HC2-GNN: Hierarchical Graph Representation Learning for Efficient Text Classification

jiejie fan[1], Xiaojuan Ban[1] *, Zhiyan Zhang[2], Xi Sun[3]

[1]School of Intelligence Science and Technology, University of Science and Technology Beijing, China
[2]Collaborative Innovation Center of Steel Technology, University of Science and Technology Beijing, China
[3]National Science Library, Chinese Academy of Sciences, Beijing, China
fanjiejie777@hotmail.com, banxj@ustb.edu.cn

Abstract

Graph Neural Networks (GNNs) offer superior modeling capabilities for text classification by capturing complex spatial features within semantic representations. However, existing graph-based approaches often suffer from computational inefficiency and limited ability to model both fine-grained local structures and the sequential nature of text. To address these challenges, we propose HC2-GNN, a Hierarchical Clustering and Coarsening Graph Neural Network, which introduces a novel lightweight graph clustering algorithm called Compromise Conductance Graph Clustering (C2GC). C2GC enables efficient graph clustering while simultaneously preserving both the textual order and the topological coherence of subgraphs. Furthermore, it incorporates a virtue cluster mechanism that expands each subgraph with semantically relevant neighbors, explicitly enabling cross-cluster information propagation without compromising local structural integrity. HC2-GNN aggregates local and global features by combining subgraph-level and full-graph representations, enhancing semantic discriminability for classification. Extensive experiments on benchmark datasets demonstrate that HC2-GNN consistently outperforms existing state-of-the-art text classification methods.

Code — https://github.com/jackeyfan/HC2-GNN

Introduction

Text classification is a fundamental task in natural language processing (NLP), with wide-ranging applications that include sentiment analysis, question answering, intent detection, and topic categorization. Due to the ambiguities and unstructured semantic relationships in textual data, graph-based methods have become prevalent in text classification, as they naturally capture complex dependencies (Yao, Mao, and Luo 2018a). Many context-aware transductive approaches (Wu et al. 2019; Liu et al. 2020; Wang et al. 2022; Zhu and Koniusz 2021) have focused on text relationships, while inductive methods (Zhang et al. 2020b; Chen, Ma, and Xiao 2018) represent text as graphs of co-occurring words. These graph approaches have set benchmarks in text classification. Furthermore, graph-based models offer a powerful representational framework for semantic learning (Wu et al. 2019; Wu, Hu, and Zhang 2021), but face the challenge of computational inefficiency. The principal strategies for graph simplification are edge dropping (Gao et al. 2021; Lin, Li, and Jia 2023; Li, Zhang, and Wu 2019), which prunes the graph by selectively removing edges while maintaining essential graph characteristics, and graph coarsening (Kang and Faloutsos 2006; Chen, Banerjee, and Li 2012; Gionis, Faloutsos, and Faloutsos 2005), which iteratively clusters nodes to form hierarchical coarse-grained graphs. Although conventional coarsening approaches can alleviate computational complexity, they often either overlook the sequential nature of text, distort local syntactic structures, or limit cross-cluster information flow.

To tackle the above issues, we propose HC2-GNN, a Hierarchical Clustering and Coarsening Graph Neural Network designed for efficient and expressive text classification. At the core of HC2-GNN is a lightweight clustering algorithm, Compromise Conductance Graph Clustering (C2GC), which balances computational efficiency with the preservation of both topological coherence and sequential text attributes by optimizing conductance scores. In addition, we introduce a virtue cluster mechanism that extends each subgraph with semantically relevant neighbors across cluster boundaries, enabling cross-cluster message passing without introducing structural noise. Finally, HC2-GNN hierarchically integrates node representations from subgraph-level and full-graph branches via a dual-GNN architecture, enhancing semantic discriminability across scales. In summary, the key contributions of this paper are as follows:

- We propose C2GC, a novel structure-aware and efficient clustering algorithm that preserves structural coherence and word order during clustering and coarsening.
- We design a virtue cluster mechanism to facilitate effective cross-cluster information propagation, addressing the message bottleneck problem in hierarchical GNNs.
- We develop HC2-GNN, a flexible and generalizable framework that integrates both subgraph-level and full-graph representations and is compatible with diverse GNN and Transformer architectures.
- We conduct extensive experiments on five public benchmark datasets, demonstrating that HC2-GNN consistently outperforms existing state-of-the-art methods in classification accuracy.

*Corresponding authors.
Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

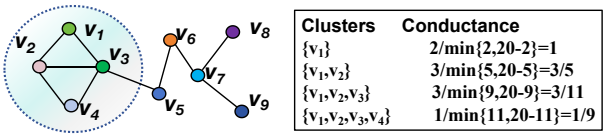

Figure 1: The conductance calculating process of the subgraphs.

Related Work

Graph-Based Text Classification

Graph-based approaches have gained prominence in text classification for capturing complex semantic and structural relationships beyond linear word sequences. Early methods such as TextGCN (Yao, Mao, and Luo 2018b) and HSGCN (Zhang et al. 2020b) constructed word-document heterogeneous graphs using co-occurrence statistics to enable effective transductive classification. Subsequent inductive models (Liu et al. 2020) generalized to unseen data and incorporated richer semantic relations via syntactic or attention-guided graphs. While these methods model long-range dependencies well, many depend on dense or static graphs, leading to high computational costs and limited adaptability. Furthermore, the sequential nature of language, which is crucial for phrase-level semantics, is often underrepresented in graph construction.

To address these issues, Graph Structure Learning (GSL) has emerged as a promising paradigm for dynamic modeling of non-local semantic relationships. Representative works such as GraphSAGE (Hamilton, Ying, and Leskovec 2017) and variational graph autoencoders (Kipf and Welling 2016b) adopt flexible aggregation mechanisms to model both local and global structures. However, most GSL methods still overlook the inherent sequentiality of natural language, which is essential for understanding. Recent efforts like BertGCN (Lin et al. 2021) integrate contextualized language models with graph topology via positional embeddings and semantic features. Parallel research in graph sparsification and coarsening (Piao et al. 2021; Pham, Thanh, and Moore 2021) seeks to improve scalability by reducing graph density, though often at the cost of local order or subgraph structure. Hierarchical models (Zhang et al. 2020a) further explore multi-granular interactions across representations, emphasizing the need for structure-aware, order-preserving mechanisms. These limitations motivate our proposed HC2-GNN, which bridges semantic hierarchy with structural fidelity in graph-based text classification.

