Learning Discriminative Text Representation for Streaming Social Event Detection

Chaodong Tong, Huailiang Peng, Xu Bai, Qiong Dai, Ruitong Zhang, Yangyang Li, Hanjie Xu, and Xianming Gu

Abstract—Event detection on social platforms can help people perceive essential events and make actionable decisions. Existing document-pivot streaming social event detection methods generally embed documents and perform text clustering. They face the challenges of constantly changing context and unknown event categories and struggle by designing compound text representation methods and various similarity measures. However, phased, well-designed methods are excessively fragile and unable to utilize the potential of text representations fully. Meanwhile, their complex threshold settings result in clustering-based event detection suffering the pain of ever-changing environments. We propose a text representation learning method namely Text Similarity Contrastive Learning Neural Network (Text-SimCLNN) to tackle these challenges. Text-SimCLNN uses smaller parts to learn the similarity probability of text pairs from semantic and structural perspectives, effectively bridging the gap between text representation learning and similarity measure in streaming event detection. Event discovery and merging in streams can be easily performed based on the learned representations, and we use various techniques to speed up such processes. Furthermore, we introduce an online update mechanism that uses heterogeneous graphs to generate high-quality samples to enable stable and reliable inductive learning. Extensive experiments on two real-world datasets demonstrate that our method far exceeds state-of-the-art (SOTA).

Index Terms—Contrastive learning, graph neural network, streaming event detection, text clustering, text representation, text similarity.

1 Introduction

The rapid growth of the Internet and changing lifestyles make people more willing to publish and share what they see and hear on social platforms. Simply observing posts related to the real world enables event detection promptly, which further enables users, organizations, and governments to acquire actionable knowledge involved with natural disasters, major social events, disease propagation [1], [2], etc.

Research on streaming social event detection, usually taking texts as the primary research object and linking multiple dimensions of information, explores text representation, similarity measure, and streaming clustering methods under time-series data. As the top priority of the task, text representation gains extensive research attentions. Many text representation methods have been developed to overcome the lack of dynamic and sparsity brought by classic methods such as TF-IDF [3], [4], [5], [6]. Most of them improve the text representation by incrementally updating the word representations or designing text representation models to obtain better word vectors. Linguistic features and knowledge bases have also been introduced to obtain better text representations [7]. Some studies construct graphs utilizing texts and closely related heterogeneous information to portray events more accurately [8], [9]. However, such structural information is costly to obtain, as is its limited coverage. Similarity measure and clustering methods are usually extended based on representation methods and are result-oriented. For the similarity measure, L^p space distance and cosine distance are widely used [3], [10], and some studies extend existing similarity measure methods based on their carefully designed text representation techniques. Becker et al. [10] calculate similarities for all their extracted contextual features and propose a weighted ensemble method to account for different measures of similarity during the clustering process. Peng et al. [8] propose a method for measuring similarity based on meta-paths called KIES as well as a nonlinear layer to support the learning of text representation. The proposed model is further used to determine whether two event instances (short texts) belong to the same category. Chen et al. [11] use the gated recurrent unit (GRU) and the attention mechanism to obtain text representations for one text pair. They normalize and multiply both text embeddings to obtain similarity and then optimize those embeddings by the backpropagation algorithm. Clustering (also named cluster analysis) is not a specific algorithm. Corresponding to event detection, it refers to collecting doc-

- *C. Tong, H. Peng, and X. Bai are with the Institute of Information Engineering, Chinese Academy of Science (CAS) and the School of Cyber Security, University of CAS, Beijing 100093, China. E-mail: {tongchaodong, penghuailiang, baixu}@iie.ac.cn.*
- *Q. Dai is with the Institute of Information Engineering, Chinese Academy of Science (CAS) and the School of Cyber Security, University of CAS, Beijing 100083, China, and also with the School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China. E-mail: daiqiong@iie.ac.cn.*
- *R. Zhang is with the the School of Cyber Science and Technology, Beihang University, Beijing 100083, China. E-mail: rtzhang@buaa.edu.cn.*
- *Y. Li is with the National Engineering Laboratory for Risk Perception and Prevention (NEL-RPP), CAEIT, Beijing 100041, China. E-mail: liyangyang@cetc.com.cn.*
- *H. Xu is with the Department of Computing and Software, McMaster University, Hamilton ON L8S6L8, Canada. E-mail: xuh113@mcmaster.ca.*
- *X. Gu is with the School of Economic Mathematics/Institute of Mathematics, Southwestern University of Finance and Economics, Chengdu 611130, China. E-mail: guxm@swufe.edu.cn.*

Manuscript received April, 2021. (Corresponding author: Qiong Dai.)

uments showing apparent subject uniformity in a cluster. Clustering methods based on similarity or distance often need to be optimized over time and space to cope with streaming event detection [12], [13].

However, when these phased, well-designed methods [10], [14], [15] work together, they face many difficulties once the environment changes. We argue that most of the existing text representation methods in streaming event detection do not fully exploit the potential for learning text representation. The vector spaces of the learned text representations are not entirely compatible with the similarity calculation subtask [6], thereby reducing the accuracy of the similarity measures. Invisible data will also challenge the learned representations. Furthermore, different methods of measuring similarity often show distinct distributions on the same dataset, and the aggregation of them makes the problem more complicated [10]. The impact of degraded performance of text representations and similarity measures will be further superimposed, which will significantly impact clustering outcomes. In addition, the existing methods rely on the manual setting of thresholds in many places, so any place with a threshold needs to be carefully adjusted to adapt for the method.

This study introduces a novel model architecture named Text Similarity Contrastive Learning Neural Network (Text-SimCLNN), which directly learns the probability of similarity of text pairs through contrastive learning. With such a method, streaming event detection is almost an end-to-end process, and there are almost no thresholds that we need to adjust dynamically. Text-SimCLNN mainly consists of a deep neural network for encoding and a simple classification header for similarity calculation. It is worth noting that the encoding part explicitly utilizes the complete structural information of texts with the help of graph neural networks (GNN). Moreover, it treats word pieces rather than words as basic units and exploits the generalization capabilities of pre-trained language models [16], [17]. This easy-to-separate structure allows to speed up similarity calculation and streaming clustering processes by various technical means. Specifically, we define the central points of an event and propose a method utilizing the central limit theorem (CLT) to sample them. We also introduce dynamic programming and pruning techniques to expedite the calculation and comparison process when dealing with streams. By the ways introduced above, there is no need to set a threshold for calculating the similarity of text pairs learned, and pairs with similarities above 0.5 are similar pairs. The same happens when we conduct ranking-based categorization in order to generate and merge events. In addition, to ensure the system's reliability and stability in streaming scenarios, we propose an update mechanism that obtains high-quality text pairs from new data fragments by constructing heterogeneous graphs and performing random walking. Fine-tuning utilizing these samples ensures that the system can behave to our expectations. We conduct extensive experiments and achieve a new state-of-the-art (SOTA) on two real-world data sets: the Twitter Event-2012 dataset and the TREC 2015 Microblog Track dataset.

In short, our contributions can be summarized as the following.

1) We propose a novel model architecture named Text-SimCLNN, which directly learns the probability of similarity of text pairs through contrastive learning. This end-to-end approach simplifies streaming event detection while making it more accurate in a constantly changing environment. Zero-shot induction can be performed by our method.
2) We apply various technologies to the single-pass algorithm to reduce the time and space complexity of our scenario. These technologies include defining the event's central points and sampling with CLT, dynamic programming, and pruning. Such improvements allow us to generate and merge events in streaming scenarios quickly.
3) An update mechanism is introduced to support reliable lifelong incremental inductive learning. Experiments with multiple time steps verify the stability of this mechanism.
4) We conduct extensive experiments while the results show that our method far exceeds SOTA. Meantime, we find that existing datasets are not labeled completely accurately, which results from the ambiguity of event definition. We hope that this will provide some insights for future related research.

The rest of this paper is organized as follows. Section 2 briefly introduces related works. Section 3 introduces some basic concepts related to our study. Section 4 elaborates on the proposed method. Section 5 presents and discusses the experimental results. We conclude this article in Section 6.

