Continual Graph Convolutional Network for Text Classification

Tiandeng Wu[1*], Qijiong Liu[2*], Yi Cao[1], Yao Huang[1], Xiao-Ming Wu[2†], Jiandong Ding[1†]

[1] Huawei Technologies Co., Ltd., China
[2] The Hong Kong Polytechnic University, Hong Kong
{wutiandeng1, caoyi23, huangyao11, dingjiandong2}@huawei.com,
jyonn.liu@connect.polyu.hk, xiao-ming.wu@polyu.edu.hk

Abstract

To capture global non-consecutive and long-distance semantic information, graph convolutional network (GCN) has been widely used for text classification. While GCN-based methods have achieved great success in offline evaluations, they usually construct fixed document-token graphs and cannot perform inference on new documents. It is still a challenge to apply GCNs in online systems which need to infer continual text data. In this work, we present a Continual GCN model, short as ContGCN, to generalize inferences from observed documents to unobserved documents. Concretely, we propose a novel *global-token-local-document* paradigm to dynamically update the document-token graph in every batch for any online system during both training and testing phases. Moreover, we design an occurrence memory module and a self-supervised contrastive learning objective to update the proposed ContGCN in any online system in a label-free manner. Extensive offline experiments conducted on five public datasets demonstrate that our proposed ContGCN can significantly improve inference quality. A 3-month A/B test on our internal online system shows ContGCN achieves 8.86% performance gain compared with state-of-the-art methods.

Introduction

As one of the fundamental tasks in natural language processing, text classification, for decades, has been extensively studied and successfully applied in various application scenarios (Xu et al. 2019a; Abaho et al. 2021). To capture the global non-consecutive and long-distance semantic information such as token co-occurrence in a corpus, graph convolutional network (GCN) has been widely used for text classification (Yao, Mao, and Luo 2019; Lin et al. 2021).

Some GCN-based methods (Li et al. 2019) build a homogeneous graph for document classification by taking each document as a node and modeling non-semantic inter-document relations such as citation links. To further exploit document-token semantic information, another line of GCN-based methods constructs heterogeneous document-token graphs, where each node represents a document or a token, and each edge is a correlation factor between two nodes. However, they commonly follow a *local-token-global-document* (**LTGD**) paradigm to construct a *fixed* graph with all labeled training documents, unlabeled test documents, and observed tokens, and perform transductive inference. Given a new document with unobserved tokens, the trained GCN model cannot make inference on it because neither the document nor the unobserved tokens are included in the graph. Therefore, while these methods are effective for offline evaluations, they cannot be deployed in online systems to infer streaming text data.

To address this challenge, in this paper, we propose a new *global-token-local-document* (**GTLD**) paradigm to dynamically construct a document-token graph, and based on which we present a continual GCN model (**ContGCN**) for text classification. Specifically, we take the vocabulary of a pretrained language model (PLM) such as BERT (Devlin et al. 2019) as the *global* token set, so a new document can be tokenized into seen tokens from the vocabulary. We further form a *local* document set with the present documents (e.g., those in the current batch). The document-token graph then consists of tokens in the global token set and documents in the local document set. The edge weights of the graph are dynamically calculated according to an occurrence memory module with historical token correlation information, and document embeddings are generated with pretrained semantic knowledge. In this way, ContGCN is empowered to perform inductive inference on streaming text data.

Furthermore, to address data distribution shift (Luo et al. 2022) which is prevalent in online services, we design a label-free online updating mechanism for ContGCN, saving the cost and effort for periodical offline updates of the model with new text data. Specifically, we fine-tune the occurrence memory module according to the distribution shift of streaming text data and update the network parameters with a carefully designed self-supervised contrastive learning objective.

We conduct extensive experiments to evaluate the effectiveness of our ContGCN model on five real-world public datasets, i.e., 20NG, R8, R52, Ohsumed, and MR. Compared with state-of-the-art models, ContGCN achieves favorable performance in both offline and online evaluations due to the proposed GTLD paradigm and label-free online updating mechanism. Moreover, we have deployed ContGCN in our online text classification system that processes

[*] Equal contribution (co-first authors). Author ordering determined by dice rolling.
[†] Corresponding authors.

thousands of textual comments daily, which further verifies its effectiveness and efficiency. To summarize, our contributions are listed as follows:

- We propose a novel global-token-local-document paradigm and a continual GCN model to infer unobserved streaming text data, which, to our knowledge, is the first attempt to use GCN for online text classification.
- We design a label-free updating mechanism based on an occurrence memory module and a self-supervised contrastive learning objective, which enables to update our proposed ContGCN online with unlabeled documents.
- Extensive offline experiments conducted on five real-world datasets and online A/B tests demonstrate the effectiveness of our proposed ContGCN model.

Preliminary

Graph Convolutional Network

A GCN (Welling and Kipf 2016) is a graph encoder that aggregates the knowledge from node neighborhoods. It is composed of a stack of graph convolutional layers. Formally, we use $G = (V, E)$ to denote a graph, where $V(\mathtt{n} = |V|)$ and E are sets of nodes and edges, respectively. Note that each node $v \in V$ is self-connected, i.e., $(v, v) \in E$. We use $\mathbf{X} \in \mathbb{R}^{\mathtt{n} \times \mathtt{d}}$ to represent node representations, where $\mathtt{d}$ is the embedding dimension. To capture the information from neighborhoods, a symmetric adjacency matrix $\mathbf{A} \in \mathbb{R}^{\mathtt{n} \times \mathtt{n}}$ is introduced, where $\mathbf{A}_{ij}$ is the correlation score of node v_i and v_j and $\mathbf{A}_{ii} = 1$. Before passing into the convolutional layer, a normalization operation is performed on the adjacency matrix, formulated as:

$$\widetilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{\frac{1}{2}}, \tag{1}$$

where $\mathbf{D}$ is a degree matrix and $D_{ii} = \sum_j A_{ij}$. For k-th of total $\mathtt{h}$ convolutional layers, the latent node embedding can be calculated as:

$$\mathbf{H}^{(k)} = \rho\left(\widetilde{\mathbf{A}}\mathbf{H}^{(k-1)}\mathbf{W}_k\right), \tag{2}$$

where $k \in \{1, 2, \cdots, \mathtt{h}\}$, ρ is the activation function, $\mathbf{W}_k \in \mathbb{R}^{\mathtt{d} \times \mathtt{d}}$ is a trainable matrix of k-th layer, and specifically, $\mathbf{H}^{(0)} = \mathbf{X}$.