Conductance-Based Graph Clustering

When confronting large-scale graphs, the computational demands escalate dramatically with the square magnitude of the adjacency matrix. A prevalent strategy is to partition or coarsen the graph hierarchically. Essentially, graph partitioning endeavours to separate the entire graph into multiple non-overlapping subgraphs characterized by dense intra-subgraph and sparse inter-subgraph connections. This process is synonymous with graph clustering in this paper, where clusters are treated as subgraphs, blurring the lines of the traditional definition. Over the past few years, several critical techniques for graph clustering have been well-established, including modularity-based graph clustering (M. et al. 2004; Newman 2006), structural graph clustering (Xu et al. 2007), and cohesive subgraph-based clustering (Chang and Qin 2019). However, these clustering methods are prone to two challenges: achieving desirable structural properties and maintaining a rigorous theoretical foundation. Apart from them, the *Conductance-Based* clustering (Andersen, Chung, and Lang 2006; Yang et al. 2019) has emerged as the most representative and favoured for its excellent structural properties and strong theoretical underpinnings.

For example, given an undirected graph $G(V, E)$, and a subgraph $C \subseteq V$, we define $\bar{C} = V \setminus C$ as the complement of C in V. The conductance of the cluster C is defined as $\phi(C) = \frac{|E(C,\bar{C})|}{\min\{\text{vol}(C), 2m-\text{vol}(C)\}}$, where $|E(C, \bar{C})|$ is the number of edges between C and $\bar{C}$, $\text{vol}(C)$ is the sum of the degrees of all vertices in C, and m is the total number of edges in G. $\phi(C)$ is the ratio of edges crossing C to the smaller volume side of the cut. Obviously, a smaller $\phi(C)$ indicates better clustering quality (Yang and Leskovec 2012; Leskovec, Lang, and Mahoney 2010). As depicted in Figure 1, (Lin, Li, and Jia 2023), calculating the conductance of clusters is outlined in the box. The objective of conductance-based graph clustering is to identify a cluster C that minimizes the conductance value $\phi(C)$. A lower conductance value signifies a cluster that is well-separated from the rest.

Method

Preliminaries

Before introducing our model, we formally define the undirected graph constructed from a given text. Let $\mathbf{A}$ denote the adjacency matrix representing the connectivity and topological structure of a graph $\mathcal{G}$ with $N = |V|$ vertices and $M = |E|$ edges. For presentation purposes, let $\mathcal{G}_S = (V_S; E_S)$ be the subgraph induced by S if $S \subseteq V$, where $E_S = \{(u, v) \in E | u, v \in S\}$. We use $N_S(v) = \{u \in S | (u, v) \in E\}$ to denote the neighbors of vertex v in S. The degree of v is given by $d_S(v) = |N_S(v)|$, and the volume of S is defined as $\text{vol}(S) = \sum_{u \in S} d_S(u)$, representing the total degree of all vertices in S.

Overall Architecture

An overview of the HC2-GNN is illustrated in Figure 2. We first construct a word co-occurrence graph for each input text following the method in (Nikolentzos, Tixier, and Vazirgiannis 2020). Within these graphs, nodes representing words are ordered sequentially according to their positions in the text. In the clustering module, we partition the graph into subgraphs via the C2GC algorithm, which leverages conductance to ensure partition quality and computational efficiency. The red dashed lines in the figure indicate the additional edges introduced to maintain inter-subgraph connectivity. The coarsening module then produces adjacency matrices for both the resulting subgraphs and the coarsened global graph. In the graph learning module, we apply a