2 Related Works

In this section, we will briefly introduce the present situation and progress of event detection. We will also introduce text representation, similarity measure, and clustering methods related to social event detection.

2.1 Event Detection

Event detection has long been addressed in the Topic Detection and Tracking (TDT) program and is divided into two categories: retrospective event detection (RED) and new event detection (NED) [18]. Event detection methods can be classified into the specified and the unspecified [19], depending on the accessibility of prior knowledge of events. The related tasks are usually named as specified event detection (SED) and unspecified event detection (UED). Besides, according to different research emphasis, event detection techniques can be divided into the document-pivot, and the feature-pivot [19], [20]. Document-pivot techniques represent documents as vectors and classify or cluster them based on the representations, while feature-pivot techniques detect topic trends or areas in data streams. In this article, we focus on UED tasks and develop document-pivot techniques.

Distinct methods have been proposed to perform event detection. Clustering is the most popular technique in event detection systems [21]. Specifically, classical methods such as topic models, hierarchical clustering, and density-based clustering account for offline event detection; incremental clustering methods (e.g., single-pass and batch clustering) handle the streaming situation; another kind of method named graph partitioning, which takes advantage of graphs

and graph algorithms, is also widely researched [22]. Dimension reduction techniques are also used in most cases to reduce calculation time and space cost.

2.2 Text Representation Learning

Social event detection research has been involved in many perspectives, including text features, temporal features, spatial characteristics, social network structures, etc [20], [22], [23], [24], [25]. Among these elements, text features usually perform the primary role during event detection, and therefore developing text representation methods attracts the attention of many researchers. Early event detection approaches use some variants of TF-IDF to represent documents and then adopt clustering methods based on document similarity with time distance penalization [26]. Brants et al. [3] further improve this kind of method, proposing an incremental TF-IDF model to acquire dynamic text representations. This approach enables models to change with environments. Hu et al. [4] aim to ease the impact of the curse of dimensionality and feature independence assumption by splitting the document representation generation procedure into two steps. The method first clustering word vectors learned by Word2Vec into a latent semantic space. After that, documents are represented as a distribution across these latent topic clusters. Single-pass clustering is conducted with such representations. Dhiman et al. [6] propose JoSE, a spherical vector representation method. It leverages the co-occurrence information among words and paragraphs in a spherical generative model. Experiments show better word similarity and text clustering results in the final steps compared to Word2Vec and Doc2Vec. Recently, deep learning models have been widely used to learn text representations, including convolutional neural networks (CNN) and recurrent neural networks (RNN). Feng et al. [5] use Bi-LSTM and two CNNs with convolution kernel sizes of 2 and 3 to obtain word representations. Their downstream tasks are slightly different from ours, namely, trigger identification and trigger classification, and they will be further introduced in section 3. However, these model architectures can hardly capture non-consecutive and long-distance semantics [27].

Pre-trained language models such as ELMo [28] and BERT [17], have shown powerful generalization abilities and reached SOTA on multiple natural language processing (NLP) tasks. Shi et al. [29] propose a sentence-level feature-based event detection model to detect 14 types of meteorological events in the social network while the model consists of a fine-tuned BERT and a wide-grained capsule network. This method is essentially still a classification model working on specified event detection.

However, an essential factor determining the success or failure of streaming event detection is whether text representations are learned properly to handle uncertain categories, especially similarity calculation. Pre-trained language models are trained with specified pre-training tasks. Thus they are not suitable to be directly used under unsupervised scenarios.

2.3 Similarity Measure

Most clustering algorithms rely on distance or similarity measures. The classic similarity measure methods are widely used in social event detection, such as cosine distance, Hellinger distance, Kullback-Leibler divergence, Jensen-Shannon distance, and Clarity-based distance [3], [30]. Becker et al. [10] represent texts as a combination of various features (e.g., titles, descriptions, tags) and propose a learnable ensemble-based model to obtain text similarities with these features. Precisely, for one kind of feature, cosine similarity is calculated with the corresponding TF-IDF vectors. Fedoryszak et al. [14] design an entity similarity measure method. It selectively links entities based on frequencies and co-occurrences, thereby constructing entity graphs for graph-based clustering. Peng et al. [8] build an event-based heterogeneous information network and design a meta-path-based event similarity measure method called KIES. They further construct the adjacency matrix of event instances utilizing KIES. Corresponding event instance representations are obtained utilizing GCN. They additionally design a non-linear classification module to determine whether event instances belong to the same category. Liu et al. [31] define an event as an 8-tuple and design a variety of similarity measures to support investigating multi-lingual event detection and evolution.

2.4 Streaming Clustering

Due to the enormous amount of data in social streams and lack of prior knowledge of event categories, classical clustering methods such as K-means cannot work well in such situations. Many researchers use incremental clustering methods or perform batch clustering [4], [9], [32]. Single-pass incremental clustering is widely used because of its simpleness. In addition to designing proper similarity measures, quickness and effectiveness compared with existing categories is also the key to applying such a method. Becker et al. [10] define the centroid of a cluster as the average of its feature representations to speed up the comparison process. Zhang et al. [33] propose a cluster centroid update strategy. For a specific cluster, if the average similarity of the new text with all points is larger than the similarity between the new text and the old centroid, a new centroid will be formed by recalculating; otherwise, the centroid remains unchanged. Moreover, as for batch clustering, multiple processes of merging events are inevitable, which is not easy [34].

There are some other clustering methods based on graph partitioning. Zhao et al. [2] construct a hypergraph structure with the heterogeneous data in microblogs, and highly related ones are partitioned into subgraphs named microblog cliques (MCs). A bipartite graph is constructed, and then they perform a transfer cut partition to detect events. Existing MCs are used to guide new event detection. Sayyadi et al. [35] consider that events and stories can be characterized by a set of descriptive, collocated keywords. Thus they construct graphs utilizing keywords and entities and use graph algorithms such as graph partitioning and community detection to gain critical documents. Documents, which are similar to the critical documents in the original corpus, form a cluster.

3 Preliminaries

In the following subsections, we will introduce the highly related concepts with this study, including what can be

TABLE 1
The descriptions of primary notations.

Notation	Description
$\mathcal{C}$	event category space
$\mathcal{D}$	document instance space
$\mathcal{W}$	word space
C	event category set
D	document time series
O	observed data points conforming to f
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	text graph
$\mathcal{V}$	the node set of $\mathcal{G}$
$\mathcal{E}$	the edge set of $\mathcal{G}$
$\boldsymbol{X} \in \mathbb{R}^{N \times 1}$	the node feature matrix of $\mathcal{V}$
$\boldsymbol{A} \in \mathbb{R}^{N \times N}$	the adjacency matrix of $\mathcal{G}$
$\mathcal{N}(i)$	the entry neighbors of node $v_i \in \mathcal{V}$
$S \in \mathbb{R}^{N}$	the word piece sequence
$\boldsymbol{S} \in \mathbb{R}^{N \times 1}$	the word piece feature matrix of S
$\boldsymbol{H_{st}^{(l)}} \in \mathbb{R}^{D \times D}$	the l-th layer hidden states of the structural encoder
$\boldsymbol{H_{se}^{(l)}} \in \mathbb{R}^{D' \times D'}$	the l-th layer hidden states of the semantic encoder
$\boldsymbol{E_{se}} \in \mathbb{R}^{D}$	the output matrix of the semantic encoder
$\boldsymbol{E_{st}} \in \mathbb{R}^{D'}$	the output matrix of the structural encoder
N	the size of $\mathcal{V}$
D	the dimension of the embedding layer in the structural encoder, equal to D_{st}
D'	the dimension of the embedding layer in the semantic encoder, equal to $D_{se}^{'}$
D_{st}	the dimension of hidden states in the structural encoder
$D_{se}^{'}$	the dimension of hidden states in the semantic encoder

called an event and the definition of the problem being studied.

