Semi-Supervised Hierarchical Graph Classification

Jia Li, Yongfeng Huang, Heng Chang and Yu Rong

Abstract—Node classification and graph classification are two graph learning problems that predict the class label of a node and the class label of a graph respectively. A node of a graph usually represents a real-world entity, e.g., a user in a social network, or a document in a document citation network. In this work, we consider a more challenging but practically useful setting, in which a node itself is a graph instance. This leads to a hierarchical graph perspective which arises in many domains such as social network, biological network and document collection. We study the node classification problem in the hierarchical graph where a "node" is a graph instance. As labels are usually limited, we design a novel semi-supervised solution named SEAL-CI. SEAL-CI adopts an iterative framework that takes turns to update two modules, one working at the graph instance level and the other at the hierarchical graph level. To enforce a consistency among different levels of hierarchical graph, we propose the Hierarchical Graph Mutual Information (HGMI) and further present a way to compute HGMI with theoretical guarantee. We demonstrate the effectiveness of this hierarchical graph modeling and the proposed SEAL-CI method on text and social network data.

Index Terms—graph representation, graph mutual information, hierarchical graph, semi-supervised learning.

1 Introduction

Graph has been widely used to model real-world entities and the relationship among them. Two graph learning problems have received a lot of attention recently, i.e., node classification and graph classification. Node classification is to predict the class label of nodes in a graph, for which many studies [1]–[3] in the literature make use of the connections between nodes to boost the classification performance. For example, [4] enhances the recommendation precision in LinkedIn by taking advantage of the interaction network, and [5] improves the performance of document classification by exploiting the citation network. Graph classification, on the other hand, is to predict the class label of graphs, for which various graph kernels [6]–[9] and deep learning approaches [10]–[12] have been designed. In this work, we consider a more challenging but practically useful setting, in which a node itself is a graph instance. This leads to *a hierarchical graph in which a set of graph instances are interconnected via edges*. This is a very expressive data representation, as it considers the relationship between graph instances, rather than treating them independently. The hierarchical graph model applies to many real-world data, for example, a social network can be modeled as a hierarchical graph, in which a user group is represented by a graph instance and treated as a node in the hierarchical graph, and then a number of user groups are interconnected via interactions or common members. As another example, a document collection can be modeled as a hierarchical graph, in which a document is regarded as a graph-of-words [13], and then a set of documents are interconnected via the citation relationship. In this paper, we study *hierarchical graph classification, which predicts the class label of graph instances in a hierarchical graph*.

To represent a hierarchical graph, there is a natural and important question:"given multiple levels of inputs and representations of a hierarchical graph, how can we enforce a consistency among different levels of the graph?" In this work, we propose to use *mutual information* (MI) to enforce this consistency. We are motivated by recent developments of graph MI maximization methods [14][15], and generalize the MI computation to hierarchical graphs, which is named Hierarchical Graph Mutual Information (HGMI). Our theoretical derivations show that HGMI can be decomposed into a linear combination of node-level MI [14] and graph-level MI [15]. In this regard, we can use non-hierarchical graph MI computational methods to compute HGMI. More specifically, for graph instances, we compute MI between nodes and instance representations via graph-level MI computational methods (e.g., INFOGRAPH [15]); for connections between graph instances, we compute MI between instances and hierarchical graph representations via node-level MI computational methods (e.g., GMI [14]).

Another challenge is that the amount of available class labels is usually very small in real-world data, which limits the classification performance. To address this challenge, we take a semi-supervised learning approach to solving the graph classification problem. We design an iterative algorithm framework which takes turns to update two modules: Instance Classifier (IC) and Hierarchical Classifier (HC). We start with the limited labeled training set and build IC, which produces the embedding vectors of graph instances. HC takes the embedding vectors as input and produces predictions. We cautiously select a subset of predicted labels with high confidence to enlarge the training set. The enlarged training set is then fed into IC in the next iteration

- *Jia Li and Yongfeng Huang are with Hong Kong University of Science and Technology (Guangzhou) and Hong Kong University of Science and Technology.*
  E-mail: jialee@ust.hk
- *Heng Chang is with Tsinghua University.*
- *Yu Rong is the corresponding author and with Tencent AI Lab.*
  E-mail: yu.rong@hotmail.com

to update its parameters in the hope of generating more accurate embedding vectors. HC further takes the new embedding vectors for model update and class prediction. This is our proposed solution, called SEmi-supervised grAph cLassification via Cautious Iteration (SEAL-CI), to the graph classification problem.

Our contributions are summarized as follows.

- We study semi-supervised hierarchical graph classification, which is scarcely studied in the literature. Our proposed solution SEAL-CI achieves superior classification performance to the state-of-the-other graph kernel and deep learning methods, even when given very few labeled training instances.
- We generalize the MI estimation to the hierarchical graph domain and propose a new concept named Hierarchical Graph Mutual Information (HGMI). We show HGMI can be decomposed by a sum of mutual information in non-hierarchical graph domain. Upon this, HGMI maximization can be achieved with the help of node/graph-level MI maximization.
- We present the HIERARCHICAL GRAPH BENCHMARK [1] with both text data and social network data, with the goal of facilitating reproducible hierarchical graph research. From the social networking platform Tencent QQ, we collect 37,836 QQ groups with 18,422,331 unique anonymized users, in which the hierarchical graph is constructed with common memberships (hierarchy-level) and friendships (instance-level). From the arXiv papers, we collect 4,666 papers, in which the hierarchical graph is constructed with citations (hierarchy-level) and semantics (instance-level)

The remainder of this paper is organized as follows. Section 2 discusses the related works and Section 3 gives the problem definition. Section 4 describes the design of SEAL-CI. We report the experimental results in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

Hierarchical graph representation In the literature, most graph representation methods focus on node-level representations [16][17][18][19] or graph-level representations [20][21][22]. One connection between these two kinds of representations is that graph-level representations could be obtained by integrating node-level representations with graph pooling operations, e.g., DIFFPOOL [23], SAGPool [24], Attention-Pool [25], MinCut-Pool [26]. Until recently Li et al. [25] firstly formulate the hierarchical graph representation problem and evaluate the effectiveness of hierarchical graph representation on social network classification tasks. Wang et al. [27] propose a Graph of Graphs Neural Network (GoGNN) and evaluate the proposed GoGNN on entity interaction prediction tasks. Xu et al. [28] consider city-level graph representations and station-level graph representations to facilitate the air quality forecasting tasks. MIRACLE [29] utilizes the multi-view contrast learning method to derive hierarchical graph representations and validated its effectiveness on Drug-Drug Interactions (DDI). However, currently there are still several problems within the hierarchical graph representation research. Firstly, most previous works are based on heuristics and lack the theoretical derivations of consistent representations among different levels of the hierarchical graph. Secondly, most works mildly evaluate the expressness of hierarchical graph representations on one task, which makes the whole story less convincing.

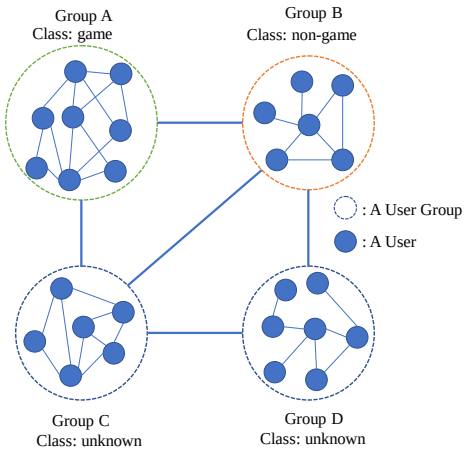

Fig. 1: A hierarchical graph example with four graph instances A, B, C, D, each of which corresponds to a user group in a social network. This figure is adapted from [25].

Graph mutual information maximization There has been a surging interest in using MI to derive unsupervised graph representations recently. For node-level representations, DGI [30] maximizes MI between a graph summery representation and node-level representations and shows that maximizing this kind of MI is equivalent to maximizing the one between the node features and node-level representations. Later, GMI [14] directly approaches MI computation by comparing the node input and node-level representations, without the overheads of an extra graph summery representation. For graph-level representations, INFOGRAPH [15] and MVGRL [31] extend the idea of DGI to the field of graph-level representation and contrasts graph-level and subgraph-level representations. Our work advances this research area by proposing hierarchical graph MI and develops an tractable computation method with guarantees.

