

Article

DAETG: Dual-Stage Attention Embedding for Community Detection on Textual-Edge Graphs

Yuchen Liu ^{1,*}  and Hongchang Chen ²¹ Institute of Information Technology, Information Engineering University, Zhengzhou 450002, China² National Digital Switching System Engineering and Technological R&D Center, Zhengzhou 450002, China; ndscchc@139.com

* Correspondence: liuyuchen2202@163.com

Abstract

Textual-edge graphs represent interactions through both connectivity and relation-specific text. Learning community structure from these graphs requires an encoder that preserves the relevance of individual relations when combining semantic and structural information. We propose Dual-Stage Attention Embedding for Community Detection on Textual-Edge Graphs (DAETG). The first attention stage summarizes incident-edge text, and the second transfers its coefficients into neighboring-node aggregation as a learnable prior. Following topology-reconstruction pretraining, Kullback–Leibler self-training and differentiable modularity jointly refine the encoder and a centroid-based community assignment. Training uses fixed node and edge text embeddings and a supplied community count, without category-membership supervision. Experiments on four benchmark graphs with base and large Bidirectional Encoder Representations from Transformers (BERT) embeddings show competitive community-detection performance, with modest mean improvements that vary across datasets and evaluation metrics. Ablation studies on two graphs support the contributions of correctly associated edge text, the dual-stage encoder, and joint community refinement. Together, these results support using relation-specific text alongside topology to learn community representations.

Keywords: community detection; textual-edge graph; graph attention network; network embedding; unsupervised learning



Academic Editor: Ferdinando Di Martino

Received: 23 September 2026

Revised: 6 October 2026

Accepted: 7 October 2026

Published: 10 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Community detection identifies groups of related entities in networks, supporting the analysis of scientific literature, online interactions, and recommendation data. Structural methods infer these groups from connectivity, often by seeking more within-community edges than expected under a network null model [1]. In text-rich networks, however, connectivity captures only part of the information relevant to grouping nodes. Node documents describe entities, whereas reviews, citation contexts, and messages describe individual relations between them. Textual-edge graphs (TEGs) retain both forms of text alongside the graph structure [2].

Edge text distinguishes relations that share the same unweighted topological form, as illustrated by the citation contexts in Figure 1. Citations surrounding Paper C may refer to adopting a method, comparing alternatives, or evaluating a result. A binary adjacency entry records each citation identically, although the contexts express different relationships. When communities reflect shared research interests, these relationships need

not provide equally informative evidence of membership. Edge text therefore offers a basis for weighting neighboring information beyond the presence of a link or the similarity of node documents.

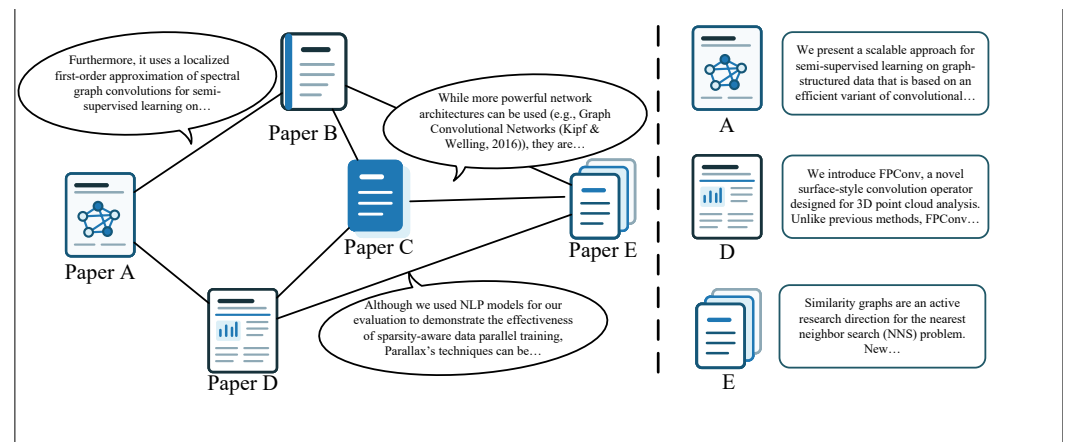


Figure 1. Illustration of a textual-edge citation graph. Papers form the nodes, while citation contexts provide relation-specific text on the edges. The contexts distinguish method adoption, comparison, and evaluation even when these relations have the same unweighted topological form. The corresponding node documents and their text are shown on the right.

Attributed graph clustering addresses the complementary problem of aligning node representations with a clustering objective. Deep Attentional Embedded Graph Clustering (DAEGC) combines graph-attention autoencoding with assignment self-training, while Community Detection Based on Unsupervised Attributed Network Embedding (CDBNE) adds a modularity-based community criterion [3,4]. Deep Modularity Networks (DMoN) directly optimize a differentiable modularity objective for attributed graph clustering [5]. These methods learn partitions from node attributes and connectivity. Their formulations, however, do not explicitly represent the text of individual relations as a separate input.

Relation information enters community models through joint statistical inference or neural message passing. Statistical models infer communities and patterns in textual or vector-valued edge content [6–9]. Edge-aware graph neural networks (GNNs) instead use edge features within message passing, including alternating node and edge attention [10]. Edgeformers further contextualizes edge text with endpoint representations and aggregates textual relations for node representation learning [11].

The question is how to preserve each relation's importance when moving from edge-text aggregation to neighboring-node aggregation. Summarizing incident edge texts into a node state supplies semantic context, but does not separately retain each edge's contribution for subsequent neighbor weighting. We connect these decisions by reusing the edge-text attention coefficient as a prior in the next attention stage. Training this encoder with a community objective allows the partition to influence both the semantic summary and its use in message passing.

We propose Dual-Stage Attention Embedding for Community Detection on Textual-Edge Graphs (DAETG). The first stage forms an edge-enriched node state and retains normalized edge-text attention coefficients. The second stage combines these coefficients with neighboring-node compatibility through a learnable transfer strength. After topology-reconstruction pretraining, a shared centroid-based assignment is refined by Kullback–Leibler (KL) self-training and modularity. DAETG learns a partition without category-membership supervision, conditional on a supplied community count; in the experiments, this count is obtained from the reference categories. The contributions are as follows:

- We develop a dual-stage encoder that explicitly transfers edge-text attention coefficients into node-neighbor attention, retaining relation-specific weighting across the two aggregation stages.
- We couple the encoder to a shared centroid-based community assignment using topology and fixed node- and edge-text embeddings. Self-training and modularity refine the assignment used for final inference.
- We evaluate the framework on four graphs from the textual-edge graph dataset and benchmark (TEG-DB) with two text-embedding settings and five seeds. Comparisons with a shared community head isolate encoder differences, while edge-text, transfer, and objective controls examine how relation semantics contribute to category alignment.

2. Related Work

2.1. Structure-Based Community Detection

Structure-based methods identify communities from network connectivity. Modularity compares within-community connectivity with the expectation under a degree-preserving null model [1]. Louvain optimizes this criterion through local node moves and successive network aggregation [12]. Leiden introduces a refinement step that provides connectedness guarantees for the resulting communities [13]. DAETG retains modularity as its structural criterion and learns representations from both node and edge text vectors.

2.2. Deep Clustering of Node-Attributed Graphs

Deep clustering jointly learns representations and cluster assignments. Deep embedded clustering (DEC) introduced a soft Student- t assignment and a sharpened target distribution for this purpose [14]. DAEGC adapts this formulation to attributed graphs through graph-attention autoencoding [3]. CDBNE combines reconstruction, assignment self-training, and modularity maximization to learn community-oriented node representations [4]. DMoN takes a different route, learning soft assignments through a differentiable modularity objective with a regularizer against degenerate solutions [5]. These methods establish assignment self-training and differentiable modularity as objectives for graph clustering.

Recent methods refine how attribute and structural information interact. The deep dual graph attention auto-encoder (DDGAE) learns from attribute and higher-order modularity views, with self-training guiding community assignment [15]. BCDAN, a balanced method for community detection in attributed networks, balances attribute and topology representations through separate encoders and their fusion [16]. Dual-view entropy-regularized nonnegative matrix factorization (DV-ERNMF) couples structural factorization with a learned attribute-similarity view through entropy regularization [17]. These methods learn from node attributes and community structure. DAETG also encodes relation text and carries its attention weights into neighbor aggregation. The resulting representations determine the centroid-based assignment used by both community losses and final inference.

2.3. Relation Text and Edge-Aware Graph Representation

Probabilistic models connect relation content with community structure through joint inference. Qi et al. jointly model social links and content attached to edges [6]. The stochastic topic block model (STBM) associates each pair of node groups with a distribution over topics in their exchanged texts [7]. Embedded Topics in the Stochastic Block Model (ETSBM) incorporates word embeddings into this joint modeling approach [18]. The deep latent position topic model (Deep-LPTM) combines a variational graph autoencoder with a probabilistic topic model to learn node groups and representations of textual interactions [8].

For vector-valued edge covariates, the stochastic block model with vectorial edge covariates (VEC-SBM) studies community recovery through iterative refinement and establishes conditions under which edge information improves recovery [9].

Edge-aware GNNs provide a complementary, message-passing formulation. Edge-conditioned convolution generates a neighborhood filter from each edge label [19], while the graph isomorphism network with edge features (GINE) adds an edge representation to the neighboring-node message [20]. The edge-enhanced graph neural network (EGNN) propagates multidimensional edge information across layers [21]; convolution with edge-node switching (CensNet) switches between node and edge convolutions using the line-graph structure [22]. Attention-based methods also jointly process both types of information. The model incorporating node and edge features in graph neural networks (NENN) alternates node-level and edge-level attention layers [10], and the edge-featured graph attention network (EGAT) updates node and edge states in parallel through mutually dependent operations [23]. Building on these established node–edge attention designs, DAETG retains the first-stage edge coefficient and inserts its logarithm into the second-stage neighbor-attention logit.

Textual-graph encoders additionally model language within its graph context. GraphFormers interleaves node-text encoding with neighboring-node aggregation [24]. Edgeformers incorporates endpoint states when encoding edge text and aggregates edge representations within a node’s ego graph [11]. The Link2Doc preprint constructs documents from local relations and transfers language-model information to a GNN for link prediction [25]. TEG-DB supplies common datasets and text embeddings for evaluating such representations in node classification and link prediction [2]. More recently, Sruthi et al. combine text embeddings and GNNs to assign semantic categories to nodes in text-rich heterogeneous networks [26]. DAETG addresses unsupervised partition learning with fixed text embeddings and a supplied K , using explicit coefficient transfer to connect relation-aware aggregation with community optimization.

3. Materials and Methods

DAETG uses edge-text attention both to summarize incident relation semantics and to weight neighboring-node aggregation. Topology reconstruction initializes this dual-stage encoder before self-training and modularity optimization refine a shared community assignment.

3.1. Problem Definition

The input graph associates a text embedding vector with every node and every observed edge. The setting follows textual-edge graph benchmarks in which text encoders are applied before graph learning and the resulting vectors are provided as graph attributes [2].