3.1 Definition of an Event

The research so far has not yet made exact judgments on how events are defined [19], [36]. Stanford Encyclopedia of Philosophy says philosophers who agree with a conception of events as particulars distinguish events as four sorts: 1) activities; 2) accomplishments; 3) achievements; and 4) states[1]. According to different theories, this taxonomy may contain some variants. Some scholars argue that different verbs describe different types of events based on Aristotle's theory. In recent years the method of dividing event detection into two subtasks which are trigger identification (TI) and trigger classification (TC) [37] coincides with such philosophical ideas. In data mining, computer science, researchers have concretized the concept of an event into a feasible definition. McMinn et al. [38] define that an event as "something significant that happens at a specific time and place." Focusing on Twitter, Weng and Lee [20] describe an event as "a set of posts sharing the same topic and words within a short time." Becker et al. [39] give a more formal definition, "an event is a real-world occurrence e with a period T_e and a stream of Twitter messages discussing the event during the period T_e." Though differences existing more or less among the above definitions, we define an event within the scope of our research as:

Definition 1. An event is a collection of documents that refer to the same real-world activities, accomplishments, achievements, and states. There should be at least one difference in participants, places, time, etc., in two different events.

With the definition of an event, we further define what an event mention is:

Definition 2. An event mention is a document of an event that at least contains one aspect of the event and can be distinguished from mentions of other events.

Here we give an example to explain these two concepts. For example, the posts related to "the COVID-19 virus" could be called as an *event*, while each post might talk about different perspectives about "COVID-19" and can be referred to as an *event mention*. However, in another fine-grained situation, "WHO marks six-month anniversary of the COVID-19 outbreak" and "WHO experts to travel to China" could be described as two different *events*[2]. Such definitions allow for overlap between events and even conceptual upper-lower relations as long as they can be distinguished. In fact, such phenomenon is widely present in daily life and research data sets.

In addition, these two definitions ensure a certain degree of distinction between documents of different events for our research. At the same time, it is also the premise of the text representation learning method we proposed.

3.2 Problem Definition

Important notations in this article have been summarized in Table 1 while matrices and vectors are in uppercase and lowercase and both in bold. Formally, given a series $D = \{d_0, d_1, ..., d_i, ..., d_n | d_i \in \mathcal{D}\}$ and a partly known or unknown event category set $C = \{c | c \in \mathcal{C}\}$, the problem (unspecified event detection, UED) is to establish a mapping $f : \mathcal{D} \rightarrow \mathcal{C}$. Specifically, $d = \{w_0, w_1, ..., w_n | w \in \mathcal{W}, n \in \mathbb{N}\}$ and some sample points conforming to the mapping f can be observed as $O = \{c = f(d) | d \in D, c \in C\}$, note that O can be an empty set. We estimate or learn the mapping f without or with parameters.

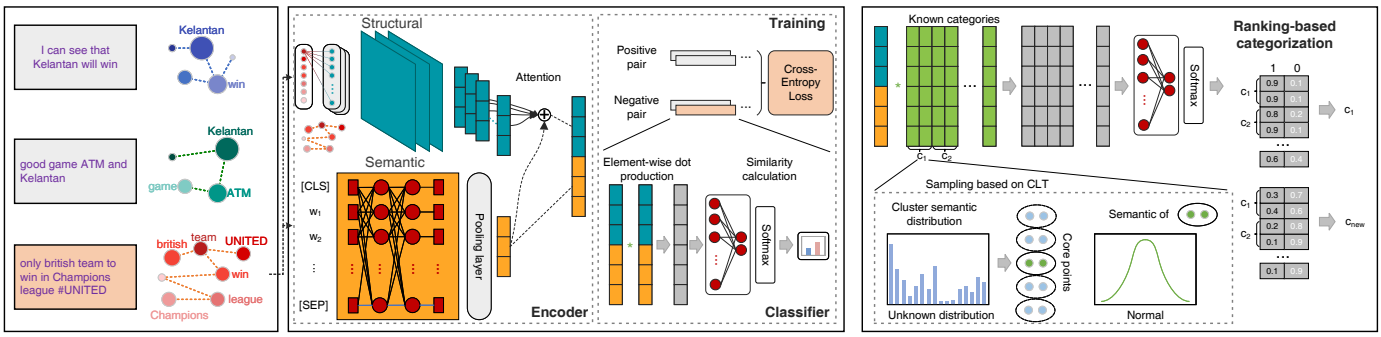
4 Methodology

In this section, we elaborate on our proposed method, including the architecture of Text-SimCLNN, the offline training procedure based on contrastive learning, a quick online inference method, and an update mechanism for stably inductive learning.

Firstly, we introduce the contrastive learning framework for text categorization and how we adapt it to UED tasks. After that, divided into details, we introduce a simple method that constructs samples for semi-supervised contrastive learning to enable offline training. Secondly, we elaborate on the proposed text similarity contrastive learning neural network, the related loss function, and its training procedure. For the inference stage, we introduce a quick inference method based on single-pass clustering with knowledge regularization. Finally, we introduce an update

1. https://plato.stanford.edu/entries/events/ (last accessed: March 25, 2021)

2. https://www.who.int/emergencies/diseases/novel-coronavirus-2019/events-as-they-happen



(a) Messages and Syntax Trees (b) Architecture of Text-SimCLNN (c) Inference with Knowledge Regularization

Fig. 1. The workflow of Text-SimCLNN. (a) illustrates messages in the data stream and their dependency syntax trees contained therein, and the word that plays a significant role in categorization is in large size. (b) shows the model architecture of Text-SimCLNN as well as its training process. (c) shows the inferring process of event merging and generating. The centroid for one event is sampled from all the documents within that event cluster, while the sampling strategy is based on the central limit theorem (CLT).

mechanism to correct the system in streaming scenarios. With the above steps, event detection can perform in a closed-loop; thus, the system falls into a stable and reliable inductive learning paradigm. Fig. 1 illustrates the workflow of Text-SimCLNN in streaming social event detection.

4.1 Contrastive Learning Framework

The essence of deep learning is to do two things: representation learning and inductive bias learning. As a recently widely concerned representation learning method, contrastive learning has proven its effectiveness in the field of computer vision [40], [41]. For an n-class classification problem, contrastive learning does not directly learn the inductive preferences. It tries to capture the features that can distinguish samples coming from different categories. We adapt contrastive learning to the UED task, and the motivations will be described as follows. The goal of a UED task is to detect which event a social post should belong to in a streaming scenario while the total number of event categories is unknown. The n-class classification techniques are not up to this task. Thus an easy-to-throwing practice is to use unsupervised learning methods such as clustering to tackle this problem. The performance of most existing clustering methods highly depends on the quality of the learned text representations. Under this premise, the text representation learning methods face the following two core challenges.

1) The methods must be able to represent out of vocabulary (OOV) words; or ignore them, which obviously loses semantics.
2) The learned text representations should have a strong generalization ability.

The first challenge is unique to natural language processing, compared to computer vision. The representation space of pixels is fixed, namely most commonly fixed to several dimensions, such as channels and RGB values. However, the word space of natural language is unfixed due to the variability of words, although most languages consist of limited graphemes and phonemes. WordPiece [16] is proposed and successfully applied on BERT [17], showing its effectiveness against this challenge which is also called the OOV problem. The second challenge has been studied for a long time, and typically several approaches have been proposed, such as transfer learning, pre-training and fine-tuning, as well as contrastive learning.

In the introduced contrastive learning framework, we regard the text categorization as multiple binary classifications. Thus we can treat any text categorization problem as a text-pair contrastive learning problem that enables texts to be embedded into a unified representation space and allows categories to change. Positive and negative samples shall be constructed in a relatively balanced manner to gain decent performance, according to Paulina et al. [42]. Text preprocessing within contrastive learning shall be applied with WordPiece or similar approaches. The representation learning model must be effective enough to handle the second challenge. Under such a framework, different-category texts can be distinguished from each other, and the categories of new texts are predictable.

4.2 Text-pair Construction Strategy

Although contrastive learning can work in a self-supervised manner, namely using the potential characteristics to generate training data such as next sentence prediction (NSP), sentence order prediction (SOP), and masked language model (MLM), we prefer to use stronger supervision signals.