Semi-supervised graph learning In many graph applications, the size of labeled instances is typically limited. In addition to labeled instances, semi-supervised graph learning seeks to leverage unlabeled instances within the network structure to obtain gains in performance. In the literature, semi-supervised graph learning techniques can be generally classified into two groups. The first group is consistency regularization. Representative works include graph Laplacian regularization [32][33], graph contrastive regularization [34][15][35]. This group aims to learn a representation function that gives consistent predictions for similar data points. The second group is pseudo-label. As the name indicates, this group uses a trained model on

1. data and code are available at https://hiergraph.github.io/

the labeled set to produce additional training examples by labeling instances of the unlabeled set [36]. Representative works include ICA [5], Cautious Iteration [37][38]. In ICA, for each node a local classifier takes the estimated labels of its neighborhood and its own features as input, and outputs a new pseudo label. The iteration continues until adjacent estimations stabilize. In this work, our proposed method SEAL-CI connects the idea of both consistency regularization and pseudo-label. We show our algorithm has a convergence property w.r.t. the empirical risk.

3 Problem Definition

We denote a set of objects as $O = \{o_1, o_2, \ldots, o_N\}$ which represent real-world entities. We use d attributes to describe properties of objects, e.g., age, gender.

We use a *graph instance* to model the relationship between objects in O, which is denoted as $g = (V, A, X)$, $V \subseteq O$ is the node set and $|V| = n$, A is an $n \times n$ adjacency matrix representing the connectivity in g, and $X \in \mathbb{R}^{n \times d}$ is a matrix recording the attribute values of all nodes in g.

A set of graph instances $G = \{g_1, g_2, \ldots\}$ can be interconnected, and the connectivity between the graph instances is represented by an adjacency matrix $\mathcal{A}$. The graph instances and their connections are modeled as a *hierarchical graph* $\mathcal{G} = (G, \mathcal{A})$.

A graph instance $g \in G$ is a *labeled graph* if it has a class label, represented by a vector $y \in \{0, 1\}^c$, where c is the number of classes. A graph instance is *unlabeled* if its class label is unknown. Then G can be divided into two subsets: labeled graphs G_l and unlabeled graphs G_u, where $G = G_l \cup G_u$, $|G_l| = L$ and $|G_u| = U$, $Y = \{y_i\}_{i=1}^{L}$ denotes the labeled ground truth. In this paper, we study the problem of ***graph classification***, which determines the class label of the unlabeled graph instances in G_u from the available class labels in G_l and the hierarchical graph topological structure. As the amount of available class labels is usually very limited in real-world data, we take a semi-supervised learning approach to solving this problem.

Our problem formulation is applicable to many real applications as illustrated below.

Application 1 Consider a social network where we want to infer the topics of user groups, as depicted in Figure 1. A, B, C, D denote four user groups. Group A has the class label of "game", B has the label of "non-game", while the class labels of C and D are unknown. These four groups are connected via some kind of relationships, e.g., interactions or common members. The internal structure of each user group shows the connections between individual users. From this hierarchical graph, we want to determine the class labels of groups C and D.

Application 2 Consider an information retrieval scenario where we want to category documents. We can model a document as a graph-of-words, i.e., graph instances in a hierarchical graph. We then connect these documents by their citation relationships, i.e., connections between graph instances in a hierarchical graph. With this setting, we can achieve better classification performance by taking advantage of both out-of-document relations and inside-of-document semantics (See details in Section 5.2).

Application 3 Protein-Protein Interaction (PPI) sheds light on biological processes, of which one important task is to predict properties of proteins. We can model a protein as a graph of amino-acid interactions [39], i.e., graph instances in a hierarchical graph. We then connect these proteins by their interactions, forming a hierarchical graph. With this formulation, we can achieve better prediction performance by considering both out-of-protein interactions and inside-of-protein interactions.

4 Methodology

4.1 Problem Formulation

In our problem setting, we have two kinds of information: graph instances and connections between the graph instances, which provide us with two perspectives to tackle the graph classification problem. Accordingly, we build two classifiers: a classifier IC constructed for graph instances and a classifier HC constructed for the hierarchical graph.

For both classifiers, one goal is to minimize the supervised loss, which measures the distance between the predicted class probabilities and the true labels. Another goal is to maximize a hierarchical graph mutual information, which measures the distance among the input hierarchical graph, IC representations and HC representations. The purpose of this hierarchical graph mutual information is to enforce a ***consistency*** among different levels of hierarchical graph.

Formally, we formulate the graph classification problem as an optimization problem:

$$\arg\min \zeta(G_l) - \xi(\mathcal{G}), \tag{1}$$

where $\zeta(G_l)$ is the empirical risk for the labeled graph instances, and $\xi(\mathcal{G})$ is the hierarchical graph mutual information for all graph instances.

Specifically, $\zeta(G_l)$ includes two parts:

$$\zeta(G_l) = \frac{1}{L} \sum_{g_i \in G_l} (\mathrm{CE}(y_i, e_i) + \mathrm{CE}(y_i, \gamma_i)), \tag{2}$$

where e_i is a vector of predicted class probabilities by IC, and γ_i is a vector of predicted class probabilities by HC. $\mathrm{CE}(\cdot, \cdot)$ is the cross-entropy loss function.

The hierarchical graph mutual information (HGMI) $\xi(\cdot)$ is defined as:

$$\xi(\mathcal{G}) = \mathrm{HGMI}(\mathcal{G}) = I(G; E; \Gamma), \tag{3}$$

where $I(\cdot; \cdot; \cdot)$ denotes the mutual information between a set of three variables, $E = \{e_i\}_{i=1}^{L+U}$ denotes the representations derived by IC and $\Gamma = \{\gamma_i\}_{i=1}^{L+U}$ denotes the representations derived by HC. In the following subsections, we first analyse HGMI and describe the way to compute HGMI; we then give our design of classifiers IC and HC, and our detailed training algorithms.

4.2 Analysis of HGMI

We first show HGMI coincides with graph mutual information between G and Γ when the hierarchical graph forms a Markov chain.

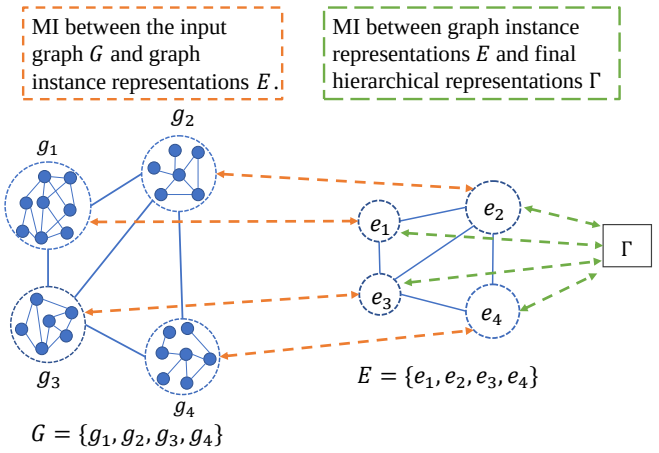

Fig. 2: Overview of the proposed hierarchical graph mutual information computation. The orange part shows the mutual information computation between the input G and graph instance representations E; the green part shows the mutual information computation between graph instance representations E and final hierarchical representations Γ.

Theorem 1. Consider the hierarchical graph forms a Markov chain, i.e., $G \to E \to \Gamma$, then

$$I(G; E; \Gamma) = I(G; \Gamma), \tag{4}$$

here $I(G; \Gamma)$ is the mutual information between the input graph instances and hierarchical graph representations.

The proof is trivial, since the three variables form a Markov chain $G \to E \to \Gamma$, then $I(G; \Gamma|E) = 0$, we have

$$I(G; E; \Gamma) = I(G; \Gamma) - I(G; \Gamma|E) = I(G; \Gamma). \tag{5}$$

It was worth mentioning that the Markov property is quite reasonable here. For example on text data, upon given document representation, it is reasonable to assume graph-of-words inputs (within documents) are irrelevant when predicting document citations (outside of documents).