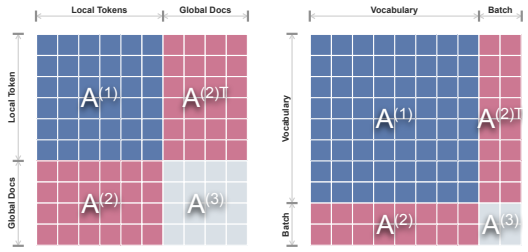
LTGD-based GCN for Text Classification

The text classification task aims to categorize documents into different classes. Formally, we use $D(\mathtt{m} = |D|)$ to denote a set of documents, which can be split into a training set $D_{train} \subset D$ and a testing set $D_{test} \subset D$. Each document can be represented as $\mathbf{s} = [t_1^{(\mathbf{s})}, t_2^{(\mathbf{s})}, \cdots, t_{|\mathbf{s}|}^{(\mathbf{s})}] \in D$, where $t_i^{(\mathbf{s})} \in T$ is a token in the global vocabulary $T(\mathtt{u} = |T|)$.

In the text classification domain, existing GCN-based methods (Yao, Mao, and Luo 2019; Qiao et al. 2018; Lin et al. 2021) mainly construct document-token heterogeneous graphs under a local-token-global-document paradigm. Precisely, they first construct a local vocabulary $T_{local} \subset T$ with size $\mathtt{u}'$ which contains all seen tokens in the document set D. Then, they construct the graph with a fixed structure, whose graph nodes include local tokens $t \in T_{local}$ and global documents $\mathbf{s} \in D$, i.e., $\mathtt{n} = \mathtt{u}' + \mathtt{m}$. In this way, they form the adjacency matrix, as shown in Figure 1a, including an inner-token symmetric matrix $\mathbf{A}^{(1)} \in \mathbb{R}^{\mathtt{u}' \times \mathtt{u}'}$, a document-token matrix $\mathbf{A}^{(2)} \in \mathbb{R}^{\mathtt{m} \times \mathtt{u}'}$, and an inner-document identity matrix $\mathbf{A}^{(3)} \in \mathbb{R}^{\mathtt{m} \times \mathtt{m}}$. It will be first initialized and then fixed in the model training. Finally, the GCN-enhanced document embedding can be formulated as:

$$\bar{\mathbf{X}} = \mathtt{GCN}(\mathbf{A}, \mathbf{X}), \tag{3}$$

After we extract the document embedding of the training set, it will next be passed into an MLP (multilayer perceptron) classifier for label prediction.



(a) LTGD (b) GTLD

Figure 1: Comparison of the adjacency matrices. Left: local-token-global-document (LTGD) paradigm (e.g., BertGCN). Right: our global-token-local-document (GTLD) paradigm.

Method

While GCN has been widely adopted in text classification due to its ability to capture high-order neighborhood information and global non-consecutive semantic knowledge, the local-token-global-document paradigm hinders the existing GCN-based methods from reasoning about unobserved documents. Thus, we propose a novel global-token-local-document paradigm where token nodes are provided by the entire vocabulary and document nodes are alterable. By dynamically updating the document-text graph, our GTLD-based continual GCN model, or ContGCN, reveals its flexibility to handle unobserved data.

Figure 2 illustrates the overview architecture of our ContGCN model, including adjacency matrix generator, node encoder, and GCN encoder. Expressly, ContGCN takes a batch of documents as input. In parallel, the adjacency matrix generator updates the adjacency matrix based on the occurrence memory module and current batch, while the node encoder produces content-based node embedding. Then, the GCN encoder is applied to capture the global-aware node representations. Finally, we take two training objectives to train the ContGCN model.

The Global-token-local-document Paradigm

Unlike the local-token-global-document paradigm which takes local tokens (T_{local}) and global documents (D) as fixed graph nodes, our proposed global-token-local-document