As we train our models on a fixed WordPiece vocabulary, more specifically its size is 30522, the training data should cover the vocabulary as much as possible. We intercept a large enough fragment from the stream and artificially label it as follows:

1) For negative sample pairs, we randomly select a sample pair without replacement from the fragment and judge whether it is a negative pair. We drop it if it is a positive one. When the fragment runs out, a new fragment is intercepted. This process goes back and forth until the number reaches our expectations.
2) For negative samples, the selecting procedure is similar.

4.3 Text Similarity Contrastive Learning Neural Network

This subsection will introduce our proposed method in detail. Firstly, we will introduce the preprocessing procedure, including constructing a text to a graph and other regular means. The Text-SimCLNN model architecture will be described in 4.3.2. The content related to training will be introduced in 4.3.3.

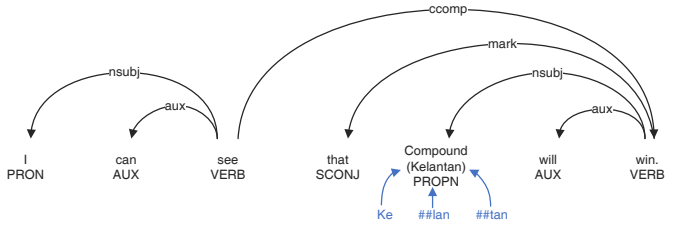


Fig. 2. Construction of a text graph. A dependency parsing tree is used to construct the nodes and edges. "ROOT" nodes, usually verbs, also called event triggers, can effectively transmit their information to other nodes without being affected by the long distance in the sequence.

4.3.1 Preprocessing

As for short social texts, we do not deal with them by conventional means like removing stop words, stemming, etc. The reason is that short texts inherently contain limited information. In the experiments, we mainly use Twitter data and remove the Twitter links and extra space separators. After that, each text is encoded into two forms: sequence encoding [17] for semantic modeling and graph encoding for structural modeling. The graph encoding procedure is: One text is first parsed using a syntactic parsing tree and then constructed into a directional graph. Tree nodes become vertexes or subgraphs depending on whether the words of nodes can be divided into smaller pieces. The subgraph is generated by firstly generating a "Compound" node and then stringing it with word pieces. As for edges, each vertex points to the vertex, which is its son in the parsing tree. Fig. 2 illustrates this construction method with an example.

4.3.2 Model Architecture

Most competitive text classification methods do not explicitly model the whole text structure [43]. Transformer-based models show powerful feature capturing capabilities, and through large-scale pre-training and fine-tuning on target tasks, they have reached or even surpassed SOTA on multiple benchmarks [17], [44]. Recently an exciting work [45] conducted a series of experiments to investigate the ability of BERT in capturing structural information. Their experimental results suggest that the intermediate layers of BERT encode a rich hierarchy of linguistic information. Such linguistic information includes surface features at the bottom, semantic features at the top, and even some syntactic features in the middle (corresponding evaluation tasks including sensitivity to word order, the depth of the syntactic tree, and the sequence of top-level constituents in the syntax tree). However, the subject-verb agreement evaluation shows that the captured structural information is limited, and deeper layers are needed to solve long-distance dependency problems.

In our method, we use GNN to model the complete structural information within one text explicitly. We further combine it with the features captured by the transformer-based encoder to obtain better generalization capabilities.

Given one text t, it is encoded into a word piece sequence $S = \{t_0, t_1, ..., t_i, ..., t_n\}$ and a text graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}), \mathcal{V} = \{v_0, v_1, ..., v_i, ..., v_N\}$, where $v_j = t_i$ *or* "$Compound$", $j = 0, 1, 2, ..., N, N \geq n$.

Semantic encoder. It mainly accounts for capturing semantic features for texts. The encoder is composed of an embedding layer, a 12-layer transformer decoder [17] and a pooling layer. We compute the output as follows:

$$\boldsymbol{H_{se}^{(0)}} = Dropout(LayerNorm(Embedding(\boldsymbol{S}))), \quad (1)$$

$$\boldsymbol{H_{se}^{(l)}} = TransformerDecoder(\boldsymbol{H_{se}^{(l-1)}}), \quad (2)$$

$$\boldsymbol{e_{se}} = AveragePooling(\boldsymbol{H_{se}^{(12)}}), \quad (3)$$

where $l = 1, 2, ..., 12$.

Structural encoder. Transformer-based models cannot fully capture the structural features of texts, and more specifically, they cannot transmit long-distance dependencies losslessly. The encoder is mainly designed to alleviate this problem. It is composed of an embedding layer, a graph neural network and a pooling layer. We compute the output as follows:

$$\boldsymbol{H_{st}^{(0)}} = Embedding(\boldsymbol{X}), \quad (4)$$

$$\boldsymbol{H_{st}^{(1)}} = Linear(\boldsymbol{H_{st}^{(0)}}), \quad (5)$$

$$\begin{aligned}\boldsymbol{H_{st_i}^{(l)}} &= GNN(\boldsymbol{H_{st_{\mathcal{N}(i)}}^{(l-1)}}, \boldsymbol{A}) \\ &= \boldsymbol{W_{l_1}} \boldsymbol{H_{st_i}^{(l)}} + \boldsymbol{W_{l_2}} \cdot mean_{j \in \mathcal{N}(i)} \boldsymbol{H_{st_j}^{(l)}}, \end{aligned} \quad (6)$$

$$\boldsymbol{H_{st}^{(l)}} \leftarrow \boldsymbol{H_{st_i}^{(l)}}, \forall v_i \in \mathcal{V}, \quad (7)$$

$$\boldsymbol{e_{st}} = Pooling(\boldsymbol{H_{st}^{(2)}}). \quad (8)$$

The embedding layer weights in 4 are initialized from the embedding layer of pre-trained BERT. The reason why we do not perform dropout [46] and layer normalization [47] in 4 is that structural information is an important part of short texts. We argue that it is not suitable for dropout while shallow neural networks hardly benefit from layer normalization. Equation 6 is the update process of node v_i in lth layer and $\mathcal{N}(i)$ are the entry neighbors of v_i. In the experiments, we find that one GNN layer is enough, and deeper GNN does not get better results, thus $l = 2$. Considering the characteristics of UED tasks, we implement the GNN layer in 6 with an inductive graph neural network [48]. As for the pooling function in 8, we employ different pooling strategies, including max pooling, averaging. To selectively leveraging structural encoding, we further employ an attention-based pooling method. This method is formalized as follows:

$$\boldsymbol{e_{st}} = MultiHeadAttention(\boldsymbol{e_{se}}, \boldsymbol{H_{st}^{(l)}}, \boldsymbol{H_{st}^{(l)}}), \quad (9)$$

where $MultiHeadAttention$ is introduced by [49]. In the experiments, we find that $MultiHeadAttention$ sometimes performs best while the average pooling achieves optimal or sub-optimal performance all the time.

The final text representation of one text is the combination of semantic and structural embeddings:

$$\boldsymbol{e} = Concatenate(\boldsymbol{e_{se}}, \boldsymbol{e_{st}}). \quad (10)$$

Classification header. As mentioned in 4.1, text pairs are encoded and then classified as positive or negative, and we further convert such similarity degree into similarity probability. The similarity probability of two text embeddings $\boldsymbol{e_1}, \boldsymbol{e_2}$ is calculated as follows:

$$output = Softmax(S(\boldsymbol{e_1}, \boldsymbol{e_2})), \quad (11)$$

where S is a mapping function with parameter θ that expresses the similarity of one text pair:

$$S = f(\boldsymbol{e_1}, \boldsymbol{e_2}; \theta). \tag{12}$$

We will further introduce 12 in detail in 4.3.3.

4.3.3 Loss Function and Training Process

Based on our text-pair construction method, the positive and negative samples are relatively balanced. We hope that the model can discern whether the two samples belong to the same category to the greatest extent; therefore, we directly optimize category probabilities. For each text in one pair, its final embedding is represented by concatenating its semantic and structural embedding, equivalent to superimposing two signals. The similarity between them is calculated as:

$$\boldsymbol{s} = Linear(\boldsymbol{e_1} \cdot \boldsymbol{e_2}), \tag{13}$$

$$\hat{y} = softmax(label = positive|\boldsymbol{s}), \tag{14}$$

where $\cdot$ is an element-wise dot product operation. We employ the element-wise dot product instead of cosine because cosine measurement loses the module length information. The linear layer enables better feature selection, while more complex non-linear structures will fit the task and thus disperse the embedding effect.

