CGraphNet: Contrastive Graph Context Prediction for Sparse Unlabeled Short Text Representation Learning on Social Media

Junyang Chen, *Member, IEEE*, Jingcai Guo, Xueliang Li, Huan Wang, *Member, IEEE*, Zhenghua Xu, Zhiguo Gong, *Senior Member, IEEE*, Liangjie Zhang, *Fellow, IEEE*, and Victor C. M. Leung, *Life Fellow, IEEE*

***Abstract*—Unlabeled text representation learning (UTRL), encompassing static word embeddings such as Word2Vec and contextualized word embeddings such as bidirectional encoder representations from transformer (BERT), aims to capture semantic word relationships in a low-dimensional space without the need for manual labeling. These word embeddings are invaluable for downstream tasks such as document classification and clustering. However, the surge of short texts generated daily on social media platforms results in sparse word cooccurrences, compromising UTRL outcomes. Contextualized models such as recurrent neural network (RNN) and BERT, while impressive, often struggle with predicting the next word due to sparse word sequences in short texts. To address this, we introduce CGraphNet, a contrastive graph context prediction model designed for UTRL. This approach converts short texts into graphs, establishing links between sequentially occurring words. Information from the next word and its neighbors informs the target prediction, a process referred to as graph context prediction, mitigating sparse word cooccurrence issues in brief sentences. To minimize noise, an attention mechanism assigns importance to neighbors, while a contrastive objective encourages more distinctive representations by comparing the target word with its neighbors. Our experiments demonstrate CGraphNet's superior performance over other baselines, particularly in classification and clustering tasks on real-world datasets.**

***Index Terms*—Contrastive graph context prediction, sequential learning, social media short text representation learning, sparsity problem, text mining.**

Received 15 November 2023; revised 19 June 2024 and 3 August 2024; accepted 27 August 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62102265 and Grant 62276089; in part by the Stable Support Project of Shenzhen under Grant 20231120161634002; in part by the Natural Science Foundation of Guangdong Province of China under Grant 2022A1515011474; in part by Guangdong Provincial Key Laboratory of Characteristic Innovation Projects under Grant 2024KTSCX060; in part by Tencent "Rhinoceros Birds"- Scientific Research Foundation for Young Teachers of Shenzhen University; in part by Hong Kong RGC General Research Fund under Grant 152211/23E and Grant 15216424/24E; in part by the National Natural Science Foundation of China under Grant 62102327; in part by PolyU Internal Fund under Grant P0043932 and Grant P0048988; in part by NVIDIA artificial intelligence (AI) Technology Center (NVAITC), Internal Fund of National Engineering Laboratory for Big Data System Computing Technology under Grant SZU-BDSC-IF2024-05; in part by the Natural Science Foundation of Hebei Province under Grant F2024202064; in part by GDST under Grant 2023A0505030013; and in part by Nansha District under Grant 2023ZD001. *(Corresponding authors: Jingcai Guo.)*

Junyang Chen, Liangjie Zhang, and Victor C. M. Leung are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: junyangchen@szu.edu.cn; zhanglj@ieee.org; vleung@ieee.org).

Jingcai Guo is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong 999077, China (e-mail: jingcai.guo@gmail.com).

Xueliang Li is with the National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Shenzhen 518060, China (e-mail: lixueliang@szu.edu.cn).

Huan Wang is with the College of Informatics, Huazhong Agricultural University, Wuhan 430070, China (e-mail: hwang@mail.hzau.edu.cn).

Zhenghua Xu is with the State Key Laboratory of Reliability and Intelligence of Electrical Equipment, Hebei University of Technology, Tianjin 300401, China (e-mail: zhenghua.xu@hebut.edu.cn).

Zhiguo Gong is with the State Key Laboratory of Internet of Things for Smart City, Department of Computer Information Science, University of Macau, Macau 999078, China (e-mail: fstzgg@umac.mo).

Digital Object Identifier 10.1109/TCSS.2024.3452695

I. Introduction

GIVEN the prevalence of short texts in real-world applications such as Twitter and Google News, the field of unlabeled text representation learning (UTRL) has gained considerable attention. Its objective is to map words to low-dimensional vectors using unsupervised learning, ensuring that semantically similar words are embedded close to each other in the vector space. These learned embeddings prove valuable for subsequent tasks such as document classification [1] and clustering [2]. However, popular methods such as Word2vec [3] and bidirectional encoder representations from transformers (BERTs) [4] are typically trained on extensive textual data, leveraging the abundant word cooccurrences present in longer texts [5]. Unfortunately, when a sentence consists of only a few words, the contextualized word embedding methods become similar to static word embedding methods. Consequently, the sparse word cooccurrence patterns in short texts hinder adequate training in UTRL. Our experimental results on TweetSet and TSet, where the average sentence lengths are approximately 6 and 8, respectively, support this perspective, as demonstrated in Tables II and III.

The task of learning representations for short texts encompasses various approaches, such as probability graph models such as gibbs sampling dirichlet mixture model (GSDMM) [6]

and biterm-based mixture model (BMM) [7], which leverage latent topic structures present in short texts to learn document- and word-topic representations, often referred to as distributions. In recent years, there have been proposed embedding learning techniques (not specifically designed for short texts) such as Word2vec [3], smooth inverse frequency (SIF) [1], and non-negative matrix factorization (NMF) [8] that aim to learn word representations by maximizing cooccurrence probabilities. However, these methods encounter two challenges [9]: 1) the lack of parameter sharing between nodes in the encoder leads to a linear growth in the number of parameters with the number of nodes, resulting in computational inefficiency; and 2) the absence of multilayer structures for capturing hierarchical patterns in the datasets. To address these challenges, contextualized word embedding learning based on deep neural networks and graph neural networks (GNNs), such as BERT [4] and IDRL [2], have been proposed. The success of BERT and IDRL can be attributed to their utilization of local word relations, shared weights, and multilayer structures.

It is a relatively new area of research to apply Transformers [4], [10] and GNN [2] for analyzing short texts, despite their groundbreaking achievements in natural language processing (NLP) and recommender systems. Unlike long texts with rich contextual information, short texts often consist of only a few words in a sequence. The original loss function in these models usually utilizes the cross-entropy loss, where the masked tokens/words in the target sentence serve as the positive sample, while the remaining tokens act as negative samples. However, this approach may result in an insufficient number of positive/negative samples in the case of short texts. As such, this sparsity in word sequences can lead to suboptimal results in short text representation learning, particularly in next-word prediction training. To address this issue, one intuitive approach is to enrich short texts by considering bigram patterns [7], [11]. Bigrams help alleviate sparsity by sampling more semantically related words within the same topic during the generative process. However, such methods are bag-of-word models that overlook word order and relations beyond the sequence. A target word may not only be related to words cooccurring in the same sequence but also to words in other sequences.

Motivation: In sequence prediction, the number of words is relatively small, such as in a short text that typically contains around 20 words. This makes it challenging to predict the next word based solely on a given short sequence. By constructing a word graph network across multiple short texts, we can reflect the word cooccurrence probabilities within the graph network into the sequence prediction task, thereby enhancing the model's sequence prediction capabilities.

In Fig. 1, we provide an example of prediction in a short sentence. Given the text "The artificial intelligence (AI) research includes," we aim to predict the probabilities of the target word "natural language processing" and associated words, such as "language recognition" and "expert system." The relevance of these words, represented by 70% and 30%, respectively, is measured using word cooccurrences from the constructed graph. We define the target word as the masked ground-truth word to be predicted, with the contextual words serving as input

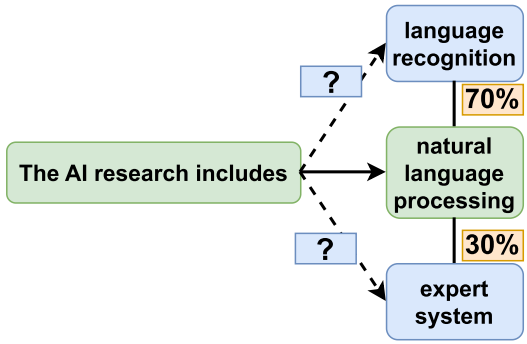

Fig. 1. In a prediction example of short texts, we take the sentence "The AI research includes" and use our model to determine the probabilities of its target word "natural language processing" and its associated words. The percentages 70% and 30% represent the relevance of these words, determined by their cooccurrences within the constructed graph.

during training. The associate words are built based on the graph constructed from words occurring consecutively in a sentence. By incorporating the neighbor words from the graph as additional contextual features, we can enhance the context during the prediction of the target word, thereby ensuring sufficient training data.

To achieve this goal, we propose a contrastive graph context prediction model called *CGraphNet*, which expands the perception areas for target prediction. Starting with raw texts, we construct a text-based graph by connecting words that occur consecutively. We modify the next-word prediction task to a next graph context prediction task by leveraging the neighbors of the next words. During sequential learning, we aggregate information from the next word and the messages exchanged with its neighbors to make the target prediction. To mitigate the impact of irrelevant words, we introduce an attention mechanism to dynamically assign importance weights to the neighbors. Additionally, we employ a contrastive objective to refine the learning process of the target word and its neighbors. The proposed method is evaluated on real-world datasets through classification and clustering experiments. The main contributions of this article can be summarized as follows.