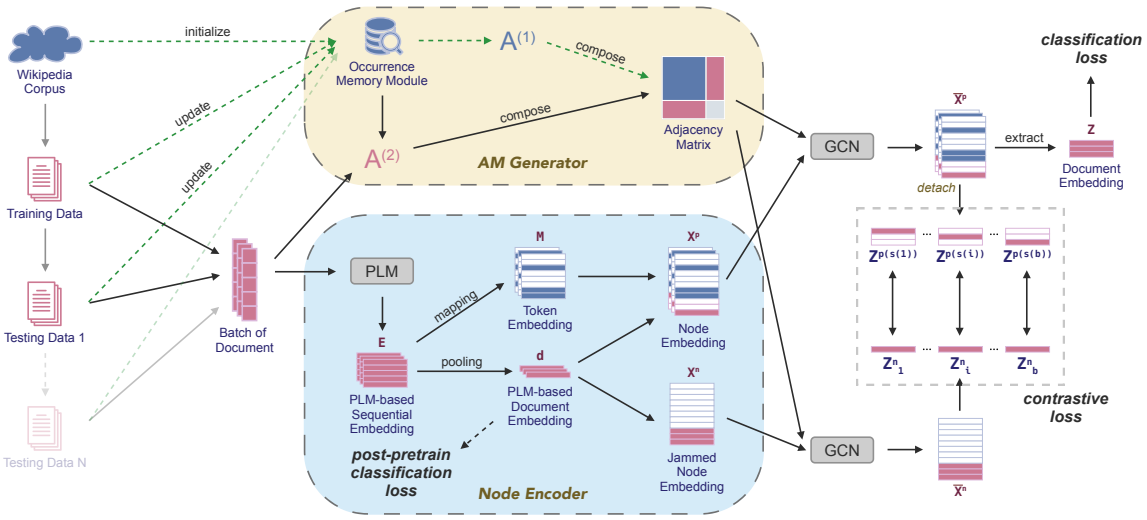


Figure 2: Framework of our ContGCN model. Green dotted lines represent operations before each phase of model training or testing. Two key components, i.e., AM Generator and Node Encoder, dynamically conduct adjacency matrix and node embedding, which are then fed into the GCN encoder. Finally, our ContGCN model is trained with a classification loss and an anti-interference contrastive loss.

paradigm constructs a document-token graph with global tokens (T) and dynamic local documents (i.e., a batch of documents $B = \{\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \cdots \mathbf{s}^{(\mathtt{b})}\} \subset D$ where $\mathtt{b}$ is the batch size).

Our ContGCN model is designed under the global-token-local-document paradigm. We set our global tokens as the vocabulary used for PLM tokenizers. Thus, unseen words can be tokenized into sub-words, i.e., several seen tokens in the vocabulary. When a new batch of data is fed into the model, the adjacency matrix and node embedding will be dynamically updated by the adjacency matrix generator and node encoder, respectively.

Adjacency Matrix Generator

As shown in Figure 1b, the GTLD-based adjacency matrix has a similar composition to the LTGD-based one: an inner-token matrix $\mathbf{A}^{(1)} \in \mathbb{R}^{\mathtt{u}\times\mathtt{u}}$ which is a *phased-fixed* matrix learned from the global token occurrence knowledge of the corpus, a document-token matrix $\mathbf{A}_2 \in \mathbb{R}^{\mathtt{b}\times\mathtt{u}}$ that is actively calculated based on the current batch, and an inner-document identity matrix $\mathbf{A}^{(3)} \in \mathbb{R}^{\mathtt{b}\times\mathtt{b}}$ to prevent each document from being influenced by other samples during model learning or reasoning. By *phased-fixed*, $\mathbf{A}^{(1)}$ will be refined when the model enters a new training or testing phase, with the emergence of new corpora in the company of new token co-occurrence knowledge, as shown in the green dotted lines of Figure 2.

Occurrence Memory Module (OMM) is an incremental historical statistics recorder, including a sentence (*not* document) counter $\mathtt{s} \in \mathbb{Z}^1$ recording the number of sentences, a token occurrence counter $\mathbf{c} \in \mathbb{Z}^{\mathtt{u}}$ recording the number of sentences where a token appears, and a token co-occurrence counter $\mathbf{C} \in \mathbb{Z}^{\mathtt{u}\times\mathtt{u}}$ recording the number of times for two tokens appearing in one sentence. OMM records the global non-consecutive semantic information and brings about the following advantages: 1) it fulfill the dynamic calculation of the adjacency matrix for any batch of documents; 2) it holds a large amount of previous general and domain-specific knowledge without re-calculating in updating. We implement the OMM updating algorithm in a simple yet efficient manner, as illustrated in Algorithm 1. As demonstrated in Figure 2, it is initialized by the Wikipedia corpus, and updated by training data or testing data before model training or testing. Thus, $\mathbf{A}^{(1)}$ will be phase-wisely updated through PPMI (positive pointwise mutual information), defined as:

$$\mathbf{A}^{(1)}_{i,j} = \begin{cases} 1, & \text{if } i = j \\ \max\left(\log\left(\mathtt{s}\frac{C_{i,j}}{c_i c_j}\right), 0\right). & \text{else} \end{cases} \tag{4}$$

For each document $\mathbf{s} \in B$, we apply TF-IDF (term frequency-inverse document frequency) to obtain document-token correlation, which is calculated by:

$$\mathbf{A}^{(2)}_{\mathbf{s},t} = \frac{\mathtt{g}(\mathbf{s},t)}{|\mathbf{s}|}\log\frac{\mathtt{s}}{c_t + 1}, \tag{5}$$

where $\mathtt{g}(\mathbf{s},t)$ represents the number of times the token t appears in the document $\mathbf{s}$. As for the inner-document matrix $\mathbf{A}^{(3)}$, it can be formulated as:

$$\mathbf{A}^{(3)}_{i,j} = \begin{cases} 1, & \text{if } i = j \\ 0. & \text{else} \end{cases} \tag{6}$$

Finally, the adjacency matrix $\mathbf{A}$ can be composed by:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}^{(1)} & \mathbf{A}^{(2)T} \\ \mathbf{A}^{(2)} & \mathbf{A}^{(3)} \end{pmatrix}. \tag{7}$$

Algorithm 1: Continual OMM updating algorithm

```
def update(corpus_or_dataset, omm):
  omm.load()
  omm.s += len(corpus_or_dataset)
  for doc in corpus_or_dataset:
    for sent in doc:
      for ti in set(sent):
        omm.c[ti] += 1
      for ti in sent:
        for tj in sent:
          omm.C[ti][tj] += ti != tj
  omm.store()
```