We use cross-entropy for optimization. With the above formulations, the loss function on the training dataset forms as follows:

$$L = \sum_{i=1}^{n} -(y \cdot \log \hat{y} + (1-y) \cdot \log(1-\hat{y})), \tag{15}$$

where n is the number of text pairs constructed from the training dataset. To speed up the training process, we pack text graphs and text sequences to enable batch training. Parallelization over a graph mini-batch is achieved by creating sparse block diagonal adjacency matrices and concatenating feature and target matrices in the node dimension[3]. This composition allows a differing number of nodes and edges over examples in one batch. To achieve joint learning of the two encoders, we use AdamW [50] for training and set the optimized step size to the smaller one [17], namely 1.0×10^{-5}.

4.4 Quick Inference with Knowledge Regularization

In the inference stage, we employ the single-pass clustering algorithm to detect events. Such an algorithm can perform streaming clustering by calculating the similarities between the current document and all existing clusters. With our representation method, the similarity between text points is determined by 14, namely $\hat{y}$. Unlike the classic clustering algorithms, it is not necessary to carefully adjust the thresholds (such as the maximum reachable distance and minimum points in DBSCAN [51]), which determine whether two points belong to the same cluster. The reason is that 14 represents the probability that two samples belong to the same category. Thus, an obvious choice is to set the threshold to 0.5. Accordingly, the core task is how to quickly and effectively generate and merge clusters (events). We start from two aspects: representing events as core point sets and introducing the ranking technique for the single-pass clustering.

Event core points. Clustering algorithms are usually of high complexity. We represent a cluster as a small set of core points sampled from the cluster. We perform clustering with the single-pass clustering technique: Each document compares with the existing cluster core points; a new cluster will be formed if it is not similar to all existing clusters.

Assuming that the semantics of *events* and *event mentions* are measurable, and the semantic of an *event* is the average of all corresponding *event mentions*, core points are defined as follows: The core points are a set of points sampled from one event cluster, and their average semantic should be equal to the semantic of the corresponding event on the premise that they obey the distribution of event mentions. We then offer a sampling strategy to form this core point set. The sampling strategy first performs n sampling with replacement while each sample size is k ($k \geq 30$), then randomly choosing one sample from $k_1, k_2, ..., k_n$ as the core point set. With such a method, the sample set is basically in line with the original data distribution according to CLT, and the event core points will not produce semantic drift in the statistical sense.

We will give a simple explanation of how this is achieved. Assume there is an event with m documents. Each of them owns a certain semantic value v where $v \in \mathbb{R}$, and we have reason to believe that v obeys a certain distribution (may wish to set the mean as μ). The mean of the distribution can represent the semantic of the event cluster. According to CLT, the distribution of the mean μ_k of the sampled set $m_1, m_2, ..., m_k, k \geq 30$ is approximately normally distributed and $\bar{\mu}_k = \mu$. Thus the semantic of the event cluster can be represented by the mean of samples that are randomly sampled from the distribution of μ_k.

In our experiments, n is set to 100, and k is set to 30. It is worth noting that this sampling strategy enables the computational cost to increase linearly with the number of categories. And for $(D_{st} + D_{se}^{'})$-dimensional space, where $D_{st} = D_{se}^{'} = 768$ in our experiments, it can easily support the distinction of thousands of event categories.

Ranking-based categorization. One document d can be similar with multiple event core point sets and may wish to set as $s_1 = \{d_{11}, d_{12}, d_{13}, d_{14}, d_{15}\}, s_2 = \{d_{21}, d_{22}, d_{23}, d_{24}, d_{25}\}, s_3 = \{d_{31}, d_{32}, d_{33}, d_{34}, d_{35}\}$. The similarities between d and s_1, s_2, s_3 involve with all the documents $d_{11}, d_{12}, ..., d_{35}$ thus the embeddings of $d_{11}, d_{12}, ..., d_{35}$ can be precomputed with 10. Five similarity scores q_1, q_2, q_3, q_4, q_5 can be calculated for d and documents within each set utilizing 13 and 14, and the final similarity score between d and each set is the average of them, i.e., $q_{final} = \frac{1}{5}\sum_{i=1}^{5} q_i$. With such a method a ranking-based categorization process can be easily performed. When the average similarity score between d and the core point set is less than 0.5, we regard them as the *density-unreachable* (the concept is borrowed from DBSCAN), and a new cluster will be formed with d. It is worth noting that once a document is classified into a cluster, the corresponding core points will be resampled with a small probability. This infrequent updating process for core points ensures the dynamic char-

3. Implementation referenced in https://pytorch-geometric.readthedocs.io/en/latest/notes/introduction.html#mini-batches

Algorithm 1: Text-SimCLNN Based Single Pass Text Clustering

Input: Known event clusters C_{known}, cluster labels L_{known} and corresponding embedding matrix $\boldsymbol{E_{known}}$; a streaming fragement including n documents $F = \{d_0, d_1, ..., d_n\}$, precomputed document embeddings $\boldsymbol{E} = \{\boldsymbol{e_{d_1}}, \boldsymbol{e_{d_2}}, ..., \boldsymbol{e_{d_n}}\}$.

Output: Document labels $L_{new} = \{l_1, l_2, ..., l_n\}$, detected event clusters C_{known}.

D = $descSortByLength(F)$;
for d_i *in* D **do**
  $\boldsymbol{e_{d_i}} \leftarrow search(\boldsymbol{E}, d_i)$;
  $\boldsymbol{W_{sim}} \leftarrow Sim(\boldsymbol{e_{d_i}}, \boldsymbol{E_{known}})$ by Eq. (14);
  $\boldsymbol{W_{sim}}[:\boldsymbol{k}] \leftarrow$ ranking (Sec. 4.4);
  $\boldsymbol{W_{sim}}[:\boldsymbol{k'}] \leftarrow$ pruning (Sec. 4.4);
  $q_{most} = max(sum(\boldsymbol{W_{sim}}[:\boldsymbol{k'}]))$ (Sec. 4.4);
  if $q_{most} < 0.5$ **then**
    $C_{known} \leftarrow c_{new}$;
    $l_i \leftarrow l_{new}$;
    $L_{new} \leftarrow add(L_{new}, l_i)$;
    $\boldsymbol{E_{known}} \leftarrow stack(\boldsymbol{E_{known}}, \boldsymbol{e_{d_i}})$;
  else
    $c_{most} \leftarrow q_{most}$;
    $l_i \leftarrow l_{most}$;
    $L_{new} \leftarrow add(L_{new}, l_i)$;
    $c_{most} \leftarrow add(c_{most}, d_i)$;
    $C_{known} \leftarrow update(C_{known}, c_{most})$;
    if $random() < \epsilon$ *(Sec. 4.4)* **then**
      $d_{cores} \leftarrow$ resampling c_{most};
      $\boldsymbol{E_{cores}} \leftarrow search(\boldsymbol{E}, d_{cores})$;
      $\boldsymbol{E_{known}} \leftarrow update(\boldsymbol{E_{known}}, \boldsymbol{E_{cores}})$;
    else
      do nothing;

acteristic of event clusters as well as calculations at a high speed.

We further accelerate this process with the idea of pruning. Assuming $q_{21}+q_{22}+3.0 < q_{11}+q_{12}+q_{13}+q_{14}+q_{15}$, s_2 will no longer need to participate in comparison calculations and the computational cost is reduced. Similarly, we can also prune the calculation process further according to the distribution of the data, e.g., when the q_{final} would not be unlikely to exceed 0.95, discard the current compared event category.

