merged as person entity since both sets overlap. Each movie is associated with a set of keywords. By collecting these words into a vocabulary of 80 most frequent keywords, we construct an 80-dimensional one-hot feature vector for each movie. The Genre and Actor entities have no feature representations. Each movie is given $[t_v, \infty]$ time range, where t_v is the release data of the respective movie. Edges are constructed as ”person directed movie”, ”person acted in movie”, and ”movie has genre” pairs.

A.1.2. DBLP

The DBLP dataset is constructed in a similar way as IMDB since the dataset consists of Papers with their respective citations, authors, and venues. To construct the representation vector for each paper, we use pre-trained sentence transformers [73] to embed the abstract text. Authors and venues have no features. Each paper is given $[t_v, \infty]$ time range, where t_v is the publication date of the respective movie. Edges are constructed as ”paper cites paper”, ”author wrote paper” and, ”paper was published in venue” entity pairs.

A.1.3. ICEWS

The ICEWS datasets consists of triplets between subject, predicate, and object entities. As ICEWS is a temporal knowledge graph, each triplet is associated with a timestamp. We model this dataset by combining subject and object into one single entity type. Between these entities, we create typed edges corresponding to the respective predicate. The name of each entity is embedded into a feature vector using a pre-trained language transformer [73]. Each edge is given $[t_v, t_v]$ time range, where t_v represents the timestamp when the corresponding triplet was valid.

A.2. Model parameter count analysis

In Eq. (A.1) till Eq. (A.8) we provide a set of equations to calculate the exact number of trainable parameters for the *MGTCOM* model. We use parameters $d^{(l)}$ and h to denote latent vector dimensionality at layer l and number of attention heads respectively. The largest number of parameters come from the Heterogeneous graph transformer layer consisting of four node type specific transformation layers (A, Q, K, V) layers and three relation type specific transformation layers (A, P, M). The attention Aux-Emb and Feat-Lin are dependent on dataset properties such as node count $|\mathcal{V}|$ and the input feature dimensionality $d^{\mathcal{X}(t)}$. Finally, we include cluster parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ as learnable parameters, though not through gradient descent.

$$\text{Node-Lin}^{(l)} = \sum_{t \in \mathcal{A}} d^{(l-1)} d^{(l)} + d^{(l)} \tag{A.1}$$

$$\text{Rel-Lin}^{(l)} = \sum_{t \in \mathcal{R}} h^l (d^{(l)}/h^l)^2 + d^{(l)} \tag{A.2}$$

$$\text{HGT}^{(l)} = 4 \cdot \text{Node-Lin}^{(l)} + 3 \cdot \text{Rel-Lin}^{(l)} \tag{A.3}$$

$$\text{Aux-Emb} = d^{(0)} |\mathcal{V}| \cdot \text{embed-ratio} \tag{A.4}$$

$$\text{Att-Lin} = (d^{(L)} d^{(L)} + d^{(L)}) h \tag{A.5}$$

$$\text{Feat-Lin} = \sum_{t \in \mathcal{A}} d^{\mathcal{X}(t)} d^{(0)} + d^{(0)} \tag{A.6}$$

$$\text{Clus} = \sum_{i=0}^{K} d^{(L)} + d^{(L)} d^{(L)} \tag{A.7}$$

$$MGTCOM = \text{Aux-Emb} + \text{Att-Lin} + \text{Feat-Lin} + \text{Clus} + \sum_{l=1}^{L} \text{HGT}^{(l)} \tag{A.8}$$

A.3. Exact model parameters

Table A.1.: Complete overview of used parameters for *MGTCOM* and its variants for bench marking and tuning.