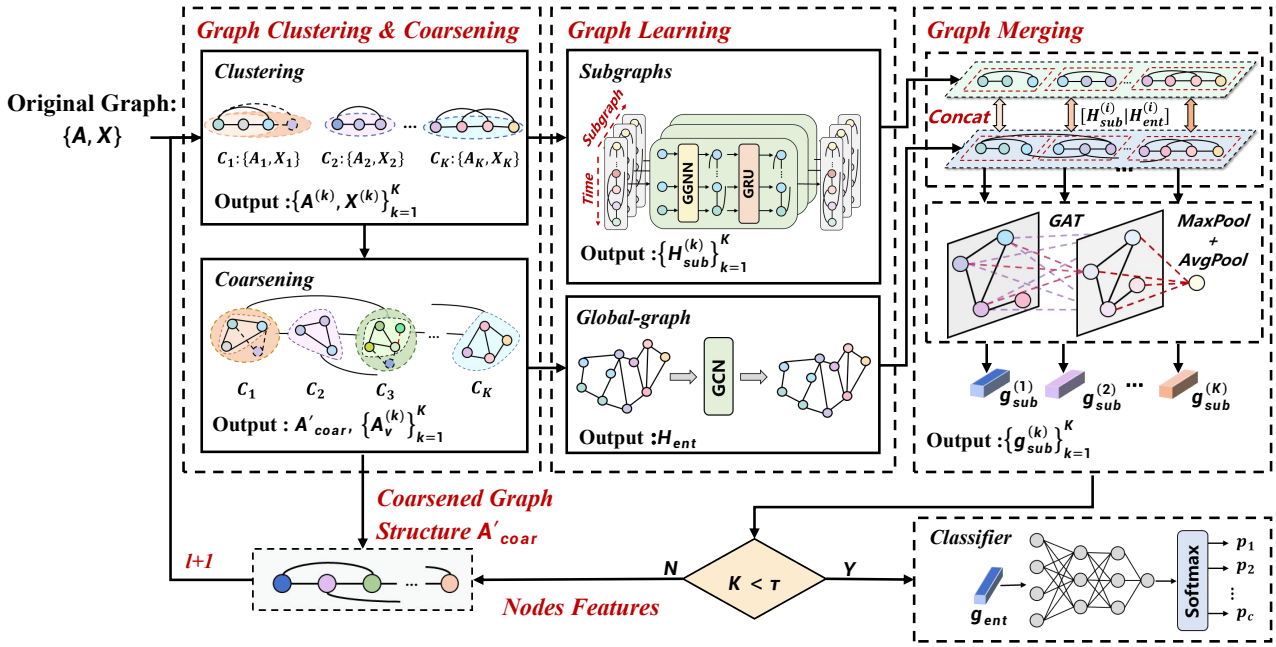

Figure 2: The HC2-GNN framework consists of three main components: graph clustering and coarsening, graph learning, and graph merging. First, the graph clustering and coarsening module partitions the input graph into subgraphs. Next, the learning module employs diverse GNNs to update the nodes within the graph. Finally, the merging module integrates the updated results into the graph/subgraph representation. This process iteratively repeats until the number of subgraphs K drops below a predefined threshold $\mathcal{T}$.

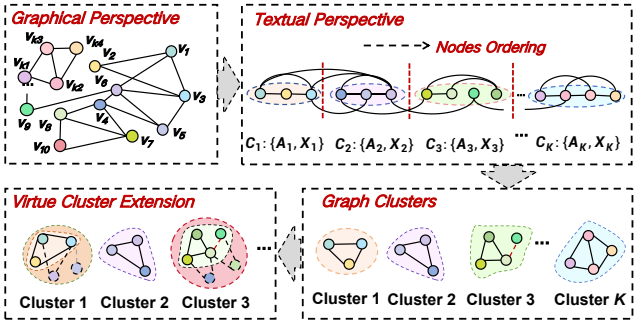

Figure 3: The workflow of the C2GC and the virtual cluster extension from both graphical and textual perspectives.

Gated GNN (GGNN) (Li et al. 2015) with Gated Recurrent Units (GRU) (Cho et al. 2014) to the subgraphs, capturing both structural and sequential information. For the global graph, we apply a Graph Convolutional Network (GCN) (Kipf and Welling 2016a) to update node embeddings. This flexible design supports multi-layer stacking and is extendable to alternative architectures such as Transformers. Subsequently, we concatenate the resulting node embeddings from the subgraph and the global graph and pass through a Graph Attention Network (GAT) (Velikovi et al. 2017) along with max-pooling and average-pooling operations to generate subgraph-level embeddings in the graph merging module. This hierarchical and iterative process produces a comprehensive, holistic graph representation that captures both local and global contextual semantics.

Graph Clustering via Compromise Conductance

Inspired by (Lin, Li, and Jia 2023; Leskovec, Lang, and Mahoney 2010), we assess the graph partitioning quality through conductance. Unlike standard graph clustering methods, we incorporate the linear word order of text, which provides additional semantic context not reflected in the graph structure. However, the pursuit of optimal conductance is known as NP-hard (Chawla et al. 2006), posing considerable computational difficulties. Moreover, strictly optimal partitions are not always necessary for semantic modeling. To balance quality and efficiency, we propose the Compromise Conductance Graph Clustering (C2GC) algorithm.

As shown in Figure 3, nodes are arranged in their original textual order. We sequentially accumulate nodes and compute conductance for each potential split point. Subsequently, partition the graph at the point of minimum conductance and update the candidate subgraphs' conductance iteratively until the target number of subgraphs is achieved. Distinct from conventional graph partitioning, our method avoids seeking global optimality and instead preserves both sequential coherence and structural integrity within each subgraph. The workflow of the graph partitioning is depicted in Figure 3, while Figure 1 provides complementary illustrations of conductance calculation from both textual and graphical perspectives. To control cluster sizes, we introduce a hyperparameter β, which restricts splitting unless a clus-

Algorithm 1: Compromise Conductance Graph Clustering

Require: Undirected graph $G(V, E)$, desired number of clusters K
Ensure: A list of clusters $subgraph\_list$

$V \leftarrow \{u_1, u_2, \ldots, u_n\}$ {arrange nodes}
$subgraph\_list \leftarrow [V]$ {initial graph}
while $len(subgraph\_list) < K$ **do**
  if $\exists\, len(T_i) > \beta \cdot avg\_len(subgraph\_list)$ **then**
    $T_k \leftarrow subgraph\_list.remove(T_i)$
  else
    $T_k \leftarrow subgraph\_list.pop()$
  end if
  $S \leftarrow [\,]$, $Temp\_c \leftarrow$ Null
  for all $u_i \in T_k$ **do**
    $S.append(u_i)$
    $Temp\_c[u_i] \leftarrow Cal\_Conduct(S)$
  end for
  $k \leftarrow Get\_Index(\min(Temp\_c))$
  $V_1 \leftarrow T_k[:k]$, $V_2 \leftarrow T_k[k:]$
  $(C_1, C_2) \leftarrow Gen\_Map\_Opt(V_1, V_2)$ {Generate mapping operators}
  $(A_1, A_2) \leftarrow Gen\_Adj\_Mat(V_1, V_2)$ {Generate adjacency matrices}
  $(A_1', A_2') \leftarrow Maint\_Con(A_1, A_2)$ {Maintain connectivity}
  $subgraph\_list.append(V_1)$
  $subgraph\_list.append(V_2)$
end while
return $subgraph\_list$

ter exceeds β times the average length. Otherwise, decisions are made based on conductance. This mechanism ensures balanced partitioning throughout the process. The details are provided in Algorithm 1.

Virtue Cluster Extension

While sequential conductance-based clustering yields coherent subgraphs, it may exclude semantically critical nodes with strong inter-cluster connections. Moreover, once subgraphs are formed, message passing tends to be confined within clusters, leading to a bottleneck in global information propagation. To address this, we employ a hop-extension algorithm to construct virtue clusters, which expand each subgraph with semantically relevant neighbors from outside.