Definition 1 (Textual-Edge Graph). *Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected graph with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}| > 0$ distinct, unordered edges. Its topology is represented by a symmetric adjacency matrix $A \in \{0, 1\}^{n \times n}$ without self-loops. The node-vector matrix is $X = [x_1^\top; \dots; x_n^\top] \in \mathbb{R}^{n \times d_x}$, and the edge-text vector matrix is $T \in \mathbb{R}^{m \times d_t}$. For an observed edge, $(i, j) \in \mathcal{E}$, $t_{ij} = t_{ji}$ denotes its text vector, whose transpose is stored as a row of T . The graph model treats the text-embedding matrices X and T as fixed input attributes.*

Definition 2 (Community Assignment). *For a given number of communities K , a soft assignment is a row-stochastic matrix $Q = [q_{ik}] \in [0, 1]^{n \times K}$ satisfying $\sum_{k=1}^K q_{ik} = 1$. The associated non-overlapping partition is $\hat{y}_i = \arg \max_k q_{ik}$ for node i .*

Problem 1 (Community Detection). Given (A, X, T) and K , textual-edge community detection learns node embeddings $Z \in \mathbb{R}^{n \times d_z}$ and a soft assignment Q without using category memberships in the optimization objective. The count K is an input; the reference- K evaluation protocol is specified in Section 4.3. The learned partition should reflect both graph connectivity and the relation semantics carried by the edge vectors.

The prescribed count K determines the number of trainable centroids and the number of columns in Q . K-means initializes these centroids after pretraining; the Student- t assignment then computes Q from the embeddings and centroids. Category memberships do not supervise these operations. In applications where K is unknown, a separate procedure must select K before DAETG is applied.

Individual feature and state vectors are column vectors; their transposes form the rows of the corresponding matrices. The neighborhood of node i is $\mathcal{N}_i = \{j \mid A_{ij} = 1\}$, and $d_i = \sum_j A_{ij}$ denotes its degree. The attention equations below assume $d_i > 0$. If an isolated node is retained, DAETG sets its edge context to zero and skips neighbor aggregation, giving $z_i = u_i$. Self-information enters through residual connections, so neither attention stage adds self-loops.

3.2. Overall Framework

DAETG aggregates edge text before aggregating neighboring-node states. The first stage produces an edge-enriched state u_i and retains the edge-semantic prior α_{ij} . The second stage combines this prior with neighboring-node compatibility to produce Z . Topology reconstruction initializes the encoder before a single centroid-based assignment Q is refined by self-training and soft modularity. Figure 2 shows the encoder and its two community objectives.

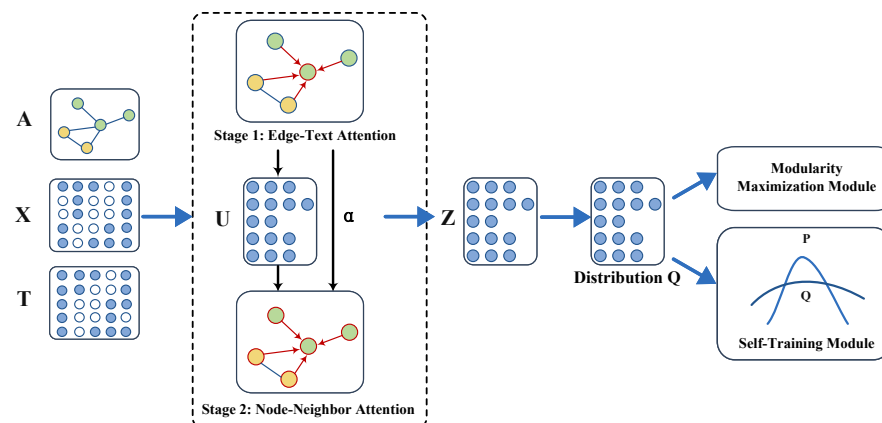


Figure 2. Overview of DAETG. The inputs are adjacency A , node vectors X , and edge-text vectors T . Edge-Text Attention produces states U and edge-semantic prior α , which Node-Neighbor Attention uses to obtain Z . The resulting Q is shared by modularity and self-training with target P .

Topology reconstruction is used only before community optimization. Joint training then updates the encoder and centroids, so the final Z differs from the embedding used to initialize K-means.

Table 1 summarizes the notation used in the method.

Table 1. Main notation.

Symbol	Description
A	adjacency matrix of the undirected graph
X	node text-embedding matrix
T	edge text-embedding matrix
U	edge-enriched node states
Z	final node-embedding matrix
α_{ij}	incident-edge attention weight used as a semantic prior
β_{ij}	node-neighbor attention coefficient
Q	soft community assignment induced by Z and μ
P	sharpened, stop-gradient target distribution
μ_k	centroid of community k
K	prescribed number of communities

3.3. Dual-Stage Attention Encoder

The encoder first scores incident edge text, then combines the resulting context and edge-semantic prior with neighboring node states.

3.3.1. Node Initialization

The node vectors are mapped to the node-state dimension before graph propagation. For each node i , the initial state is given by Equation (1)

$$h_i = \phi(W_0 x_i + b_0), \quad (1)$$

where $W_0 \in \mathbb{R}^{d_z \times d_x}$ and $b_0 \in \mathbb{R}^{d_z}$ are trainable, $\phi(\cdot)$ is an elementwise nonlinear activation, and d_z is the node-state dimension.

3.3.2. Stage 1: Edge-Text Attention

The first stage assigns a semantic weight to each edge within a node's neighborhood. For edge (i, j) incident to node i , the unnormalized score is given by Equation (2)

$$\begin{aligned} \zeta_{ij} &= [W_h h_i \parallel W_t t_{ij}], \\ s_{ij} &= \text{LeakyReLU}(a_e^\top \zeta_{ij}). \end{aligned} \quad (2)$$

Here, $W_h \in \mathbb{R}^{d_a \times d_z}$ and $W_t \in \mathbb{R}^{d_a \times d_t}$ project into an attention space of dimension d_a . The symbol $\parallel$ denotes concatenation, and $a_e \in \mathbb{R}^{2d_a}$ is a trainable scoring vector. Equations (3) and (4) give the normalized log-attention and its probability

$$\log \alpha_{ij} = s_{ij} - \log \sum_{r \in \mathcal{N}_i} s_{ir}, \quad (3)$$

$$\alpha_{ij} = \exp(\log \alpha_{ij}). \quad (4)$$

The log form provides a numerically stable prior for the next-stage compatibility logit. Stage 1 uses the receiving state h_i and edge vector t_{ij} to score each incident relation. Stage 2 introduces the neighbor state through a pair-specific query–key dot product.

Attention weights are receiver-specific even though the input graph is undirected. Although $t_{ij} = t_{ji}$, the endpoint state and local normalization can give $\alpha_{ij} \neq \alpha_{ji}$. For example, if all incident-edge scores are equal, then $\alpha_{ij} = 1/d_i$ and $\alpha_{ji} = 1/d_j$. Different endpoint degrees already produce asymmetric aggregation weights on an undirected edge. These coefficients govern conditional aggregation, while the community objective retains the original symmetric A and its degree-based null model; neither α nor β replaces its entries. Symmetrizing the first-stage weights would replace receiver-specific normalization with a different aggregation rule.

Equations (5) and (6) summarize the relation context around node i :

$$c_i = W_v \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} t_{ij} \right), \quad (5)$$

$$u_i = \text{LN}(h_i + c_i). \quad (6)$$

$W_v \in \mathbb{R}^{d_z \times d_i}$ maps edge vectors to the node-state dimension, and LN denotes layer normalization. Each message contains its own t_{ij} : linearity gives the equivalent reduction $c_i = \sum_{j \in \mathcal{N}_i} \alpha_{ij} (W_v t_{ij})$. The first-stage output is $U = [u_1^\top; \dots; u_n^\top] \in \mathbb{R}^{n \times d_z}$.

3.3.3. Stage 2: Node-Neighbor Attention

The second stage evaluates each neighbor using node compatibility and the semantic prior obtained from the same edge. Its attention logit is given by Equation (7)

$$r_{ij} = \frac{(W_q u_i)^\top (W_k u_j)}{\sqrt{d_a}} + \gamma \log \alpha_{ij}, \quad (7)$$

where $W_q, W_k \in \mathbb{R}^{d_a \times d_z}$ are trainable query and key projections. A scalar transfer strength shared across edges is parameterized as $\gamma = \text{sigmoid}(\tilde{\gamma}) \in (0, 1)$ and is initialized at $\gamma = 0.5$, corresponding to $\tilde{\gamma} = 0$. It is learned during both pretraining and community refinement. The normalized neighbor coefficient is given by Equation (8)

$$\beta_{ij} = \frac{\exp(r_{ij})}{\sum_{r \in \mathcal{N}_i} \exp(r_{ir})}. \quad (8)$$

The node update in Equation (9) aggregates edge-conditioned neighbor messages through a residual connection:

$$z_i = \text{LN} \left[u_i + \phi \left(\sum_{j \in \mathcal{N}_i} \beta_{ij} W_n u_j \right) \right]. \quad (9)$$

$W_n \in \mathbb{R}^{d_z \times d_z}$ is a trainable projection of neighbor states. It acts on each u_j before the weighted reduction; the activation is applied to the resulting sum. The second-stage output is the final embedding matrix $Z = [z_1^\top; \dots; z_n^\top] \in \mathbb{R}^{n \times d_z}$.

The transferred prior couples the two attention stages directly. Combining Equations (7) and (8) yields Equation (10)

$$\beta_{ij} \propto \alpha_{ij}^\gamma \exp \left(\frac{(W_q u_i)^\top (W_k u_j)}{\sqrt{d_a}} \right). \quad (10)$$

The first-stage log-normalizer is constant across $\mathcal{N}_i$ and cancels in Equation (8). Adding $\gamma \log \alpha_{ij}$ is therefore equivalent to adding γs_{ij} before the second-stage softmax. The same learned edge scores influence both the semantic context and the neighbor weights. The relative weight of two neighbors depends on both their node compatibility and the ratio of their edge-semantic priors.

The transfer strength controls how strongly edge semantics influence neighbor weighting. For $0 < \gamma < 1$, exponentiation tempers the prior before combining it with node compatibility. The sigmoid parameterization restricts this component to a tempered prior and supplies the chain-rule factor $\partial \gamma / \partial \tilde{\gamma} = \gamma(1 - \gamma)$ during optimization.

The transfer controls fix or relax this parameterization. Explicitly fixing $\gamma = 0$ removes transfer while retaining the edge context in U ; it is not implemented through a finite

sigmoid parameter. Fixed-grid conditions set the effective value of γ directly, including the endpoints 0 and 1. The unconstrained variant instead optimizes $\gamma \in \mathbb{R}$ directly from the same initial value 0.5. Values above 1 sharpen the transferred prior, whereas negative values reverse its ordering.

3.4. Topology-Aware Encoder Pretraining

Topology reconstruction supplies the unsupervised objective used to pretrain the dual-stage encoder. The decoder scores an observed or sampled node pair using Equation (11)

$$\hat{A}_{ij} = \text{sigmoid}(z_i^\top z_j). \quad (11)$$

Let $\mathcal{E}^-$ contain unordered, unobserved, loop-free node pairs sampled at a fixed negative ratio $\rho > 0$, so $|\mathcal{E}^-| = \rho|\mathcal{E}|$. Each positive undirected edge is counted once. The pretraining loss is given by Equation (12)

$$\begin{aligned} \mathcal{L}_{\text{pre}} = & -\frac{1}{|\mathcal{E}|} \sum_{(i,j) \in \mathcal{E}} \log \hat{A}_{ij} \\ & -\frac{1}{|\mathcal{E}^-|} \sum_{(i,j) \in \mathcal{E}^-} \log(1 - \hat{A}_{ij}). \end{aligned} \quad (12)$$

Because $\hat{A}$ depends on Z , minimizing Equation (12) updates both attention stages through the path $\mathcal{L}_{\text{pre}} \rightarrow \hat{A} \rightarrow Z \rightarrow U$. The separate positive and negative averages give the two classes equal total weight, irrespective of the number of sampled negative pairs.