Node Encoder

Since the pretrained language model (PLM) (Devlin et al. 2019; Liu et al. 2019; Yang et al. 2019a) validates its effectiveness for text modeling in various scenarios due to its general knowledge learned from a large corpus, we leverage it as a document encoder to capture semantic information for each document $\mathbf{s} \in B$:

$$\mathbf{E}^{(\mathbf{s})} = \texttt{PLM}(\mathbf{s}) \in \mathbb{R}^{\texttt{l} \times \texttt{d}}, \tag{8}$$

where $\texttt{l} \geq |\mathbf{s}|$ is the maximum document length for PLM, as short documents will append special <PAD> tokens, to align the length with documents from the same batch, according to BERT (Devlin et al. 2019). Note that we average the hidden states of the first and last transformer layers following (Li et al. 2020; Su et al. 2021). Next, we perform an average pooling operation on $\mathbf{E}^{(\mathbf{s})}$ to obtain the unified document embedding $\mathbf{d}^{(\mathbf{s})} \in \mathbb{R}^{\texttt{d}}$, and meanwhile, construct sample-specific token embedding matrix $\mathbf{M}^{(\mathbf{s})} \in \mathbb{R}^{\texttt{u} \times \texttt{d}}$, computed by:

$$\mathbf{M}_i^{(\mathbf{s})} = \begin{cases} \mathbf{E}_k^{(\mathbf{s})}, & \text{if exists } k, \text{where } i = t_k^{(\mathbf{s})} \\ \mathbf{0}. & \text{else} \end{cases} \tag{9}$$

More precisely, we map $\mathbf{E}^{(\mathbf{s})}$ into the global-token embedding space, while unseen token embedding is set to $\mathbf{0}$ vector. After we parallelly perform the above operations on the batch data, we concatenate the token embedding and document embedding to obtain node embedding. Following Bert-GCN (Lin et al. 2021), we form a batch-wise node embedding $\mathbf{X}^n \in \mathbb{R}^{(\texttt{u}+\texttt{b}) \times \texttt{d}}$, defined as:

$$\mathbf{X}_i^n = \begin{cases} \mathbf{0}, & \text{if } i \leq \texttt{u} \\ \mathbf{d}^{(\mathbf{s}^{(i-\texttt{u})})}. & \text{else} \end{cases} \tag{10}$$

However, such node embedding will be affected by document interference due to the node message passing inside the GCN network. To avoid the mutual interference within batches, we form sample-specific node embedding $\mathbf{X}^{p(\mathbf{s})} \in \mathbb{R}^{(\texttt{u}+\texttt{b}) \times \texttt{d}}$ by:

$$\mathbf{X}_i^{p(\mathbf{s})} = \begin{cases} \mathbf{M}_k^{(\mathbf{s})}, & \text{if } i \leq \texttt{u} \\ \mathbf{d}^{(\mathbf{s})}, & \text{if } \mathbf{s}^{(i-\texttt{u})} = \mathbf{s} \\ \mathbf{0}, & \text{else} \end{cases} \tag{11}$$

where $\mathbf{s}^{(i-\texttt{u})}$ is the $(i-\texttt{u})$-th sample in current batch B. In other words, we sample-wisely map $\mathbf{M}_k^{(\mathbf{s})}$ and $\mathbf{d}^{(\mathbf{s})}$ into the global-token-local-document latent space. We name $\mathbf{X}^n$ and $\mathbf{X}^{p(\mathbf{s})}$ as *jammed* and *unjammed* node embedding.

GCN Encoder

When adjacency matrix ($\mathbf{A}$) and node embeddings (unjammed $\mathbf{X}^{p(\mathbf{s})}$ and jammed $\mathbf{X}^n$) are produced, the GCN encoder is applied to capture global-aware node representations. We use $\bar{\mathbf{X}}^{p(\mathbf{s})}$ and $\bar{\mathbf{X}}^n$ to represent the GCN-enhanced unjammed and jammed node representations based by Equation 3. Then, we extract the GCN-enhanced document embedding $\mathbf{Z}_i = \bar{\mathbf{X}}_{i+\texttt{u}}^{p(\mathbf{s}^{(i)})} \in \mathbb{R}^{\texttt{b} \times \texttt{d}}$, sample-wise unjammed document embedding $\mathbf{Z}_i^{p(\mathbf{s})} = \bar{\mathbf{X}}_{i+\texttt{u}}^{p(\mathbf{s})} \in \mathbb{R}^{\texttt{b} \times \texttt{d}}$, and jammed document embedding $\mathbf{Z}_i^n = \bar{\mathbf{X}}_{i+\texttt{u}}^n \in \mathbb{R}^{\texttt{b} \times \texttt{d}}$.

Training Objectives

For training the model, we adopt a classification task and a contrastive task as our training objectives.

Document Classification Task is the primary task tailored for the text classification scenario. Following Bert-GCN (Lin et al. 2021), a MLP classifier is used to infer the probability distribution over all classes with a softmax activation function. Thus, the loss function can be defined as:

$$\texttt{CLS} : \mathbb{R}^{\texttt{d}} \to \mathbb{R}^{\texttt{c}}, \tag{12}$$

$$\mathcal{L}_{cls} = -\frac{1}{\texttt{b}} \sum_{i=1}^{\texttt{b}} \log \left(\texttt{CLS}\left(\mathbf{Z}_i\right)_{l_i} \right), \tag{13}$$

where $\texttt{c}$ is the total number of document classes, and $1 < l_i \leq \texttt{c}$ is the class label of i-th document.