We identify all n-hop neighbors of a subgraph and rank them by cumulative edge weight connecting them to the original cluster. Nodes are added in descending order, with conductance recalculated after each inclusion. When conductance stabilizes or increases, the extension halts, and the resulting node set forms the V-cluster. This strategy enables explicit cross-cluster information propagation without compromising local structural purity. The complete workflow for V-cluster generation is detailed in Algorithm 2.

Graph Coarsening

After partitioning the graph into K subgraphs, we perform graph coarsening mathematically. Let N_k denote the number

Algorithm 2: Virtue Cluster Extension

Input: A subgraph list $subgraph\_list$, hops n
Output: V-cluster list $V\_cluster\_list$, mask list $Mask\_list$
$V\_cluster\_list \leftarrow$ `Null`, $Mask\_list \leftarrow$ `Null`
while $subgraph\_list \neq$ `Null` **do**
  $V_1 \leftarrow subgraph\_list.$`pop()`, $nhop\_outside \leftarrow$ `Null`
  for $h = 1$ **to** n **do**
    for all $u \in V_1$ **do**
      $hop\_neighbors \leftarrow$ `neighbors`(u, h)
      $nhop\_outside.$`add`$(hop\_neighbors)$
    end for
    $nhop\_outside \leftarrow nhop\_outside \setminus V_1$
  end for
  `Weight_cum_Sort`$(nhop\_outside)$
  $c \leftarrow$ `Cal_Conduct`(V_1), $V\_temp \leftarrow$ `Null`
  while $nhop\_outside \neq$ `Null` **do**
    $v \leftarrow nhop\_outside.$`pop()`
    $V\_candidate \leftarrow V_1 \cup \{v\}$
    $c\_new \leftarrow$ `Cal_Conduct`$(V\_candidate)$
    if $Tolerance(c\_new, c)$ **then**
      break {beyond tolerance}
    end if
    $c \leftarrow c\_new$, $V\_temp \leftarrow V\_candidate$
  end while
  $Av \leftarrow$ `Gen_Adj_matrix`(V_temp)
  $Mask \leftarrow$ `Gen_mask_adj`(V_1, V_temp)
  $V\_cluster\_list.$`append`(V_temp)
  $Mask\_list.$`append`$(Mask)$
end while
Return: $V\_cluster\_list, Mask\_list$

of nodes in the subgraph $\mathbf{G}^{(k)}$, and let $\Gamma^{(k)}$ be the list of nodes in $\mathbf{G}^{(k)}$. For each subgraph $\mathbf{G}^{(k)}$, the mapping matrix $\mathbf{C}^{(k)} \in \mathbb{R}^{N \times N_k}$ is given by:

$$\mathbf{C}^{(k)}[i, j] = \begin{cases} 1 & \text{if } \Gamma^{(k)(j)} = v_i, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $\mathbf{C}^{(k)}[i, j]$ denotes the element in the (i, j)-th position, and $\Gamma^{(k)(j)}$ is the j-th element in the node list $\Gamma^{(k)}$. This matrix indicates the relation between nodes in the subgraph $G^{(k)}$ and the original graph. $\mathbf{C}^{(k)}$ and $\Gamma^{(k)}$ can be precalculated during graph clustering stage (Algorithm 1). Using the mapping, the induced adjacency matrix $\mathbf{A}^{(k)} \in \mathbb{R}^{N_k \times N_k}$ of the subgraph $G^{(k)}$ can be obtained as:

$$\mathbf{A}^{(k)} = \mathbf{C^{(k)}}^T \mathbf{A} \mathbf{C}^{(k)}. \tag{2}$$

The intra-subgraph adjacency matrix for the graph G is defined as the sum over all subgraphs:

$$\mathbf{A}_{\text{int}} = \sum_{k=1}^{K} \mathbf{C}^{(k)} \mathbf{A}^{(k)} \mathbf{C^{(k)}}^T. \tag{3}$$

The inter-subgraph adjacency matrix, as the complement within $\mathbf{A}$ representing edges between subgraphs can be defended by:

$$\mathbf{A}_{\text{ext}} = \mathbf{A} - \mathbf{A}_{\text{int}}. \tag{4}$$

To model the inter-subgraph connectivity in a compressed form, we introduce an assignment matrix $\mathbf{S} \in \mathbb{R}^{N \times K}$ which indicates whether a node belongs to a specific subgraph such that:

$$\mathbf{S}[i,j] = \begin{cases} 1, & \text{if } v_i \in \Gamma^{(j)}, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

With the assignment, the coarsened adjacency matrix $\mathbf{A}_{\text{coar}} \in \mathbb{R}^{K \times K}$, which captures the structural relations among clusters, is given by:

$$\mathbf{A}_{\text{coar}} = \mathbf{S}^T \mathbf{A}_{\text{ext}} \mathbf{S}. \tag{6}$$

Each entry of $\mathbf{A}_{\text{coar}}$ reflects the density of edges between clusters. Even if two clusters share only one light-weight original edge, the corresponding entry is non-zero. To improve sparsity and highlight salient connections, we apply row normalization to $\mathbf{A}_{\text{coar}}$ and set a pruning threshold α. Edges with weights below α, except for key edges (whose removal would isolate a node), are removed. The processing is as follows:

$$\mathbf{A'}_{\text{coar}} = \text{Norm}\Big(\mathbf{A}_{\text{coar}} \odot \mathbf{1}_{\{E_{ij} \geq \alpha \text{ or key edge}\}}\Big). \tag{7}$$

Since the C2GC algorithm compromises subgraph connectivity under strict sequential constraints, we ensure subgraph connectivity by adding auxiliary edges. Isolated nodes are linked to pre-existing nodes efficiently within the subgraph to form a connected structure, yielding subgraph adjacency matrix and corresponding V-cluster matrix $\mathbf{A_v}^{(k)}$.