Community refinement uses the pretrained encoder without the reconstruction decoder. DAETG follows the initialization sequence of CDBNE [4], then removes reconstruction loss after centroid initialization.

3.5. Community-Oriented Self-Training

K-means is run once on the pretrained Z to initialize trainable centroids $\{\mu_k \in \mathbb{R}^{d_z}\}_{k=1}^K$. Each experimental run repeats this fit on its own pretrained embeddings, using its run seed s and the K-means++ configuration in Appendix A. The 20 restarts belong to this single fit; the five experimental runs include variation in both encoder initialization and centroid initialization.

The Student- t kernel with one degree of freedom converts distances from node i to the centroids into a soft assignment [14], as given in Equation (13):

$$q_{ik} = \frac{(1 + \|z_i - \mu_k\|_2^2)^{-1}}{\sum_{r=1}^K (1 + \|z_i - \mu_r\|_2^2)^{-1}}. \quad (13)$$

Both community losses act through Q , which also supplies the final labels.

The target distribution emphasizes confident assignments while compensating for cluster frequency. Let $f_k = \sum_i q_{ik}$. After initialization, DAETG refreshes the target before refinement steps $\tau, 2\tau, \dots$ using Equation (14)

$$\tilde{p}_{ik} = \frac{q_{ik}^2 / f_k}{\sum_{r=1}^K q_{ir}^2 / f_r}, \quad P \leftarrow \text{stopgrad}(\tilde{P}). \quad (14)$$

The stop-gradient operation holds P fixed during encoder and centroid updates. The self-training loss is given by Equation (15)

$$\mathcal{L}_{\text{KL}} = \frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K p_{ik} \log \frac{p_{ik}}{q_{ik}}. \quad (15)$$

The node-wise mean places self-training on a per-node scale before it is combined with normalized modularity.

Soft modularity adds a graph-level community preference to the same Q . Let $\mathbf{1} \in \mathbb{R}^n$ be the all-ones vector. Then, $\mathbf{d} = A\mathbf{1}$ is the degree vector, whose i th entry is $d_i = \sum_j A_{ij}$; because the graph is undirected, $2m = \sum_i d_i$. Following the degree-preserving null model of modularity [1], DAETG uses Equation (16)

$$\mathcal{M}(A, Q) = \frac{1}{2m} \text{Tr}(Q^\top A Q) - \frac{1}{(2m)^2} \|\mathbf{d}^\top Q\|_2^2. \quad (16)$$

The first term measures soft within-community connectivity; the second subtracts the expectation determined by community degree totals. For a one-hot assignment $q_{ik} = \mathbf{1}\{\hat{y}_i = k\}$, $\sum_k q_{ik}q_{jk} = \mathbf{1}\{\hat{y}_i = \hat{y}_j\}$. Expanding both terms gives the classical score in Equation (17):

$$\mathcal{M}(A, Q) = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{d_i d_j}{2m} \right) \mathbf{1}\{\hat{y}_i = \hat{y}_j\}. \quad (17)$$

In a sparse implementation, $\text{Tr}(Q^\top A Q) = 2 \sum_{(i,j) \in \mathcal{E}} q_i^\top q_j$, where q_i is node i 's assignment vector. This form counts each undirected edge twice in the adjacency product without constructing a dense modularity matrix. Equivalently, the first term is $m^{-1} \sum_{(i,j) \in \mathcal{E}} q_i^\top q_j$ when each unordered edge occurs once; the trace form in Equation (16) retains the factor $(2m)^{-1}$.

The normalization gives zero modularity to a hard partition with all nodes in one community: the two terms are $2m/(2m) = 1$ and $(2m)^2/(2m)^2 = 1$.

The joint objective in Equation (18) combines cluster sharpening and graph-level structural cohesion:

$$\mathcal{L}_{\text{joint}} = \mathcal{L}_{\text{KL}} - \lambda_M \mathcal{M}(A, Q). \quad (18)$$

Here, $\lambda_M \geq 0$ controls the structural contribution, and the negative sign makes minimization favor higher modularity. KL divergence sharpens confident assignments, whereas modularity rewards within-community connectivity above the degree-based null expectation. Because $Q = Q(Z, \mu)$, both terms jointly optimize the encoder and centroids used for final inference.

3.6. Optimization and Inference

The experimental protocol uses $S_{\text{pre}} = 200$ pretraining updates, $S_{\text{joint}} = 200$ refinement updates, and a target-update interval of $\tau = 5$ refinement updates.

At inference, Z provides continuous node representations and Q records soft membership scores. The non-overlapping label is $\hat{y}_i = \arg \max_k q_{ik}$, equivalently the nearest learned centroid under Equation (13); no additional K-means fit is required.

For fixed representation dimensions and negative-sampling ratio, an encoder and objective pass costs $O(n + m + nKd_z + mK)$ and is therefore linear in graph size when K is fixed. The four terms account for node projection, sparse dual-stage attention, Student- t assignment, and sparse modularity, respectively. The one-time K-means initialization costs $O(I_{\text{km}} nKd_z)$ for I_{km} total iterations across restarts. Because $|\mathcal{E}^-| = O(m)$, pretraining also avoids a dense $n \times n$ reconstruction target. Algorithm 1 summarizes training and inference; Appendix A.9 reports runtime and peak GPU memory comparisons.

Algorithm 1 Training and Inference of DAETG**Require:** Graph inputs A , X , and T ; community count K **Require:** Step counts S_{pre} and S_{joint} ; update interval τ **Ensure:** Z , Q , and labels $\hat{y}$

```

1: Initialize the dual-stage encoder parameters  $\theta$ 
2: for  $t = 1, \dots, S_{\text{pre}}$  do
3:    $Z \leftarrow f_{\theta}(A, X, T)$  using Equations (1)–(9)
4:   Update  $\theta$  by minimizing  $\mathcal{L}_{\text{pre}}$ 
5: Recompute  $Z \leftarrow f_{\theta}(A, X, T)$  using the pretrained parameters
6: Initialize  $\{\mu_k\}_{k=1}^K$  by K-means on  $Z$ 
7: Compute  $Q \leftarrow Q(Z, \mu)$  and  $P \leftarrow \text{stopgrad}(\text{Target}(Q))$ 
8: for  $t = 1, \dots, S_{\text{joint}}$  do
9:   Recompute  $Z \leftarrow f_{\theta}(A, X, T)$  and  $Q \leftarrow Q(Z, \mu)$ 
10:  if  $t \bmod \tau = 0$  then
11:     $P \leftarrow \text{stopgrad}(\text{Target}(Q))$ 
12:  Update  $\theta$  and  $\mu$  by minimizing  $\mathcal{L}_{\text{joint}}$ 
13: Recompute  $Z \leftarrow f_{\theta}(A, X, T)$  and  $Q \leftarrow Q(Z, \mu)$ 
14:  $\hat{y}_i \leftarrow \arg \max_k q_{ik}$  for each node  $i$ 
15: return  $Z, Q, \hat{y}$ 

```

4. Results and Discussion**4.1. Datasets and Graph Construction**

We evaluate DAETG on four processed snapshots from TEG-DB [2]. Goodreads-Children and Amazon-Apps contain review relations between users and items; the labeled items can have multiple reference categories. Citation uses the arxiv snapshot, with papers as nodes and citation contexts as edge text. Twitter links users to tweets they author or in which they are mentioned; tweet content supplies the edge text. Tweets carry event-category labels, whereas users are unlabeled. Citation and Twitter provide a single reference category for each labeled target node. Table 2 distinguishes stored relation records from the final unique edges and reports the entity types and evaluation subsets.

Table 2. Final graph and evaluation statistics for the processed TEG-DB snapshots. R counts stored relation records; m counts distinct unordered edges after self-loop removal and merging. Evaluation uses the target entity type. Of the 123,946 evaluated books, 109,255 have nonempty reference sets; all evaluation nodes in the other datasets have nonempty reference sets. K is the number of preprocessed reference categories supplied to clustering.

(a) Graph and Evaluation Counts					
Dataset	Nodes n	Records R	Unique Edges m	Evaluation Nodes	K
Goodreads-Children	216,613	734,640	734,640	123,946	9
Amazon-Apps	100,480	752,937	752,937	13,209	61
Citation	169,343	1,166,243	1,157,799	169,343	40
Twitter	60,785	74,666	37,328	33,841	371
(b) Entity Types and Evaluation Targets					
Dataset	Node Types (Counts)	Evaluated Type		Label Type	
Goodreads-Children	Users (92,667); Books (123,946)	Books		Multi	
Amazon-Apps	Users (87,271); Items (13,209)	Items		Multi	
Citation	Papers (169,343)	Papers		Single	
Twitter	Users (26,944); Tweets (33,841)	Tweets		Single	

Preprocessing converts the relation records into a simple undirected graph. Each record is mapped to the unordered key $(\min(i, j), \max(i, j))$. Directed and reciprocal links, repeated interactions, and multiple texts for the same pair are merged into one

unweighted edge. The associated record-level text vectors are averaged before row-wise ℓ_2 normalization, so each distinct pair contributes one row to T . The preprocessing removes self-loops and retains all snapshot nodes, including any isolated nodes. The four snapshots contain no self-loop records, and their final graphs contain no isolated nodes. Node vectors are also row-normalized; zero vectors remain zero. The resulting A is binary and symmetric, with $2m$ nonzero entries; both orientations of an edge use the same vector t_{ij} . Goodreads-Children and Amazon-Apps have one record per distinct pair. Citation contains 8444 reciprocal pairs, whose merging reduces 1,166,243 directed records to 1,157,799 unordered edges. Twitter contains 37,338 repeated records beyond its 37,328 distinct user–tweet pairs.

All retained nodes participate in training. Evaluation uses the target entity subsets: 123,946 books, 13,209 items, 169,343 papers, and 33,841 tweets, respectively. A multi-label node is counted once in the evaluation subset. Both BERT conditions use the same graph and evaluation mask.

Reference-category preprocessing removes the domain-level category children from Goodreads-Children and Apps for Android from Amazon-Apps before counting categories or computing evaluation metrics. Each removed category is shared by every target node in its snapshot and does not distinguish communities within that domain. The remaining category sets contain 9 and 61 categories, respectively. This operation changes neither the graph nor the target evaluation mask. The 14,691 Children nodes whose reference sets become empty remain in the evaluation universe and contribute no reference-category memberships; no replacement category is added.

The encoder treats the snapshots as homogeneous graphs with shared parameters and no type indicators. Users and items or tweets remain in the same graph, allowing a community to span entity types. The homogeneous degree-based null model in Equation (16) rewards excess within-community connectivity in the pooled graph, while the external metrics measure category alignment on the target subset.