1) We propose a novel method, called graph context prediction, to tackle the problem of sparsity in unlabeled short text representation learning, with a specific focus on social media data.
2) In our approach, we begin by representing the short texts as a graph, where the nodes correspond to words and the edges represent word cooccurrences in the sequences. During sequential training, instead of predicting the target word and its neighbors individually, we aggregate the information of the next word and its neighbors as the target prediction. This allows us to enhance the context by incorporating the neighbor words from the graph as additional contextual features.
3) Furthermore, we employ an attention mechanism to dynamically assign importance weights to the neighbors. This ensures that more attention is given to relevant words in the prediction process. Additionally, we introduce a

contrastive loss function to optimize the learning process between the target word and its neighbors, allowing for more effective representation learning.

4) In the experiments, we thoroughly evaluate the learned word representations by applying them to various classification and clustering tasks. The results of our experiments demonstrate that our method outperforms state-of-the-art competitive models, showcasing the effectiveness and superiority of our approach. We have released a simplified version of the code, and a more comprehensive version will be made public in the revision. The code repository is at https://github.com/anonymou-git/CGraphNet.

The rest of this article is structured as follows. We commence by examining related research in Section II. Subsequently, we present the proposed model, *CGraphNet*, along with its detailed algorithm in Section III. Our experimental results are then outlined in Section IV. Ultimately, Section V offers a conclusion for our study.

II. Related Work

UTRL is widely applicable in various domains, including sentiment analysis [12], code-switching identification [13], and authorship representations [14]. However, UTRL faces a significant challenge due to the sparsity problem, especially in the case of short documents that contain limited words compared to longer documents with richer contexts. To tackle this challenge, most of researchers have explored different approaches, which can be broadly categorized into the probabilistic models and the deep neural network models.

A. Probabilistic Models

Prior to the advent of deep neural networks, statistical methods, such as probabilistic models, have been extensively explored in the field of UTRL for their interpretability and stability. In the context of short texts, several probabilistic models have been proposed. GSDMM [6] leverages a Dirichlet multinomial distribution to model the word-topic representation of a document, assuming that all words are related to the same topic. Biterm topic model (BTM) [15] introduces the concept of biterms (biterm patterns) to capture word cooccurrences in short texts. Subsequent works, such as GSDPMM [16], nonparametric mixture model (NPMM) [17], and DP-BMM [7], focus on enhancing word cooccurrence probability inference for short texts. SIF [1] proposes a relatively simple yet effective document embedding model that achieves better performance compared to classical baselines such as recurrent neural network (RNN) [18] and long short-term memory model (LSTM) [19], due to its statistical advantages. However, these methods neglect the inherent sequential structure of textual data, thereby limiting their ability to capture semantic information.

B. Deep Neural Network Models

With neural network techniques becoming increasingly popular, Word2vec [3] first includes a shallow neural network for word embedding. Then, embeddings from language models (ELMO) [20] proposes learning contextual representations through a task to predict the next words using LSTM [21]. Though these embedding approaches are context-aware, their networks are not deeply bidirectional. After that, OpenAI generative pre-trained transformer (GPT) [22] proposes a deep left-to-right architecture by incorporating the Transformer [23], while every token can only attend to previous tokens in the self-attention layers. To overcome this problem, BERT [4] improves it by introducing BERT. SimCSE [24] presents a contrastive learning framework that greatly advances state-of-the-art sentence embeddings. Nevertheless, these methods are not specially designed for short texts, as such, they could not handle well the sparsity problem during their sequential training. Additionally, some methods [25], [26] apply convolution techniques commonly used in image processing to text representation learning. However, these methods focus solely on the size of the convolutional kernel and fail to account for the global relationships within the text. In recent years, IDRL [2] proposes a GNN-based document representation learning model that can map the short text structures into a graph network and recursively aggregate the neighbor information of the words in the document representation learning. More discussions on the integration of GNN and LSTM methods can be found in [27]. However, how to incorporate GNN with Transformer in unlabeled short texts is still an unknown problem.

III. Proposed Model

The problem formulation and notations are presented in this section. Subsequently, a detailed description of each training step in the algorithm follows the overview of the proposed CGraphNet, as illustrated in Fig. 2.

A. Problem Formulation and Notations

Since our method aims to expand the scope of perception for target prediction in contextualized word embedding learning, the problem can be formulated as follows.

1) Contrastive Graph Context Prediction: We denote a corpus as $\mathbf{C} = (\mathbf{S}, \mathbf{V})$, where $\mathbf{S}$ denotes a set of sequences (or call short documents) and $\mathbf{V}$ represents the vocabulary of the sequences. We denote each sequence $\mathbf{s} \in \mathbf{S}$ contains a list $\mathbf{s} = [v_1, ..., v_t, ..., v_{|\mathbf{s}|}]$ in consecutive order, where $v_t \in \mathbf{V}$ is the word at position t and $|\mathbf{s}|$ is the length of the sequence. Given a sequence $\mathbf{s}$, the original contextualized word embedding learning aims to predict the word that would occur in position $|\mathbf{s}| + 1$, which can be formalized as modeling the probability over all possible words at next position $|\mathbf{s}| + 1$

$$p(v_{|\mathbf{s}|+1} = v|\mathbf{s}). \tag{1}$$

By contrast, we predict the next word and its neighbors in the proposed *CGraphNet*, which is formulated as follows:

$$p(v_{|\mathbf{s}|+1} = \text{Aggregation}(v, \{v_n, \forall v_n \in \mathcal{N}_v\})|\mathbf{s}) \tag{2}$$

where Aggregation denotes that we aggregate the information of the next word and the messages passing from its neighbors

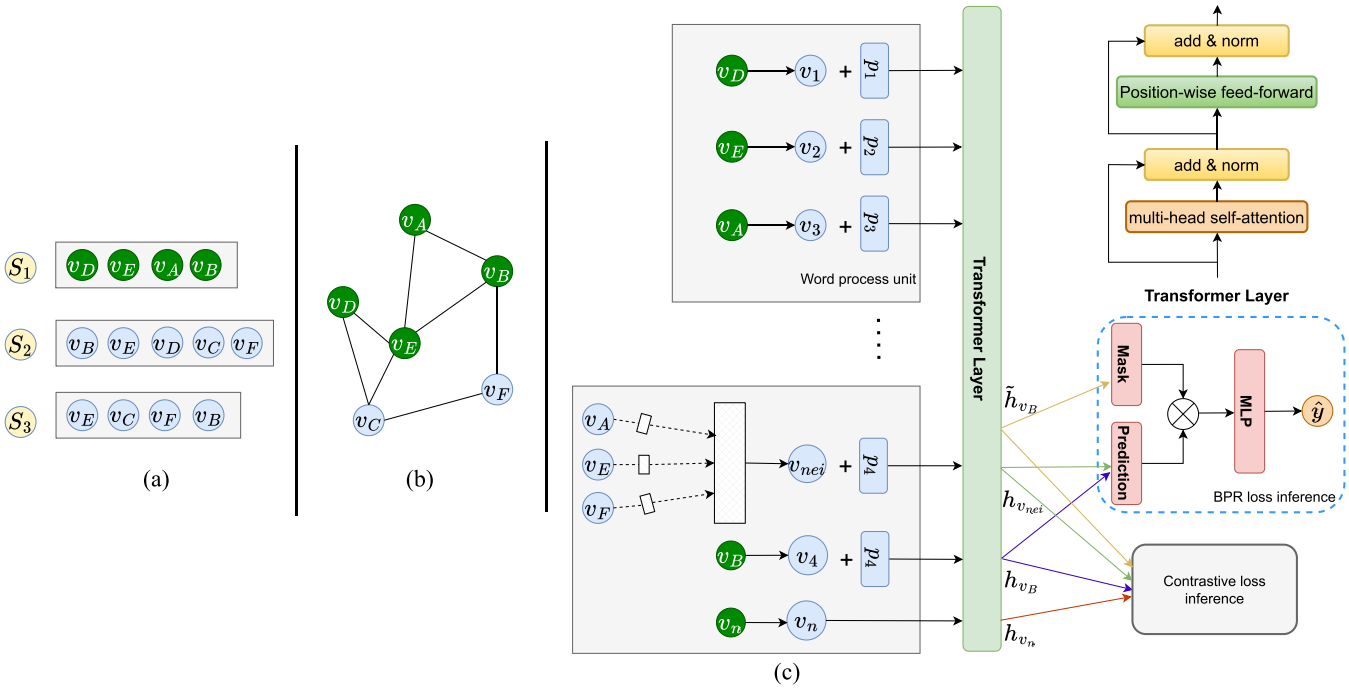

Fig. 2. Overview of the proposed CGraphNet model: (a) input word sequences where S denotes sequences and v represents words; (b) graph construction where words are connected if they occur consecutively in the sequences; and (c) graph context prediction in sequential learning. Here, we use S_1 as an example where sequence $[v_D, v_E, v_A]$ is given to predict word v_B and its sampled neighbors. Note that here $\tilde{\boldsymbol{h}}_{v_B}$ denotes the ground-truth embedding of the target word v_B, $\boldsymbol{h}_{v_{nei}}$ denotes the embedding of the aggregated neighboring nodes v_{nei}, $\boldsymbol{h}_{v_B}$ is the predicted embedding of the target word, and $\boldsymbol{h}_{v_n}$ denotes the embedding of the negative word v_n.

(referred to *Message Propagation Step* and *Message Aggregation Step* in the following section), and $\mathcal{N}_v$ denotes the set of neighbors of word v. Note that here we aim to augment the context by providing the neighbor words in the graph. However, it would be inefficient to predict the target word and its neighbors one by one. Inspired by the graph attention network [28], we aggregate them with an attention mechanism and only predict the fused information once for high efficiency.