Graph Learning

Given the distinct structural characteristics of subgraphs and the global graph, we adopt different GNN architectures. To leverage both sequential and topological structure of each subgraph, we utilize GGNN (Li et al. 2015) integrated with GRU. The GRU sequentially updates the subgraph, while GGNN enables message propagation through a gated mechanism that controls information flow across neighboring nodes. To support multi-hop message passing, we stack the GGNN-GRU block $\mathbf{t}$ times. The update equation of GGNN for a V-cluster is formulated as:

$$\begin{aligned} \mathbf{a}_t &= \mathbf{A_v}^{(k)} \mathbf{H}_{t-1}^{(k)} \mathbf{W}_a, \\ \mathbf{z}_t &= \sigma(\mathbf{W}_z \mathbf{a}_t + \mathbf{U}_z \mathbf{H}_{t-1}^{(k)} + \mathbf{b}_z), \\ \mathbf{r}_t &= \sigma(\mathbf{W}_r \mathbf{a}_t + \mathbf{U}_r \mathbf{H}_{t-1}^{(k)} + \mathbf{b}_r), \\ \hat{\mathbf{H}}_t^{(k)} &= \tanh(\mathbf{W}_a \mathbf{a}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{H}_{t-1}^{(k)}) + \mathbf{b}_h), \\ \mathbf{H}_t^{(k)} &= \mathbf{z}_t \odot \hat{\mathbf{H}}_t^{(k)} + (1 - \mathbf{z}_t) \odot \mathbf{H}_{t-1}^{(k)}, \end{aligned} \tag{8}$$

where $\mathbf{A_v}^{(k)}$ is the adjacency matrix of the V-cluster associated with the subgraph, $\sigma(\cdot)$ denotes the sigmoid function, $\odot$ denotes element-wise multiplication, and $\mathbf{W}, \mathbf{U}$, and $\mathbf{b}$ represent trainable weights and biases. The terms $\mathbf{z}$ and $\mathbf{r}$ serve as the update and reset gates, respectively. Since the V-cluster may lose the inherent sequential structure, we apply a masking operator (constructed in Algorithm 2) to recover the original subgraph and perform GRU updates accordingly.

$$\mathbf{H}_{sub}^{(k)} = \mathbf{GRU}(\mathbf{Mask}^{(k)} \odot \mathbf{H}_t^{(k)}). \tag{9}$$

For the full graph, which contains numerous nodes and edges with rich structural information and long-range dependencies, we employ GCN to learn the node embeddings:

$$\mathbf{H}_{\mathbf{ent}}^{\ell+1} = f(\mathbf{A}^{\ell} \mathbf{W}^{\ell} \mathbf{H}^{\ell}) \tag{10}$$

where $\mathbf{A}^{\ell}$ is the normalized adjacency matrix at l_{th} layer of the coarsened graph. Specifically, the first layer ($l = 0$) corresponds to the original adjacency matrix A, while subsequent layers ($l \geq 1$) correspond to the coarsened adjacency matrices $\mathbf{A'}_{\text{coar}}^{\ell}$. $\mathbf{W}^{\ell}$ is the trainable weight matrix, and f is a non-linear activation function, typically ReLU. This hierarchical dual-branch design allows HC2-GNN to capture both fine-grained local and abstract global semantics, improving expressive power for downstream classification.

Graph Merging

After updating the subgraph and global graph representations through their respective encoders, we merge the resulting node embeddings to construct a unified semantic representation accordingly. As illustrated in Figure 2, the node embeddings from the global graph and each corresponding subgraph or V-cluster are concatenated and passed through a GAT (Velikovi et al. 2017). This strategy enriches the representation space by integrating fine-grained local information with global context. The combination is defined as follows:

$$\begin{aligned} \tilde{\mathbf{H}}^{(k)} &= \text{GAT}\big(\text{concat}\big(\mathbf{H}_{\text{sub}}^{(k)}, \mathbf{H}_{\text{ent}}^{(k)}\big)\big), \\ \mathbf{g}_{\text{sub}}^{(k)} &= \frac{1}{|\mathbf{G}^{(k)}|} \sum_{v \in \mathbf{G}^{(k)}} \tilde{\mathbf{h}}_v^{(k)} + \max_{v \in \mathbf{G}^{(k)}} \tilde{\mathbf{h}}_v^{(k)}, \end{aligned} \tag{11}$$

where $\mathbf{H}_{\mathbf{ent}}^{(k)} \in \mathbb{R}^{N_k \times d_g}$ and $\mathbf{H}_{\mathbf{sub}}^{(k)} \in \mathbb{R}^{N_k \times d}$ represent the node embeddings from the global graph and the corresponding subgraph $\mathbf{G}^{(k)}$, $\tilde{\mathbf{h}}_v^{(k)}$ is the embedding of node v in $\tilde{\mathbf{H}}^{(k)}$, $\mathbf{g}_{\text{sub}}^{k}$ represents the subgraph-level embedding, obtained by combining average and max pooling results.

After stacking the model iteratively, we concatenate the subgraphs to generate a comprehensive representation $\mathbf{g}_{\text{ent}}$ for the full graph. Finally, for the downstream text classification task, we apply a fully connected (FC) layer followed by a softmax function to predict the class probabilities:

$$\hat{\mathbf{Y}}_i = \text{softmax}(\text{FC}(\mathbf{g}_{\text{ent}})), \tag{12}$$

$$\mathcal{L} = -\sum_{c=1}^{C} \mathbf{Y}_c \log(\hat{\mathbf{Y}}_c), \tag{13}$$

where C is the number of target classes. $\hat{\mathbf{Y}}_c$ denotes the predicted probability of class c, and $\mathcal{L}$ is the cross-entropy loss.

Experiments

Datasets and Baselines

To evaluate the performance of HC2-GNN, we adopt five benchmark datasets: MR, Ohsumed, 20NG, R8, and R52. We compare it against ten baselines, categorized into two groups: traditional models and transformer-enhanced models. For a fair comparison within the transformer-enhanced

category, we replace the GNN components in HC2-GNN with G-Transformer (Dwivedi and Bresson 2020) and denote it as HC2-GNN-GT. **Group 1:Traditional Models TextGCN** (Yao, Mao, and Luo 2018b); **TextRNN** (Liu, Qiu, and Huang 2016); **FastText** (Joulin et al. 2017); **SWEM** (Shen et al. 2018); **TextING** (Zhang et al. 2020b); **SGC** (Wu et al. 2019). **Group 2 (Transformer-Enhanced Models) BertGCN** (Lin et al. 2021); **BertGAT** (Lin et al. 2021); **MSABertGCN** (Yang and Liu 2023); **ConTextING-BT** (Huang, Chen, and Chen 2022).