4.2. Comparison Methods

The main comparison includes four graph encoders that use connectivity, node vectors, and edge vectors (A, X, T) .

- **GraphSAGE** aggregates neighborhood messages [27]. Our edge-text augmentation uses a multilayer perceptron (MLP) to form each message as $m_{ij} = \text{MLP}_{\text{SAGE}}([h_j \parallel t_{ij}])$ before mean aggregation and the node update.
- **GINE** adds a transformed edge vector to each neighboring-node message before aggregation [20].
- **NENN** alternates node- and edge-level attention to update both representations [10].
- **GraphTransformer** is the Transformer-style baseline used in TEG-DB, incorporating edge vectors into neighbor keys and values [2].

The comparison without edge text includes five methods developed for clustering node-attributed graphs. These methods receive the same (A, X) and retain their community objectives.

- **DAEGC** combines a graph-attention autoencoder with clustering self-training [3].
- **CDBNE** integrates reconstruction, assignment self-training, and modularity maximization [4].
- **DDGAE** learns complementary attribute and modularity-matrix views through graph-attention autoencoding [15].
- **BCDAN** balances attribute and topology representations through separate encoders and community-oriented fusion [16].
- **DMoN** learns soft assignments through graph convolution, differentiable modularity, and cluster-mass regularization [5].

4.3. Evaluation Metrics and Experimental Protocol

The main experiment compares encoders under the same community-learning protocol. BERT-base and BERT-Large supply frozen node and edge vectors of 768 and 1024 dimensions, respectively. Every encoder outputs 128-dimensional representations, receives 200 topology-reconstruction updates, and then uses K-means++ to initialize the trainable centroids. The same Student- t assignment and joint objective provide 200 refinement updates. These encoders also share the optimizer, negative-sampling, K-means, and target-update settings in Table A1. Each run initializes centroids from its own pretrained representations. This comparison controls inputs, community objectives, and update counts while retaining the encoder-specific architectures in Table A2. The specialized methods receive 400 updates with their own objectives and stage schedules.

Each condition uses seeds {11, 22, 33, 44, 55} with the same graph, evaluation mask, and corresponding frozen embeddings. Community-detection scores are reported as mean $\pm$ sample standard deviation (SD) over five runs, with the variance divisor 5 – 1. Performance scores and SDs are expressed as percentages to two decimal places; differences are in percentage points. Transfer strengths retain their native scale. Bold and underlined entries identify the highest and second-highest distinct means within each comparison. Repeated conditions reuse the same results. Paired differences, pointwise 95% intervals, and multiplicity-adjusted tests are defined in Appendix A.8. A pointwise interval excluding zero does not imply significance after correction for multiple comparisons; inferential claims use the Holm-adjusted p -values at the 0.05 threshold. Individual-run scores and paired comparisons are provided in Supplementary Tables S1 and S2, respectively.

Training uses a supplied community count without category-membership supervision. The reference categories retained after the preprocessing in Section 4.1 set K to 9, 61, 40, and 371 for Goodreads-Children, Amazon-Apps, Citation, and Twitter. Membership labels do not enter optimization, centroid initialization, stopping, or checkpoint selection. Every run uses its final prescribed update. The default modularity weight $\lambda_M = 0.1$ was fixed before examining reference-category scores. Common settings are shared across datasets, including an initial learned transfer strength of $\gamma = 0.5$. The sensitivity experiments assess the effect of transfer strength and modularity weighting on performance.

Overlapping normalized mutual information (ONMI) and pairwise F1 (Pair-F1) assess agreement with the multi-label reference categories on Goodreads-Children and Amazon-Apps. For ONMI [28], the argmax partition is represented as a cover with one membership per node. Nodes with empty reference sets remain in the evaluation universe and contribute no reference memberships. If $\mathcal{P}$ contains unordered node pairs in the same predicted community and $\mathcal{R}$ contains those sharing a reference category, then $\text{Pair-F1} = 2|\mathcal{P} \cap \mathcal{R}|/(|\mathcal{P}| + |\mathcal{R}|)$. Appendix A.6 also evaluates a fixed overlapping decoder. Citation and Twitter use adjusted Rand index (ARI) [29] and clustering accuracy (ACC). ARI measures chance-adjusted pairwise agreement; ACC uses one-to-one Hungarian matching, with a rectangular contingency matrix when community counts differ. Unmatched predicted communities contribute no correct assignments, and all evaluation nodes enter the denominator. Higher values are better for all four metrics.

4.4. Community Detection with Edge-Text Information

The shared-head comparison tests whether the encoder improves community representations when inputs and optimization are held fixed. Table 3 compares the five encoders under both embedding settings.

Table 3. Shared-head encoder comparison with edge text: five-run mean $\pm$ SD (%). Best and second-best means in each metric column are bold and underlined, respectively, within each panel.

Encoder	Goodreads-Children		Amazon-Apps		Citation		Twitter	
	ONMI	Pair-F1	ONMI	Pair-F1	ARI	ACC	ARI	ACC
(a) BERT-base								
GraphSAGE	47.22 $\pm$ 2.36	73.05 $\pm$ 1.23	54.68 $\pm$ 2.03	36.12 $\pm$ 0.56	69.26 $\pm$ 1.12	66.77 $\pm$ 1.36	71.20 $\pm$ 2.23	72.53 $\pm$ 1.68
GINE	49.51 $\pm$ 0.69	74.32 $\pm$ 2.13	57.06 $\pm$ 1.78	37.84 $\pm$ 2.51	70.55 $\pm$ 0.36	68.32 $\pm$ 0.98	72.94 $\pm$ 1.11	73.56 $\pm$ 1.09
NENN	49.22 $\pm$ 1.10	74.34 $\pm$ 1.02	58.55 $\pm$ 3.21	38.17 $\pm$ 1.10	71.82 $\pm$ 0.32	69.34 $\pm$ 0.57	72.26 $\pm$ 0.36	73.26 $\pm$ 0.28
GraphTransformer	<u>51.43 $\pm$ 0.23</u>	<u>75.92 $\pm$ 0.95</u>	<u>59.74 $\pm$ 0.08</u>	<u>38.34 $\pm$ 0.13</u>	<u>72.67 $\pm$ 1.56</u>	<u>69.77 $\pm$ 2.08</u>	72.51 $\pm$ 3.44	73.38 $\pm$ 1.96
DAETG	52.57 $\pm$ 0.47	77.07 $\pm$ 0.60	60.39 $\pm$ 0.24	39.12 $\pm$ 1.15	72.96 $\pm$ 1.49	71.22 $\pm$ 1.15	<u>72.57 $\pm$ 0.39</u>	73.75 $\pm$ 0.17
(b) BERT-Large								
GraphSAGE	46.36 $\pm$ 1.69	73.23 $\pm$ 1.21	55.43 $\pm$ 2.63	33.96 $\pm$ 1.79	65.03 $\pm$ 0.78	64.18 $\pm$ 1.29	72.16 $\pm$ 1.23	73.36 $\pm$ 1.24
GINE	48.53 $\pm$ 3.69	74.56 $\pm$ 2.36	57.84 $\pm$ 1.36	35.59 $\pm$ 2.51	66.73 $\pm$ 0.63	65.46 $\pm$ 1.18	73.31 $\pm$ 0.11	74.34 $\pm$ 1.29
NENN	48.36 $\pm$ 1.97	75.46 $\pm$ 2.30	59.31 $\pm$ 1.21	36.88 $\pm$ 2.10	67.99 $\pm$ 0.23	66.88 $\pm$ 0.39	74.02 $\pm$ 0.66	75.05 $\pm$ 0.58
GraphTransformer	<u>50.51 $\pm$ 0.37</u>	<u>76.36 $\pm$ 1.36</u>	<u>60.71 $\pm$ 2.03</u>	<u>37.84 $\pm$ 1.01</u>	<u>69.05 $\pm$ 1.13</u>	<u>67.92 $\pm$ 2.08</u>	<u>74.43 $\pm$ 1.44</u>	<u>75.55 $\pm$ 1.35</u>
DAETG	51.68 $\pm$ 0.36	77.30 $\pm$ 0.60	61.26 $\pm$ 0.29	38.06 $\pm$ 0.85	69.56 $\pm$ 1.19	68.75 $\pm$ 0.25	74.77 $\pm$ 0.49	75.88 $\pm$ 0.97

DAETG's clearest mean advantage occurs on Goodreads-Children, where both metrics improve under both embedding settings. The BERT-base gains over GraphTransformer are 1.14 points in ONMI and 1.15 in Pair-F1; the BERT-Large comparison follows the same pattern. Amazon-Apps shows smaller gains, indicating that the benefit of relation-specific aggregation varies even between recommendation graphs. On Citation, the larger gain in ACC than ARI suggests a stronger change in category alignment than in pairwise agreement. Twitter is more closely contested: GINE gives the highest BERT-base ARI, while DAETG gives the highest ACC.

Embedding size changes the balance between the two measures of category agreement. On Apps, every encoder gains ONMI but loses Pair-F1 with BERT-Large; on Citation, both metrics favor BERT-base. These shared trends indicate that text representations affect the comparison beyond the choice of graph encoder.

The ONMI gains over GraphTransformer occur in all five runs on Children with both embeddings and on Apps with BERT-base. Children's mean-difference intervals are [0.50, 1.78] and [0.40, 1.94] points for BERT-base and BERT-Large, respectively; Apps gives [0.31, 0.99] with BERT-base. Their Holm-adjusted p -values are 0.360, 0.527, and 0.282. The remaining GraphTransformer comparisons have intervals spanning zero (Appendix A.8). None of the 16 comparisons against GraphTransformer reaches the Holm-adjusted 0.05 threshold within the 64-comparison shared-head family.

The BERT-base cost comparison in Appendix A.9 reports lower peak GPU memory use for DAETG on all four graphs and shorter runtimes on Children, Apps, and Citation. On Twitter, its runtime matches GraphTransformer at the reported precision. Appendix A.9 gives the measurement environment and results.

4.5. Community Detection Without Edge-Text Information

Removing edge text tests the community framework without relation semantics. Setting $T = 0$ gives $c_i = 0$ and $u_i = \text{LN}(h_i)$; uniform first-stage attention makes the transferred prior constant, which cancels in the neighbor softmax. Table 4 compares this variant with methods using topology and BERT-base node vectors.

Table 4. Community methods without edge text, using BERT-base: five-run mean $\pm$ SD (%). Best and second-best means in each metric column are bold and underlined, respectively.