B. Proposed CGraphNet Algorithm

For further clarification, we present the proposed Algorithm 1. Initially, a preprocessing step is applied to the input short texts (line 2). Subsequently, a graph is constructed to represent the text data (line 3). Following this, a batch of sequences is sampled (line 5), and message propagation is conducted to obtain the aggregated embedding of the target word and its neighbors for target prediction (line 8). Additionally, position embeddings are added to the words of the sequence (line 9). Finally, stochastic gradient descent [29] is utilized for optimization based on the BPR loss and the contrastive loss (line 12). The preceding steps (lines 5–12) are repeated until convergence is achieved, indicated by the stability of the loss values.

C. Step One: Graph Construction

In this part, we introduce the graph construction of the input word sequences. As shown in Fig. 2(a), we present each short document as a sequence, e.g., S_1 contains four words. When two words appear consecutively in a sequence, they are connected by an undirected edge. For example, as shown in Fig. 2(b), words D and E are connected since they occur sequentially in sequence S_1. By capturing transitive dependencies between words across all sequences, CGraphNe will predict graph context for words based on their semantic neighbors. Note that here we use a window size of 2 because we found that using larger window sizes results in an exponential increase in data volume. Therefore, we opted for the simplest construction method.

D. Step Two: Embedding Fusion

To broaden the scope of target prediction, we predict the next word and its neighboring words during sequential training. However, calculating the loss for every neighbor is not an efficient approach. Therefore, we introduce the steps of *message propagation* and *message aggregation* to consolidate the information of the target word and its neighbors. This way, we only need to calculate the prediction loss once, as is typically done. The specific details are outlined as follows.

1) Message Propagation: This step involves interactions between a target word (e.g., word B) and its neighboring words (e.g., words A, E, and F), as illustrated in Fig. 2(b). Drawing inspiration from prior research [30] focused on extracting features from locally constructed

Algorithm 1: The Training Process for CGraphNet

Input: Short text corpus $\mathbf{C} = (\mathbf{S}, \mathbf{V})$, word embedding dimension d, number of multi-head K, and number of the Transformer layer L
Result: Vertex embedding $\mathbf{v}, \forall v \in \mathbf{V}$

```
begin
    Preprocessing: removing stopwords and stemming words;
    Step one: construct a graph based on the words consecutively occurring in the sequences;
    while not converge do
        Sample a batch 𝒯 of sequences from S;
        for s ∈ S do
            for each target word v ∈ s do
                Step Two: aggregate the representations of this word and its neighbors as h_{v_i} according to Eq. (6) for the target prediction;
                Step Three: add the position embedding into the words according to Eq. (7);
            end
            Step Four: obtain the output h_{v_i} as the target prediction based on Eq. (10);
            Step Five: construct the BPR loss and the contrastive loss, as shown in Eq. (14) for optimization;
        end
    end
end
```

neighbors, we define the message passing from word v_j to v_i as follows:

$$\boldsymbol{m}_{v_i \leftarrow v_j} = \mathcal{F}(n_{v_j v_i}, \boldsymbol{h}_{v_j}) \cdot \boldsymbol{h}_{v_j} \tag{3}$$

where $\boldsymbol{m}_{v_i \leftarrow v_j}$ represents the message passed from nodes v_j to v_i in a d-dimensional space, $n_{v_j v_i}$ stands for the neighbor type (e.g., one-hop or multihop neighbors), $\boldsymbol{h}_v \in \mathbb{R}^d$ is the hidden representation of word v, and $\mathcal{F}(\cdot)$ takes both the representations of the neighbor word $\boldsymbol{h}_{v_j}$ and the neighbor type $n_{v_j v_i}$ as inputs, yielding a output matrix $\in \mathbb{R}^{d \times d}$. Here, $n_{v_j v_i}$ is a one-hot encoded neighbor type. We begin by concatenating $n_{v_j v_i}$ and $\boldsymbol{h}_{v_j}$, and then pass them through a multilayer perception (MLP) for further processing. The details of $\mathcal{F}(\cdot)$ are provided as follows:

$$\mathcal{F}(n_{v_j v_i}, \boldsymbol{h}_{v_j}) = \text{MLP}(n_{v_j v_i} || \boldsymbol{h}_{v_j}) \tag{4}$$

where symbol $||$ denotes the concatenation operation.

2) Message Aggregation: During this step, we aggregate the information associated with the target word and the messages passing from its neighbors, as illustrated in Fig. 2(c). To accomplish this, we utilize an *Attention* aggregator, which incorporates the attention mechanism [23]. This mechanism helps in learning the importance weights of words, which is essential for reducing the noise information propagated from neighbors. We calculate the weight coefficient α_{v_i, v_j} between two words using the following equation:

$$\alpha_{v_i, v_j} = \frac{\exp\Big(\sigma(\mathbf{a}^T \cdot [\boldsymbol{F}\boldsymbol{h}_{v_i} || \boldsymbol{F}\boldsymbol{m}_{v_i \leftarrow v_j}])\Big)}{\sum_{v_k \in \mathcal{N}_{v_i}} \exp\Big(\sigma(\mathbf{a}^T \cdot [\boldsymbol{F}\boldsymbol{h}_{v_i} || \boldsymbol{F}\boldsymbol{m}_{v_i \leftarrow v_k}])\Big)} \tag{5}$$

where $\boldsymbol{F} \in \mathbb{R}^{d \times d}$ is a trained weight matrix used to map words into the same feature space. $\mathbf{a} \in \mathbb{R}^{2d}$ represents the relation weights associated with the target word and its neighbors, and $\mathcal{N}_{v_i}$ denotes the set of neighbors of word v_i. Subsequently, with the acquired weight coefficients and the information from neighbors, the final representations of word v_i are formulated as follows:

$$\boldsymbol{h}_{v_i} = \sigma\Big(\sum_{v_j \in \mathcal{N}_{v_i}} \alpha_{v_i, v_j} \boldsymbol{F}\boldsymbol{m}_{v_i \leftarrow v_j}\Big) \tag{6}$$

where σ is the ReLU [31] function for activation.

E. Step Three: Position Injection

Up to now, we have introduced how we construct the graph and aggregate the information of words. The next step is to compute the positions of the words for the input sequence before feeding the embeddings into the Transformer layer, as shown in Fig. 2(c).

To account for the sequential order of words in the sequences, we introduce position embeddings into the word representations. In this regard, a learnable embedding matrix $\boldsymbol{P} \in \mathbb{R}^{L \times d}$ is employed to encode the words, where L represents the maximum length of the sequences. For instance, when dealing with a word embedding $\boldsymbol{h}_{v_i}$ within a sequence, the following formulation is applied:

$$\boldsymbol{h}_{v_i} = \boldsymbol{h}_{v_i} + \boldsymbol{p}_i \tag{7}$$

where $\boldsymbol{p}_i \in \boldsymbol{P}$ is the d-dimensional position embedding with index i, for example, $\boldsymbol{p}_1$ denotes the first word in a sequence.

F. Step Four: Transformer Layer in Contextualized Learning

Following the described steps, the word representations $\boldsymbol{h}_{v_i}$ of a sequence are passed through the Transformer Layer [23]. Unlike the original Transformer layer, we only utilized the encoder module of the Transformer, as shown in Fig. 2, to learn the relationships between words in a sentence. This layer typically comprises two main components: *multihead self-attention* and *positionwise feed forward*. We will provide a brief overview of these components in the following sections.

1) Multihead Self-Attention: We represent an input sequence using a matrix $\boldsymbol{H}^l \in \mathbb{R}^{|S| \times d}$, where $|S|$ is the length of the sequence, l is the layer number, and $\boldsymbol{h}_{v_i}^l \in \boldsymbol{H}^l$ denotes the embedding of the word v_i within the sequence. To ensure a stable training process and enable the model to capture word information from multiple subspaces and positions simultaneously, we employ multihead self-attention. This projection of $\boldsymbol{H}^l$ into n embedding subspaces can be expressed as follows:

$$\boldsymbol{H}^l = \text{Concat}(\text{head}_1, ..., \text{head}_n)\boldsymbol{W}^O \tag{8}$$

$$\text{where head}_i = \text{Attention}(\boldsymbol{H}^l\boldsymbol{W}_i^Q, \boldsymbol{H}^l\boldsymbol{W}_i^K, \boldsymbol{H}^l\boldsymbol{W}_i^V)$$

where $\boldsymbol{W}_i^Q \in \mathbb{R}^{d\times d/n}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{d\times d/n}$, $\boldsymbol{W}_i^V \in \mathbb{R}^{d\times d/n}$, and $\boldsymbol{W}_i^O \in \mathbb{R}^{d\times d}$ are learnable projection matrices. Moreover, the Attention$(\cdot)$ function in (8) is given by

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d/n}}\right)V \tag{9}$$

where Q, K, and V correspond to $\boldsymbol{H}^l\boldsymbol{W}_i^Q$, $\boldsymbol{H}^l\boldsymbol{W}_i^K$, and $\boldsymbol{H}^l\boldsymbol{W}_i^V$, respectively, while $\sqrt{d/n}$ serves as a temperature factor used for scaling to prevent extremely small gradients [23].

2) Positionwise Feed Forward: We incorporate a positionwise feed-forward network (FFN) layer [23] into the model to enhance the power of nonlinear expression and learn the interactions between dimensions. Two linear transformations are contained in the FFN layer, and in between that is an activation function for the ReLU, which is defined as follows:

$$\text{FNN}(\boldsymbol{x}) = \max(0, \text{ReLU}(\boldsymbol{x}\boldsymbol{F}_1 + \boldsymbol{b}_1))\boldsymbol{F}_2 + \boldsymbol{b}_2 \tag{10}$$

where $\boldsymbol{F}_1 \in \mathbb{R}^{d\times 4d}$, $\boldsymbol{F}_2 \in \mathbb{R}^{4d\times d}$, $\boldsymbol{b}_1 \in \mathbb{R}^{4d}$, and $\boldsymbol{b}_2 \in \mathbb{R}^d$ are learnable parameters, $\boldsymbol{x}$ represents the output of (9).