Experimental Settings

To ensure consistency, we randomly split the training set of each dataset into a ratio of 9:1 for training and validation. The clustering and coarsening modules are stacked twice by default. We also set an early stopping threshold of 25 nodes, terminating further stacking when the graph becomes sufficiently small. Other hyperparameters are tuned according to validation performance. We use the Adam optimizer with a learning rate of 0.005 and a dropout rate of 0.5, choose the pre-trained GloVe (Pennington, Socher, and Manning 2014) with 300 dimensions as the word embedding. The out-of-vocabulary (OOV) words are randomly initialized from a uniform distribution in the range [-0.01, 0.01].

Performance Comparison

Group 1: Traditional Models We compare our HC2-GNN against several baseline models in terms of text classification accuracy. Table 1 summarizes the results, where the baseline values listed are state-of-the-art on these datasets. HC2-GNN consistently outperforms existing models across the board. Specifically, compared to TextING (an inductive model that leverages individual text structures), HC2-GNN achieves improvements by margins of 2.7% on Ohsumed and 4.1% on 20NG, along with slight gains on the remaining datasets. For the MR dataset, which contains short texts averaging only 18 words, clustering and coarsening are less effective due to the concentration on a few meaningful characters and the sparse semantic structure. As a result, most models that leverage the individual text structure show marginal differences in performance. Since R8 and R52 are simple, most GNN-based baselines perform satisfactorily. The Ohsumed dataset, which contains moderate-length texts but with a limited training set and medical domain complexity,

Model	MR	R8	R52	Ohsumed	20NG
FastText	75.14 ± 0.20	96.13 ± 0.21	92.81 ± 0.09	57.70 ± 0.49	79.67 ± 0.29
TextRNN	77.68 ± 0.86	96.31 ± 0.33	90.54 ± 0.91	49.27 ± 1.07	75.43 ± 1.72
SWEM	76.65 ± 0.63	95.32 ± 0.26	92.94 ± 0.24	63.12 ± 0.55	85.16 ± 0.29
TextGCN	76.74 ± 0.20	97.07 ± 0.10	93.56 ± 0.18	68.36 ± 0.56	86.34 ± 0.06
TextING	79.82 ± 0.20	98.04 ± 0.25	95.48 ± 0.19	69.70 ± 0.45	85.47 ± 0.56
SGC	75.34 ± 0.65	97.15 ± 0.32	93.90 ± 0.28	68.24 ± 0.69	88.32 ± 0.33
HC2-GNN	**80.22 ± 0.31**	**98.21 ± 0.27**	**95.97 ± 0.35**	**72.47 ± 0.51**	**89.66 ± 0.39**

Table 1: Text Classification accuracies(%) on group 1. It illustrates the accuracy in the mean ± standard deviation. We ran the model 10 times. The best performance per column is in bold. Some baseline results are from (Yao, Mao, and Luo 2018a).

Model	MR	R52	Ohsumed	20NG
BertGAT	86.24 ± 0.40	95.70 ± 0.33	70.84 ± 0.42	87.22 ± 0.53
BertGCN	85.35 ± 0.72	96.55 ± 0.48	72.40 ± 0.46	88.75 ± 0.19
ConTextING-BT	86.01 ± 0.35	96.45 ± 0.62	71.28 ± 0.25	86.19 ± 0.37
MSABertGCN	86.87 ± 0.24	**96.68 ± 0.36**	74.72 ± 0.65	–
HC2-GNN-GT	**87.25 ± 0.32**	96.40± 0.55	**75.32 ± 0.40**	**91.26 ± 0.39**

Table 2: Text Classification accuracies(%) on group 2. We run each model three times to ensure reliability. Some baseline results are from (Lin et al. 2021) and (Li et al. 2025).

constrains the performance of most models, except GNN-based baselines with strong generalization ability. On the long-text 20NG dataset, which is rich in structural and sequential information, HC2-GNN achieves the most significant improvements, validating its effectiveness in modeling complex dependencies.

Group 2: Transformer-Enhanced Models Transformer-enhanced models significantly outperform traditional approaches by leveraging pre-trained language representations and integrating transformer architectures with GNNs to capture both long-range dependencies and graph-based structures. As shown in Table 2, HC2-GNN-GT incorporates clustering and coarsening to emphasize local semantics, while global merging enhances the integration of contextual features. This design leads to substantial performance improvements, particularly on 20NG and Ohsumed. For the short-text dataset MR, however, the scarcity of structural information limits the benefits of graph-based modeling. On R52, all models achieve comparable results due to the dataset's relative simplicity. Although transformer-enhanced approaches require longer training time to converge compared to traditional models, they consistently deliver superior accuracy due to the powerful feature extraction capability of transformer encoders, which remain effective even on datasets with limited contextual or structural information.

Ablation Study

Effect of Graph Clustering

To evaluate the effectiveness of the clustering component in HC2-GNN, we conduct ablation experiments using several model variants. In one variant, we disregard the node sequencing and partition the graph using the conductance approximation algorithm proposed in (Lin, Li, and Jia 2023), denoted as **HC2-GNN-G**, which retains only the graph topology. In another variant, we removed the entire clustering module, resulting in a simplified model **HC-GNN** that operates directly on the original graph structure. To isolate the effect of the virtue cluster (V-cluster) mechanism in C2GC, we created another variant **HC2-GNN-sub**, which disables the cluster expansion phase, restricting message passing within the original subgraph boundaries. For simplicity, all variants adopt a single layer of graph clustering and coarsening. Furthermore, to highlight the contribution of structural and sequential information, we construct a modified dataset, **20NG-L**, by selecting the 2000 longest texts from the original 20NG dataset and retaining approximately 1500 instances after label balance filtering. As demonstrated in Table 3, HC2-GNN significantly outperforms HC-GNN