Config key	Value
loss	HINGE
hinge_margin	0.1
metric	DOTP
lr	0.02
repr_dim	64
lp_max_pairs	5000
ballroom_params/context_size	10
ballroom_params/num_neg_samples	1
ballroom_params/walk_length	20
ballroom_params/walks_per_node	10
n2v_params/context_size	10
n2v_params/num_neg_samples	1
n2v_params/p	1
n2v_params/q	0.5
n2v_params/walk_length	20
n2v_params/walks_per_node	10
embed_node_ratio	1.0
conv_num_heads	2
conv_num_layers	2
num_samples	[3, 2]
sampler_method	HGT
split_num_test	0.1
split_num_val	0.1
tempo_hidden_dim	32
tempo_repr_dim	64
tempo_weight	1.0
topo_hidden_dim	32
topo_repr_dim	64
topo_weight	1.0
use_tempo	True
use_topo	True
infer_k	3
prior_alpha	1.0
prior_kappa	1.0
prior_nu	65
prior_sigma_scale	0.1

Bibliography

[1] IMDB 5000 Movie Dataset. https://kaggle.com/carolzhangdc/imdb-5000-movie-dataset.

[2] Mihael Ankerst, Markus M. Breunig, Hans-peter Kriegel, and Jörg Sander. OPTICS: Ordering Points To Identify the Clustering Structure. pages 49–60. ACM Press, 1999.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, 2006.

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

[5] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral Networks and Locally Connected Networks on Graphs, May 2014.

[7] Jinxin Cao, Di Jin, Liang Yang, and Jianwu Dang. Incorporating network structure with node contents for community detection on large networks using deep learning. *Neurocomputing*, 297:71–81, July 2018.

[8] Yuwei Cao, Hao Peng, Jia Wu, Yingtong Dou, Jianxin Li, and Philip Yu. Knowledge-Preserving Incremental Social Event Detection via Heterogeneous GNNs. pages 3383–3395, April 2021.

[9] Sandro Cavallari, Vincent W. Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. Learning Community Embedding with Community Detection and Node Embedding on Graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 377–386, Singapore Singapore, November 2017. ACM.

[10] Jason Chang. Sampling in Computer Vision and Bayesian Nonparametric Mixtures. page 241.

[11] Jason Chang and John W Fisher III. Parallel Sampling of DP Mixture Models using Sub-Cluster Splits. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[12] Weijian Chen, Fuli Feng, Qifan Wang, Xiangnan He, Chonggang Song, Guohui Ling, and Yongdong Zhang. CatGCN: Graph Convolutional Networks with Categorical Node Features. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021.

[13] Jun Jin Choong, Xin Liu, and Tsuyoshi Murata. Learning Community Structure with Variational Autoencoder. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 69–78, November 2018.

[14] Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. HyTE: Hyperplane-based Temporally aware Knowledge Graph Embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2001–2011, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[15] CSIRO's Data61. Stargazers · stellargraph/stellargraph. https://github.com/stellargraph/stellargraph, 2018.

[16] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. Metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 135–144, New York, NY, USA, August 2017. Association for Computing Machinery.

[17] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231, Portland, Oregon, August 1996. AAAI Press.

[18] Hossein Fani, Eric Jiang, Ebrahim Bagheri, Feras Al-Obeidat, Weichang Du, and Mehdi Kargar. User community detection via embedding of social network structure and temporal content. *Information Processing & Management*, 57(2):102056, March 2020.

[19] Illés Farkas, Dániel Ábel, Gergely Palla, and Tamás Vicsek. Weighted network modules. *New Journal of Physics*, 9(6):180–180, June 2007.

[20] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, February 2010.

[21] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. HIN2Vec: Explore Meta-paths in Heterogeneous Information Networks for Representation Learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1797–1806, New York, NY, USA, November 2017. Association for Computing Machinery.

[22] Alberto García-Durán, Sebastijan Dumancic, and Mathias Niepert. Learning Sequence Encoders for Temporal Knowledge Graph Completion. pages 4816–4821, January 2018.

[23] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, June 2002.

[24] Marko Gosak, Rene Markovič, Jurij Dolenšek, Marjan Slak Rupnik, Marko Marhl, Andraž Stožer, and Matjaž Perc. Network science of biological systems at different scales: A review. *Physics of Life Reviews*, 24:118–135, March 2018.

[25] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. Dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, 187:104816, January 2020.