Method	Goodreads-Children		Amazon-Apps		Citation		Twitter	
	ONMI	Pair-F1	ONMI	Pair-F1	ARI	ACC	ARI	ACC
DAEGC	42.81 $\pm$ 1.21	66.62 $\pm$ 0.92	51.62 $\pm$ 1.32	32.62 $\pm$ 1.36	64.65 $\pm$ 1.33	62.07 $\pm$ 0.11	56.03 $\pm$ 1.15	64.93 $\pm$ 0.66
CDBNE	45.62 $\pm$ 2.13	71.42 $\pm$ 1.71	52.36 $\pm$ 2.25	33.32 $\pm$ 2.32	65.44 $\pm$ 1.28	<u>64.86 $\pm$ 0.92</u>	66.87 $\pm$ 1.01	68.30 $\pm$ 1.66
DDGAE	46.21 $\pm$ 1.11	72.01 $\pm$ 0.89	53.06 $\pm$ 0.13	34.05 $\pm$ 0.02	62.24 $\pm$ 0.33	63.04 $\pm$ 0.69	68.64 $\pm$ 1.21	65.55 $\pm$ 0.99
BCDAN	47.03 $\pm$ 1.00	72.74 $\pm$ 1.54	53.88 $\pm$ 1.36	34.06 $\pm$ 2.01	66.23 $\pm$ 1.33	64.11 $\pm$ 0.45	71.12 $\pm$ 3.44	72.03 $\pm$ 2.25
DMoN	46.22 $\pm$ 2.65	71.69 $\pm$ 3.44	53.22 $\pm$ 1.69	33.88 $\pm$ 1.36	65.98 $\pm$ 1.74	63.66 $\pm$ 1.47	70.56 $\pm$ 0.83	70.98 $\pm$ 1.44
DAETG (no edge text)	47.36 $\pm$ 1.15	73.05 $\pm$ 0.78	54.16 $\pm$ 0.24	34.23 $\pm$ 0.59	67.08 $\pm$ 0.14	65.02 $\pm$ 1.35	<u>71.00 $\pm$ 0.29</u>	72.74 $\pm$ 1.18

Without edge text, DAETG performs close to the leading specialized methods. Its margins over the strongest alternative range from -0.12 to 0.85 points, with none significant after Holm correction. DMoN remains competitive using node attributes and regularized modularity alone. Restoring edge text produces larger within-DAETG gains, especially on Children, Apps, and Citation. The contrast identifies relation semantics as an important source of the full model's improvement; the following ablations examine how the encoder uses them.

4.6. Ablation Study

The ablations separate edge-text encoding from community refinement. We use Goodreads-Children and Twitter to cover contrasting relation and evaluation settings: user–book reviews with multi-label categories, and user–tweet relations with single-label event categories. The graphs also differ in node count, edge count, and reference community count (Table 2). Within each dataset, all variants use the same BERT-base embeddings. Table 5 reports the controls; objective variants share each seed's pretrained full-model checkpoint and initial centroids.

Table 5. BERT-base ablations: five-run mean $\pm$ SD (%). Best and second-best means in each metric column are bold and underlined within each panel. Pretrain + K-means uses 200 updates; the other variants use 200+200.

Variant	Goodreads-Children		Twitter	
	ONMI	Pair-F1	ARI	ACC
(a) Encoder and edge-text controls				
DAETG	52.57 $\pm$ 0.47	77.07 $\pm$ 0.60	72.57 $\pm$ 0.39	73.75 $\pm$ 0.17
No edge text	47.36 $\pm$ 1.15	73.05 $\pm$ 0.78	71.00 $\pm$ 0.29	72.74 $\pm$ 1.18
$\gamma = 0$	<u>48.48 $\pm$ 2.34</u>	<u>74.60 $\pm$ 1.76</u>	<u>71.65 $\pm$ 1.40</u>	<u>72.88 $\pm$ 1.30</u>
Shuffled edge text	44.20 $\pm$ 1.20	70.90 $\pm$ 1.10	63.89 $\pm$ 2.36	66.59 $\pm$ 1.50
Single-stage encoder	41.83 $\pm$ 1.23	68.30 $\pm$ 1.11	61.20 $\pm$ 1.63	65.20 $\pm$ 0.95
(b) Training strategy and objectives				
$\mathcal{L}_{\text{joint}}$	<u>52.57 $\pm$ 0.47</u>	<u>77.07 $\pm$ 0.60</u>	<u>72.57 $\pm$ 0.39</u>	73.75 $\pm$ 0.17
$\mathcal{L}_{\text{KL}}$	49.62 $\pm$ 0.90	74.80 $\pm$ 0.83	68.63 $\pm$ 1.33	71.36 $\pm$ 1.21
$-0.1\mathcal{M}$	39.85 $\pm$ 1.83	64.10 $\pm$ 1.71	59.87 $\pm$ 2.20	64.39 $\pm$ 1.26
$\mathcal{L}_{\text{joint}} + 0.1R_{\text{col}}$	52.60 $\pm$ 0.60	77.13 $\pm$ 0.50	72.74 $\pm$ 1.02	73.78 $\pm$ 0.57
$\mathcal{L}_{\text{KL}} + 0.1R_{\text{col}}$	50.10 $\pm$ 0.82	74.88 $\pm$ 0.76	69.70 $\pm$ 1.20	71.94 $\pm$ 0.88
$\mathcal{L}_{\text{joint}} + 0.1\mathcal{L}_{\text{pre}}$	52.46 $\pm$ 0.56	76.90 $\pm$ 0.60	72.40 $\pm$ 1.12	73.80 $\pm$ 1.05
Pretrain + K-means	48.89 $\pm$ 1.10	73.20 $\pm$ 1.13	67.80 $\pm$ 1.50	70.58 $\pm$ 1.34

Separate controls test edge context and attention transfer. Fixing $\gamma = 0$ in both training stages preserves T in c_i and u_i while removing its prior from neighbor attention. This variant improves over no edge text, whereas enabling transfer gives further mean gains

of 4.09 ONMI points on Children and 0.92 ARI points on Twitter. Shuffling edge vectors within relation types instead lowers performance below the no-text condition, showing that their association with node pairs matters (Appendix A). The single-stage encoder also gives lower scores at a closely matched parameter count (287,552 versus 287,489; Appendix A.2). This comparison evaluates the complete encoder designs, including their different attention functions and the single-stage bottleneck; the $\gamma = 0$ control specifically tests coefficient transfer. The full-model comparisons with shuffled text and the single-stage control reach Holm-adjusted significance for both primary metrics, whereas the $\gamma = 0$ comparisons do not ($p_{\text{Holm}} = 0.370$ on Children and 1.000 on Twitter; Appendix A.8). Thus, these controls support correctly associated edge text and the complete encoder, while the incremental benefit of coefficient transfer remains unestablished under the five-run evaluation.

Community refinement improves the representations learned by reconstruction. Direct K-means after 200 pretraining updates gives lower scores than the full 200+200 schedule. KL-only refinement recovers part of the gain, while the joint objective improves on either KL or modularity alone. These controls distinguish confident assignment refinement from the connectivity constraint imposed by modularity. Their combination benefits category alignment under the tested loss weights.

Additional regularization changes the results less than the main components. Adding the DMoN cluster-mass penalty or retaining reconstruction produces small mean changes whose paired intervals include zero. Replacing modularity with the mass penalty maintains low empty-cluster rates but lowers category alignment: controlling community mass does not encode which nodes are connected. Appendix A.7 reports the assignment diagnostics, and Appendix A.5 provides the Louvain control using connectivity alone. On these datasets, DAETG's KL-modularity objective captures the main refinement gains without an additional regularizer.

4.7. Transfer Strength and Parameter Sensitivity

Learned transfer adjusts the strength of the edge prior to each graph. Table 6 compares sigmoid-constrained, fixed, and free-scalar transfer with $\lambda_M = 0.1$. Both learned variants start at effective $\gamma = 0.5$ and update it in both stages; the fixed variant retains 0.5 throughout.

Table 6. Learned and fixed transfer with BERT-base and $\lambda_M = 0.1$. Performance is five-run mean $\pm$ SD (%); final γ retains its native scale. Best and second-best means in each performance column are bold and underlined.

(a) Community-Detection Performance								
Transfer	Goodreads-Children		Amazon-Apps		Citation		Twitter	
	ONMI	Pair-F1	ONMI	Pair-F1	ARI	ACC	ARI	ACC
Learned sigmoid	52.57 $\pm$ 0.47	77.07 $\pm$ 0.60	60.39 $\pm$ 0.24	39.12 $\pm$ 1.15	72.96 $\pm$ 1.49	71.22 $\pm$ 1.15	72.57 $\pm$ 0.39	73.75 $\pm$ 0.17
Fixed $\gamma = 0.5$	52.20 $\pm$ 0.62	76.80 $\pm$ 0.54	60.11 $\pm$ 0.67	38.98 $\pm$ 0.62	72.74 $\pm$ 0.55	70.90 $\pm$ 0.63	72.23 $\pm$ 1.07	73.51 $\pm$ 0.93
Free scalar	<u>52.41 $\pm$ 0.93</u>	<u>76.90 $\pm$ 0.75</u>	60.42 $\pm$ 0.88	<u>39.04 $\pm$ 0.82</u>	<u>72.85 $\pm$ 0.74</u>	<u>71.11 $\pm$ 0.94</u>	<u>72.41 $\pm$ 1.40</u>	<u>73.61 $\pm$ 1.17</u>
(b) Final Transfer Strength								
Transfer	Children		Apps		Citation		Twitter	
Learned sigmoid	0.43 $\pm$ 0.07		0.61 $\pm$ 0.08		0.34 $\pm$ 0.06		0.58 $\pm$ 0.10	
Fixed $\gamma = 0.5$	0.50 (fixed)		0.50 (fixed)		0.50 (fixed)		0.50 (fixed)	
Free scalar	0.46 $\pm$ 0.12		0.68 $\pm$ 0.18		0.29 $\pm$ 0.15		0.64 $\pm$ 0.22	

Starting from 0.5, sigmoid transfer ends at 0.43 $\pm$ 0.07 on Children, 0.61 $\pm$ 0.08 on Apps, 0.34 $\pm$ 0.06 on Citation, and 0.58 $\pm$ 0.10 on Twitter. Each value summarizes one global scalar per run. The means decrease from initialization on Children and Citation and increase on Apps and Twitter, reflecting different learned weighting of the edge prior. Its effect also

depends on edge-score dispersion and node-compatibility scale, so these values are not direct cross-graph measures of text importance.

Learning sigmoid transfer improves primary-metric means over fixing $\gamma = 0.5$ by only 0.22–0.37 points. These descriptive differences do not firmly establish an incremental benefit from learning transfer strength under the five-run evaluation. The free scalar performs similarly and leads on Apps ONMI. Sigmoid transfer has lower final-parameter SD on all four graphs, although score SD does not follow this ordering.

Figure 3 examines the two parameters separately. The transfer sweep fixes $\lambda_M = 0.1$, and the modularity sweep fixes $\gamma = 0.5$; their shared condition reuses the fixed-transfer result in Table 6.

Transfer strength and modularity weight act at different points in the learning procedure. The exponent γ changes the relative importance of the edge-semantic prior before neighbor-attention normalization, while leaving the adjacency and modularity null model unchanged. The coefficient λ_M instead weights the topology criterion during refinement of the same assignments used by KL self-training. The two sweeps therefore probe how relation semantics enter the representation and how connectivity subsequently guides the partition.

Moderate weighting gives the most favorable region of the tested settings. Transfer scores vary little between 0.25 and 1; Twitter’s dip at 0.1 shows that a weak prior can perform worse than disabling transfer. Modularity has a more pronounced effect: omitting it or using the largest weight lowers every primary score relative to 0.1, although Citation peaks at 0.01. The loss weight balances assignment preferences induced by the embeddings with degree-adjusted connectivity. A stronger modularity coefficient can favor structurally cohesive partitions even when agreement with reference categories decreases. The decline at the largest tested weight is consistent with this tradeoff. DAETG’s default modularity weight lies in this moderate region, while learned transfer adjusts the edge prior to each graph.