Discussion. There are two main differences from the original transformer: 1) the original transformer is designed for the translation tasks with the encoder for the source domain and the decoder for the target domain. Different from it, we remove the decoder since our method only needs to model a sentence as a sequence in the next-token prediction task. 2) The original loss function is based on the cross-entropy loss: the masked token in the target sentence is the positive sample, and the other ones as the negative samples. This may cause an insufficient number of negative samples in short texts. By contrast, our method uses the BPR loss and the contrastive loss (the details will be introduced in the following section). Compared with the original one, the main difference is to take the words from other sequences in the training batch as the negative samples. In this way, more negative words can be involved in the training to achieve better performance.

G. Step Five: Contrastive Objective

The goal is to predict the likelihood of the next word and its neighbors, giving the words at positions less than that of the predicted one. We formulate the objective with the two following parts.

1) Bayesian Personalized Ranking (BPR) Loss: After feeding the words of the sequence into the Transformer layer, we can obtain the final output $\boldsymbol{h}_{v_i}$ as the representation of the target prediction, e.g., $\boldsymbol{h}_{v_B}$ as shown in Fig. 2(c). Besides, we have the masked graph context embedding $\tilde{\boldsymbol{h}}_{v_i}$ as the ground truth, e.g., $\tilde{\boldsymbol{h}}_{v_B}$, the fused information of v_B and its neighbors by (6). Next, we concatenate the embeddings of $\boldsymbol{h}_{v_i}$ and $\tilde{\boldsymbol{h}}_{v_i}$, and apply a MLP to calculate the prediction score

$$\hat{y}_{s,v_i} = \sigma(\text{MLP}(\boldsymbol{h}_{v_i}||\tilde{\boldsymbol{h}}_{v_i})) \tag{11}$$

where σ represents an activation function, $||$ signifies concatenation, and $\hat{y}_{s,v_i}$ stands for the prediction score of the next word v_i and its neighbors within a sequence s. Last, we employ a commonly used BPR loss [32] to train our model. This loss function is a pairwise loss designed to encourage the model to rank positive samples higher than negative ones. The BPR loss can be expressed as

$$\mathcal{L}_{\text{BPR}} = \sum_{(s,v_i,v_n)\in\mathcal{T}} -\log\,\sigma(\hat{y}_{s,v_i} - \hat{y}_{s,v_n}) \tag{12}$$

where $\mathcal{T}$ denotes a training batch, $\hat{y}_{s,v_i}$ is the prediction score of a positive sample (the aggregated information from the target word and its neighbors), and $\hat{y}_{s,v_n}$ is the score of a negative sample (a nontarget word from other sequences in the training batch).

2) Contrastive Loss: When computing the above loss, we find that the margin between a target word and its neighbors may become vague. Therefore, apart from the general target prediction with the BPR loss, we further exploit a contrastive loss to refine the learning process of the target word and its neighbors. To be specific, we maximize the difference between the positive sample (the aggregated information of the target word and its neighbors) and the negative sample. Meanwhile, we pull the predictions of the target word and its neighbors away to learn more discriminative representations. Inspired by the contrastive learning [33], [34], we define the pairwise contrastive loss as follows:

$$\mathcal{L}_{\text{cont}} = -\log \frac{\exp(\boldsymbol{h}_{v_i}^T\tilde{\boldsymbol{h}}_{v_i})}{\exp(\boldsymbol{h}_{v_i}^T\tilde{\boldsymbol{h}}_{v_i}) + \exp(\boldsymbol{h}_{v_i}^T\boldsymbol{h}_{v_{nei}}) + \exp(\boldsymbol{h}_{v_i}^T\boldsymbol{h}_{v_n})} \tag{13}$$

where $\boldsymbol{h}_{v_i}$ is the representation of the target-word prediction, $\tilde{\boldsymbol{h}}_{v_i}$ is the word embedding of the ground truth, $\boldsymbol{h}_{v_{\text{nei}}}$ denotes the embedding of the aggregated information from neighboring nodes, v_n denotes the negative samples of words randomly drawn from the training batch $\mathcal{T}$. Based on (12) and (13), we arrive at the total loss as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{BPR}} + \mathcal{L}_{\text{cont}}. \tag{14}$$

With the contrastive objective as shown in the above equation, we can not only predict the likelihood of the next word and its neighbors but also can refine the learning process that is maximizing the difference between the positive and negative samples while pulling the predictions of the target word and its neighbors away to learn more discriminative embeddings.

H. Time Complexity Analysis

The computational complexity of our method consists of two main parts: the embedding fusion from Section III-D and the transformer layer from Section III-F. For the first part, considering the number of edges $\mathbf{E}$ contained in the constructed graph, the time complexity can be estimated as $O(|\mathbf{E}| \times n_g \times d)$, where $|\mathbf{E}|$ denotes the edge size, n_g is the number of hidden layers contained in the GNN (generally set to 2), and d is the dimension of the hidden layers. It is important to note that the word cooccurrence matrix is sparse in short texts, and in the

TABLE I
STATISTICS OF DATASETS

Dataset	Vocabulary	Avg. Len	Sequences	Labels
TweetSet	5098	8.56	2472	89
TSet	8111	6.23	11 109	152
SSet	18 478	22.20	11 109	152
TSSet	19 672	28.43	22 218	152
Trends-T	45 404	8.43	200 000	10

implementation, we use sparse-dense matrix multiplications. This means that the computational complexity is linear with respect to the number of graph edges. For the second part, the transformer layer, the time complexity can be estimated as $O((|\mathbf{S}|^2 \times d + |\mathbf{S}| \times d^2) \times L)$, where $|\mathbf{S}|$ is the input sequence size, d is the dimension of the hidden layers, and L is the number of transformer layers. The overall time complexity of our method is $O(|\mathbf{E}| \times n_g \times d + (|\mathbf{S}|^2 \times d + |\mathbf{S}| \times d^2) \times L)$. The first part is linear with respect to the sparse word cooccurrences, while the second part is the same as the original Transformer.

IV. EXPERIMENTS

To assess the effectiveness of the acquired word representations, we utilized real-world datasets for downstream tasks, such as text classification and clustering. Additionally, we conducted a comprehensive ablation analysis to ascertain the most impactful component within our method.

A. Datasets

The experiment was conducted using five widely used short text datasets, and their statistics are reported in Table I.

1) TweetSet[1]*:* This dataset consists of tweets from the Text REtrieval Conference microblog tracks for 2011 and 2012, comprising a total of 109 queries. The tweets were evaluated and categorized by assessors into four groups: spam, irrelevant, relevant, and highly relevant. Following the approach described in [6], each query is treated as a cluster, with its highly relevant tweets forming short documents. After removing queries with no highly relevant tweets, our dataset comprises 2472 tweets and 89 clusters (topic labels). We refer to this dataset as TweetSet.

2) Google News[2]*:* Following a similar approach as described in [6] and [35], our algorithm retrieves the titles and snippets of 11109 news articles, naturally grouped into 152 cluster labels (stories) based on a snapshot taken on 27 November 2013. The dataset comprises three subsets: TitleSet (*TSet*), SnippetSet (*SSet*), and TitleSnippetSet (*TSSet*). TSet and SSet contain titles and snippets separately, while TSSet contains titles and snippets together. These three datasets, varying in length, are utilized to evaluate the performance of different clustering methods.

3) Trends-T: As described in [36], hashtag/trend search results for each individual hashtag/trend represent a single topic distribution. The authors utilized the Twitter API to collect labeled data from the dedicated Academic API. To classify tweets belonging to the same hashtag, they queried Tweets (sequences) on the day the hashtags appeared (covering 24 h). A total of 200 000 Tweets were collected across ten different topics or trends.

A simplified version of all datasets after preprocessing and codes have been published in the GitHub.[3]

B. Baselines

In this study, we incorporate various state-of-the-art approaches that have been designed to effectively learn text representations from unlabeled text (UTRL). Specifically, we consider methods such as contextualized word embedding learning, static word embedding learning, and probabilistic graph learning. While there are numerous other learning methods available, we have chosen to focus on these selected baselines for specific reasons. Some methods have been excluded because they have been shown to perform worse than the chosen baselines, as demonstrated in their respective articles. Others are supervised models that are not suitable for UTRL, making them inappropriate for our study. Here are brief descriptions of the selected baselines.

BERT [4] has gained a great deal of attention for its success in learning sentence representations in recent years. Based on multihead self-attention [23], it achieves state-of-the-art results when modeling text sequences.

SimCSE [24] presents a framework for advancing state-of-the-art sentence embeddings by blending contrastive learning and sentence embedding techniques. Using only standard dropout as noise, it reconstructs an input sentence from its embedding in a contrastive objective.

Word2vec [3] is the first study to utilize a two-layer neural network to learn word embeddings and is one of the most popular models. For the sake of computational efficiency, the designed structure is shallow.

NMF [37] proposes a nonnegative matrix factorization approach to learn topic-word distributions of corpora. The learned distributions can be regarded as the word representations, which improve the interpretability of the results.

GSDMM [6] is a traditional well-known probabilistic graph model that is designed for the short text modeling. The word and document representations are learned by combining Dirichlet and multinomial distributions.

BTM [15] alleviates the sparse word cooccurrence problem through aggregated patterns (i.e., biterms) when modeling short texts.