[26] Derek Greene, Dónal Doyle, and Pádraig Cunningham. Tracking the Evolution of Communities in Dynamic Social Networks. In *2010 International Conference on Advances in Social Networks Analysis and Mining*, pages 176–183, August 2010.

[27] Aditya Grover and Jure Leskovec. Node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 855–864, New York, NY, USA, August 2016. Association for Computing Machinery.

[28] Loni Hagen, Thomas Keller, Stephen Neely, Nic DePaula, and Claudia Robert-Cooperman. Crisis Communications in the Age of Social Media: A Network Analysis of Zika-Related Tweets. *Social Science Computer Review*, 36(5):523–541, October 2018.

[29] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 1025–1035, Red Hook, NY, USA, December 2017. Curran Associates Inc.

[30] Xiaofei He, Deng Cai, Yuanlong Shao, Hujun Bao, and Jiawei Han. Laplacian Regularized Gaussian Mixture Model for Data Clustering. *IEEE Trans. Knowl. Data Eng.*, 23:1406–1418, September 2011.

[31] Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs), July 2020.

[32] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous Graph Transformer. In *Proceedings of The Web Conference 2020*, pages 2704–2710, Taipei Taiwan, April 2020. ACM.

[33] Mingqing Huang, Qingshan Jiang, Qiang Qu, Lifei Chen, and Hui Chen. Information fusion oriented heterogeneous social network for friend recommendation via community detection. *Applied Soft Computing*, 114:108103, January 2022.

[34] Yuting Jia, Qinqin Zhang, Weinan Zhang, and Xinbing Wang. CommunityGAN: Community Detection with Generative Adversarial Nets. In *The World Wide Web Conference*, pages 784–794, San Francisco CA USA, May 2019. ACM.

[35] Yoonsuk Kang, Jun-Seok Lee, Won-Yong Shin, and Sang-Wook Kim. Community reinforcement: An effective and efficient preprocessing method for accurate community detection. *Knowledge-Based Systems*, page 107741, November 2021.

[36] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation Learning for Dynamic Graphs: A Survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020.

[37] Thomas N. Kipf and Max Welling. Variational Graph Auto-Encoders. *arXiv:1611.07308 [cs, stat]*, November 2016.

[38] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]*, February 2017.

[39] Mark Kozdoba and Shie Mannor. Community Detection via Measure Space Embedding. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[40] Jussi M. Kumpula, Mikko Kivelä, Kimmo Kaski, and Jari Saramäki. Sequential algorithm for fast clique percolation. *Physical Review E*, 78(2):026109, August 2008.

[41] Ye Li, Chaofeng Sha, Xin Huang, and Yanchun Zhang. Community Detection in Attributed Graphs: An Embedding Approach. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), April 2018.

[42] Hongtao Liu, Hui Chen, Mao Lin, and Yu Wu. Community Detection Based on Topic Distance in Social Tagging Networks. *Indonesian Journal of Electrical Engineering and Computer Science*, 12(5):4038–4049, May 2014.

[43] Linhao Luo, Yixiang Fang, Xin Cao, Xiaofeng Zhang, and Wenjie Zhang. Detecting Communities from Heterogeneous Graphs: A Context Path-based Graph Neural Network Model. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1170–1180. Association for Computing Machinery, New York, NY, USA, October 2021.

[44] Sedigheh Mahdavi, Shima Khoshraftar, and Aijun An. Dynamic Joint Variational Graph Autoencoders. In Peggy Cellier and Kurt Driessens, editors, *Machine Learning and Knowledge Discovery in Databases*, Communications in Computer and Information Science, pages 385–401, Cham, 2020. Springer International Publishing.

[45] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, 3(2):127–163, July 2000.

[46] Miller Mcpherson, Lynn Smith-Lovin, and James Cook. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology*, 27:415, January 2001.

[47] Nikhil Mehta, Lawrence Carin Duke, and Piyush Rai. Stochastic Blockmodels meet Graph Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4466–4474. PMLR, May 2019.

[48] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. *arXiv:1310.4546 [cs, stat]*, October 2013.