In Text-SimCLNN, the time consumption of text encoding is high. Thus by implementing the clustering algorithm with the dynamic programming technique, each text will be calculated only once during inference. Considering that not all events in the newly arrived stream are unseen, we pre-calculate the embeddings of already known clusters to detect them quickly. We call this inference process with prior knowledge as knowledge regularization, and it is referred to as KR in the rest of this article. The whole clustering process is shown in Algorithm 1.

4.5 Update Mechanism

Inductive learning in streaming scenarios without feedback and correction will result in unstable and unreliable system performance. An example shows such a phenomenon:

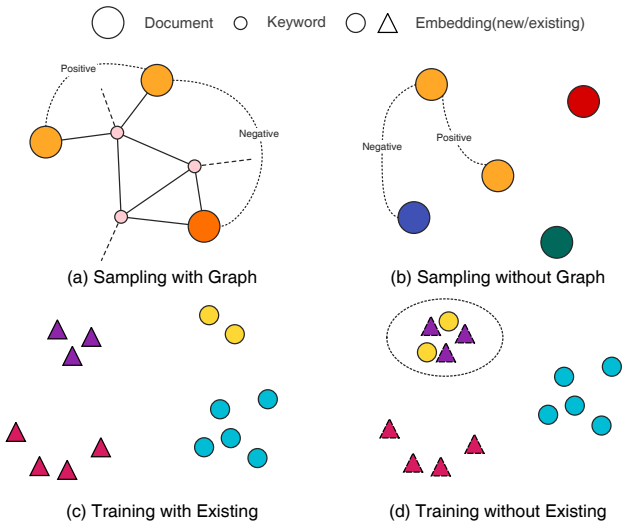


Fig. 3. Advantages of the proposed update mechanism. Samples gained with the graph are more indistinguishable and learning from them is more efficient. Adding existing event core points can avoid overlapping in learned vector spaces.

Example 1. **P1 (C8):** Crazy political junkie here, listening to the debates.#Debate2012

C1 Description: 12 Oct 2012 – Paul Ryan spoke for 40 of the 90 minutes during Thursday night's vice presidential debate and managed to tell at least 24 myths during that time.

C8 Description: During US presidential debate, President Barack Obama tells candidate Mitt Romney he is the last person to get tough on China.

Within this example, the document is predicted as **P1** which refers to the ground truth **C1**, while it belongs to a new category **C8**. Too fine event granularity definition and event evolution will cause much overlap between documents, resulting in such a phenomenon.

We propose an update mechanism, i.e., regular fine-tuning, to overcome this problem. When the system exhibits performance degradation or when it comes to the update cycle time, usually weekly or monthly, the update module will activate to correct the system. Firstly, a sample set will be generated from current streams following the sampling strategy described in 4.4. Keywords are then extracted from these documents utilizing TF-IDF, and a heterogeneous graph is constructed. Top-10 keywords within each document and document itself become vertices while keyword co-occurrence forms edges. Some start document nodes are randomly selected with the graph constructed, and n paths are generated with k-step random walking. Supposing m document nodes are filtered out from n paths, positive and negative pairs are automatically labeled utilizing hashtags, or other platform-specific symbols [9]. The proportion of data based on this explicit similarity judgment is not significant. Therefore, the text-pair construction strategy mentioned in 4.2 is applied to prevent the size of sample pairs from being too small. Parts of the already known event cluster core points are added to m to avoid spatial overlap between the learned representations and the existing ones.

Text-SimCLNN is then fine-tuned with the newly generated data.

Note that: 1) Graph-based sampling makes the construction of positive sample pairs very efficient, while the construction of negative samples can be independent of the graph; 2) Keyword co-occurrence-based edges ensure that the positive samples obtained are of high quality; 3) 2-6 is enough for the k value selection.

Fig. 3 gives an intuitive explanation of why the sample pairs selected in this way are of high quality and why the core points of known event categories should be added.

5 Experiments

This section introduces the experiment setups, including datasets, baselines, metrics, and hyper-parameter settings. A pilot experiment is conducted to investigate the influence of the ratio of positive and negative samples on contrastive learning performance. After that, we compare our method with baselines in an offline mode and further investigate the effectiveness of our model under the streaming scenario. We also conduct further experiments to explore the generalization ability of the proposed model. At last, we visualize the learned text representations and show the benefits. We use a *16G ASPEED Graphics Family (rev 30)* card for training.

5.1 Experiment Setups

5.1.1 Datasets

Twitter Event-2012 Dataset [38]: The Twitter Event-2012 dataset was collected from the 10th of October 2012 to the 7th of November 2012. It covers many interesting and significant events, including Hurricane Sandy and the U.S. Presidential Elections. Wikipedia Current Events Portal and some other event detection approaches are used to create a pool of events. Specifically, after filtering duplicate tweets, the dataset contains 66935 tweets distributed across 504 event categories.

TREC 2015 Microblog Track Dataset[4]: This dataset is a part of the TREC 2015 Microblog Track, which consists of 46 manually produced equivalence classes of tweets. The event categories defined within it own a coarser granularity compared with the Twitter Event-2012 dataset, for instance, Iran nuclear agreement, Greek international debt crisis, California drought agricultural effects, etc. The total number of documents in this dataset is 3,488. Both datasets show a long-tail distribution.

5.1.2 Baselines

We compare our method with general text representation models with common similarity measuring methods, namely cosine and Euclidean distance. Besides, we compare our method to PP-GCN and KPGNN, which are SOTA event detection methods and supervised text classification methods, i.e., GCN [52] and BERT [17], in the offline evaluation experiment. The text representation models are briefly described below. Word2Vec [53], namely Skip-Gram and CBOW, which establish a self-supervised learning method between words and context, directly generates word embeddings; thus, sentences can be embedded with the average of all the words[5]. GloVe obtains word vector representations based on global word-word co-occurrence statistics. ELMo [28] uses stacked Bi-LSTM to obtain dynamic word vectors, and we take the average of the last hidden states of all words as sentence embeddings. BERT [17] performs pre-training on large corpora with the masked language model and next sentence prediction tasks while outputs the embedded word piece sequence vector-matrices in the language model hidden layers. Sentence embedding can be obtained with the average on some hidden layer, and we choose the last in our experiments. In order to distinguish from the BERT for text classification, it is referred to as BERT$_{emb}$. PP-GCN [8] builds an event-based heterogeneous information network and proposes a novel graph neural network as well as a similarity calculation method named KIES to perform event clustering. KPGNN [9] models complex social messages into unified social graphs and obtains dynamic sentence embeddings with a constantly changing heterogeneous knowledge graph.

5.1.3 Metrics

We mainly use three metrics to evaluate the performance, and they are briefly described below[6].

ARI: The adjusted Rand index is a function that measures the similarity of the two assignments, ignoring permutations.

NMI: The normalized mutual information uses information entropy to measure the similarity of two distributions.

AMI: The adjusted mutual information adjusts NMI utilizing expectations.

5.1.4 Hyper-parameter Settings

As for the contrastive learning samples construction stage, we test a series of different positive and negative sample ratios to see how does the sample distributions affect the performance, and it will be discussed in 5.2. Other customized hyper-parameter settings are shown in Table 2.

TABLE 2
Customized hyper-parameter settings.

Parameter	Value
Embedding size of the structural encoder, D	768
GNN hidden size, D_{st}	768
Number of layers of GNN	1
Learning rate	1e-5
Batch size	60
Early stop threshold	3
Cluster core points	30

5.2 Sample Distribution Influence

Statistics-based (machine) learning methods are driven by data, and their performance is significantly affected by sample distributions and sample sizes. In this part, we explore

4. https://trec.nist.gov/data/microblog2015.html

5. In the experiments, we find that CBOW with multiple preprocessing is a solid baseline. These preprocessing methods include lowercasing, removing multiple spaces, short tokens, digits, and punctuations, replacing dash between words, and filtering stopping words.

6. https://scikit-learn.org/stable/modules/clustering.html#overview-of-clustering-methods

TABLE 3
Sample distributions and quantities.