Anti-interference Contrastive Task is an auxiliary task designed to enable the GCN encoder to avoid the interference between documents. Specifically, for each jammed document embedding $\mathbf{Z}_i^n$, it is encouraged to approximate the unjammed document embedding $\mathbf{Z}_i^{p(\mathbf{s}^{(i)})}$. Meanwhile, we use other documents in the batch as negative samples. Note that $\mathbf{Z}^{p(\mathbf{s})}$ is detached. Thus, the loss function is:

$$\mathcal{L}_{aic} = -\frac{1}{\texttt{b}} \sum_{i=1}^{\texttt{b}} \log \left(\mathbf{y}_i^{(\mathbf{s}^{(i)})} \right), \text{where} \tag{14}$$

$$\mathbf{y}^{(\mathbf{s}^{(i)})} = \texttt{softmax} \left(\mathbf{Z}^{p(\mathbf{s}^{(i)})} \mathbf{Z}_i^{nT} \right) \in \mathbb{R}^{\texttt{b}}. \tag{15}$$

The overall loss function is the combination of the classification and contrastive tasks:

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda \mathcal{L}_{aic}, \tag{16}$$

where λ is a balancing parameter.

Label-free Updating Mechanism (LUM). The occurrence memory module and the anti-interference contrastive task enables to continually update our ContGCN model with incoming unlabeled text data during inference. Hence, we name it label-free updating mechanism (LUM).

Model Training and Update

Our ContGCN will be continually tuned in different stages. **Stage I: Before training.** Following BertGCN (Lin et al. 2021), we first post-pretrain the PLM by the classification task on the PLM-enhanced document embeddings $\mathbf{d}^{(\mathbf{s})}$ to speed up model convergence, as indicated in Figure 2 (the black dotted line). **Stage II: During training.** We train the ContGCN model with the multi-task training objective (Eq. 16). **Stage III: During inference.** When new test data arrives, we first update the occurrence memory module with Algorithm 1, and then employ the auxiliary anti-interference contrastive task (Eq. 14) to finetune the ContGCN model.

Experiments

Experimental Setups

Datasets. Following (Lin et al. 2021), we conduct experiments on five real-world text classification datasets, i.e., 20-Newsgroups (20NG), Ohsumed, R52 Reuters, R8 Reuters, and Movie Review(MR) datasets. The statistics of datasets are summarized in Table 1. For all datasets, we randomly selected 10% of the training set as the validation set.

Baselines and Variants of Our Method. To verify the effectiveness of our proposed ContGCN model, we compare with three types of state-of-the-art models: 1) traditional GCN-based models without leveraging pretrained general semantic knowledge, including TextGCN (Yao, Mao, and Luo 2019) and TensorGCN (Liu et al. 2020); 2) transformer-based PLMs, including BERT (Devlin et al. 2019), RoBERTa (Liu et al. 2019) and XLNet (Yang et al. 2019b); 3) models combining GCN with PLM, including TG-Transformer (Zhang and Zhang 2020), BertGCN (Lin et al. 2021) and RoBERTaGCN (Lin et al. 2021). As our ContGCN can be plugged by different PLMs, we adopt BERT, XLNet, and RoBERTa as our variants, namely $\text{ContGCN}_{\texttt{BERT}}$, $\text{ContGCN}_{\texttt{XLNet}}$, and $\text{ContGCN}_{\texttt{RoBERTa}}$.

Implementation Details We adopt the Adam optimizer (Kingma and Ba 2015) to train the network of our ContGCN model and baseline models, which are consistent in the following parameters: the number of graph convolutional layers (if using) is set to 3, the embedding dimension is set to 768, and the batch size is set to 64. In the post-pretraining phase, we set the learning rate for PLM to 1e-4. During training, we set different learning rates for PLM and other randomly initialized parameters (including the GCN network) following (Lin et al. 2021). Precisely, we set 1e-5 to finetune RoBERTa and BERT, 5e-6 to finetune XLNet, and 5e-4 to other parameters. We average ten experimental results as the final evaluation results. We release the source code for reproducibility [1].

Comparison with Offline State-of-the-art Methods

We conduct an offline comparison between our ContGCN method with state-of-the-art baselines on five datasets. Table 2 summarizes the overall performance of all methods, from which we can make the following observations:

[1] https://github.com/Jyonn/ContGCN

Dataset	20NG	R8	R52	Ohsumed	MR
# Docs	18,846	7,674	9,100	7,400	10,662
# Training	11,314	5,485	6,532	3,357	7,108
# Test	7,532	2,189	2,568	4,043	3,554
# Classes	20	8	52	23	2
Avg. Length	221	66	70	136	20

Table 1: Dataset statistics.

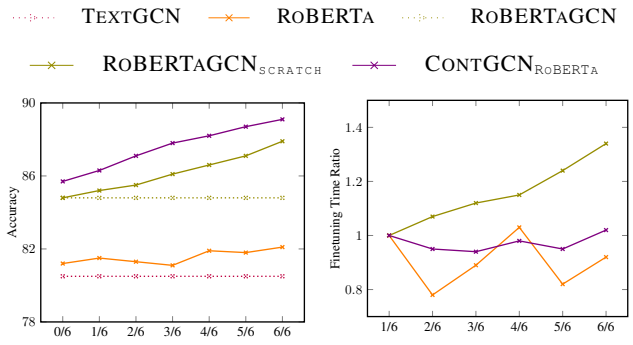
Models	20NG	R8	R52	Ohsumed	MR
TextGCN	86.3	97.1	93.6	68.4	76.7
TensorGCN	87.7	98.0	95.0	70.1	77.9
BERT	85.3	97.8	96.4	70.5	85.7
RoBERTa	83.8	97.8	96.2	70.7	89.4
XLNet	85.1	98.0	96.6	70.7	87.2
TG-Transformer	-	98.1	95.2	70.4	-
BertGCN	89.3	98.1	96.6	72.8	86.0
RoBERTaGCN	89.5	98.2	96.1	72.8	89.7
$\text{ContGCN}_{\texttt{BERT}}$	89.4	98.3	96.9	73.1	86.4
$\text{ContGCN}_{\texttt{XLNet}}$	89.7	98.5	**97.0**	73.1	88.7
$\text{ContGCN}_{\texttt{RoBERTa}}$	**90.1**	**98.6**	96.6	**73.4**	**91.3**

Table 2: Comparison of our ContGCN model with state-of-the-art baselines which are underlined. The best results are in boldface.