Model	20NG	20NG-L
HC-GNN (w/o clustering)	86.5± 0.3	88.3± 0.2
HC2-GNN-G (w/o sequentiality)	87.9± 0.3	90.7± 0.6
HC2-GNN-sub (w/o V-cluster)	88.4± 0.6	92.6± 0.3
HC2-GNN	89.6± 0.4	94.3± 0.5

Table 3: Accuracies(%) of the variant HC2-GNN models for clustering ablation

on both datasets, confirming the necessity of graph clustering. While the performance of HC2-GNN-G is comparable to HC2-GNN on the 20NG dataset, it drops significantly on 20NG-L. It indicates that sequential information becomes increasingly critical as document length and structural complexity grow. The variant HC2-GNN-sub also underperforms compared to the HC2-GNN model, with margins of 1.2% and 1.7% on 20NG and 20NG-L, respectively, indicating the importance of the virtue cluster mechanism, which enables effective and reasonable clustering, and facilitates information flow across clusters.

Effect of Graph Merging

To investigate the graph merging mechanism, we conduct a set of ablation experiments on additional model modifications. First, we remove the global graph branch that prevents the model from leveraging global contextual information, denoted as **HC2-local**. In the second variant, we simplify the HC2-local model by replacing the GRU-GGNN block with GCN, termed **HC2-Simp**(w/o sequentiality), disregarding the sequential modeling capability within each subgraph. Finally, we modify HC2-Simp by applying the rigid conductance-based partitioning algorithm from (Lin, Li, and Jia 2023), resulting in **HC2-Cond**. It replaces our proposed C2GC module, thereby removing both sequential preservation and semantic-aware expansion.

For consistency, each variant uses a single layer of clustering and coarsening, and evaluations were conducted on both the 20NG and 20NG-L datasets. As shown in Table 4, HC2-local outperforms HC2-Simp across both datasets, confirming that GRU-GGNN provides stronger representation power by modeling sequential dependencies within subgraphs, which highlights the necessity of preserving sequential information. The variant HC2-Cond performs comparably to HC2-Simp or slightly better, likely due to its more principled graph partitioning strategy. However, it still falls short of HC2-GNN, which benefits from the full integration of sequence, structure, global context, and facile information

Model	20NG	20NG-L
HC2-local(w/o global)	86.8± 0.4	90.7± 0.5
HC2-Simp(w/o global&sequentiality)	85.7± 0.2	86.4± 0.3
HC2-Cond(rigid conductance)	86.2± 0.3	87.2± 0.6
HC2-GNN	89.6± 0.4	94.3± 0.5

Table 4: Accuracies(%) of the variant HC2-GNN models for merging ablation

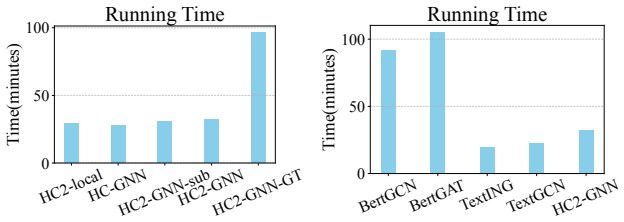

Figure 4: Running time of two epoches on dataset 20NG of various models. (a) The variant HC2-GNN models (left). (b) Representative baslines in both transformer-enhanced and traditional models (right).

flow. Notably, HC2-GNN achieves the highest performance on both datasets, surpassing HC2-Simp by a margin of 3.9% on 20NG, with even greater gains on 20NG-L. These improvements highlight that combining global graph features, locally subgraph structure, sequential awareness, and cross-cluster message passing is essential for effective text classification, especially in long-text sets.

Performance Analysis

Efficiency Analysis

Our evaluation focuses on training costs, which are critical for real-world scalability. We measure the running time from two perspectives: among different HC2-GNN variants and in comparison with representative baselines. All experiments were conducted on a workstation equipped with 64 GB memory and an Nvidia 4090 GPU, using a batch size of 128. We report the runtime per two training epochs on the 20NG dataset to ensure comparability.

As shown in Figure 4(a), the time magnitude of the HC2-GNN variants remains consistent, except for HC2-GNN-GT, which incurs noticeably higher costs due to the transformer modules. It confirms that the clustering and coarsening mechanisms introduce minimal overhead. Figure 4(b) illustrates the clear computation cost gap between transformer-enhanced and traditional models. Transformer-enhanced models, while achieving higher accuracy, generally require substantially longer training time and more epochs to converge. In contrast, HC2-GNN achieves a favorable trade-off between training efficiency and predictive performance, making it competitive and practical.

Conclusion

We present HC2-GNN, a novel graph-based model for efficient and expressive text classification. Core to our framework is C2GC, a lightweight clustering algorithm that preserves both structural coherence and word order, along with a virtue cluster mechanism that facilitates cross-cluster information flow. Extensive experiments on five public datasets show that HC2-GNN outperforms traditional GNNs and transformer-enhanced baselines, especially on long and structurally rich texts. Ablation and efficiency studies further validate the effectiveness and scalability of each component.

Acknowledgments

This work was supported by National Major Science and Technology Projects of China under Grant No. 2024ZD0608100 and the National Natural Science Foundation of China under Grant No. U22A2022.

References

Andersen, R.; Chung, F.; and Lang, K. 2006. Local graph partitioning using pagerank vectors. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, 475–486.

Chang, L.; and Qin, L. 2019. Cohesive Subgraph Computation Over Large Sparse Graphs. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 2068–2071.

Chawla, S.; Krauthgamer, R.; Kumar, R.; Rabani, Y.; and Sivakumar, D. 2006. ON THE HARDNESS OF APPROXIMATING MULTICUT AND SPARSEST-CUT. In *Computational Complexity*, p.94–114.

Chen, J.; Ma, T.; and Xiao, C. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. *arXiv preprint arXiv:1801.10247*.

Chen, W.; Banerjee, A.; and Li, S.-Y. 2012. Efficient Graph-Based Clustering via Node Sampling and Coarsening. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 541–549. ACM.