Negative & positive ratio	0.1	0.5	1	2	5	10	20	full
Neg. category coverage	39/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46
Num of avaliable categories (avg)/category	5.32	11.09	15.78	20.13	27.00	31.20	35.01	-
Num of neg. samples (avg)/category	6.42	27.13	54.27	108.53	271.33	542.67	1085.33	-

Note: The data used for statistics comes from sampling the training dataset with different ratios. For each sample in the dataset, one positive sample pair will be constructed. "Number" is abbreviated as "num", the same with "negative" and "average". The last column "full" represents the training dataset.

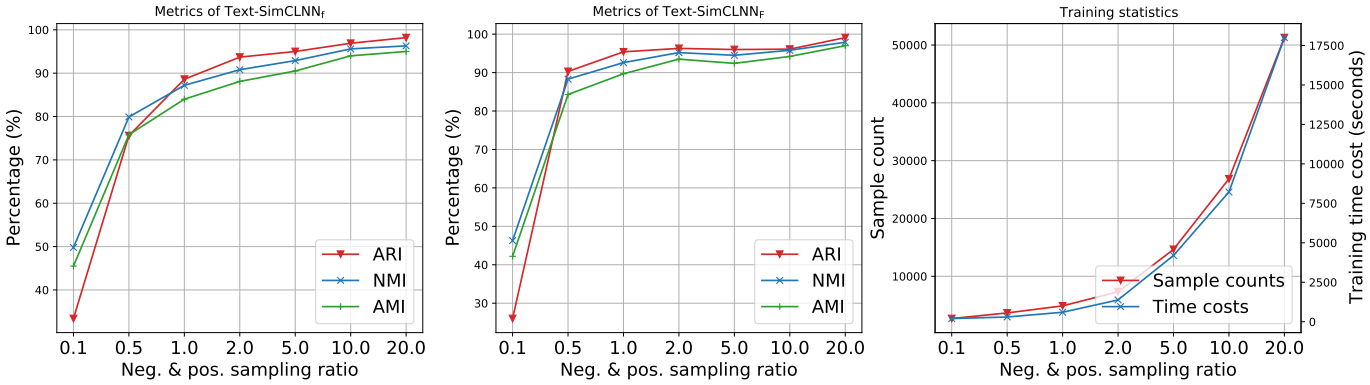


Fig. 4. Metrics and training statistics under different negative and positive sampling ratios.

TABLE 4
Offline evaluation metrics of our method compared with baselines on the **Twitter** Event-2012 dataset and the **TREC** 2015 Microblog Track dataset.

		Word2Vec	GloVe	ELMo	BERT$_{emb}$	PP-GCN	KPGNN	GCN	BERT	Text-SimCLNN$_{p/P}$	Text-SimCLNN$_{f/F}$
Twitter	ARI	0.152	0.030	0.061	0.041	0.207	0.223	*0.784*	*0.860*	0.759/0.821	0.797/**0.862**
	NMI	0.693	0.478	0.342	0.484	0.695	0.710	*0.931*	***0.959***	0.838/0.913	0.853/0.928
	AMI	0.579	0.305	0.285	0.326	0.517	0.521	*0.904*	***0.923***	0.771/0.856	0.787/0.881
TREC	ARI	0.732	0.113	0.129	0.147	0.768	0.814	*0.969*	*0.936*	0.944/**0.970**	0.942/0.968
	NMI	0.793	0.322	0.358	0.356	0.807	0.833	*0.947*	*0.937*	0.929/0.956	0.930/**0.958**
	AMI	0.727	0.227	0.260	0.248	0.801	0.820	*0.929*	*0.914*	0.907/0.939	0.908/**0.942**

Note: Each experiment is carried out three times, and the displayed metrics take the average. The best results are marked in bold. The metrics obtained by supervised classification methods are in *italic*.

how much the sample distribution and sample quantity will affect the performance of the text representation method in our framework. At the same time, the prior knowledge derived from the experimental results further guides the following experiments. We carry out our experiments on the TREC dataset and choose a series of different negative and positive sampling ratios to construct the training text pairs while details are shown in Table 3. Evaluations are conducted in an offline scenario in which Text-SimCLNN is trained on the sampled text pairs generated from the training dataset, and ARI, NMI, AMI metrics are calculated on the test dataset.

Fig. 4 shows the performance across the series. ARI, NMI, and AMI rise quickly and then enter a lower slope zone with the ratio increased. In addition, the more training data, the performance is getting better, and no performance bottleneck is observed in our parameter settings. As shown in the middle subfigure, the curve has an inflection point at "2.0". The training time cost linearly increases with the ratio growth. Following the experimental results, we choose the ratio as "2.0" in the latter experiments to balance training costs and performance.

5.3 Offline Evaluation

The offline evaluation is conducted on both datasets and follows the 7:2:1 approach to construct the training, evaluation, and test datasets. We use the CBOW form of Word2Vec and train it following the hyper-parameters mentioned in [53]. GloVe[7] and ELMo[8] are implemented with the open released sources. Considering no next sentence relations existing in the training data, we use the popular released source[9] for BERT and fine-tune it with masked language model task only. For one document, we directly use the average pooling strategy to get its embedding with the model outputs, and specific to BERT, we choose the last hidden layer states as the model output. Density clustering with Euclidean and cosine distance is performed with these text presentation models, and the better is reported. We train them with the same epochs as our method for classification methods, i.e., GCN and BERT, to get relatively fair judgments. PP-GCN and KPGNN provide full-stack solutions. Therefore we compare with them, utilizing the released sources described

7. https://github.com/maciejkula/glove-python
8. https://github.com/ltgoslo/simple_elmo
9. https://huggingface.co/

TABLE 5
The distribution and vocabulary coverage of the streaming fragments.

Periods	Category number (partial/total)	New category number	Document number (cate. existing/new)	Vocabulary coverage up to now	Vocabulary coverage increase
$\boldsymbol{F_1}$ **(2012-10-08_2012-10-15)**	112/112	112/100%	0/14,524	10,488/30,522	1.385/doc
$\boldsymbol{F_{1,2}}$ **(2012-10-16_2012-10-22)**	141/229	117/83%	1,845/18,747	14,495/30,522	0.195/doc
$\boldsymbol{F_{1,2,3}}$ **(2012-10-23_2012-10-29)**	150/361	132/88%	1,086/13,526	16,161/30,522	0.114/doc
$\boldsymbol{F_{1,2,3,4}}$ **(2012-10-30_2012-11-05)**	134/475	114/85%	1,767/12,217	17,266/30,522	0.079/doc
$\boldsymbol{F_{1,2,3,4,5}}$ **(2012-11-06_2012-11-12)**	47/504	29/61%	536/2,687	17,517/30,522	0.078/doc

The basic word piece vocabulary consists of 30,522 word pieces. Here is an example to illustrate this table. For $F_{1,2}$, "141/229" means there are 141 categories in F_2 ($F_{1,2} - F_1$) and "up to now" category number is "229". Among the "141" categories, there are "117" categories (i.e., 83% of "141") we have never seen. Document numbers corresponding to the "seen" and "unseen" categories are "1,845" and "18,747".

TABLE 6
Streaming evaluation **ARIs**. The best results in one step are marked in bold. Metric differences before and after fine-tuning are shown in brackets.