First, PLM-only methods mostly outperform GCN-only methods due to their pre-learned semantic knowledge. As the document lengths of the 20NG dataset are incredibly long, GCN-only methods can construct more document-token edges for better semantic comprehension. In contrast, for the MR dataset, GCN-only methods reveal their weakness in handling documents with short lengths . **Second**, PLM-empowered GCN methods enjoy the strength of both PLM and GCN models, which outperform PLM-only and GCN-only methods. **Third**, our ContGCN achieves state-of-the-art performance in five datasets; chances are that: 1) by employing the proposed global-token-local-document paradigm, our ContGCN model reaps the benefits from general semantic knowledge initialized from a large Wikipedia corpus; 2) proposed contrastive learning objective attenuates inter-document interference. Specifically, $\text{ContGCN}_{\texttt{RoBERTa}}$ achieves the state-of-the-art performance on four datasets.

Comparison on Online Learning Scenario

Figure 3 illustrates the performance in online learning scenario where training/updating data is incremental while testing data is fixed. Based on the results, we can draw the following observations: **First**, since existing GCN-based methods (i.e., TextGCN and RoBERTaGCN) construct fixed graphs by original corpus, they are incapable of being updated with or reasoning about unobserved data. Hence, their performance remains unchanged over time, as the dotted lines demonstrate in Figure 3a. **Second**, as the updating data increments, the performance of all updateable models shows an overall upward trend. **Third**, our ContGCN method outperforms all methods at the different proportions of updat-



(a) Reasoning Quality (b) Finetuning Efficiency

Figure 3: Comparison between our ContGCN model and baselines in an online learning scenario. We divide the 20NG dataset into training, testing set, and updating set according to 2:2:6. For each subsequent training or finetuning, we take 10% of the training set as the validation set. For each model, we first use the training set to learn an initial version. Then, we divide the updating set into 6 proportions equally, and feed each proportion one-by-one to finetune the model. The *finetuning time ratio* is calculated by current proportion finetuning time over the first proportion finetuning time.

Models	20NG	R8	Ohsumed
ContGCN$_{\text{RoBERTa}}$	90.1	98.6	73.4
- Wikipedia Init	89.9	98.2	73.1
- OMM Updating	89.6	98.3	73.0
- Contrastive Loss	89.7	98.5	73.2
ContGCN$_{\text{XLNet}}$	89.7	98.5	73.1
- Wikipedia Init	89.8	98.3	72.8
- OMM Updating	89.4	98.2	72.7
- Contrastive Loss	89.5	98.2	73.0

Table 3: Influence of the Wikipedia initialization on OMM, OMM updating, and anti-interference contrastive objective.

ing stage. **Fourth**, due to the flaws of LTGD-based GCN methods, we design a RoBERTaGCN$_{\text{scratch}}$ model to retrain from scratch with all previous data in each updating stage. As demonstrated in Figure 3b, the finetuning time ratio of ContGCN and RoBERTa fluctuates around 1, indicating that they are updated at similar times for each proportion. However, due to the retraining strategy, the updating time of RoBERTaGCN$_{\text{scratch}}$ tends to grow linearly as data grows, which hinders the possibility of its online learning.

Ablation Study

First, we study the effect of different components of ContGCN, including the Wikipedia initialization, OMM updating, and anti-interference contrastive task on the offline performance. Based on the results from Table 3, we can conclude that: **First**, for the 20NG dataset, we notice that the Wikipedia initialization serves a little function, probably due to the long length of documents that carries enough non-consecutive knowledge during OMM updating. Apart from this, after removing each components, most performance of both ContGCN$_{\text{RoBERTa}}$ and ContGCN$_{\text{XLNet}}$ on three datasets declines, which verifies the effectiveness of these components. **Second**, models without OMM updating get the worst performance, which indicates that non-consecutive semantic information is vital to model training and testing.

Next, we study the updating strategy of ContGCN on the online learning scenario. As depicted in Table 4, we can make the following observations. **First**, compared ContGCN with ContGCN$^{\alpha}$ and ContGCN$^{\beta}$, both OMM updating and contrastive loss reveal their effectiveness to boost the performance in online scenario. **Second**, compared with retraining from scratch, i.e., ContGCN*, ContGCN is updated with less time and computational resources, while achieves competitive and utterly acceptable performance.

Variants	1/6	2/6	3/6	4/6	5/6	6/6
ContGCN*	86.4	87.3	88.1	88.6	89.0	89.6
ContGCN	86.3	87.1	87.8	88.2	88.7	89.1
ContGCN$^{\alpha}$	86.1	86.9	87.5	87.9	88.3	88.7
ContGCN$^{\beta}$	86.0	86.2	86.4	86.6	86.9	87.1

Table 4: Variants of ContGCN$_{\text{RoBERTa}}$ in online learning scenario on the 20NG dataset. ContGCN* is retrained from scratch with all previous data in each updating stage. ContGCN$^{\alpha}$ is updated without the contrastive loss. ContGCN$^{\beta}$ is updated without LUM.