[49] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture. *IEEE Access*, 6:39501–39514, 2018.

[50] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics*, 69(6 Pt 2):066133, June 2004.

[51] Giang Hoang Nguyen, John Boaz Lee, Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-Time Dynamic Network Embeddings. In *Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18*, pages 969–976, Lyon, France, 2018. ACM Press.

[52] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 2609–2615, Stockholm, Sweden, July 2018. AAAI Press.

[53] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:5363–5370, April 2020.

[54] Leto Peel, Daniel B. Larremore, and Aaron Clauset. The ground truth about metadata and community detection in networks. *Science Advances*, 3(5):e1602548, 2017.

[55] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, August 2014. Association for Computing Machinery.

[56] Stephen D. Pryke. Analysing construction project coalitions: Exploring the application of social network analysis. *Construction Management and Economics*, 22(8):787–797, October 2004.

[57] Meitar Ronen, Shahaf E. Finder, and Oren Freifeld. DeepDPM: Deep Clustering With an Unknown Number of Clusters. *arXiv:2203.14309 [cs, stat]*, March 2022.

[58] Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. GEMSEC: Graph embedding with self clustering. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '19, pages 65–72, New York, NY, USA, August 2019. Association for Computing Machinery.

[59] Philipp Schuetz and Amedeo Caflisch. Multistep greedy algorithm identifies community structure in real-world and computer-generated networks. *Physical Review E*, 78(2):026112, August 2008.

[60] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective Classification in Network Data. *AI Magazine*, 29(3):93–93, September 2008.

[61] Xing Su, Shan Xue, Fanzhen Liu, Jia Wu, Jian Yang, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Di Jin, Quan Z. Sheng, and Philip S. Yu. A Comprehensive Survey on Community Detection With Deep Learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, 2022.

[62] Yizhou Sun, Charu Aggarwal, and Jiawei Han. Relation Strength-Aware Clustering of Heterogeneous Information Networks with Incomplete Attributes. *Proceedings of the VLDB Endowment*, 5, January 2012.

[63] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1067–1077, Republic and Canton of Geneva, CHE, May 2015. International World Wide Web Conferences Steering Committee.

[64] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101(476):1566–1581, December 2006.

[65] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning Deep Representations for Graph Clustering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1), June 2014.

[66] Stefano F. Tonellato. Bayesian nonparametric clustering as a community detection problem. *Computational Statistics & Data Analysis*, 152:107044, December 2020.

[67] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[68] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural Deep Network Embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1225–1234, San Francisco California USA, August 2016. ACM.

[69] Fei Wang, Tao Li, Xin Wang, Shenghuo Zhu, and Chris Ding. Community discovery using nonnegative matrix factorization. *Data Min. Knowl. Discov.*, 22:493–521, May 2011.

[70] Hongwei Wang and Jure Leskovec. Combining Graph Convolutional Neural Networks and Label Propagation. *ACM Transactions on Information Systems*, 40(4):73:1–73:27, November 2021.

[71] Peizhuo Wang, Lin Gao, and Xiaoke Ma. Dynamic community detection based on network structural perturbation and topological similarity. *Journal of Statistical Mechanics: Theory and Experiment*, 2017(1):013401, January 2017.

[72] Shihan Wang, Marijn Schraagen, Erik Tjong Kim Sang, and Mehdi Dastani. Public Sentiment on Governmental COVID-19 Measures in Dutch Social Media. In *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*, Online, December 2020. Association for Computational Linguistics.

[73] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers, April 2020.

[74] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community Preserving Network Embedding. page 7.

[75] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous Graph Attention Network. In *The World Wide Web Conference*, WWW '19, pages 2022–2032, New York, NY, USA, May 2019. Association for Computing Machinery.

[76] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1112–1119, Québec City, Québec, Canada, July 2014. AAAI Press.

[77] Chengyuan Wu and Carol Hargreaves. *Topological Machine Learning for Mixed Numeric and Categorical Data*. March 2020.