SIF [1] provides a relatively simple, yet effective LSTM [19] and RNN [18] based sentence embedding model that achieves better results than baselines. Since this method requires pretrained word embedding, we exploit GloVe [38] that is trained on several large social network corpora.

[1]https://trec.nist.gov/data/microblog.html
[2]https://news.google.com/news/
[3]https://github.com/anonymou-git/CGraphNet

TABLE II
SEQUENCE CLASSIFICATION RESULTS (%) OF TWEETSET

Method	10%		20%		40%		60%		80%	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
GSDMM	29.95	13.31	29.85	13.32	35.84	16.25	31.13	14.69	36.65	19.58
BTM	30.41	13.74	30.89	14.26	36.67	17.86	32.91	17.19	40.73	26.43
NMF	34.90	17.80	36.29	20.09	37.64	22.95	41.72	26.42	43.14	28.95
Word2vec	49.50	44.75	52.71	46.87	53.29	48.07	55.23	48.94	56.34	50.65
SimCSE	43.91	40.19	47.03	44.19	52.62	50.55	55.03	52.56	56.15	53.47
BERT	51.48	49.16	52.36	49.97	55.90	53.50	55.82	53.10	56.15	52.49
SIF	57.30	54.13	60.30	57.02	62.27	59.25	62.74	60.43	64.50	62.32
IDRL	59.41	50.89	65.60	60.80	69.56	66.03	71.66	69.08	72.57	70.48
CGraphNet	**65.59**	**62.75**	**68.82**	**66.45**	**71.77**	**69.33**	**73.22**	**71.03**	**75.18**	**72.82**

Note: The bold values indicate statistically significant results ($p < 0.05$) compared to baseline values.

IDRL [2] is a GNN-based document representation learning model. It utilizes recursive aggregation of neighbor information from short text structures by mapping them into a graph network.

C. Implementation Settings

The proposed method is implemented using TensorFlow [39]. The embedding size is uniformly set to 128 for all models. For BERT, Word2Vec, IDRL, and our CGraphNet, model parameters are randomly initialized with a Gaussian distribution, and the models are optimized using mini-batch Adam [29]. We set the learning rate to 1e-3, the batch size to 512, and the maximum sequence length to 50. Additionally, we set the numbers of Transformer layers and attention heads to 10 and 2, respectively. For SIF, which requires pretrained embeddings to initialize parameters, we use the GloVe word embedding [38], trained on several large social network corpora. To obtain the best performance for NMF, GSDMM, BTM, and IDRL, which require predefined topic numbers, we use the ground-truth setting of the datasets. Regarding our proposed method, when computing the aggregation function in (2), we follow established engineering practices, as seen in well-accepted graph-based methods such as GraphSAGE [40] and PinSage [41]. Specifically, we randomly sample a small number of neighbors (typically around 5–10 in our cases) for each node during training. This approach reduces the parameter complexity from O(N) to O(m), where N represents the average number of neighbors, and m denotes the predefined sampling number.

D. Evaluation Metrics

Note that each sample in the dataset has only one label, indicating that the classification task is a single-label classification. To assess the discriminative power of the proposed model, we utilize the micro-F1 and macro-F1 metrics [42]. We build the sequence classifier using the Liblinear package [43] with default settings. These metrics are widely used for evaluating classification performance. For evaluating the clustering performance of sequences, we employ two common metrics: clustering accuracy (ACC) [44] and normalized mutual information (NMI) [45]. When the clustering results perfectly match the ground-truth topics, both ACC and NMI can reach a value of one. Conversely, if the results are generated randomly, the values are expected to be close to zero.

E. Evaluation on Classification

A set of five real-world datasets is utilized in this section to demonstrate the downstream classification tasks' performance. It is essential to note that all baselines are trained unsupervised here, without utilizing labeled data. We begin by inputting all datasets to obtain the pretrained models. Subsequently, we execute downstream classification tasks by employing the Liblinear package [43] with default settings to build the classifier, enabling a comparison of the performance of all models after pretraining. To perform the experimental data split, we randomly select 10%, 20%, 40%, 60%, and 80% of sequences as the training sets with labels to build the classifier. The remaining sequences are used as the testing sets to predict their labels. In this scenario, we do not consider the validation setting since promoting test performance for all baselines will not affect the classification results' ranking. The micro-F1 and macro-F1 results are presented in Tables II–V, where the highest scores are indicated in bold type. From these tables, the following observations can be made.

1) Across all datasets with varying training ratios, the proposed CGraphNet consistently outperforms the other models, demonstrating its remarkable effectiveness and superiority in the downstream classification tasks.
2) Based on the overall test results, the baselines are ranked in the following order: CGraphNet > IDRL > BERT > SimCSE > SIF > Word2vec > NMF > BTM > GSDMM. On average, our CGraphNet achieves substantial performance gains over the second-best model, IDRL, with improvements of 6.23%, 5.87%, 33.10%,

TABLE III
SEQUENCE CLASSIFICATION RESULTS (%) OF TSET

Method	10%		20%		40%		60%		80%	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
GSDMM	30.69	11.20	32.07	13.54	42.35	25.42	39.36	23.69	44.42	28.65
BTM	32.51	14.39	36.00	19.54	45.60	29.52	44.83	30.33	48.59	33.99
NMF	33.98	16.73	35.53	19.70	39.85	27.26	39.52	27.98	40.80	28.96
Word2vec	60.13	55.87	62.94	58.83	64.67	60.81	65.25	61.60	65.39	61.03
SimCSE	56.45	54.24	60.65	58.60	62.53	60.34	62.65	60.43	62.52	59.86
BERT	60.02	58.00	64.18	62.01	65.67	63.64	65.13	63.18	65.26	63.32
SIF	57.96	54.01	58.97	55.20	60.26	56.60	60.33	56.97	60.16	56.51
IDRL	60.68	54.33	64.58	59.79	67.80	64.22	69.01	65.87	69.66	66.14
CGraphNet	**65.24**	**62.81**	**68.27**	**65.99**	**70.40**	**68.42**	**70.70**	**68.87**	**71.15**	**67.95**

Note: The bold values indicate statistically significant results ($p < 0.05$) compared to baseline values.

TABLE IV
SEQUENCE CLASSIFICATION RESULTS (%) OF SSET

Method	10%		20%		40%		60%		80%	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
GSDMM	31.67	12.16	31.87	12.64	32.05	12.42	31.37	12.00	31.98	12.28
BTM	31.46	12.15	31.57	12.08	31.58	12.05	31.74	12.83	31.76	12.08
NMF	31.94	12.79	32.87	14.66	34.49	16.51	34.42	17.33	35.49	18.23
Word2vec	52.01	44.19	53.75	46.60	55.66	48.67	55.85	48.89	56.98	49.91
SimCSE	59.62	56.34	62.83	59.60	65.05	61.85	64.94	61.55	65.89	62.20
BERT	64.52	61.64	67.06	64.12	68.70	65.64	68.76	65.47	69.58	65.99
SIF	57.46	52.92	59.32	54.90	60.25	56.12	60.45	56.07	59.78	55.64
IDRL	53.36	40.50	57.65	47.45	61.63	53.93	62.26	55.32	63.40	56.75
CGraphNet	**72.15**	**69.86**	**74.27**	**72.03**	**75.33**	**73.07**	**75.20**	**73.08**	**76.19**	**73.85**

Note: The bold values indicate statistically significant results ($p < 0.05$) compared to baseline values.

TABLE V
SEQUENCES CLASSIFICATION RESULTS (%) OF TSSET

Method	10%		20%		40%		60%		80%	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
GSDMM	33.01	12.40	32.96	12.39	33.15	12.50	32.58	12.34	32.81	12.41
BTM	31.67	12.15	31.57	12.08	31.58	12.05	32.21	13.49	31.93	12.46
NMF	35.54	11.46	36.28	13.30	37.61	15.69	39.76	19.06	40.90	21.38
Word2vec	57.20	51.06	58.81	52.72	60.05	54.06	60.23	53.88	61.40	54.85
SimCSE	67.82	64.69	70.97	73.04	72.91	70.17	73.04	70.31	73.29	70.32
BERT	72.68	69.97	74.12	71.40	75.66	73.12	75.71	73.15	77.13	74.63
SIF	48.38	42.85	50.07	45.16	51.13	46.21	51.06	46.50	52.35	47.05
IDRL	55.00	44.95	58.12	49.33	62.25	54.77	63.65	57.32	65.64	60.55
CGraphNet	**77.67**	**75.55**	**80.01**	**78.16**	**80.87**	**79.18**	**81.65**	**80.04**	**82.47**	**80.92**

Note: The bold values indicate statistically significant results ($p < 0.05$) compared to baseline values.

and 39.35% on TweetSet, TSet, SSet, and TSSet, respectively. Furthermore, our method demonstrates significant advantages over the contextualized word embedding model, BERT, with improvements of 31.52%, 7.83%, 11.12%, and 7.99% on the respective datasets. These results strongly suggest that our proposed graph context prediction method is remarkably effective in learning more powerful and discriminative representations.

TABLE VI
CLASSIFICATION RESULTS (%) OF TRENDS-T

Method	10%		50%	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1
SimCSE	81.47	80.22	81.70	80.50
BERT	81.80	80.52	81.97	80.74
IDRL	72.63	71.72	72.63	74.28
CGraphNet	**84.46**	**83.20**	**84.69**	**83.47**

Note: The bold values indicate statistically significant results ($p < 0.05$) compared to baseline values.