We have the following hierarchical graph mutual information decomposition theorem to compute HGMI.

Theorem 2. Consider the hierarchical graph forms a Markov chain, the hierarchical graph mutual information can be decomposed by a sum of non-hierarchical graph mutual information, namely,

$$I(G; E; \Gamma) = \alpha(I(G; E) + I(E; \Gamma)), \tag{6}$$

here $\alpha \in [0, \frac{1}{2}]$, $I(G; E)$ is the mutual information between graph instance input and graph representation on instance level, $I(E; \Gamma)$ is the mutual information between instance representation and hierarchical graph representation on hierarchical level.

To prove Theorem 2, we first introduce two lemmas.

Lemma 1. For any random variable Z_1, Z_2, Z_3, we have

$$I(Z_1; Z_2, Z_3) \geq \frac{1}{2}(I(Z_1; Z_2) + I(Z_1; Z_3)), \tag{7}$$

here $I(Z_1; Z_2, Z_3)$ is the mutual information between variable Z_1 and the joint distribution of Z_2 and Z_3 .

To prove Lemma 1, we make use of the chain rule for mutual information.

$$\begin{aligned} I(Z_1; Z_2, Z_3) &= I(Z_1; Z_3) + I(Z_1; Z_2|Z_3) \\ &\geq I(Z_1; Z_3). \end{aligned} \tag{8}$$

The last inequality holds as mutual information is non-negative. Accordingly, we have

$$I(Z_1; Z_2, Z_3) \geq I(Z_1; Z_2). \tag{9}$$

Based on Eq. (8) and Eq. (9), we complete the proof of Lemma 1.

Lemma 2. For a Markov chain $G \to E \to \Gamma$, we have

$$I(E; G, \Gamma) \leq I(E; G) + I(E; \Gamma). \tag{10}$$

Proof. According to [40], if the conditional distribution $P(Z_3|Z_1, Z_2)$ is multiplicative, i.e., $\exists$ two functions s_1 and s_2, s.t., $P(Z_3|Z_1, Z_2) = s_1(Z_3, Z_1)s_2(Z_3, Z_2)$, then $I(Z_1; Z_2) \geq I(Z_1; Z_2|Z_3)$. Since $G \to E \to \Gamma$ is a Markov chain, we have $P(\Gamma|G, E) = P(\Gamma|E)$, which means $P(\Gamma|G, E)$ is multiplicative. Thus, $I(G; E) \geq I(G; E|\Gamma)$ holds in our case.

$$\begin{aligned} I(E; G, \Gamma) &= I(E; \Gamma) + I(E; G|\Gamma) \\ &\leq I(E; G) + I(E; \Gamma). \end{aligned} \tag{11}$$

Thus, we complete the proof of Lemma 2. We then prove Theorem 2:

$$\begin{aligned} I(G; E; \Gamma) &= I(G; E) - I(G; E|\Gamma) \\ &= I(G; E) - (I(E; G, \Gamma) - I(E; \Gamma)) \\ &= I(G; E) + I(E; \Gamma) - I(E; G, \Gamma) \\ &= \alpha(I(G; E) + I(E; \Gamma)). \end{aligned} \tag{12}$$

The last equality holds as we have $\frac{1}{2}(I(G; E) + I(E; \Gamma)) \leq I(E; G, \Gamma) \leq I(G; E) + I(E; \Gamma)$, based on Lemma 1 and Lemma 2. Thus, Theorem 2 is proved.

We use Figure 2 to illustrate the idea to compute HGMI. One naive method to compute HGMI is according to the definition, which inevitably involves with the computation of joint distribution $P(G, E, \Gamma)$ and is not tractable in general [41]. With Theorem 2, we decouple the computation of HGMI into a sum of the computation of graph mutual information without hierarchies, which is tractable as used in many previous works [30][14]. Next we present the design of IC and HC classifier, and then give the detailed way to maximize HGMI.

4.3 Design of Classifiers

Classifier IC takes a graph instance as input. As different graph instances have different numbers of nodes, IC is expected to handle graph instances of arbitrary size. Classifier HC takes the hierarchical graph as input, in which individual graph instances are the "nodes". We propose to embed a graph instance $g_i \in G$ into a fixed-length vector e_i via IC first. Then HC can take as input the embedding vectors of graph instances and the adjacency matrix $\mathcal{A}$. In particular, IC takes as input the adjacency matrix A_i and attribute matrix X_i of an arbitrary-sized graph instance g_i, and outputs an instance representation e_i, i.e., $e_i = \text{IC}(A_i, X_i)$. HC takes the instance representations $E = \{e_i\}_{i=1}^{L+U}$ and $\mathcal{A}$, and

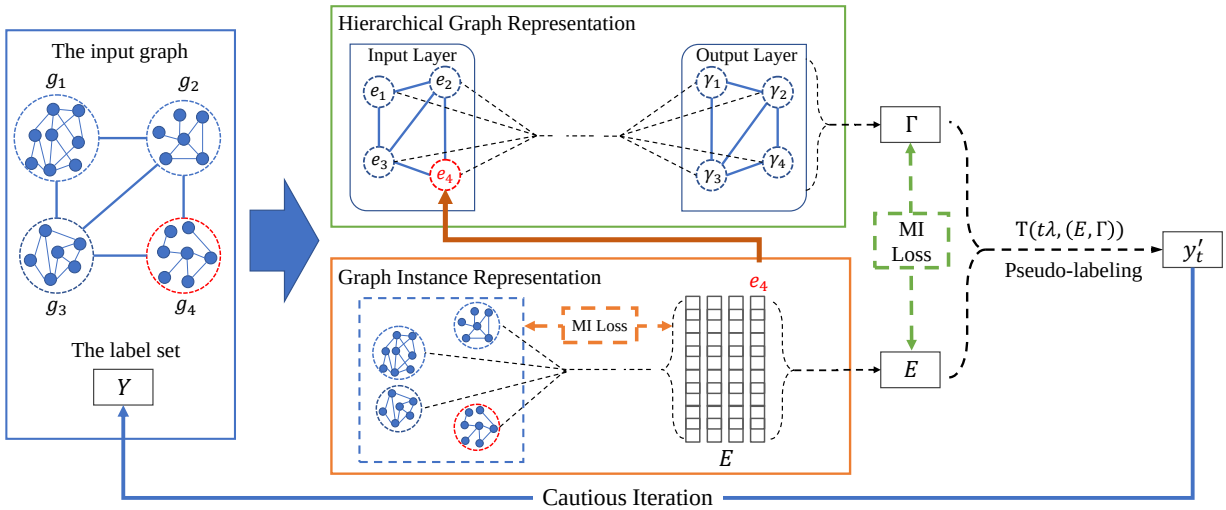

Fig. 3: Schematic diagram of the learning framework SEAL-CI. There are two subroutines: graph instance representation (in the orange box) and hierarchical graph representation (in the green box).

outputs the predicted hierarchical graph representations $\Gamma = \{\gamma_i\}_{i=1}^{L+U}$, i.e., $\Gamma = \text{HC}(E, \mathcal{A})$. In the following, we illustrate the design of IC which performs instance graph representation, and then the design of HC which performs hierarchical graph representation.

4.3.1 Graph instance representation

The task of graph instance representation is to produce a fixed-length embedding vector of a graph instance, for which, however, we identify two challenges:

- *Size invariance*: How to design the network structure to flexibly take an arbitrary-sized graph instance and produce a fixed-length embedding vector?
- *Permutation invariance*: How to derive the representation regardless of the permutation of nodes?

While the concept of size invariance is straightforward, we specifically provide the definition of permutation invariance on graphs here:

Definition 1 (Permutation Invariance on graphs). *Let $P \in \{0,1\}^{n\times n}$ be any permutation matrix of order n, then the permutation to a graph instance g is defined as a mapping of the node indices, i.e.:*

$$Pg = (PAP^\top, PX).$$

Then a function f is invariant to permutation on g if

$$f(A, X) = f(PAP^\top, PX).$$

Recently *graph pooling* has emerged as a research topic that can be used to tackle the above challenges. Some common practices include DIFFPOOL [23], Attention-based Pool [25][24][42], MinCut-Pool [26]. In particular, [42] shows Attention-based Pool is permutation-invariant by connecting with Weisfeiler-Lehman test. To this end, in IC, we first utilize a multi-layer Graph Neural Network (GNN) [18] to smooth each node's features over the graph's topology. Then we adopt a *graph pooling* method and transform a variable number of smoothed nodes into a fixed-length graph instance representation e.