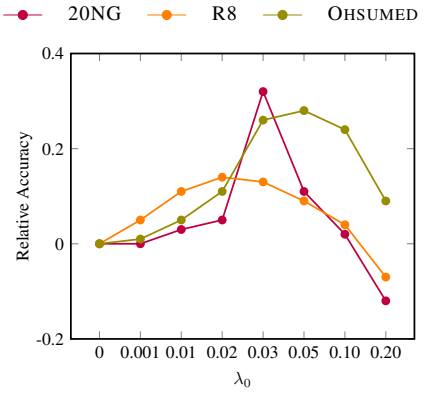


Figure 4: Influence of the λ, i.e., the proportion of auxiliary anti-interference contrastive task used for loss calculation. The relative accuracy represents the difference between the accuracy at $\lambda = \lambda_0$ and that at $\lambda = 0$.

Impact of Anti-interference Contrastive Learning

Here, we study the balancing parameter λ which weights the auxiliary anti-interference contrastive loss. We conduct the experiments on the 20NG, R8 and Ohsumed datasets with ContGCN$_{\text{RoBERTa}}$ model, considering the λ from $\{0.001, 0.01, 0.02, 0.03, 0.05, 0.10, 0.20\}$. As demonstrated in Figure 4, we can make the following observations: **First**, datasets vary in their dependence on auxiliary tasks. Typi-

Models	0th	1st	2nd	3rd
RoBERTaGCN	91.7	N/A	N/A	N/A
RoBERTa	87.6	86.8	85.2	83.5
ContGCN$^{\beta}$	**92.8**	90.3	89.9	88.2
ContGCN	**92.8**	**92.5**	**92.0**	**90.9**

Table 5: Comparison of our ContGCN model with RoBERTa baseline used in the industrial scenario. The percentage represents the ratio of the current performance to the initial (0th) performance. All models are first trained offline in the 0th month based on initial labeled dataset. After deployed, our ContGCN perform online learning with LUM. ContGCN$^{\beta}$ is a static network with parameters fixed after training on 0-th month data.

cally, when λ is set to 0.03, 0.02, and 0.05, it achieves the best performance on the 20NG, R8, and Ohsumed datasets, respectively. **Second**, for each dataset, the evaluation performance increases first and then decreases as λ increases. For some cases (e.g., $\lambda = 0.20$ on the 20NG dataset), the performance would be worse than that when not applying the auxiliary task. Hence, it is critical to make a trade-off for selecting the λ value.

Online A/B Testing

We have deployed the ContGCN on an industrial system for huawei public opinion analyzing system, serving millions of documents monthly. The task is modeled via an optimized variant of the RoBERTa (Liu et al. 2019) called RoBERTa$_{\texttt{wwm-ext}}$ (Xu 2021), tailored for Chinese text classification, which will still be abbreviated as RoBERTa below and in Table 5. To leverage global non-consecutive occurrence information, we introduce GCN into online text classification systems. Due to the inherent shortage, RoBERTaGCN fails (i.e., gets N/A results in the table) to reason about unobserved documents despite obtaining better performance than RoBERTa initially. Thus, we apply the GTLD-based ContGCN model to learn document-token knowledge. After these two models are trained offline in the 0th month, we deploy them for online comparison. As illustrated in Table 5, our ContGCN (specifically, ContGCN$_{\texttt{RoBERTa}}$) has achieved accuracy gains of 5.94%, 6.57%, 7.98%, and 8.86% over the baseline RoBERTa model in the 0th, 1st, 2nd, and 3rd month, respectively. Besides, due to the *distribution shift* of public opinions, the accuracy will drop slightly over time. However, our ContGCN model with the still maintains 98.0% performance after three months, much higher than 95.3% of the baseline model. Furthermore, by removing the *label-free update mechanism*, the performance will drop largely, which authenticates the online learning capability of our ContGCN model.

Related Work

Graph neural networks (Scarselli et al. 2008) (GNNs) have achieved growing applications for arbitrarily graph-structured data via extracting dependencies on graph nodes and transferring messages through graph edges (Hamilton, Ying, and Leskovec 2017; Xu et al. 2019b). Formally, GNN-based methods can be divided (Wu et al. 2020) into graph auto-encoder (Kipf and Welling 2016; Cao, Lu, and Xu 2016) (GAEs), graph convolutional networks (Yao, Mao, and Luo 2019; Lin et al. 2021), graph spatial-temporal networks (Yu, Yin, and Zhu 2018), graph attention networks (Zhang et al. 2018), and graph generative networks (De Cao and Kipf 2018). As one of the most successful variant, graph convolutional networks (GCNs) has been widely explored in various scenarios in natural language processing domain, such as question answering (Song et al. 2018) and relation extraction (Zhang et al. 2018).

Both academia and industry are devoted to the exploration of text classification. Early work (Jacovi, Sar Shalom, and Goldberg 2018; Sari, Rini, and Malik 2019) employs traditional language models such as convolutional neural networks (Krizhevsky, Sutskever, and Hinton 2012) or long-short term memory networks (Hochreiter and Schmidhuber 1997) to capture text sequences. Recently, Transformer-based pretrained language models (PLMs) achieve remarkable success over existing methods in diverse domains such as recommender system (Liu et al. 2022) and text classification (Devlin et al. 2019; Liu et al. 2019). Despite the rich semantic information of individual text sequence being effectively captured by these methods, the global non-consecutive and long-distance semantic information, such as token co-occurrence in a corpus is not leverage. In recent years, the GCNs have attracted much attention in the text classification domain (Yao, Mao, and Luo 2019; Lin et al. 2021), due to their ability to model non-structured data and capture global dependence, such as high-order neighborhood information. Unlike existing GCN-based methods, which mainly construct fixed graphs and are incapable of reasoning about unobserved documents, we propose a global-token-local-document paradigm to empower our ContGCN model to test new data in online systems effortlessly. We find some GNN-based methods (Li et al. 2019; Xie et al. 2021; Wang, Han, and Poon 2022) construct homogeneous graph for documents, however, we focus on constructing graph nodes with documents and token (Yao, Mao, and Luo 2019) to capture their semantic relations.