3) Moreover, for the datasets TSet, SSet, and TSSet, which share the same background knowledge but have varying sequence lengths, we observe that the classification performances of BERT and CGraphNet increase proportionally with the sequence length (please refer to the statistics in Table I). This observation underscores the necessity and potential of enhancing unlabeled short text representation learning through sequential training. The ability of CGraphNet to consistently improve its performance along with the sequence length highlights its capacity to effectively capture and leverage context in the learning process.

Additional observations: There are substantial differences between the probability statistical models (GSDMM and BTM) and other neural network-based models. Furthermore, despite NMF and Word2Vec belonging to relatively similar algorithmic categories, Word2Vec's performance is enhanced by its word cooccurrence augmentation operation. By constructing more word pairs through a sliding window context, Word2Vec achieves superior performance compared to NMF.

1) Evaluation on Large-Scale Datasets: To ensure the consistency of experimental performance, we conduct an evaluation on a large-scale dataset containing 200 000 tweets. For comparison, we select SimCSE, BERT, and IDRL, as they have demonstrated good performance on previous small-scale datasets. The results are presented in Table VI. Notably, our method achieves average improvements of 3.46% and 3.35% on the 10% and 50% training sets, respectively, when compared to the second-best method. These results affirm that our method continues to perform effectively on large-scale datasets.

F. Evaluation on Clustering

To further evaluate the learned representations of our method, we perform downstream clustering tasks by comparing the performance of the top-3 baselines (IDRL, SIF, and BERT), as illustrated in Fig. 3. For this evaluation, we uniformly run K-means[4] on the learned representations, adopting the ground-truth setting for the number of clusters, as indicated in Table I. The observations from the clustering results are as follows.

[4]https://scikit-learn.org/stable/

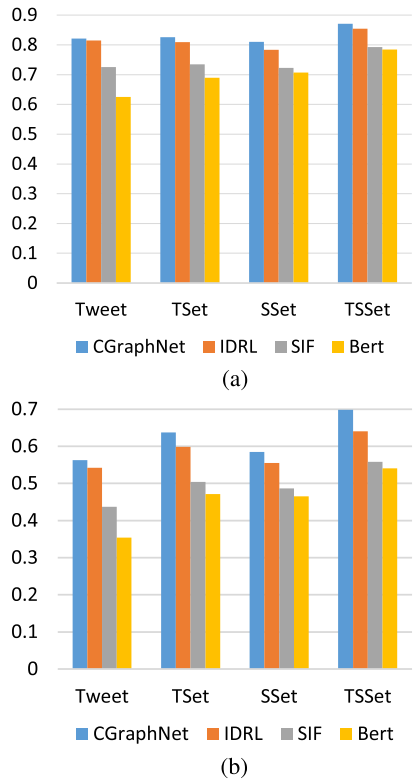

Fig. 3. Clustering performance with (a) NMI and (b) ACC metrics.

1) In general, the overall clustering performance is as follows: CGraphNet > IDRL > BERT > SIF. Specifically, from Fig. 3(a), it can be observed that CGraphNet achieves improvements of 0.79%, 2.12%, 3.38%, and 1.96% in terms of NMI performance over IDRL on Tweet, TSet, SSet, and TSSet, respectively. These results provide strong validation for the effectiveness of our designed model.
2) Furthermore, from Fig. 3(b), it is evident that CGraphNet consistently outperforms IDRL. Specifically, CGraphNet achieves improvements of 3.78%, 6.54%, 5.40%, and 9.03% in terms of ACC performance on Tweet, TSet, SSet, and TSSet, respectively. These results further confirm the superiority of our method in learning more discriminative representations.

G. Ablation Analysis

To analyze the effectiveness of the various components proposed in our method, we conduct an ablation analysis as presented in Tables VII and VIII. The highest scores are marked in boldface, and the second-best performance results are denoted with an asterisk (*). It is important to note that the method without graph construction means only the target and the context are used as input to the transformer layer.

From these tables, we observe that our proposed method CGraphNet consistently outperforms the other variants in terms of both classification and clustering metrics. More specifically, CGraphNet achieves improvements of 1.75%, 1.77%, 0.80%,

TABLE VII
ABLATION ANALYSIS ON TWEETSET AND TSET

Method	TweetSet				TSet			
	Micro-F1	Macro-F1	NMI	ACC	Micro-F1	Macro-F1	NMI	ACC
w/o graph construction	0.5593	0.5285	0.6251	0.3539	0.6643	0.6340	0.6893	0.4711
w/o Transformer	0.7128	0.6791	0.8150*	0.5424*	0.6955*	0.6631	0.8090*	0.5984*
w/o BPR loss	0.3520	0.2751	0.3231	0.1646	0.3069	0.2027	0.1890	0.0571
w/o contrastive loss	0.7180*	0.6965*	0.7179	0.4785	0.6915	0.6786*	0.7439	0.5306
CGraphNet	**0.7306**	**0.7088**	**0.8215**	**0.5630**	**0.7087**	**0.6887**	**0.8262**	**0.6375**
Improv.	1.75%	1.77%	0.80%	3.80%	1.90%	1.49%	2.13%	6.53%

Note: The bold values indicate statistically significant results ($p < 0.05$) compared to baseline values.

TABLE VIII
ABLATION ANALYSIS ON SSET AND TSSET

Method	SSet				TSSet			
	Micro-F1	Macro-F1	NMI	ACC	Micro-F1	Macro-F1	NMI	ACC
w/o graph construction	0.6882	0.6572	0.7070	0.4650	0.7575	0.7318	0.7841	0.5406
w/o Transformer	0.6071	0.5427	0.7832*	0.5550*	0.6291	0.5590	0.8541	0.6407
w/o BPR loss	0.3045	0.1682	0.1706	0.0383	0.3060	0.1648	0.1704	0.0367
w/o contrastive loss	0.7172*	0.6909*	0.7580	0.5354	0.8081*	0.7797*	0.8641*	0.6827*
CGraphNet	**0.7530**	**0.7305**	**0.8097**	**0.5849**	**0.8123**	**0.7950**	**0.8709**	**0.6986**
Improv.	4.99%	5.73%	3.38%	5.39%	0.52%	1.96%	0.79%	2.33%

Note: The bold values indicate statistically significant results ($p < 0.05$) compared to baseline values.

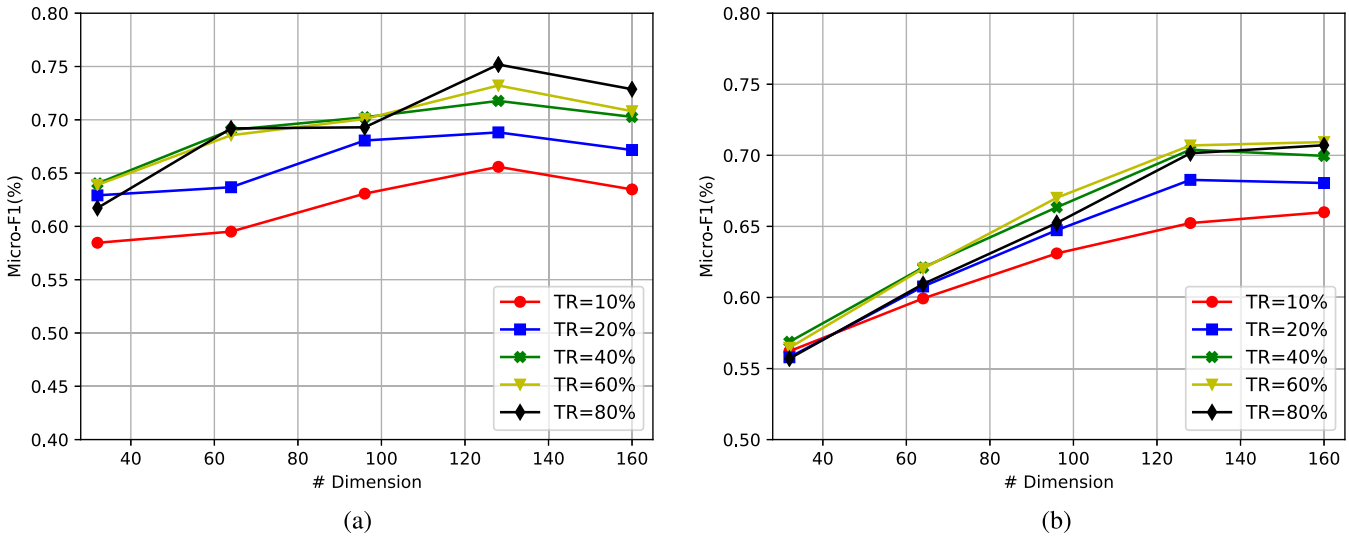

Fig. 4. Influence of dimension setting on downstream classification tasks in (a) Tweet and (b) TSet datasets. Here, "TR" represents training ratios.

and 3.80% over the second-best results in terms of micro-F1, macro-F1, NMI, and ACC on TweetSet, respectively. Moreover, it obtains gains of 1.90%, 1.49%, 2.13%, and 6.53% on TSet, 4.99%, 5.73%, 3.38%, and 5.39% on SSet, and 0.52%, 1.96%, 0.79%, and 2.33% on TSSet, respectively. These results provide strong evidence validating the effectiveness of