Formally, we are given the adjacency matrix $A \in \mathbb{R}^{n\times n}$ and the attribute matrix $X \in \mathbb{R}^{n\times d}$ of a graph instance g as inputs. We apply a multi-layer GNN network:

$$H = \text{GNN}(A, X), \tag{13}$$

here we get a set of representation $H \in \mathbb{R}^{n\times v}$ for nodes in g. Note that the representation H is size variant, i.e., its size is still determined by the number of nodes n. So next we utilize the *graph pooling* mechanism:

$$e = \text{Pooling(H)}, \tag{14}$$

here consider Attention-based Pool is adopted, as the attention weight is used to multiply with H and flatten the representation, e is size invariant and does not depend on the number of nodes n any more, which solves the first challenge.

While for the second challenge of the node permutations, the following positive Proposition shows that our constructed instance-level classifier IC is permutation invariant as long as the selected component graph pooling method and GNN are permutation invariant:

Proposition 1. *Let $P \in \{0,1\}^{n\times n}$ be any permutation matrix, then $IC(A, X) = IC(PAP^\top, PX)$ as long as the chosen graph pooling method and GNN are permutation invariant.*

Proof. Consider a permutation invariant GNN, e.g., GIN [21], we have:

$$H = \text{GNN}(A, X) = \text{GNN}(PAP^\top, PX). \tag{15}$$

For pooling operation, consider the permutation invariant Attention-based Pooling function used in [25]:

$$S = \text{softmax}(W_{s2}\text{tanh}(W_{s1}H^T)), \quad e = SH, \tag{16}$$

where W_{s1} and W_{s2} are two weight matrices. H is the same after nodes permutation from Eq. (15) and our example pooling function Eq. (16) is only relevant to H, which reveals the permutation invariance of IC. Thus we finish the proof. □

To summarize, we use GNN and *graph pooling* that are permutation invariant to construct the instance-level classifier IC. It produces graph instance representations $E = \{e_i\}_{i=1}^{L+U}$ while successfully dealing with the aforementioned two challenges, which is the input for classifier HC in the next.

4.3.2 Hierarchical graph representation

Given E and the adjacency matrix $\mathcal{A} \in \mathbb{R}^{(L+U)\times(L+U)}$, our next task is to infer the parameters of classifier HC and derive the predicted probabilities $\Gamma = \{\gamma_i\}_{i=1}^{L+U}$. This problem falls into the setting of non-hierarchical graph setting where E can be treated as the set of node features. In this context, we consider again a multi-layer GNN. Thus the model becomes:

$$\Gamma = HC(E, \mathcal{A}) = \text{softmax}(\text{GNN}(E, \mathcal{A})), \tag{17}$$

where $\Gamma \in \mathbb{R}^{(L+U)\times c}$ is the derived hierarchical graph representation. With Γ and E, next we introduce the way to maximize HGMI.

4.4 Maximization of HGMI

With Theorem 2, the computation of HGMI is reduced into MI between graph input and graph-level representation ($I(G;E)$) and MI between graph input and node-level representation ($I(E;\Gamma)$), in which we have many choices such as INFOGRAPH [15] and GMI [14].

To compute $I(G;E)$, we use the method of INFOGRAPH [15], where MI is computed between the graph input G and graph-level representation E. More specifically, it is shown in [15] that maximizing the global $I(G;E)$ can be estimated by MI between node representations and graph-level representations,

$$I(G;E) = \sum_{i}^{L+U} \frac{1}{L+U} \sum_{j}^{n} \frac{1}{n} I(h_{ij}; e_i), \tag{18}$$

where h_{ij} is a node representation sampled from H_i. The core thus becomes how we compute $I(h_{ij}; e_i)$. In this work, we resort to Jensen-Shannnon MI estimator (JSD),

$$I(h_{ij}; e_i) = -sp(-\mathbb{D}_{\mathbb{I}}(h_{ij}, e_i) - \mathbb{E}_{\hat{h}_{ij}} sp(\mathbb{D}_{\mathbb{I}}(\hat{h}_{ij}, e_i)), \tag{19}$$

where $\mathbb{D}_{\mathbb{I}} : \mathbb{R}^v \times \mathbb{R}^m \to \mathbb{R}$ is a discriminator constructed by a neural network with parameters D_I, i.e., $\mathbb{D}_{\mathbb{I}}(h_{ij}, e_i) = \sigma(h_{ij}^\top W_{D_I} e_i)$ and W_{D_I} is a learnable scoring matrix. $\hat{h}_{ij}$ is a negative example. $sp(x) = \log(1 + \exp(x))$ is the softplus function. For a detailed graph-level negative sampling process, please see INFOGRAPH [15].

To compute $I(E;\Gamma)$, we adopt the method of GMI [14], where MI is computed between the graph feature input and node-level representation. More specifically, it is shown in [14] that, for a node-level representation γ_i, maximizing the global $I(E;\gamma_i)$ can be decomposed as a weighted sum of local MIs,

$$I(E;\gamma_i) = \sum_{j}^{L+U} w_{ij} I(e_j; \gamma_i), \tag{20}$$

here we adopt the mean version of GMI, meaning $w_{ij} = \frac{1}{L+U}$. To compute $I(e_j;\gamma_i)$, we use JSD again,

$$I(e_j;\gamma_i) = -sp(-\mathbb{D}_{\mathbb{H}}(e_j, \gamma_i) - \mathbb{E}_{\hat{e}_j} sp(\mathbb{D}_{\mathbb{H}}(\hat{e}_j, \gamma_i)), \tag{21}$$

Algorithm 1: SEAL-CI

Input: $\mathcal{G} = \{G, \mathcal{A}\}$.
Output: Γ^{t+1},E^{t+1}.

Initial: $G_{tmp} = \emptyset$, $G_l^0 = G_l$, $t = 0$;
while $t\lambda \leq U$ **do**
  $\mathcal{W}^{t+1} \leftarrow \arg\min \zeta(G_l^t|\mathcal{W}^t) - \xi(\mathcal{G}^t|\mathcal{W}^t)$;
  $E^{t+1} \leftarrow \mathsf{IC}(A, X|\mathcal{W}^{t+1})$;
  $\Gamma^{t+1} \leftarrow \mathsf{HC}(E^{t+1}, \mathcal{A}|\mathcal{W}^{t+1})$;
  $G_{tmp} \leftarrow T(t\lambda, (E_{G_u}^{t+1}, \Gamma_{G_u}^{t+1}))$;
  $G_l^{t+1} \leftarrow G_l \cup G_{tmp}$;
  $G_{tmp} = \emptyset$;
Return Γ^{t+1},E^{t+1};

where $\mathbb{D}_{\mathbb{H}} : \mathbb{R}^m \times \mathbb{R}^c \to \mathbb{R}$ is a discriminator constructed by a neural network with parameters D_H. $\hat{e}_j$ is a negative example. For a detailed node-level negative sampling process, please see GMI [14].

4.5 The Proposed SEAL-CI Model

In this subsection, we present our method to minimize the objective function Eq. (1). A naive way would be directly minimizing Eq. (1) in an end-to-end fashion with the limited labels. This algorithm is called SEmi-supervised grAph cLassification (SEAL). However, in real-world scenarios, the number of labeled graph instances L can be quite small compared to the number of unlabeled instances U. In this context, neural network based classifiers may suffer from the problem of overfitting.

Following previous works [37][5], we use the idea of iterative algorithm to alternate optimizing the two modules of IC and HC by trusting a subset of predictions. To be more specific, we combine the instance representation in Section 4.3.1 and hierarchical representation in Section 4.3.2 into one iterative algorithm. We build IC to produce instance representation E^t for all graph instances in iteration t, and then feed E^t into HC to get the predicted probabilities Γ^t. We then make use of pseudo-labeling strategy to update the parameters of IC and generate E^{t+1}, which is then used as the input of HC in iteration $t+1$. Figure 3 depicts the overall framework of this iterative process.