Conclusion

To deploy GCN-based text classification methods to online industrial systems, we propose a ContGCN model with a novel global-token-local-document paradigm and a label-free updating mechanism, which endow the model an capability of inferring unobserved documents and enable to continually update the model during inference time. To the best of our knowledge, this is the first attempt to use GCN for online text classification. Extensive online and offline evaluations validate the effectiveness of our proposed ContGCN model, which achieves favorable performance compared with various state-of-the-art methods.

Acknowledgement

The authors would like to thank the anonymous reviewers for their helpful comments. Q. Liu and X. Wu were supported by GRF No.15222220 funded by the UGC and ITS/359/21FP funded by the ITC of Hong Kong.

References

Abaho, M.; Bollegala, D.; Williamson, P.; and Dodd, S. 2021. Detect and Classify – Joint Span Detection and Classification for Health Outcomes. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 8709–8721. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Cao, S.; Lu, W.; and Xu, Q. 2016. Deep neural networks for learning graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

De Cao, N.; and Kipf, T. 2018. MolGAN: An implicit generative model for small molecular graphs. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Jacovi, A.; Sar Shalom, O.; and Goldberg, Y. 2018. Understanding Convolutional Neural Networks for Text Classification. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 56–65. Brussels, Belgium: Association for Computational Linguistics.

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Kipf, T. N.; and Welling, M. 2016. Variational graph auto-encoders. *Conference and Workshop on Neural Information Processing Systems*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Li, B.; Zhou, H.; He, J.; Wang, M.; Yang, Y.; and Li, L. 2020. On the Sentence Embeddings from Pre-trained Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 9119–9130. Online: Association for Computational Linguistics.

Li, Q.; Wu, X.-M.; Liu, H.; Zhang, X.; and Guan, Z. 2019. Label efficient semi-supervised learning via graph filtering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9582–9591.

Lin, Y.; Meng, Y.; Sun, X.; Han, Q.; Kuang, K.; Li, J.; and Wu, F. 2021. BertGCN: Transductive Text Classification by Combining GNN and BERT. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 1456–1462.

Liu, Q.; Zhu, J.; Dai, Q.; and Wu, X. 2022. Boosting Deep CTR Prediction with a Plug-and-Play Pre-trainer for News Recommendation. In *Proceedings of the 29th International Conference on Computational Linguistics*, 2823–2833. Gyeongju, Republic of Korea: International Committee on Computational Linguistics.

Liu, X.; You, X.; Zhang, X.; Wu, J.; and Lv, P. 2020. Tensor graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 8409–8416.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*, abs/1907.11692.

Luo, R.; Sinha, R.; Hindy, A.; Zhao, S.; Savarese, S.; Schmerling, E.; and Pavone, M. 2022. Online distribution shift detection via recency prediction. *arXiv preprint arXiv:2211.09916*.

Qiao, C.; Huang, B.; Niu, G.; Li, D.; Dong, D.; He, W.; Yu, D.; and Wu, H. 2018. A New Method of Region Embedding for Text Classification. In *International Conference on Learning Representations*.

Sari, W.; Rini, D.; and Malik, R. 2019. Text Classification Using Long Short-Term Memory With GloVe Features. *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika*, 5(2): 85–100.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2008. The graph neural network model. *IEEE transactions on neural networks*, 20(1): 61–80.

Song, L.; Wang, Z.; Yu, M.; Zhang, Y.; Florian, R.; and Gildea, D. 2018. Exploring Graph-structured Passage Representation for Multi-hop Reading Comprehension with Graph Neural Networks. *ArXiv*, abs/1809.02040.

Su, J.; Cao, J.; Liu, W.; and Ou, Y. 2021. Whitening sentence representations for better semantics and faster retrieval. *arXiv preprint arXiv:2103.15316*.

Wang, K.; Han, S. C.; and Poon, J. 2022. InducT-GCN: Inductive Graph Convolutional Networks for Text Classification. *arXiv preprint arXiv:2206.00265*.

Welling, M.; and Kipf, T. N. 2016. Semi-supervised classification with graph convolutional networks. In *J. International Conference on Learning Representations (ICLR 2017)*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24.

Xie, Q.; Huang, J.; Du, P.; Peng, M.; and Nie, J.-Y. 2021. Inductive Topic Variational Graph Auto-Encoder for Text Classification. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4218–4227. Online: Association for Computational Linguistics.

Xu, G.; Meng, Y.; Qiu, X.; Yu, Z.; and Wu, X. 2019a. Sentiment analysis of comment texts based on BiLSTM. *Ieee Access*, 7: 51522–51532.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019b. How powerful are graph neural networks? *International Conference on Learning Representations*.

Xu, Z. 2021. RoBERTa-wwm-ext Fine-Tuning for Chinese Text Classification. *arXiv preprint arXiv:2103.00492*.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R. R.; and Le, Q. V. 2019a. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in Neural Information Processing Systems*, volume 32.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R. R.; and Le, Q. V. 2019b. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Yao, L.; Mao, C.; and Luo, Y. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7370–7377.

Yu, B.; Yin, H.; and Zhu, Z. 2018. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *International Joint Conference on Artificial Intelligence*.

Zhang, H.; and Zhang, J. 2020. Text graph transformer for document classification. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zhang, J.; Shi, X.; Xie, J.; Ma, H.; King, I.; and Yeung, D. 2018. GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, 339–349.