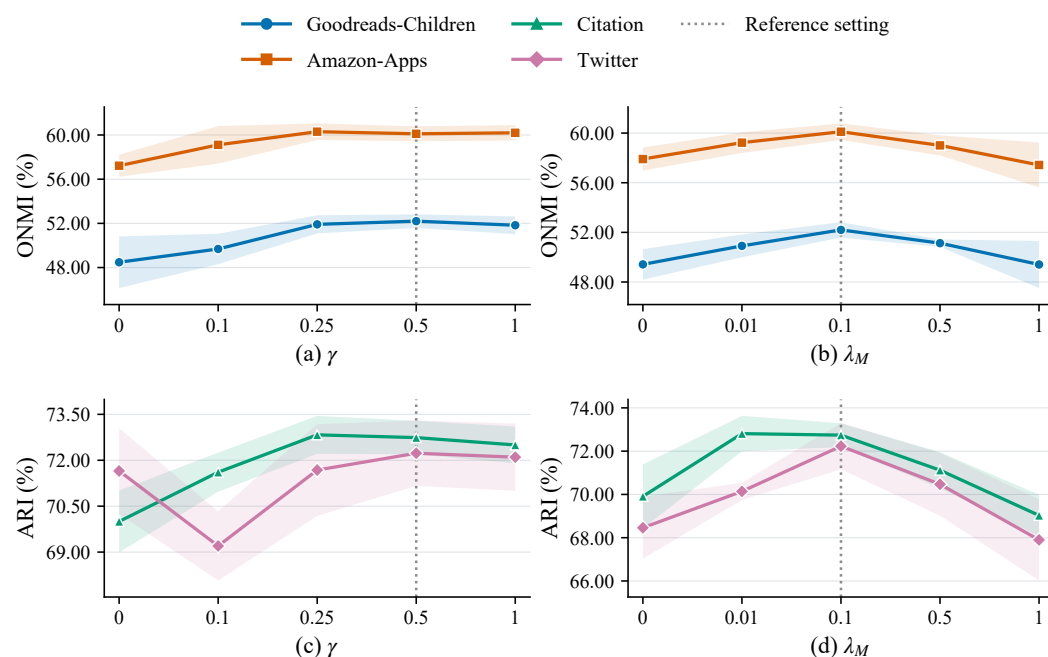


Figure 3. Parameter sensitivity with BERT-base (five seeds). Lines show mean performance (%); shaded bands show one sample SD. (a) ONMI on Goodreads-Children and Amazon-Apps as fixed γ varies at $\lambda_M = 0.1$; (b) ONMI on the same datasets as λ_M varies at $\gamma = 0.5$; (c) ARI on Citation and Twitter as fixed γ varies at $\lambda_M = 0.1$; (d) ARI on the same datasets as λ_M varies at $\gamma = 0.5$. Gray lines mark these reference values. Horizontal positions enumerate the tested settings.

4.8. Limitations

DAETG learns partitions at a supplied community granularity. The experiments obtain K from reference categories while excluding membership labels from training; automatic selection of K remains outside this evaluation. Argmax decoding gives a non-overlapping partition. The fixed overlapping decoder improves both metrics on Children but trades higher Pair-F1 for lower ONMI on Apps (Appendix A.6).

Practical use therefore requires an externally specified community granularity. A known catalog or an analytical objective can supply it; when K is unknown, a separate selection procedure is needed. The present reference- K results do not establish performance under misspecified K . Choosing too few or too many communities can merge or fragment semantic groups, with metric-dependent effects that also depend on community sizes and separation. Applications with overlapping communities also require an explicit decoding rule, since the observed decoder tradeoff differs between Children and Apps.

The evidence concerns the four TEG-DB snapshots, two embedding models, and the stated training protocol. Component ablations cover two datasets with BERT-base, and five seeds give limited precision for small differences or for assessing the distributional assumptions of paired tests. The homogeneous modularity null model also ignores restrictions on type-to-type connections. Application to a new textual-edge domain therefore requires validation of community-count choice, decoding, and the contribution of relation text.

5. Conclusions

DAETG connects relation semantics with community learning through explicit attention transfer. Its dual-stage encoder first summarizes incident-edge text and then reuses the resulting coefficients when aggregating neighboring nodes. Topology pretraining supplies an initial representation, and KL self-training with soft modularity jointly refines the encoder and centroids for a supplied community count.

Experiments on four TEG-DB graphs show competitive community-detection performance under matched community-learning objectives. The mean improvements are modest and depend on the dataset and evaluation metric. Ablations on Goodreads-Children and Twitter support the contributions of correctly associated edge text, the complete dual-stage encoder, and joint community refinement. Within the BERT-base cost comparison, DAETG has the lowest mean peak GPU memory on all four graphs and the shortest mean training time on three.

These findings support combining relation semantics with structural community learning. Future work will examine community-count selection, adaptive overlapping decoding, and evaluation on additional types of textual-edge graphs.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/electronics15204624/s1>, Table S1: Individual-run community-detection scores for 164 experimental conditions across five seeds; Table S2: Seed-paired performance comparisons, pointwise 95% confidence intervals, and Holm-adjusted statistical tests for 144 planned contrasts.

Author Contributions: Conceptualization, Y.L.; methodology, Y.L.; software, Y.L.; validation, Y.L.; formal analysis, Y.L.; investigation, Y.L.; data curation, Y.L.; writing—original draft preparation, Y.L.; writing—review and editing, Y.L. and H.C.; visualization, Y.L.; supervision, H.C. All authors have read and agreed to the published version of the manuscript.

Funding: Publication of this article is supported by the Top Talent Training Project of Henan Province (Grant No. 244500510012).

Data Availability Statement: Publicly available datasets were analyzed in this study. The TEG-DB benchmark data are available at <https://github.com/Zhuofeng-Li/TEG-Benchmark> (accessed on 6 October 2026), as described in Ref. [2]. The experimental summaries are included in the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Implementation and Supplementary Analyses

Table A1 separates the shared-head training settings from DAETG-specific encoder settings. For run seed s , negative sampling uses seed $10^6 + s$.

The shuffled-text control operates on unique undirected edges after record merging and vector normalization. Within each relation type, complete edge vectors are randomly permuted once using seed $2 \times 10^6 + s$. The permutation is fixed throughout pretraining and refinement, and both orientations of an edge share the assigned vector. This preserves adjacency, node degrees, relation-type edge counts, and each type's multiset of text vectors, while randomizing their association with node pairs. Node features and reference labels remain unchanged.

Table A1. Shared-head training protocol and DAETG-specific encoder settings. The shared protocol applies to all five encoders in Table 3.

Component	Setting
Shared inputs, optimization, and community head	
Input vectors	Frozen, row- ℓ_2 normalized; BERT-base 768 or BERT-Large 1024 dimensions
Encoder output	128-dimensional node representations
Optimizer	Adam; learning rate 10^{-3} in both stages; $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$
Regularization	Weight decay 10^{-5} ; centroids, biases, and any LayerNorm or transfer parameters excluded; gradient norm clipped at 5
Training schedule	200 full-batch pretraining and 200 refinement updates; float32; Adam state reset at refinement
Stopping	Final prescribed update; no scheduler or early stopping
Negative sampling	One unordered non-edge per positive edge; no self-pairs; resampled at each pretraining update
Reconstruction loss	Mean positive binary cross-entropy (BCE) plus mean negative BCE
Community head	Student- t with $\nu = 1$; trainable centroids; node-mean KL; initial stop-gradient target refreshed before updates 5, 10, ...
Refinement objective	KL self-training and modularity, $\lambda_M = 0.1$; reconstruction used only during pretraining
K-means initialization	K-means++, 20 restarts, maximum 300 iterations, tolerance 10^{-4} , Lloyd algorithm, random state s
Replications	$s \in \{11, 22, 33, 44, 55\}$; mean and sample SD with denominator 5 – 1
DAETG encoder only	
Attention	$d_a = 64$; one attention head per stage
Activations	Rectified linear unit (ReLU) for node initialization and neighbor update; LeakyReLU slope 0.2 for edge scores
Normalization	LayerNorm, $\epsilon = 10^{-5}$; no dropout
Projection initialization	Xavier uniform; only W_0 has a bias (initialized to zero); LayerNorm scale 1 and bias 0
Transfer	One global learned sigmoid scalar; effective initial $\gamma = 0.5$, initial logit 0; stable log-softmax prior

Appendix A.1. Baseline Inputs and Adaptations

Table A2 specifies the comparison methods. The shared-head encoders replace supervised classification outputs with 128-dimensional representations and use the shared protocol in Table A1. Their internal widths and message operations remain method-specific. The no-edge-text comparison instead retains the specialized methods' native objectives and representation widths, with frozen BERT-base node vectors as attributes and a total budget of 400 updates.

Table A2. Baseline inputs and method-specific adaptations. The upper block uses the shared training protocol in Table A1; the lower block retains native community objectives with 400 total updates.

Method	Input	Method-Specific Adaptation and Optimization
GraphSAGE	A, X, T	Three layers, widths 256–256–128, mean aggregation. Neighbor messages are $\text{MLP}_{\text{SAGE}}([h_j t_{ij}])$, followed by GraphSAGE aggregation and the shared community head.
GINE	A, X, T	Three GINEConv layers, widths 256–256–128, linear update maps and fixed $\epsilon = 0$. Each layer projects T to its input width before message aggregation; output feeds the shared head.
NENN	A, X, T	Three attention layers in node–edge–node order; the final node representation is adapted to 128 dimensions for the shared community head.
GraphTransformer	A, X, T	Three TransformerConv layers, widths 256–256–128, one head. Projected edge vectors enter keys and values; a projected root-state skip is retained, without a learned gating coefficient.
DAEGC	A, X	Two attention layers, widths 256–16, second-order transition proximity; topology reconstruction plus clustering self-training with coefficient 10.
CDBNE	A, X	Attention encoder with widths 256–16 and reconstruction decoder; published objective $L_R - 0.1L_M + L_C$. Reconstruction remains active during clustering.
DDGAE	A, X	Two attention autoencoders, each with widths 256–16, for the attribute and modularity-matrix views. Second-order transition proximity and mean fusion are retained.
BCDAN	A, X	Attribute and structural autoencoders with widths $d_{\text{in}}-500-500-2000-10$, aligned graph convolutional network (GCN) layers, and delivery coefficient 0.5. Structural features are Cluster Affiliation Model for Big Networks (BIGCLAM) affiliations; loss weights are (0.1, 0.01, 0.1).
DMoN	A, X	$\text{GCN}(d_x, 128)-\text{ReLU}-\text{GCN}(128, K)-\text{softmax}$; $-\mathcal{M} + R_{\text{col}}$, without reconstruction pretraining.

DDGAE uses $B = A - dd^T / (2m)$ as its modularity view and $(D^{-1}A + (D^{-1}A)^2)/2$ as its transition proximity, with $D = \text{diag}(d)$ [15]. These are distinct inputs to its representation and attention computations. BCDAN's weights in the table correspond to attribute clustering, GCN, and structure clustering, respectively; d_{in} is the input width of the attribute or BIGCLAM-affiliation branch [16]. All methods use the common evaluation masks and seeds; the main comparisons decode a single membership by argmax.