4.5.1 Pseudo-labeling

To update the instance representations, a naive approach is feeding the whole set of (E^t,Γ^t) as pseudo labels for the parameter update in IC, which is the idea of the original ICA [5]. However, not all (E^t,Γ^t) are correct in their predictions. The false predictions may lead the update of embedding neural network to the wrong direction. Within the set of unlabeled graphs, different (E^t,Γ^t) could contribute differently to the update of embedding neural network. To this end, we make use of the idea of *cautious iteration* [37] and *curriculum learning* [43], and cautiously exploit a subset of (E^t,Γ^t) to update the parameters of IC in each iteration. Specifically, in iteration t, we choose the $t\lambda$ most confident predicted labels in both IC and HC while ignoring the less confident predicted labels. This operation continues until all the unlabeled samples have been utilized or until a predefined iteration number. This algorithm is

called SEmi-supervised grAph cLassification via Cautious Iteration (SEAL-CI) and is presented in Algorithm 1. Note here $\mathcal{W}$ is the set of all the parameters of IC and HC. In line 3, SEAL is well trained to provide a foundation for SEAL-CI to find informative and accurate pseudo labels thereafter. In line 6, the training set for IC has been enlarged by $t\lambda$ instances and it is done by "committing" these instances' labels from their maximum probability. In other words, the newly enrolled training instances are found by:

$$T(t\lambda,(E,\Gamma)) = \mathsf{top}(\max((E,\Gamma)), t\lambda), \tag{22}$$

here function $\mathsf{top}(\cdot,\lambda)$ is used to select the top $t\lambda$ instances and function $\max(\cdot)$ is used to select the maximum value in a set.

4.5.2 Analysis of SEAL-CI

In this part, we discuss about the convergence properties of proposed SEAL-CI algorithm. We first show the empirical risk is a monotonically decreasing function with respect to the iteration step.

Theorem 3. Let $\zeta(\mathcal{W})$ be the empirical risk, $\mathcal{W}^t(t = 1, 2, \cdots)$ be the parameter set of algorithm 1 at iteration step t, $\zeta(\mathcal{W}^t)$ be the empirical risk at the corresponding iteration step. Then $\zeta(\mathcal{W}^t)$ is a monotonically decreasing function, namely,

$$\zeta(\mathcal{W}^{t+1}) \leq \zeta(\mathcal{W}^t), \tag{23}$$

here $\zeta(\mathcal{W}^t)$ is short for $\zeta(G_l|\mathcal{W}^t)$ and the size of G_l will be dynamically increased as we add pseudo-labeled instances.

The *cautious selection function* T in the pseudo labeling process provides a Bayesian prior for incorporating unlabeled instances, which can be formalized as follows:

$$\begin{aligned}\zeta(\mathcal{W}^{t+1}) = \frac{1}{L+1}(&\mathrm{CE}(Y,E_l) + \mathrm{CE}(Y,\Gamma_l) + \mathrm{CE}(y'_t,e_t)\\ &+ \mathrm{CE}(y'_t,\gamma_t)),\end{aligned} \tag{24}$$

here $E_l = \{e_i\}_{i=1}^L$ and $\Gamma_l = \{\gamma_i\}_{i=1}^L$, $\mathrm{CE}(Y,E_l) = \sum_{g_i\in G_l}\mathrm{CE}(y_i,e_i)$, $\mathrm{CE}(\cdot,\cdot)$ is the cross-entropy function. y'_t denotes the pseudo label that we incorporate with $T(e_i,\gamma_i)$ and $\lambda = 1$, i.e., we consider increasing the size of training set by one at each iteration. Let T be short for $T(e_i,\gamma_i)$, we have

$$\mathrm{CE}(y_t,\gamma_t) = (L+U)\mathrm{Cov}[\mathrm{CE}(y'_t,\gamma_t),T] + \mathbb{E}[\mathrm{CE}(Y,\Gamma_l)], \tag{25}$$

$$\mathrm{CE}(y_t,e_t) = (L+U)\mathrm{Cov}[\mathrm{CE}(y'_t,e_t),T] + \mathbb{E}[\mathrm{CE}(Y,E_l)]. \tag{26}$$

Based on Eq. (25) and Eq. (26), we can rewrite $\zeta(\mathcal{W}^{t+1})$:

$$\begin{aligned}\zeta(\mathcal{W}^{t+1}) &= \beta\{\mathrm{Cov}[\mathrm{CE}(y'_t,e_t),T] + \mathrm{Cov}[\mathrm{CE}(y'_t,\gamma_t),T]\}\\ &\quad + \frac{1}{L}(\mathrm{CE}(Y,E_l) + \mathrm{CE}(Y,\Gamma_l))\\ &= \beta\{\mathrm{Cov}[\mathrm{CE}(y'_t,e_t),T] + \mathrm{Cov}[\mathrm{CE}(y'_t,\gamma_t),T]\}\\ &\quad + \zeta(\mathcal{W}^t) \leq \zeta(\mathcal{W}^t),\end{aligned} \tag{27}$$

here $\beta = \frac{L+U}{L+1}$. The last inequality holds as the labeling strategy T is similar to a discrete sign function. It is easy to see the positive correlation between γ_t and T. Besides, γ_t and the pseudo labels y'_t are positively correlated. As cross entropy term $\mathrm{CE}(y'_t,\gamma_t)$ is negatively correlated to γ_t, the correlation between T and $CE(y'_t,\gamma_t)$ is negative. Similarly, the correlation between T and $CE(y'_t,e_t)$ is negative. We thus finish the proof of Theorem 3.

As the empirical risk is a monotonically decreasing function and bounded (i.e., non-negative), we conclude that the proposed SEAL-CI algorithm has a convergence property.

TABLE 1: Statistics of generated graph instances

Type	Number	Nodes	Edges	Density
Watts-Strogatz	351	173	347	2.3%
Tree	217	127	120	1.5%
Erdős-Rényi	418	174	3045	20%
Barbell	818	169	2379	16.3%
Bipartite	426	144	1102	10.6%
Barabási-Albert	298	173	509	3.4%
Path	180	175	170	1.1%

5 EXPERIMENTS

We evaluate SEAL-CI method on synthetic, text and social network data sets.

5.1 Synthetic Data

We evaluate the performance of SEAL-CI on synthetic data. We first give a description of the synthetic generator, then visualize the learned embeddings. Finally we compare our methods with baselines in terms of classification accuracy.

5.1.1 Synthetic Data Generation

The benchmark data set Cora [44] contains 2708 papers which are connected by the citation relationship. We borrow the topological structure of Cora to provide the skeleton (i.e., edges) of our synthetic hierarchical graph. Then we generate a set of graph instances, which serve as the nodes of this hierarchical graph. Since there are 7 classes in Cora, we adopt 7 different graph generation algorithms, that is, Watts-Strogatz [45], Tree graph, Erdős-Rényi [46], Barbell [47], Bipartite graph, Barabási-Albert graph [48] and Path graph, to generate 7 different types of graph instances, and connect them in the hierarchical graph.

Specifically, to generate a graph instance g, we randomly sample a number from $[100, 200]$ as its size n. Then we generate its structure and assign the class label according to the graph generation algorithm. In this step, the parameter p in Watts-Strogatz, Erdős-Rényi, Bipartite graph and Barabási-Albert graph is randomly sampled from $[0.1, 0.5]$, the branching factor for Tree graph is randomly sampled from $[1, 3]$. At last, to make this problem more challenging, we randomly remove 1% to 20% edges in the generated graph g. The statistics of the generated graph instances are listed in Table 1.

5.1.2 Baselines and Metrics

We use the following approaches as our baselines:

- GK-SVM [8], which calculates the graphlet count kernel (GK) matrix, then GK-SVM feeds the kernel matrix into SVM [49].
- WL-SVM [9], which is similar as above but using the Weisfeiler-Lehman subtree kernel (WL).