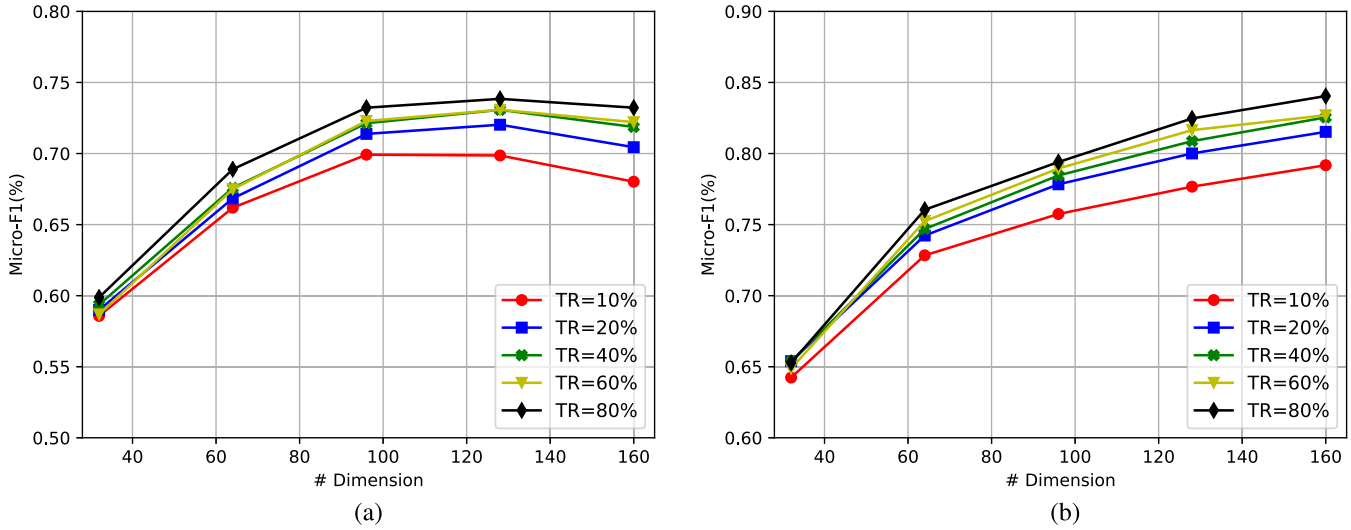

Fig. 5. Influence of dimension setting on downstream classification tasks in (a) SSet and (b) TSSet datasets. Here, "TR" represents training ratios.

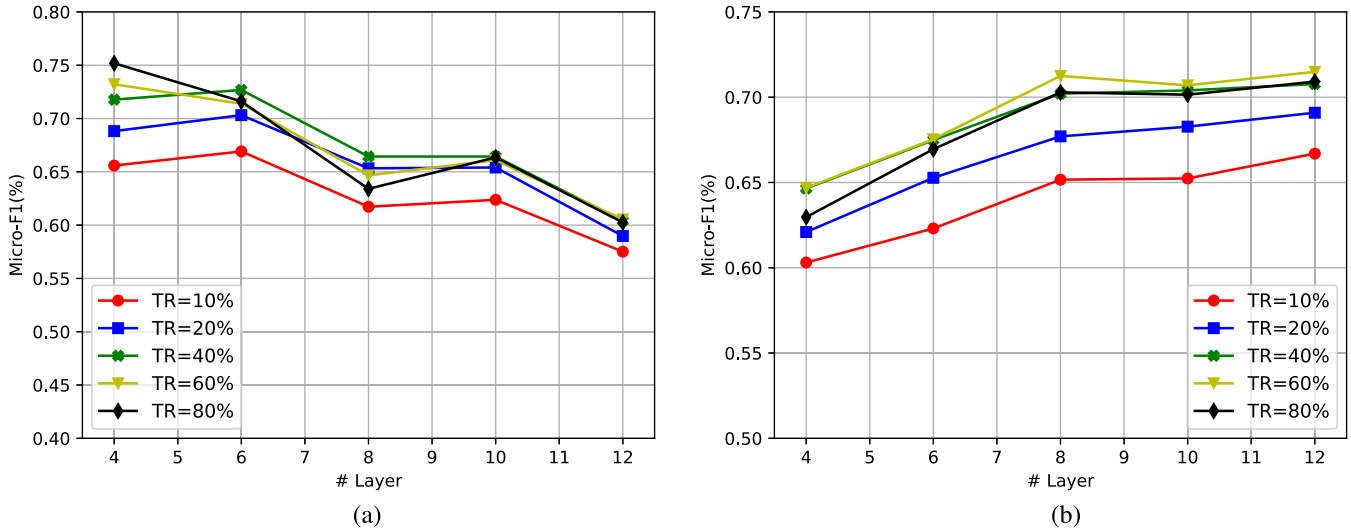

Fig. 6. Influence of the Transformer layer number on downstream classification tasks in (a) Tweet and (b) TSet datasets. Here, "TR" represents training ratios.

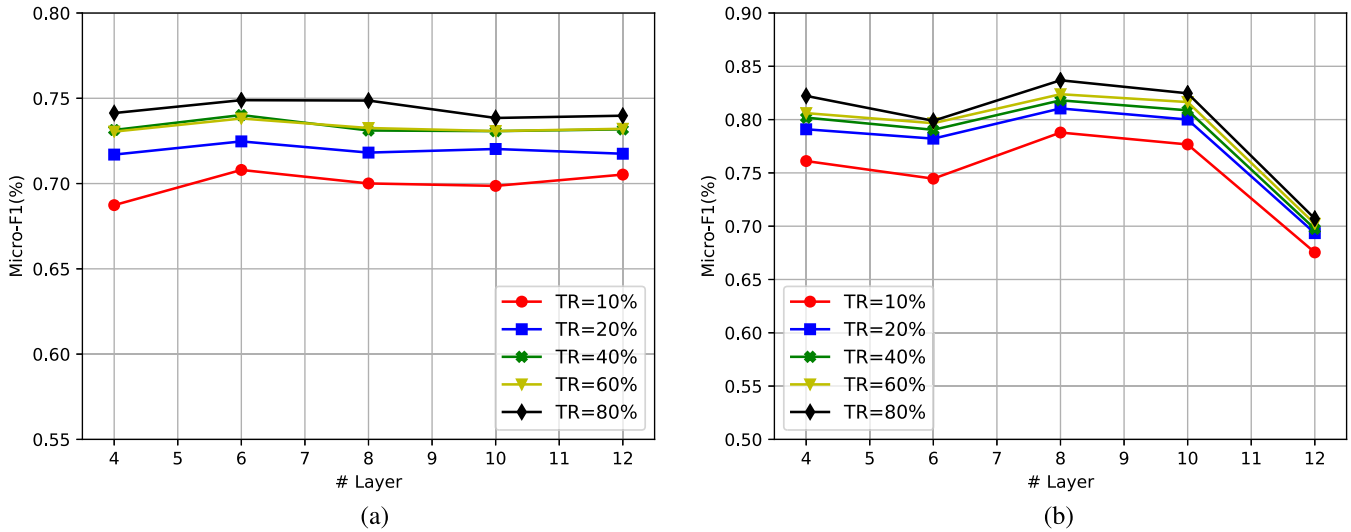

Fig. 7. Influence of the Transformer layer number on downstream classification tasks in (a) SSet and (b) TSSet datasets. Here, "TR" represents training ratios.

TABLE IX
CLASSIFICATION RESULTS (%) FOR VARIOUS GRAPH LAYERS ON TRENDS-T

CGraphNet	Micro-F1	Macro-F1
Graph Layer = 1	81.23	79.71
Graph Layer = 2	84.46	83.20
Graph Layer = 3	85.04	83.93
Graph Layer = 4	83.14	81.43

our model design. The comprehensive improvement across various evaluation metrics reaffirms the superiority of CGraphNet in representation learning for both classification and clustering tasks.

H. Parameter Sensitivity

In this section, we analyze the sensitivity of CGraphNet to the key parameters of downstream classification.

1) Embedding Size: We investigate how the representation dimension size affects the performance of CGraphNet by varying its numbers in 32, 64, 96, 128, and 160. The experimental results are reported in Figs. 4 and 5. Specifically, on Tweet, TSet, and SSet datasets, the curves tend to rise along with the dimension size and achieve the peaks at $d = 128$. However, on the TSSet dataset, the curve shows a boost at $d = 64$ and achieves a slowly growing performance for larger dimension sizes. One possible reason for these different trends in the curves is the average length of sequences in the datasets: a larger dimension size may be required to map more complex word relations into the embedding space, which can influence the model's performance.

2) Layer Number of the Transformer: Next, we evaluate how the setting of the layer number affects the performance of downstream tasks by setting the number in 4, 6, 8, 10, and 12 on four datasets. From Figs. 6 and 7, we can observe that CGraphNet obtains the best classification results at $l = 4$ on the Tweet dataset. Moreover, the performance of CGraphNet tends to rise with slight fluctuations as the layer number increases and achieves relatively stable performance around $l = 10$ on TSet, SSet, and TSSet. In general, the optimal layer number setting is dependent on the background knowledge of the datasets, with $l = 4$ being optimal for Tweet, and $l = 8$ for the other datasets.

3) Layer Number of the Graph: We evaluated the impact of different graph network layers on model performance using the Trends-T dataset, with 10% of the data as training sets. The experimental results are shown in Table IX. As can be seen, our method performs better when the graph layer is set to 2 or 3. However, as the number of layers increases, the performance begins to decline.

V. CONCLUSION

In this article, we propose CGraphNet, a GNN-based model for unlabeled social media short text representation learning. While current deep neural network models such as BERT and IDRL have shown significant improvements in NLP tasks, they face challenges with sparse word sequences in short texts, leading to suboptimal performance in next-word prediction training. To address this issue, we introduce a graph context prediction method that enhances the perception areas of the target prediction. Specifically, we aggregate information from the next word and messages passed from its neighbors during prediction training. Additionally, we incorporate an attention mechanism to reduce irrelevant words and minimize noise propagation. The Transformer layer, a deep bidirectional sequential network, is used to predict the generated graph context. Furthermore, we employ a contrastive loss to refine the learning process by leveraging the relation between the target word and its neighbors. The experimental results demonstrate that our method outperforms other baselines on real-world datasets for classification and clustering tasks.