Cho, K.; Van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734.

Dwivedi, V. P.; and Bresson, X. 2020. A Generalization of Transformer Networks to Graphs.

Gao, Z.; Bhattacharya, S.; Zhang, L.; Blum, R. S.; Ribeiro, A.; and Sadler, B. M. 2021. Training Robust Graph Neural Networks with Topology Adaptive Edge Dropping. *arXiv preprint arXiv:2106.02892*.

Gionis, A.; Faloutsos, M.; and Faloutsos, C. 2005. Graph Coarsening for Community Detection. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, 563–564. ACM.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NIPS)*, 1024–1034.

Huang, Y.-H.; Chen, Y.-H.; and Chen, Y.-S. 2022. ConTextING: Granting Document-Wise Contextual Embeddings to Graph Neural Networks for Inductive Text Classification. In Calzolari, N.; Huang, C.-R.; and Kim, H. a., eds., *Proceedings of the 29th International Conference on Computational Linguistics*, 1163–1168. Gyeongju, Republic of Korea: International Committee on Computational Linguistics.

Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2017. Bag of Tricks for Efficient Text Classification. *arXiv preprint arXiv:1607.01759*.

Kang, U.; and Faloutsos, C. 2006. Node Dropping: A New Sampling-Based Approach to Graph Clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 367–376. ACM.

Kipf, T. N.; and Welling, M. 2016a. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations (ICLR)*.

Kipf, T. N.; and Welling, M. 2016b. Variational Graph Auto-Encoders. ArXiv:1611.07308.

Leskovec, J.; Lang, K. J.; and Mahoney, M. W. 2010. Empirical comparison of algorithms for network community detection. *ACM*.

Li, P.; Fu, X.; Chen, J.; and Hu, J. 2025. CoGraphNet for enhanced text classification using word-sentence heterogeneous graph representations and improved interpretability. *Scientific Reports*, 15(1): 1–15.

Li, P.; Zhang, Y.; and Wu, J. 2019. Edge-Dropping for Graph Clustering: A Theoretical and Experimental Study. *arXiv preprint arXiv:1901.05555*.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. 2015. Gated Graph Sequence Neural Networks. *Computer Science*.

Lin, L.; Li, R.; and Jia, T. 2023. Scalable and effective conductance-based graph clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 37, 4471–4478.

Lin, Y.; Meng, Y.; Sun, X.; Han, Q.; and Wu, F. 2021. BertGCN: Transductive Text Classification by Combining GNN and BERT. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP*.

Liu, P.; Qiu, X.; and Huang, X. 2016. Recurrent Neural Network for Text Classification with Multi-Task Learning. *AAAI Press*.

Liu, X.; You, X.; Zhang, X.; Wu, J.; and Lv, P. 2020. Tensor Graph Convolutional Networks for Text Classification. *World Wide Web*, (Aug).

M.; E.; J.; and Newman. 2004. Fast algorithm for detecting community structure in networks. *Physical Review E*.

Newman, M. E. J. 2006. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103(23): 8577–8582.

Nikolentzos, G.; Tixier, A.; and Vazirgiannis, M. 2020. Message Passing Attention Networks for Document Understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 8544–8551.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Pham, H. V.; Thanh, D. H.; and Moore, P. 2021. Hierarchical Pooling in Graph Neural Networks to Enhance Classification Performance in Large Datasets. *Sensors*, (18).

Piao, Y.; Lee, S.; Lee, D.; and Kim, S. 2021. Sparse Structure Learning via Graph Neural Networks for Inductive Document Classification.

Shen, D.; Wang, G.; Wang, W.; Renqiang, M.; and Carin, L. 2018. Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. In *Association for Computational Linguistics (ACL)*.

Velikovi, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2017. Graph Attention Networks. *arXiv preprint arXiv:1710.10903*.

Wang, K.; Han, S. C.; Long, S.; and Poon, J. 2022. ME-GCN: Multi-dimensional Edge-Embedded Graph Convolutional Networks for Semi-supervised Text Classification. *arXiv preprint arXiv:2204.04618*.

Wu, F.; Zhang, T.; Souza, A. H. D.; Fifty, C.; Yu, T.; and Weinberger, K. Q. 2019. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Wu, Z.; Hu, W.; and Zhang, Y. 2021. GraphBERT: Only Attention is Needed for Learning Graphs. In *Proceedings of the 37th International Conference on Machine Learning*, 5236–5245.

Xu, X.; Yuruk, N.; Feng, Z.; and Schweiger, T. A. J. 2007. SCAN: a structural clustering algorithm for networks. In *ACM*, 824–833.

Yang, J.; and Leskovec, J. 2012. Defining and Evaluating Network Communities Based on Ground-Truth. In *IEEE International Conference on Data Mining*, 1–8.

Yang, R.; Xiao, X.; Wei, Z.; Bhowmick, S. S.; Zhao, J.; and Li, R. H. 2019. Efficient Estimation of Heat Kernel PageRank for Local Clustering. In *Proceedings of the 2019 International Conference on Management of Data*.

Yang, X.; and Liu, W. 2023. Maximal-Semantics-Augmented BertGCN for Text Classification. *International Journal of Asian Language Processing*, 33(02).

Yao, L.; Mao, C.; and Luo, Y. 2018a. Graph Convolutional Networks for Text Classification. In *The Association for the Advancement of Artificial Intelligence*.

Yao, L.; Mao, C.; and Luo, Y. 2018b. Graph Convolutional Networks for Text Classification. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*.

Zhang, X.; Wang, B.; Wang, X.; and Sun, X. 2020a. Hierarchical Graph Representation Learning for Text Classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Zhang, Y.; Yu, X.; Cui, Z.; Wu, S.; Wen, Z.; and Wang, L. 2020b. Every Document Owns Its Structure: Inductive Text Classification via Graph Neural Networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Zhu, H.; and Koniusz, P. 2021. Simple Spectral Graph Convolution. In *International Conference on Learning Representations (ICLR)*.