- graph2vec-GCN [11], which embeds the graph instances by graph2vec and then feeds the embeddings to GCN.
- SAGE [25], which ignores the connections between graph instances and treats them independently.
- GAT [22], which uses attention mechanisms in node-level neighborhood aggregation.
- GIN [21], which is the GNN model with state-of-the art performance in graph classification.
- MIRACLE [29], which uses the multi-view contrast learning method to exploit relation between the structure of graph instance level and the one at the hierarchical graph level.
- SEAL is the base of SEAL-CI, which differs with SEAL-CI in two ways: 1) it does not consider label enlargement, and 2) it is trained in an end-to-end fashion with the given class labels.

We use 300 graph instances as the training set for all methods. We use 1000 graph instances as the testing set. We run each method 5 times and report its average accuracy. The number of epochs for graph2vec is 1000 and the learning rate is 0.3. For SEAL-CI, we use SAGE [25] as the graph pooling method. We use a two-layer GCN in IC, in which the first GCN layer has 32 output channels and the second GCN layer has 4 output channels. The dense layer has 48 units with a dropout rate of 0.3.

5.1.3 Results

Table 2 shows the experimental results for semi-supervised graph classification. Among all approaches, SEAL-CI achieves the best performance. In the following, we analyze the performance of all methods categorized into 3 groups.

Group *1: All the embedding-based methods perform better than these two kernel methods, which proves that embedding vectors are effective representations for graph instances and are suitable input for graph neural networks.

Group *2: graph2vec-GCN achieves 85.2% accuracy, which is comparable to that of SAGE, but lower than that of SEAL-CI. One possible explanation is that graph2vec is an unsupervised embedding method, which fails to generate discriminative embeddings for classification. Another possibility is that the 300 training instances do not include very informative ones. These limitations of graph2vec motivate us to use supervised graph representation modules such as SAGE and the label enlargement framework in SEAL-CI. MIRACLE and SEAL generate the graph instance representations in a supervised way, and they outperform graph2vec-GCN by more than 1.5%.

Group *3: SEAL-CI outperforms MIRACLE and SEAL, which proves the effectiveness of our hierarchical graph perspective and the label enlargement algorithm for semi-supervised graph classification.

5.2 Text Data

We evaluate SEAL-CI on the arXiv paper dataset. Firstly, We introduce the compositions of arXiv paper dataset and present how to construct hierarchical graph from text data. Then, we show the evaluation results and give some insights on how to construct a hierarchical graph for text data.

TABLE 2: Comparison of different methods on the synthetic data set for semi-supervised graph classification

	Algorithm	Accuracy
*1	**GK-SVM**	77.8%
	WL-SVM	83.4%
	SAGE	85.7%
	GAT	85.0%
	GIN	86.7%
*2	**graph2vec-GCN**	85.2%
	MIRACLE	86.7%
	SEAL	87.8%
*3	**SEAL-CI**	91.2%

TABLE 3: Statistics of arXiv paper data

Class	Number	Length of title	Length of abstract
AI	232	8.57	158.04
CL	648	9.25	140.36
IT	909	10.12	170.29
LG	1157	8.69	160.63
CV	1720	9.41	170.54

5.2.1 Data Description

arXiv is an open-access repository of electronic preprints. We collect 4666 Computer Science (CS) arXiv papers indexed by Microsoft Academic Graph (MAG) [50]. These papers belong to five subject areas including AI, CL, IT, LG and CV. The arXiv papers data forms a citation network which indicates citation relationships. Each paper consists of two parts: title and abstract. The statistics of the arXiv paper dataset is listed in Table 3.

The connections among graph instances (i.e., papers) are provided by citation relations. Each paper is a semantic graph instance constructed from its title and abstract. There are three type of nodes in a semantic graph instance: abstract node, title node and argument nodes. Specifically, for each sentence in abstract, we parse it into a tuple with Semantic Role Labeling (SRL) toolkit developed by AllenNLP [2]. For each tuple, we regard its elements as the argument nodes of the semantic graph. The node features of the semantic graph are extracted by pretrained Bert model [51]. The connections within the semantic graph are created as the following: 1) same tuple, if two nodes are from the same tuple, we connect them, and 2) vocabulary overlapping, we connect the node pair if the size of overlapped words is larger than half of the minimum size of any text node. In the Figure 4 we show an example of hierarchical graph of arXiv papers data.

5.2.2 Baselines and Metrics

In addition to the set of baselines in Section 5.1.2, we compare with the following Bert-feature baselines:

- Bert-MLP, which classifies arXiv papers by feeding average features of title and abstract into a multi-layer perceptron (MLP).
- Bert-IC, which feeds semantic graph instances into IC and classifies arXiv papers in an independent graph classification fashion.

2. https://demo.allennlp.org/semantic-role-labeling

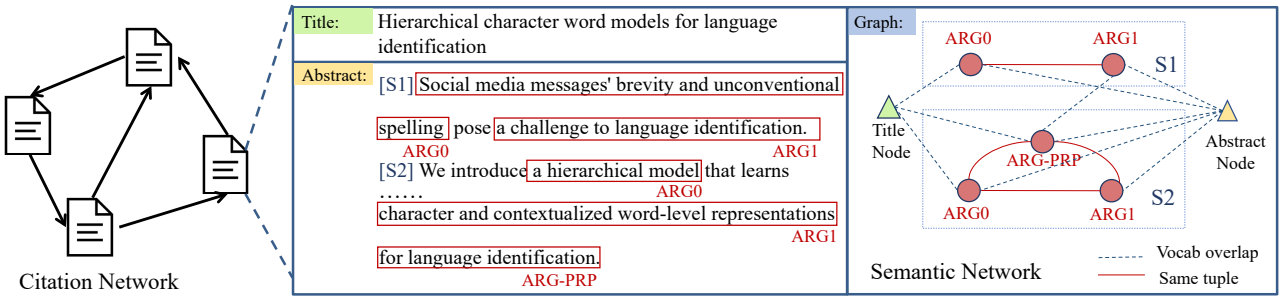

Fig. 4: An illustration of the hierarchical graph of arXiv papers data

TABLE 4: Comparison of different methods on arXiv Paper data for semi-supervised graph classification

	Algorithm	Accuracy
*1	**GK-SVM**	38.7%
	WL-SVM	35.1%
	graph2vec-GCN	42.7%
	GAT	69.3%
	GIN	69.1%
*2	**Bert-MLP**	63.7%
	Bert-IC	70.1%
	Bert-HC	75.2%
*3	**MIRACLE**	78.4%
	SEAL	78.8%
*4	**SEAL-CI**	79.4%

- Bert-HC, which feeds average features of title and abstract into HC and classifies arXiv papers in a transductive node classification fashion.

400 graph instances are used as labeled training instances for all methods. We use 2000 instances for testing for all methods. We set the dimension of Bert node feature to 768. We use average pool in IC. We set $\lambda = 1$ and updated times $t = 100$ for SEAL-CI. We run each method 3 times and report its average accuracy.

5.2.3 Results

Table 4 shows the experimental results for semi-supervised graph classification on arXiv paper data. SEAL-CI obtains the state-of-the-art performances among all baselines. To analyze the results of these baselines, all methods are divided into four group as the following.

Group *1: Both GK-SVM and WL-SVM only utilize the structural information of semantic graph instances. And their classification accuracies are lower than 40%, which indicates that structural information of semantic graph instances is not enough to obtain satisfied performance on text data. Graph2vec-GCN tries to exploit both structure of semantic graph instances and the hierarchical graph but still has a weak performance with an accuracy of 42.7%.

Group *2: In this group, we show the performance of several baselines based on Bert. Bert-MLP only uses Bert features extracted from title and abstract, and obtains an accuracy of 63.7%. Bert-IC outperforms Bert-MLP by about 6.4%, by combing both structural and Bert features of text data. Bert-HC, by utilizing the connections between Bert features of title and abstract, achieves the best performance among baselines within Group *2.