Appendix A.2. Single-Stage Encoder

The single-stage control combines neighbor and edge messages in one attention operation, using the same initialized states h_i in Equations (A1)–(A3):

$$a_{ij} = \text{softmax}_{j \in \mathcal{N}_i} \left(\text{LeakyReLU} \left(a^\top [W_h h_i \parallel W_j h_j \parallel W_t t_{ij}] \right) \right), \quad (\text{A1})$$

$$v_i = \sum_{j \in \mathcal{N}_i} a_{ij} (W_n h_j + W_v t_{ij}), \quad (\text{A2})$$

$$z_i = \text{LN}(h_i + W_2 \text{ReLU}(W_1 v_i)). \quad (\text{A3})$$

Attention and output widths are 64 and 128. The bias-free maps $W_1 \in \mathbb{R}^{w \times 128}$ and $W_2 \in \mathbb{R}^{128 \times w}$ use the integer $w \in [1, 512]$ minimizing the relative parameter-count difference from DAETG, without label-based selection. With BERT-base, the counts are $N_{\text{DAETG}} = 287,489$ and $N_{\text{single}}(w) = 279,104 + 256w$; thus, $w = 33$ gives 287,552 parameters, a difference of 0.022%. Counts include biases and affine LayerNorm, exclude frozen embeddings, and gain the same 128K trainable centroid parameters during refinement. The decoder has no parameters; isolated nodes use $z_i = \text{LN}(h_i)$, and this variant has no transfer scalar.

Appendix A.3. Message Reductions

Both edge orientations carry the same text vector but normalize attention by receiving node. For message record ℓ with receiver i_ℓ and neighbor j_ℓ , the reductions are

$$c_i = \sum_{\ell: i_\ell=i} \alpha_\ell W_v t_\ell, \quad g_i = \sum_{\ell: i_\ell=i} \beta_\ell W_n u_{j_\ell}, \quad z_i = \text{LN}(u_i + \phi(g_i)).$$

Each message retains its vector inside the sum, and the neighbor projection precedes activation, as in Equations (5) and (9).

Appendix A.4. Objective Controls and DMoN

The DMoN penalty in Equation (A4) balances total soft community mass independently of adjacency:

$$R_{\text{col}}(Q) = \frac{\sqrt{K}}{n} \left\| \sum_{i=1}^n q_i \right\|_2 - 1. \quad (\text{A4})$$

It is minimized by equal column sums [5]. Table 5 uses coefficient 0.1 when adding the penalty, replacing modularity with it, or retaining $\mathcal{L}_{\text{pre}}$. Retained reconstruction resamples negatives at every refinement update. Objective variants share each seed's pretrained encoder and initial centroids; Pretrain + K-means returns that initial partition directly.

DMoN uses a 128-channel GCN with ReLU followed by a K -channel GCN and row-softmax. It minimizes $-\mathcal{M} + R_{\text{col}}$ for 400 updates without reconstruction or dropout. GCN propagation uses $\tilde{D}^{-1/2}(A + I)\tilde{D}^{-1/2}$, where $\tilde{D}$ contains the degrees of $A + I$; modularity uses the original loop-free A .

Appendix A.5. Topology-Only Control

Louvain [12] tests category alignment from connectivity alone (Table A3). NetworkX 3.4.2 uses the complete deduplicated graph, unit edge weights, resolution 1, gain threshold 10^{-7} , singleton initialization, and no level cap. Node and edge insertion orders are fixed; the five common seeds control node-order randomization. The same evaluation mask is applied after convergence, without text embeddings or a prescribed K .

Graph fragmentation helps explain the difference between Louvain's partitions and the reference categories. Children has 3204 connected components, and Twitter has 23,682; Louvain returns community counts close to these structural scales. An event category can span disconnected components, which local modularity moves cannot merge, and one-to-one ACC penalizes the excess communities. Apps is connected and yields 9–12 communities, whereas Citation yields 133–159. This control compares complete systems with different inputs and community-count rules; Section 4.4 isolates encoder differences.

Table A3. Topology-only Louvain: five-run mean $\pm$ SD (%) and the range of discovered community counts. Dashes denote unused metrics; an SD of 0.00 can result from rounding.

Dataset	ONMI	Pair-F1	ARI	ACC	K_{out}
Goodreads-Children	2.32 ± 0.03	17.46 ± 0.81	–	–	3584–3756
Amazon-Apps	2.29 ± 0.12	23.98 ± 3.02	–	–	9–12
Citation	–	–	25.22 ± 1.36	36.82 ± 1.31	133–159
Twitter	–	–	0.36 ± 0.00	3.02 ± 0.00	23,688–23,689

Appendix A.6. Overlapping Decoding

A fixed relative threshold converts the same BERT-base assignments into overlapping memberships: $O_{ik} = \mathbf{1}[q_{ik} \geq 0.75 \max_{k'} q_{ik'}]$, where $O \in \{0, 1\}^{n \times K}$ records the decoded memberships. The threshold is fixed without label tuning; training, K , features, and evaluation nodes are unchanged. ONMI compares the resulting cover with the reference cover; Pair-F1 counts each unordered pair sharing any predicted community once. Table A4 compares the decoders.

Table A4. Decoding the same BERT-base assignments: five-run mean $\pm$ SD (%). Bold and underlined entries mark the better and lower means for each dataset and metric.

Dataset	Decoder	ONMI	Pair-F1
Goodreads-Children	Argmax	<u>52.57 ± 0.47</u>	<u>77.07 ± 0.60</u>
	Relative threshold 0.75	<u>54.56 ± 0.65</u>	<u>79.28 ± 0.79</u>
Amazon-Apps	Argmax	<u>60.39 ± 0.24</u>	<u>39.12 ± 1.15</u>
	Relative threshold 0.75	<u>58.32 ± 0.35</u>	<u>41.16 ± 1.32</u>

The fixed decoder produces different tradeoffs on the two recommendation graphs. Children has 65.01% multi-label evaluation nodes and 2.16 memberships per node; both metrics improve. Apps has only 8.26% multi-label nodes and 1.10 memberships per node; Pair-F1 increases while ONMI decreases. Additional assignments can recover shared pairs while weakening agreement with individual categories, providing a plausible explanation for this tradeoff. The same learned Q therefore supports both partition and overlapping evaluation through a change in decoding alone.

Appendix A.7. Assignment Entropy and Empty Clusters

Assignment diagnostics measure soft-column dispersion and hard-cluster occupancy on the full graph. For each run, define

$$\pi_{ik} = \frac{q_{ik}}{\sum_j q_{jk}}, \quad H_k = -\frac{\sum_i \pi_{ik} \log \pi_{ik}}{\log n}, \quad E = \frac{1}{K} \sum_k \mathbf{1} \left[\sum_i \mathbf{1}[\arg \max_{k'} q_{ik'} = k] = 0 \right].$$

We average H_k across columns within each run. The BERT-base diagnostics use all $n = 216,613$, $K = 9$ nodes/communities for Children and $n = 60,785$, $K = 371$ for Twitter, independently of the evaluation mask. Table A5 summarizes the five runs.

Table A5. Assignment diagnostics: five-run mean $\pm$ SD (%). Column entropy is normalized by $\log n$ and averaged across communities within each run; empty-cluster rates use hard assignments.

Variant	Goodreads-Children		Twitter	
	Column Entropy	Empty (%)	Column Entropy	Empty (%)
$\mathcal{L}_{\text{joint}}$	84.56 ± 0.84	0.00 ± 0.00	70.80 ± 1.10	0.59 ± 0.23
$\mathcal{L}_{\text{KL}}$	86.12 ± 1.21	2.22 ± 4.97	73.52 ± 1.55	4.31 ± 0.85
$-0.1\mathcal{M}$	87.24 ± 1.50	17.78 ± 6.09	75.65 ± 1.86	11.32 ± 0.85
$\mathcal{L}_{\text{joint}} + 0.1R_{\text{col}}$	86.77 ± 0.88	0.00 ± 0.00	73.88 ± 1.90	0.11 ± 0.15
$\mathcal{L}_{\text{KL}} + 0.1R_{\text{col}}$	87.70 ± 1.05	0.00 ± 0.00	75.12 ± 1.20	0.22 ± 0.12
$\mathcal{L}_{\text{joint}} + 0.1\mathcal{L}_{\text{pre}}$	85.40 ± 0.95	0.00 ± 0.00	72.10 ± 1.15	0.65 ± 0.24

Column entropy measures the dispersion of each community's normalized mass; it does not measure category agreement or balance between communities. Adding the mass penalty further lowers Twitter's empty-cluster rate but changes category scores little. Replacing modularity also keeps few empty clusters yet loses category alignment. DAETG's joint objective uses the prescribed centroids more fully than either single-loss variant while combining assignment refinement with a connectivity criterion that improves category alignment in these comparisons.

Appendix A.8. Paired Performance Differences

Paired analysis preserves the seed correspondence between methods. For scores a_s, b_s , let $\Delta_s = a_s - b_s$; the pointwise 95% interval is $\bar{\Delta} \pm t_{0.975, 4s} s_{\Delta} / \sqrt{5}$, with $s_{\Delta}^2 = \sum_s (\Delta_s - \bar{\Delta})^2 / 4$. Two-sided paired- t tests use Holm correction within the shared-head, no-edge-text, and ablation families of 64, 40, and 40 contrasts. Intervals are pointwise, not simultaneous. Table A6 covers both embedding settings against GraphTransformer; Table A7 shows primary-metric ablations.

The larger component effects are more clearly separated than the leading methods' small margins. Shuffled text, the single-stage encoder, and no edge text give Holm-adjusted paired- t $p < 0.05$ for both displayed metrics; the transfer-only comparisons do not reach significance after Holm correction. None of the 16 GraphTransformer comparisons in Table A6 reaches the corrected threshold either, including those with pointwise intervals excluding zero. Continued reconstruction and the additional mass penalty have intervals spanning zero. In the no-edge-text family, 7 of 40 contrasts pass Holm correction, but none of the margins over the strongest alternative for each metric do.

Table A6. Paired differences between DAETG and GraphTransformer. Differences and pointwise 95% paired- t confidence intervals (CI) are in percentage points. W/T/L counts seed-wise wins, ties, and losses; p_H denotes Holm adjustment across the 64 shared-head comparisons.

Dataset	Metric	BERT-Base					BERT-Large				
		$\bar{\Delta}$	95% CI	W/T/L	p_H		$\bar{\Delta}$	95% CI	W/T/L	p_H	
Children	ONMI	+1.14	[0.50, 1.78]	5/0/0	0.360		+1.17	[0.40, 1.94]	5/0/0	0.527	
Children	Pair-F1	+1.15	[−0.48, 2.78]	4/0/1	1.000		+0.94	[−0.99, 2.87]	4/0/1	1.000	
Apps	ONMI	+0.65	[0.31, 0.99]	5/0/0	0.282		+0.55	[−1.84, 2.94]	3/0/2	1.000	
Apps	Pair-F1	+0.78	[−0.68, 2.24]	4/0/1	1.000		+0.22	[−1.29, 1.73]	3/0/2	1.000	
Citation	ARI	+0.29	[−2.49, 3.07]	3/0/2	1.000		+0.51	[−2.03, 3.05]	3/0/2	1.000	
Citation	ACC	+1.45	[−1.50, 4.40]	3/0/2	1.000		+0.83	[−1.87, 3.53]	2/0/3	1.000	
Twitter	ARI	+0.06	[−4.18, 4.30]	3/0/2	1.000		+0.34	[−1.71, 2.39]	2/0/3	1.000	
Twitter	ACC	+0.37	[−1.98, 2.72]	2/0/3	1.000		+0.33	[−2.39, 3.05]	2/0/3	1.000	

Table A7. Paired differences between the full DAETG model and each ablation, using BERT-base. Each cell reports the mean difference and its pointwise 95% paired-*t* interval in percentage points. Holm correction covers all 40 ablation contrasts, including both metrics on both datasets.