REFERENCES

[1] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," in *Proc. 5th Int. Conf. Learn. Representations, (ICLR)*, 2017, pp. 1–16.
[2] J. Chen et al., "Inductive document representation learning for short text clustering," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, Springer, 2020, pp. 600–616.
[3] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.
[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
[5] Q. Li et al., "A survey on text classification: From shallow to deep learning," 2020, *arXiv:2008.00364*.
[6] J. Yin and J. Wang, "A Dirichlet multinomial mixture model-based approach for short text clustering," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining,* New York, NY, USA: ACM, 2014, pp. 233–242.
[7] J. Chen, Z. Gong, and W. Liu, "A Dirichlet process biterm-based mixture model for short text stream clustering," *Appl. Intell.*, vol. 50, pp. 1609–1619, Feb. 2020.
[8] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proc. 26th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, New York, NY, USA: ACM, 2003, pp. 267–273.
[9] J. Zhou et al., "Graph neural networks: A review of methods and applications," 2018, *arXiv:1812.08434*.
[10] A. Rogers, O. Kovaleva, and A. Rumshisky, "A primer in BERTology: What we know about how BERT works," 2020, *arXiv:2002.12327*.
[11] X. Cheng, X. Yan, Y. Lan, and J. Guo, "BTM: Topic modeling over short texts," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 12, pp. 2928–2941, Dec. 2014.
[12] M. Pota, M. Ventura, R. Catelli, and M. Esposito, "An effective BERT-based pipeline for Twitter sentiment analysis: A case study in Italian," *Sensors*, vol. 21, no. 1, 2020, Art. no. 133.
[13] Y. Samih, S. Maharjan, M. Attia, L. Kallmeyer, and T. Solorio, "Multilingual code-switching identification via LSTM recurrent neural networks," in *Proc. 2nd Workshop Comput. Approaches Code Switching*, 2016, pp. 50–59.
[14] R. A. Rivera-Soto et al., "Learning universal authorship representations," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 913–919.
[15] X. Yan, J. Guo, Y. Lan, and X. Cheng, "A biterm topic model for short texts," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 1445–1456.
[16] J. Yin and J. Wang, "A model-based approach for text clustering with outlier detection," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE),* Piscataway, NJ, USA: IEEE Press, 2016, pp. 625–636.
[17] J. Chen, Z. Gong, and W. Liu, "A nonparametric model for online topic discovery with word embeddings," *Inf. Sci.*, vol. 504, pp. 32–47, Dec. 2019.
[18] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," 2015, *arXiv:1506.00019*.

[19] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *J. Mach. Learn. Res.*, vol. 3, pp. 115–143, Aug. 2002.
[20] M. E. Peters, W. Ammar, C. Bhagavatula, and R. Power, "Semi-supervised sequence tagging with bidirectional language models," 2017, *arXiv:1705.00108*.
[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[22] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding with unsupervised learning," Tech. Report, OpenAI, 2018.
[23] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[24] T. Gao, X. Yao, and D. Chen, "SimCSE: Simple contrastive learning of sentence embeddings," 2021, *arXiv:2104.08821*.
[25] S. Soni, S. S. Chouhan, and S. S. Rathore, "Textconvonet: A convolutional neural network based architecture for text classification," *Appl. Intell.*, vol. 53, no. 11, pp. 14249–14268, 2023.
[26] Y. Zhou, J. Li, J. Chi, W. Tang, and Y. Zheng, "Set-CNN: A text convolutional neural network based on semantic extension for short text classification," *Knowl.-Based Syst.*, vol. 257, 2022, Art. no. 109948.
[27] Z. Wang, "Research on classification algorithm of graph neural network based on combination of graph neural network and LSTM," in *Proc. Int. Conf. Elect. Drives, Power Electron. Eng. (EDPEE),* Piscataway, NJ, USA: IEEE Press, 2024, pp. 597–602.
[28] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.
[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[30] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. Eur. Semantic Web Conf.,* Springer, 2018, pp. 593–607.
[31] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.
[32] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," 2012, *arXiv:1205.2618*.
[33] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Deep graph contrastive representation learning," 2020, *arXiv:2006.04131*.
[34] M. Jin, Y. Zheng, Y.-F. Li, C. Gong, C. Zhou, and S. Pan, "Multi-scale contrastive siamese networks for self-supervised graph representation learning," 2021, *arXiv:2105.05682*.
[35] S. Banerjee, K. Ramanathan, and A. Gupta, "Clustering short texts using Wikipedia," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2007, pp. 787–788.
[36] D. Assenmacher and H. Trautmann, "Textual one-pass stream clustering with automated distance threshold adaption," in *Proc. Asian Conf. Intell. Inf. Database Syst.,* Cham, Switzerland: Springer, 2022, pp. 3–16.
[37] D. Kuang, C. Ding, and H. Park, "Symmetric nonnegative matrix factorization for graph clustering," in *Proc. SIAM Int. Conf. Data Mining,* Philadelphia, PA, USA: SIAM, 2012, pp. 106–117.
[38] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.
[39] M. Abadi et al., "TensorFlow: A system for large-scale machine learning." in *Proc. 14th USENIX Conf. Operat. Syst. Design Implementation.*, vol. 16, Savannah, GA, USA, no. 2016, 2016, pp. 265–283.
[40] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Advances Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.
[41] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 974–983.
[42] H. Gao, J. Pei, and H. Huang, "ProGAN: Network embedding via proximity generative adversarial network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 1308–1316.
[43] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Aug. 2008.
[44] P. Huang, Y. Huang, W. Wang, and L. Wang, "Deep embedding network for clustering," in *Proc. 22nd Int. Conf. Pattern Recognit.,* Piscataway, NJ, USA: IEEE Press, 2014, pp. 1532–1537.
[45] J. Xu, B. Xu, P. Wang, S. Zheng, G. Tian, and J. Zhao, "Self-taught convolutional neural networks for short text clustering," *Neural Netw.*, vol. 88, pp. 22–31, Apr. 2017.

Junyang Chen (Member, IEEE) received the Ph.D. degree in computer and information science from the University of Macau, Macau, China, in 2020.

He is currently an Assistant Professor with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include graph neural networks, text mining, and recommender systems.

Jingcai Guo received the B.E. degree from Sichuan University, Chengdu, China, in 2013, the M.E. degree from Waseda University, Tokyo, Japan, in 2015, and the Ph.D. degree from The Hong Kong Polytechnic University, Hong Kong, China, in 2021, all in computer science.

He is currently a Research Assistant Professor with the Department of Computing, The Hong Kong Polytechnic University. His research interests include machine learning and data mining.

Xueliang Li received the Ph.D. degree in computer science from Roskilde University, Roskilde, Denmark, in 2017.

He is currently an Assistant Professor with the National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Shenzhen, China. His research interest includes deep learning.

Huan Wang (Member, IEEE) received the Ph.D. degree in computer application from Wuhan University, Wuhan, China, in 2019.

He is currently an Associate Professor with the College of Informatics, Huazhong Agricultural University, Wuhan, China. He conducted postdoctoral research under the supervision of Prof. Zhiguo Gong at the University of Macau, Macau, China. His research interests include social network analysis, network optimization and reconstruction, as well as artificial intelligence for drug discovery.

Zhenghua Xu received the M.Phil. degree from The University of Melbourne, Melbourne, Australia, in 2012, and the D.Phil. degree from the University of Oxford, Oxford, U.K., in 2018, both in computer science.

He is currently a Professor with Hebei University of Technology, Tianjin, China. From 2017 to 2018, he worked as a Research Associate with the Department of Computer Science, University of Oxford.

Dr. Xu is an awardee of the "100 Talents Plan" at Hebei province.

Zhiguo Gong (Senior Member, IEEE) received the Ph.D. degree in computer science from the Institute of Mathematics, Chinese Academy of Science, Beijing, China, 1998.

He is currently a Professor with the Faculty of Science and Technology, University of Macau, Macau, China. His research interests include machine learning, data mining, database, and information retrieval.

Liangjie Zhang (Fellow, IEEE) received the B.S. degree in electrical engineering from Xidian University, Xi'an, China, in 1990, the M.S. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 1992, and the Ph.D. degree in computer engineering from Tsinghua University, Beijing, China, in 1996.

He is currently a Distinguished Professor in computer science and software engineering with Shenzhen University, Shenzhen, Shenzhen, China. He is the Senior Vice President, the Chief Scientist, and the Director of Research, Kingdee International Software Group Company Limited, Shenzhen, China, and the Director of the Open Group, San Francisco, USA.

Dr. Zhang chaired the IEEE Computer Society's Technical Committee on Services Computing from 2003 to 2011.

Victor C. M. Leung (Life Fellow, IEEE) received the B.A.Sc. degree in electrical engineering from the University of British Columbia (UBC), Vancouver, Canada, in 1977, and the Ph.D. degree in electrical engineering from the University of British Columbia, in 1981. He is currently the Distinguished Professor in computer science and software engineering with Shenzhen University, Shenzhen, China. He is also an Emeritus Professor in electrical and computer engineering and the Director of the Laboratory for Wireless Networks and Mobile Systems, University of British Columbia (UBC), Vancouver, BC, Canada. He is named in the current Clarivate Analytics list of "Highly Cited Researcher." His research interests include networks and systems.

Dr. Leung is a Fellow of the Royal Society of Canada, Canadian Academy of Engineering, and Engineering Institute of Canada.