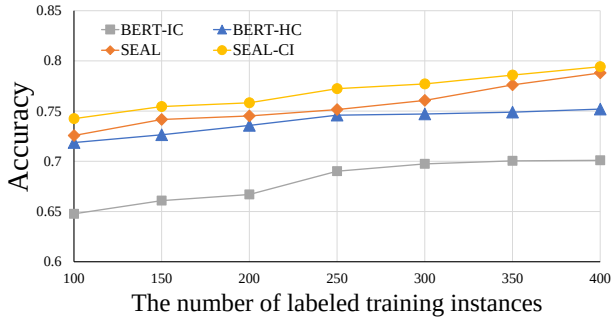

Fig. 5: Accuracy with different number of labeled training instances on text data for semi-supervised graph classification.

Group *3: Both SEAL and MIRACLE aim to minimize mutual information between graph instance representations and hierarchical representations and differ in the measurement of consistency. SEAL and MIRACLE outperform baselines in Group *1 with a margin of 30% and Group *2 with a margin of more than 3%.

Group *4: SEAL-CI outperforms MIRACLE and SEAL, which demonstrates the effectiveness of the cautious iteration algorithm 1.

5.2.4 Influence of the number of labeled training instances

We examine how the number of labeled training instances affects the performance of our methods. We train Bert-IC, Bert-HC, SEAL and SEAL-CI with a label size of $\{100, 150, 200, 250, 300, 350, 400\}$. We set $\lambda = 1$ in SEAL-CI. We run all methods 5 times, and plot their average accuracy in Figure 5. As we can see from Figure 5, SEAL-CI performs the best since it can utilize more training samples. SEAL outperform Bert-HC and Bert-IC in all cases, which makes sense because SEAL makes good use of the hierarchical graph setting and consider the connections between the graph instances for classification. Meanwhile, as SEAL-CI is based on SEAL, the good performance of SEAL provides a foundation for SEAL-CI to find informative and accurate pseudo labels, which could further boost the performance.

TABLE 5: Statistics of collected Tencent QQ groups

Class label	Number	Nodes	Edges	Density
game	1,773	147	395	5.48%
non-game	36,063	365	1586	3.28%

5.2.5 Discussion

How to construct hierarchical graphs from text data is an open question. In the above experiment, we use semantic parsing techniques to construct graph instances for papers, and citation relationships to connect these papers. We think hierarchical graph structures could be a good paradigm to boost the performance of Natural Language Processing classification tasks. Moreover, it is worth mentioning that there are several other ways to construct semantic graphs such as Name Entity Recognition (NER).

5.3 Social Network Data

In this section, we evaluate SEAL-CI on Tencent QQ group data. We describe the characteristics of this data set and then present the experimental results. Finally, we have some open discussions on how to construct a hierarchical graph for social network data.

5.3.1 Data Description

Tencent QQ is a social networking platform in China with nearly 800 million monthly active users. There are around 100 million active online QQ groups. In this experiment, we select 37,836 QQ groups with 18,422,331 unique anonymized users. For each user, we extract seven personal features:

- number of days ever since the registration day;
- most frequently active area code in the past 90 days;
- number of friends;
- number of active days in the past 30 days;
- number of logging in the past 30 days;
- number of messages sent in the past 30 days;
- number of messages sent within QQ groups in the past 30 days.

We have 298,837,578 friend relationships among these users. 1,773 groups are labeled as "game" and the remaining groups are labeled as "non-game".

We construct the hierarchical graph from this Tencent QQ group data as follows. A user is treated as an object, and a QQ group as a graph instance. The users in one group are connected by their friendship. The attribute matrix X is filled with the attribute values of the users. The statistics of the graph instances are listed in Table 5. We build the hierarchical graph from the graph instances via common members across groups, that is, if two groups have more than one common member, we connect them.

5.3.2 Baselines and Metrics

We use the same set of baselines as in Section 5.1.2. 1000 graph instances are used as labeled training instances for all methods. We use 10,000 instances for testing for all methods. We run each method 3 times and report its average performance. For IC, we use SAGE and set the parameters as the same in Section 5.1. Since the class distribution is extremely imbalanced in this data set, we report the Macro-F1 instead of accuracy.

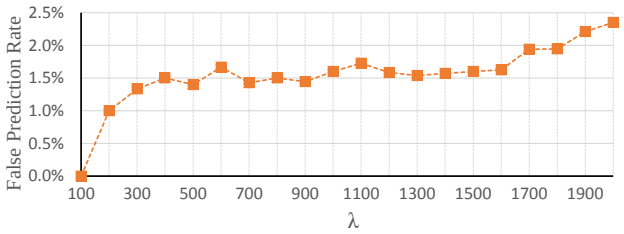

Fig. 6: The false prediction rate of IC with λ in SEAL-CI.

TABLE 6: Comparison of different methods on Tencent QQ group data for semi-supervised graph classification

	Algorithm	Macro-F1
*1	**GK-SVM**	48.8%
	WL-SVM	47.8%
	SAGE	75.0%
	GAT	79.9%
	GIN	79.5%
*2	**graph2vec-GCN**	48.1%
	MIRACLE	80.3%
	SEAL	84.3%
*3	**SEAL-CI**	85.5%

5.3.3 Results

Table 6 shows the experimental results. SEAL-C/AI outperform GK, WL and graph2vec by at least 36% in Macro-F1. SEAL beats MIRACLE by about 4%. SEAL-CI outperforms SEAL and MIRACLE with a margin of more than 1.2%, which demonstrates the effectiveness of the cautious iteration algorithm 1 on social network data. Next we consider the relation between λ and the false prediction rate (i.e., the percentage of misclassified instances) in the pseudo-labeling. Figure 6 shows the false prediction rate within the λ most confident predictions of IC. As we can see, the false prediction rate increases as λ increases and it reaches 2.4% when $\lambda = 2000$. In the framework of SEAL-CI, as the iteration goes on, we shall bring in more noise to the parameter update, while all the training samples in SEAL are informative and correct. This observation tells us when using SEAL-CI, we need to be "cautious" enough, i.e., λ should be small enough, to ensure the false prediction rate is small.

5.3.4 Visualization

We provide visualization of a "game" group and its neighborhood in Figure 7. The left part is the ego network of the center "game" group. In the one-hop neighborhood of this "game" group, there are 10 "game" groups and 19 "non-game" groups. "Game" groups are densely interconnected with a density of 34.5%, whereas "non-game" groups are sparsely connected with a density of 8.8%. The much higher density among "game" groups validates that common membership is an effective way to relate them in a hierarchical graph for classification. The right part depicts the internal structure of the ego "game" group with 22 users. A bigger node indicates a larger importance obtained by SAGE, and a darker green color implies a larger node degree. These 22 members are loosely connected and there are no triangles. This makes sense because in reality online "game" groups

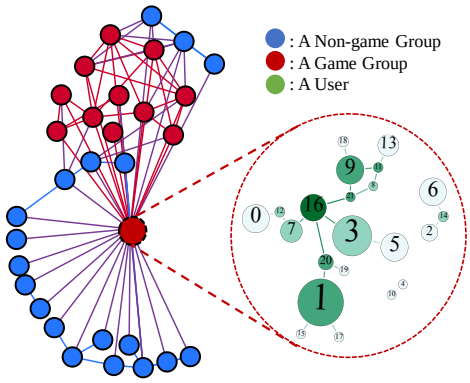

Fig. 7: The ego network of a "game" group. The left side is the ego network, in which "game" groups are in red and "non-game" groups are in blue. The right side is the internal structure of the ego "game" group, in which a bigger node indicates a larger importance, and a darker color implies a larger node degree.

are not acquaintance networks. Regarding the learned node importance, node 1 has the highest importance as it is the second active member and has a large degree in this group. Node 16 is also important since it has the highest degree in this group. The "border" member 5 has a big attention weight since it has the largest number of days ever since the registration day and is quite active in this group.

5.3.5 Discussion

How to construct a hierarchical graph from social network is an open question. In the above experiment, we connect two QQ groups if they have more than one common member (i.e., > 1). When we change the threshold, it directly affects the edge density in the hierarchical graph, and may influence the classification performance. For example, if we connect two QQ groups when they have one common member or more (i.e., ≥ 1), the edge density is 2.8% compared with 0.27% in the first setting. A proper setting of this threshold is data dependent, and can be determined through a validation set.