Comparator	Children ONMI	p_{Holm}	Twitter ARI	p_{Holm}
No edge text	+5.21 [3.59, 6.83]	0.021	+1.57 [1.13, 2.01]	0.016
$\gamma = 0$	+4.09 [0.70, 7.48]	0.370	+0.92 [−0.77, 2.61]	1.000
Shuffled text	+8.37 [7.10, 9.64]	0.002	+8.68 [6.04, 11.32]	0.020
Single-stage	+10.74 [9.01, 12.47]	0.002	+11.37 [9.20, 13.54]	0.004
KL-only	+2.95 [1.83, 4.07]	0.041	+3.94 [2.11, 5.77]	0.083
Modularity-only	+12.72 [10.67, 14.77]	0.002	+12.70 [10.32, 15.08]	0.004
Joint +0.1 R_{col}	−0.03 [−0.64, 0.58]	1.000	−0.17 [−1.56, 1.22]	1.000
KL +0.1 R_{col}	+2.47 [0.93, 4.01]	0.181	+2.87 [1.12, 4.62]	0.178
Joint +0.1 $\mathcal{L}_{\text{pre}}$	+0.11 [−0.50, 0.72]	1.000	+0.17 [−1.37, 1.71]	1.000
Pretrain + K-means	+3.68 [2.68, 4.68]	0.015	+4.77 [2.46, 7.08]	0.087

The paired-*t* calculation assumes approximately normal run differences. An exact two-sided sign-flip check requires symmetry under the null. With five pairs, there are 2^5 sign configurations and a minimum attainable two-sided $p = 2/32 = 0.0625$.

Appendix A.9. Computational Cost

Table A8 compares the runtime and peak GPU memory of the five shared-head encoders using frozen BERT-base inputs. The cost measurements were obtained on an NVIDIA A100 PCIe GPU with 40 GB of memory and an Intel Core i9-9900X CPU at 3.50 GHz. The encoders use the common 200-update pretraining and 200-update refinement protocol in Section 4.3. Runtime includes both training stages and the intervening K-means initialization, excluding BERT feature generation, data loading, and external evaluation metrics. Each entry averages five runs; memory is the mean of the per-run GPU memory peaks.

Table A8. Runtime and peak GPU memory for the BERT-base shared-head comparison on an NVIDIA A100 PCIe 40 GB GPU and an Intel Core i9-9900X CPU at 3.50 GHz. Each cell gives mean runtime in minutes/mean peak GPU memory in GiB over five runs.

Method	Children	Apps	Citation	Twitter
GraphSAGE	9.3/19.8	10.5/19.1	14.9/26.4	6.3/2.6
GINE	8.0/16.8	8.8/19.3	12.5/28.9	6.5/2.6
NENN	4.5/9.3	5.3/12.9	6.9/16.9	6.3/2.2
GraphTransformer	6.6/17.2	7.7/15.9	10.5/24.3	6.2/2.5
DAETG	3.5/6.3	3.8/6.0	4.5/8.6	6.2/1.6

Relative to GraphTransformer, DAETG’s runtime is approximately 47%, 51%, and 57% lower on the first three graphs. Both methods take 6.2 min on Twitter at the reported precision. DAETG’s peak GPU memory use is approximately 36–65% lower across the four graphs. These measurements characterize practical costs on the evaluated graphs; the linear-complexity statement in Section 3.6 is conditional on fixed representation dimensions and K .

References

1. Newman, M.E.J. Modularity and Community Structure in Networks. *Proc. Natl. Acad. Sci. USA* **2006**, *103*, 8577–8582. <https://doi.org/10.1073/pnas.0601602103>.

2. Li, Z.; Gou, Z.; Zhang, X.; Liu, Z.; Li, S.; Hu, Y.; Ling, C.; Zhang, Z.; Zhao, L. TEG-DB: A Comprehensive Dataset and Benchmark of Textual-Edge Graphs. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 10–15 December 2024; Volume 37, pp. 60980–60998. <https://doi.org/10.52202/079017-1949>.
3. Wang, C.; Pan, S.; Hu, R.; Long, G.; Jiang, J.; Zhang, C. Attributed Graph Clustering: A Deep Attentional Embedding Approach. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019; pp. 3670–3676. <https://doi.org/10.24963/ijcai.2019/509>.
4. Zhou, X.; Su, L.; Li, X.; Zhao, Z.; Li, C. Community Detection Based on Unsupervised Attributed Network Embedding. *Expert Syst. Appl.* **2023**, *213*, 118937. <https://doi.org/10.1016/j.eswa.2022.118937>.
5. Tsitsulin, A.; Palowitch, J.; Perozzi, B.; Müller, E. Graph Clustering with Graph Neural Networks. *J. Mach. Learn. Res.* **2023**, *24*, 1–21.
6. Qi, G.J.; Aggarwal, C.C.; Huang, T.S. Community Detection with Edge Content in Social Media Networks. In Proceedings of the 28th IEEE International Conference on Data Engineering, Arlington, VA, USA, 1–5 April 2012; pp. 534–545. <https://doi.org/10.1109/ICDE.2012.77>.
7. Bouveyron, C.; Latouche, P.; Zreik, R. The Stochastic Topic Block Model for the Clustering of Vertices in Networks with Textual Edges. *Stat. Comput.* **2018**, *28*, 11–31. <https://doi.org/10.1007/s11222-016-9713-7>.
8. Boutin, R.; Latouche, P.; Bouveyron, C. The Deep Latent Position Topic Model for Clustering and Representation of Networks with Textual Edges. *Scand. J. Stat.* **2025**, *52*, 1975–2013. <https://doi.org/10.1111/sjos.70016>.
9. Braun, G.; Sugiyama, M. VEC-SBM: Optimal Community Detection with Vectorial Edges Covariates. In Proceedings of the 27th International Conference on Artificial Intelligence and Statistics, Valencia, Spain, 2–4 May 2024; Volume 238, pp. 532–540.
10. Yang, Y.; Li, D. NENN: Incorporate Node and Edge Features in Graph Neural Networks. In Proceedings of the 12th Asian Conference on Machine Learning, Bangkok, Thailand, 18–20 November 2020; Volume 129, pp. 593–608.
11. Jin, B.; Zhang, Y.; Meng, Y.; Han, J. Edgeformers: Graph-Empowered Transformers for Representation Learning on Textual-Edge Networks. In Proceedings of the International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
12. Blondel, V.D.; Guillaume, J.L.; Lambiotte, R.; Lefebvre, E. Fast Unfolding of Communities in Large Networks. *J. Stat. Mech. Theory Exp.* **2008**, *2008*, P10008. <https://doi.org/10.1088/1742-5468/2008/10/P10008>.
13. Traag, V.A.; Waltman, L.; van Eck, N.J. From Louvain to Leiden: Guaranteeing Well-Connected Communities. *Sci. Rep.* **2019**, *9*, 5233. <https://doi.org/10.1038/s41598-019-41695-z>.
14. Xie, J.; Girshick, R.; Farhadi, A. Unsupervised Deep Embedding for Clustering Analysis. In Proceedings of the 33rd International Conference on Machine Learning, New York City, NY, USA, 19–24 June 2016; Volume 48, pp. 478–487.
15. Wu, X.; Lu, W.; Quan, Y.; Miao, Q.; Sun, P.G. Deep Dual Graph Attention Auto-Encoder for Community Detection. *Expert Syst. Appl.* **2024**, *238*, 122182. <https://doi.org/10.1016/j.eswa.2023.122182>.
16. Liu, Y.; Peng, Y.; Chen, H.; Zhang, J.; Li, S. BCDAN: A Balanced Method for Community Detection in Attributed Networks. *Neurocomputing* **2025**, *644*, 130411. <https://doi.org/10.1016/j.neucom.2025.130411>.
17. Mohammadi, M.; Berahmand, K.; Forouzandeh, S.; Zhou, X.; Khosravi, H. Dual-View Entropy-Regularized Nonnegative Matrix Factorization for Attributed Graph Clustering. *Inf. Sci.* **2025**, *721*, 122566. <https://doi.org/10.1016/j.ins.2025.122566>.
18. Boutin, R.; Bouveyron, C.; Latouche, P. Embedded Topics in the Stochastic Block Model. *Stat. Comput.* **2023**, *33*, 95. <https://doi.org/10.1007/s11222-023-10265-9>.
19. Simonovsky, M.; Komodakis, N. Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 29–38. <https://doi.org/10.1109/CVPR.2017.11>.
20. Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V.; Leskovec, J. Strategies for Pre-training Graph Neural Networks. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
21. Gong, L.; Cheng, Q. Exploiting Edge Features for Graph Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; IEEE: New York, NY, USA, 2019; pp. 9203–9211. <https://doi.org/10.1109/CVPR.2019.00943>.
22. Jiang, X.; Ji, P.; Li, S. CensNet: Convolution with Edge-Node Switching in Graph Neural Networks. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019; pp. 2656–2662. <https://doi.org/10.24963/ijcai.2019/369>.
23. Chen, J.; Chen, H. Edge-Featured Graph Attention Network. *arXiv* **2021**, arXiv:2101.07671.
24. Yang, J.; Liu, Z.; Xiao, S.; Li, C.; Lian, D.; Agrawal, S.; Singh, A.; Sun, G.; Xie, X. GraphFormers: GNN-Nested Transformers for Representation Learning on Textual Graph. In Proceedings of the 35th International Conference on Neural Information Processing Systems, Online, 6–14 December 2021; Curran Associates Inc.: Red Hook, NY, USA, 2021; pp. 28798–28810.
25. Ling, C.; Li, Z.; Hu, Y.; Zhang, Z.; Liu, Z.; Zheng, S.; Pei, J.; Zhao, L. Link Prediction on Textual Edge Graphs. *arXiv* **2024**, arXiv:2405.16606 <https://doi.org/10.48550/arXiv.2405.16606>.

26. Sruthi K S.; Sreekumar, A.; Balakrishnan, K. Text-Attributed Community Detection in Complex Networks through LLMs and GNNs: A Powerful Fusion of Language and Graphs. *Neurocomputing* **2025**, *647*, 130573. <https://doi.org/10.1016/j.neucom.2025.130573>.
27. Hamilton, W.L.; Ying, R.; Leskovec, J. Inductive Representation Learning on Large Graphs. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 1024–1034.
28. Lancichinetti, A.; Fortunato, S.; Kertész, J. Detecting the Overlapping and Hierarchical Community Structure in Complex Networks. *New J. Phys.* **2009**, *11*, 033015. <https://doi.org/10.1088/1367-2630/11/3/033015>.
29. Hubert, L.; Arabie, P. Comparing Partitions. *J. Classif.* **1985**, *2*, 193–218. <https://doi.org/10.1007/BF01908075>.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.