[78] Jiaming Wu, Meng Liu, Jiangting Fan, Yong Liu, and Meng Han. SageDy: A Novel Sampling and Aggregating Based Representation Learning Approach for Dynamic Networks. In Igor Farkaš, Paolo Masulli, Sebastian Otte, and Stefan Wermter, editors, *Artificial Neural Networks and Machine Learning – ICANN 2021*, Lecture Notes in Computer Science, pages 3–15, Cham, 2021. Springer International Publishing.

[79] Xiaodong Wu, Weizhe Lin, Zhilin Wang, and Elena Rastorgueva. Author2Vec: A Framework for Generating User Embedding. *arXiv:2003.11627 [cs, stat]*, March 2020.

[80] Jierui Xie, Stephen Kelley, and Boleslaw K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys*, 45(4):43:1–43:35, August 2013.

[81] Shan Xue, Jie Lu, and Guangquan Zhang. Cross-domain network representations. *Pattern Recognition*, 94:135–148, October 2019.

[82] Carl Yang, Mengxiong Liu, Zongyi Wang, Liyuan Liu, and Jiawei Han. Graph Clustering with Dynamic Embedding. *arXiv:1712.08249 [physics]*, December 2017.

[83] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. Heterogeneous Network Representation Learning: A Unified Framework with Survey and Benchmark. *IEEE Transactions on Knowledge and Data Engineering*, PP:1–1, December 2020.

[84] Jaewon Yang and Jure Leskovec. Community-Affiliation Graph Model for Overlapping Network Community Detection. In *2012 IEEE 12th International Conference on Data Mining*, pages 1170–1175, December 2012.

[85] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, MDS '12, pages 1–8, New York, NY, USA, August 2012. Association for Computing Machinery.

[86] Liang Yang, Xiaochun Cao, Dongxiao He, Chuan Wang, Xiao Wang, and Weixiong Zhang. Modularity based community detection with deep learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2252–2258, New York, New York, USA, July 2016. AAAI Press.

[87] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 40–48, New York, NY, USA, June 2016. JMLR.org.

[88] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 974–983, New York, NY, USA, July 2018. Association for Computing Machinery.

[89] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An efficient data clustering method for very large databases. *ACM SIGMOD Record*, 25(2):103–114, June 1996.

[90] Tianqi Zhang, Yun Xiong, Jiawei Zhang, Yao Zhang, Yizhu Jiao, and Yangyong Zhu. CommDGI: Community Detection Oriented Deep Graph Infomax. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, pages 1843–1852, New York, NY, USA, October 2020. Association for Computing Machinery.

[91] Kai Zhao, Ting Bai, Bin Wu, Bai Wang, Youjie Zhang, Yuanyu Yang, and Jian-Yun Nie. Deep Adversarial Completion for Sparse Heterogeneous Information Network Embedding. In *Proceedings of The Web Conference 2020*, pages 508–518. Association for Computing Machinery, New York, NY, USA, April 2020.

[92] Yifeng Zhao, Xiangwei Wang, Hongxia Yang, Le Song, and Jie Tang. Large Scale Evolving Graphs with Burst Detection. pages 4412–4418, 2019.

[93] Yujing Zhou, Weile Liu, Yang Pei, Lei Wang, Daren Zha, and Tianshu Fu. Dynamic Network Embedding by Semantic Evolution. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2019.

[94] Jiong Zhu, Mark Heimann, Yujun Yan, Lingxiao Zhao, Leman Akoglu, and Danai Koutra. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. page 12.

[95] Ruimin Zhu and Wenxin Jiang. Combining Random Walks and Nonparametric Bayesian Topic Model for Community Detection, August 2016.

[96] Ruimin Zhu and Wenxin Jiang. Bayesian Complex Network Community Detection Using Nonparametric Topic Model. In Luca Maria Aiello, Chantal Cherifi, Hocine Cherifi, Renaud Lambiotte, Pietro Lió, and Luis M. Rocha, editors, *Complex Networks and Their Applications VII*, Studies in Computational Intelligence, pages 280–291, Cham, 2019. Springer International Publishing.