6 Conclusion

In this paper, we study the problem of semi-supervised hierarchical graph classification. To enforce a consistency among different levels of the hierarchical graph, we propose Hierarchical Graph Mutual Information (HGMI) and show HGMI can be computed via non-hierarchical graph mutual information computation methods. We build two classifiers IC and HC at the graph instance level and the hierarchical graph level respectively to fully exploit the available information. Our semi-supervised solution SEAL-CI adopts an iterative framework to update IC and HC alternately with an enlarged training set. We present two hierarchical graph benchmarks and demonstrate that SEAL-CI outperforms other competitors by a significant margin.

7 Acknowledgements

The work described in this paper was supported by grants from HKUST(GZ) under a Startup Grant, HKUST-GZU Joint Research Collaboration Fund (Project No.: GZU22EG05) and Tencent AI Lab Rhino-Bird Focused Research Program RBFR2022008.

References

[1] H. Chang, Y. Rong, T. Xu, W. Huang, S. Sojoudi, J. Huang, and W. Zhu, "Spectral graph attention network with fast eigen-approximation," in *CIKM*, 2021, pp. 2905–2909.
[2] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," *ICLR*, 2020.
[3] J. Tang, J. Li, Z. Gao, and J. Li, "Rethinking graph neural networks for anomaly detection," in *International Conference on Machine Learning*, 2022.
[4] R. Ramanath, H. Inan, G. Polatkan, B. Hu, Q. Guo, C. Ozcaglar, X. Wu, K. Kenthapadi, and S. C. Geyik, "Towards deep and representation learning for talent search at linkedin," in *CIKM*, 2018, pp. 2253–2261.
[5] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–106, 2008.
[6] K. M. Borgwardt and H.-P. Kriegel, "Shortest-path kernels on graphs," in *ICDM*, 2005, pp. 74–81.
[7] T. Gärtner, P. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," in *Learning theory and kernel machines*, Springer, 2003, pp. 129–143.
[8] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. M. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *AISTATS*, 2009, pp. 488–495.
[9] N. Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *Journal of Machine Learning Research*, vol. 12, no. Sep, pp. 2539–2561, 2011.
[10] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *ICML*, 2016, pp. 2014–2023.
[11] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "Graph2vec: Learning distributed representations of graphs," *arXiv preprint arXiv:1707.05005*, 2017.
[12] J. Li, T. Yu, D.-C. Juan, A. Gopalan, H. Cheng, and A. Tomkins, "Graph autoencoders with deconvolutional networks," *arXiv preprint arXiv:2012.11898*, 2020.
[13] F. Rousseau, E. Kiagias, and M. Vazirgiannis, "Text categorization as a graph classification problem," in *ACL-IJCNLP*, 2015, pp. 1702–1712.
[14] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, and J. Huang, "Graph representation learning via graphical mutual information maximization," in *Proceedings of The Web Conference*, 2020, pp. 259–270.
[15] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," *ICLR*, 2020.
[16] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD*, 2014, pp. 701–710.
[17] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *KDD*, 2016, pp. 855–864.
[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[19] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NIPS*, 2016, pp. 3844–3852.
[20] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *KDD*, 2015, pp. 1365–1374.
[21] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *ICLR*, 2019.
[22] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *ICLR*, 2018.
[23] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *NeurIPS*, 2018, pp. 4800–4810.
[24] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," *ICML*, 2019.
[25] J. Li, Y. Rong, H. Cheng, H. Meng, W. Huang, and J. Huang, "Semi-supervised graph classification: A hierarchical graph perspective," in *The World Wide Web Conference*, 2019, 972–982.
[26] F. M. Bianchi, D. Grattarola, and C. Alippi, "Spectral clustering with graph neural networks for graph pooling," in *ICML*, 2020, pp. 2729–2738.

[27] H. Wang, D. Lian, Y. Zhang, L. Qin, and X. Lin, "Gognn: Graph of graphs neural network for predicting structured entity interactions," *IJCAI*, 2020.

[28] J. Xu, L. Chen, M. Lv, C. Zhan, S. Chen, and J. Chang, "Highair: A hierarchical graph neural network-based air quality forecasting method," *arXiv preprint arXiv:2101.04264*, 2021.

[29] Y. Wang, Y. Min, X. Chen, and J. Wu, "Multi-view graph contrastive representation learning for drug-drug interaction prediction," in *The World Wide Web Conference*, 2021.

[30] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," *ICLR*, 2019.

[31] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *ICML*, 2020, pp. 3451–3461.

[32] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML*, 2003, pp. 912–919.

[33] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *ICML*, 2016, pp. 40–48.

[34] S. Wan, S. Pan, J. Yang, and C. Gong, "Contrastive and generative graph convolutional networks for graph-based semi-supervised learning," in *AAAI*, 2021, pp. 10 049–10 057.

[35] K. Sun, Z. Lin, and Z. Zhu, "Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes," in *AAAI*, vol. 34, 2020, pp. 5892–5899.

[36] Y. Ouali, C. Hudelot, and M. Tami, "An overview of deep semi-supervised learning," *arXiv preprint arXiv:2006.05278*, 2020.

[37] L. K. McDowell, K. M. Gupta, and D. W. Aha, "Cautious inference in collective classification," in *AAAI*, 2007, pp. 596–601.

[38] P. Cascante-Bonilla, F. Tan, Y. Qi, and V. Ordonez, "Curriculum labeling: Revisiting pseudo-labeling for semi-supervised learning," 2021.

[39] V. Gligorijević, P. D. Renfrew, T. Kosciolek, J. K. Leman, D. Berenberg, T. Vatanen, C. Chandler, B. C. Taylor, I. M. Fisk, H. Vlamakis, *et al.*, "Structure-based protein function prediction using graph convolutional networks," *Nature communications*, vol. 12, no. 1, pp. 1–14, 2021.

[40] R. Renner and U. Maurer, "About the mutual (conditional) information."

[41] L. Paninski, "Estimation of entropy and mutual information," *Neural computation*, vol. 15, no. 6, pp. 1191–1253, 2003.

[42] J. Baek, M. Kang, and S. J. Hwang, "Accurate learning of graph representations with graph multiset pooling," in *ICLR*, 2021.

[43] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *ICML*, 2009, pp. 41–48.

[44] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.

[45] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, 1998.

[46] P. Erdős and A Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.

[47] M. Herbster and M. Pontil, "Prediction on a graph with a perceptron," in *NIPS*, 2007, pp. 577–584.

[48] B. Bollobás and O. M. Riordan, "Mathematical results on scale-free random graphs," *Handbook of graphs and networks: from the genome to the internet*, pp. 1–34, 2003.

[49] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[50] I. Beltagy, K. Lo, and A. Cohan, "Scibert: A pretrained language model for scientific text," *EMNLP*, 2019.

[51] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019.

Jia Li is an assistant professor at Hong Kong University of Science and Technology (Guangzhou) and an affiliated assistant professor at Hong Kong University of Science and Technology. He received the Ph.D. degree at Chinese University of Hong Kong in 2021. His research interests include machine learning, data mining and deep graph learning. He has published several papers as the leading author in top conferences such as KDD, WWW and NeurIPS.

Yongfeng Huang is a research assistant at The Hong Kong University of Science and Technology (Guangzhou). Previously, he received master degree in Tsinghua University. His research interests include natural language processing and deep graph learning.

Heng Chang is currently pursuing a Ph.D. Degree at Tsinghua University. He received his B.S. from the Department of Electronic Engineering, Tsinghua University in 2017. His research interests focus on graph representation learning and machine learning on graph data, especially in the development of efficient graph learning models, and the explanation and robustness of graph neural networks. He has published several papers in prestigious conferences including NeurIPS, AAAI, CIKM, etc.

Yu Rong is a senior researcher of Machine Learning Center in Tencent AI Lab. He received the Ph.D. degree from The Chinese University of Hong Kong in 2016. He joined Tencent AI Lab in June 2017. His main research interests include social network analysis, graph neural networks, and large-scale graph systems. In Tencent AI Lab, he is working on building the large-scale graph learning framework and applying the deep graph learning model to various applications, such as ADMET prediction and malicious detection. He has published several papers on data mining, machine learning top conferences, including KDD, WWW, NeurIPS, ICLR, CVPR, ICCV.