	Init (F_1)	$\boldsymbol{F_2}$	$\boldsymbol{F_3}$	$\boldsymbol{F_4}$	$\boldsymbol{F_5}$
Word2Vec	0.613±.072	0.562±.079 (+0.127)	0.385±.048 (+0.170)	0.392±.065 (+0.112)	0.299±.051 (+0.176)
GloVe	0.533±.022	0.472±.031 (+0.277)	0.342±.040 (+0.138)	0.328±.037 (+0.125)	0.285±.035 (+0.123)
ELMo	0.143±.040	0.121±.038 (+0.021)	0.106±.012 (-0.008)	0.085±.011 (-0.024)	0.071±.012 (-0.015)
BERT$_{emb}$	0.148±.017	0.119±.020 (-0.013)	0.093±.010 (+0.003)	0.088±.009 (+0.014)	0.081±.010 (-0.008)
PP-GCN	0.698±.037	0.477±.035 (+0.319)	0.413±.027 (+0.213)	0.368±.028 (+0.257)	0.313±.029 (+0.205)
KPGNN	0.732±.054	0.589±.048 (-)	0.451±.037 (-)	0.442±.044 (-)	0.445±.035 (-)
Text-SimCLNN$_p$	0.969±.012	0.867±.010 (+0.121)	0.632±.011 (+0.198)	**0.800±.011** (+0.237)	0.875±.007 (+0.250)
Text-SimCLNN$_P$	0.983±.002	0.869±.003 (+0.153)	0.626±.006 (+0.271)	0.675±.005 (+0.214)	0.723±.002 (+0.282)
Text-SimCLNN$_f$	0.975±.005	0.871±.007 (+0.147)	0.739±.007 (+0.230)	0.783±.006 (+0.259)	**0.882±.007** (+0.296)
Text-SimCLNN$_F$	**0.990±.001**	**0.882±.002** (+0.174)	**0.752±.003** (+0.253)	0.727±.001 (+0.221)	0.619±.002 (+0.208)

in [8], [9]. We further evaluate Text-SimCLNNs with partial or full information available and with or without knowledge regularization (KR), referred to as Text-SimCLNN$_p$, Text-SimCLNN$_P$, Text-SimCLNN$_f$, and Text-SimCLNN$_F$.

As shown in Table 4, Text-SimCLNNs are better than all the others while Text-SimCLNN$_F$ performs best. The structural information has brought some improvements, and knowledge regularization magnifies this improvement. Intuitively, if some events are known, the judgment of new documents can rely on specific knowledge that appeared within the existing categories when comparing with them. We argue that more formal or structural texts would benefit more from our method than the short social ones. As we can see, Word2Vec and GloVe gain pretty high metrics on the TREC 2015 Microblog Track dataset. However, they gain low ARIs on the Twitter Event-2012 dataset, which means the clustering results are far from the ground truths. It reveals that Word2Vec and GloVe can learn discriminative text representations with a small amount of data, but cannot work well with many samples and categories. Pre-trained language models such as ELMo and BERT do not work well on both of the two datasets. In other words, the average of their hidden layer outputs cannot be directly used for clustering. That is reasonable because their training tasks focus on language modeling, and text differences are not explicitly modeled. Thus the representational capacity of distances within the learned characterization space is limited. PP-GCN and KPGNN leverage the global structure information of streaming data. However, the representation of language sequences is inadequate. Besides, all these compared methods need to be adjusted carefully during clustering and will suffer the constantly changing distance thresholds during streaming clustering.

5.4 Streaming Scenario Evaluation

This subsection evaluates Text-SimCLNN in a streaming scenario.

Experiments are conducted on the Twitter Event-2012 dataset only mainly because of the size and period of the other one. The Twitter Event-2012 dataset spans five natural weeks, and we split it into five fragments, namely as F_1, F_2, F_3, F_4, F_5. The distribution and vocabulary coverage are shown in Table 5. High coverage of the vocabulary is beneficial to migrate our methods to unseen documents. The first fragment F_1 is used for training Text-SimCLNNs from scratch. After that, we perform incremental learning on the rest fragments. At each step, we perform the update mechanism with a sampled set (1000 new samples and 1000 existing event core points) which consists of some samples of current fragment and part of core points of known clusters and then fine-tune Text-SimCLNNs to repeat. Metrics are calculated on the whole current fragment. Streaming evaluation with the update mechanism is conducted in full steps, and ARI, NMI, AMI metrics are reported. Text representation learning baselines such as Word2Vec, GloVe, ELMo, and BERT$_{emb}$ are fine-tuned at each step to enable comparisons. PP-GCN and KPGNN are evaluated following the instructions mentioned in [8], [9] and Text-SimCLNN$_p$, Text-SimCLNN$_P$, Text-SimCLNN$_f$, and Text-SimCLNN$_F$ are involved.

Streaming evaluation results are shown in Table 6, 7, 8. KPGNN constructs heterogeneous graphs for the predicted documents. Thus, it can not be evaluated before fine-tuning. Basically, all models will benefit from fine-tuning. The metrics of Word2Vec have been declining after it starts with high metric values. Metrics of Text-SimCLNNs reduce by 10-20% after initialization. With step increasing, the three metrics

TABLE 7
Streaming evaluation **NMIs**. The best results in one step are marked in bold. Metric differences before and after fine-tuning are shown in brackets.

	Init (F_1)	F_2	F_3	F_4	F_5
Word2Vec	0.806±.017	0.756±.031 (+0.215)	0.738±.026 (+0.173)	0.729±.018 (+0.192)	0.490±.013 (+0.091)
GloVe	0.625±.029	0.571±.025 (+0.249)	0.510±.019 (+0.223)	0.483±.022 (+0.208)	0.453±.021 (+0.183)
ELMo	0.507±.011	0.463±.017 (+0.087)	0.370±.014 (-0.011)	0.298±.010 (-0.055)	0.216±.008 (-0.074)
BERT$_{emb}$	0.453±.019	0.401±.013 (-0.029)	0.338±.012 (+0.033)	0.305±.007 (+0.006)	0.281±.008 (-0.013)
PP-GCN	0.787±.014	0.657±.012 (+0.383)	0.631±.016 (+0.279)	0.628±.017 (+0.302)	0.614±.007 (+0.258)
KPGNN	0.811±.011	0.739±.010 (-)	0.688±.016 (-)	0.677±.011 (-)	0.652±.009 (-)
Text-SimCLNN$_p$	0.931±.007	**0.770±.006** (+0.096)	0.748±.006 (+0.123)	**0.795±.004** (+0.141)	0.855±.005 (+0.159)
Text-SimCLNN$_P$	0.957±.002	0.767±.001 (+0.064)	**0.751±.002** (+0.083)	0.791±.001 (+0.127)	0.811±.003 (+0.104)
Text-SimCLNN$_f$	0.935±.003	0.719±.002 (+0.072)	0.718±.003 (+0.088)	0.773±.001 (+0.125)	**0.858±.003** (+0.162)
Text-SimCLNN$_F$	**0.966±.001**	0.750±.002 (+0.080)	0.739±.001 (+0.059)	0.779±.002 (+0.115)	0.796±.001 (+0.102)

TABLE 8
Streaming evaluation **AMIs**. The best results in one step are marked in bold. Metric differences before and after fine-tuning are shown in brackets.

	Init (F_1)	F_2	F_3	F_4	F_5
Word2Vec	0.793±.021	0.740±.017 (+0.176)	0.714±.015 (+0.148)	0.703±.022 (+0.122)	0.449±.011 (+0.134)
GloVe	0.611±.017	0.488±.019 (+0.218)	0.492±.019 (+0.180)	0.440±.025 (+0.167)	0.453±.018 (+0.199)
ELMo	0.476±.008	0.383±.012 (+0.014)	0.318±.014 (-0.005)	0.260±.009 (-0.018)	0.196±.006 (-0.013)
BERT$_{emb}$	0.429±.011	0.401±.010 (-0.031)	0.312±.008 (+0.007)	0.287±.009 (+0.012)	0.275±.005 (-0.025)
PP-GCN	0.787±.014	0.657±.012 (+0.341)	0.631±.016 (+0.258)	0.628±.017 (+0.229)	0.614±.007 (+0.189)
KPGNN	0.811±.011	0.739±.010 (-)	0.688±.016 (-)	0.677±.011 (-)	0.652±.009 (-)
Text-SimCLNN$_p$	0.927±.004	**0.754±.005** (+0.124)	0.723±.001 (+0.091)	**0.779±.003** (+0.137)	0.844±.002 (+0.136)
Text-SimCLNN$_P$	0.954±.001	0.747±.002 (+0.088)	**0.726±.001** (+0.076)	0.762±.001 (+0.114)	0.782±.003 (+0.092)
Text-SimCLNN$_f$	0.932±.003	0.704±.002 (+0.061)	0.692±.001 (+0.057)	0.756±.004 (+0.109)	**0.848±.003** (+0.159)
Text-SimCLNN$_F$	**0.964±.001**	0.731±.001 (+0.099)	0.700±.001 (+0.081)	0.748±.002 (+0.111)	0.765±.002 (+0.084)

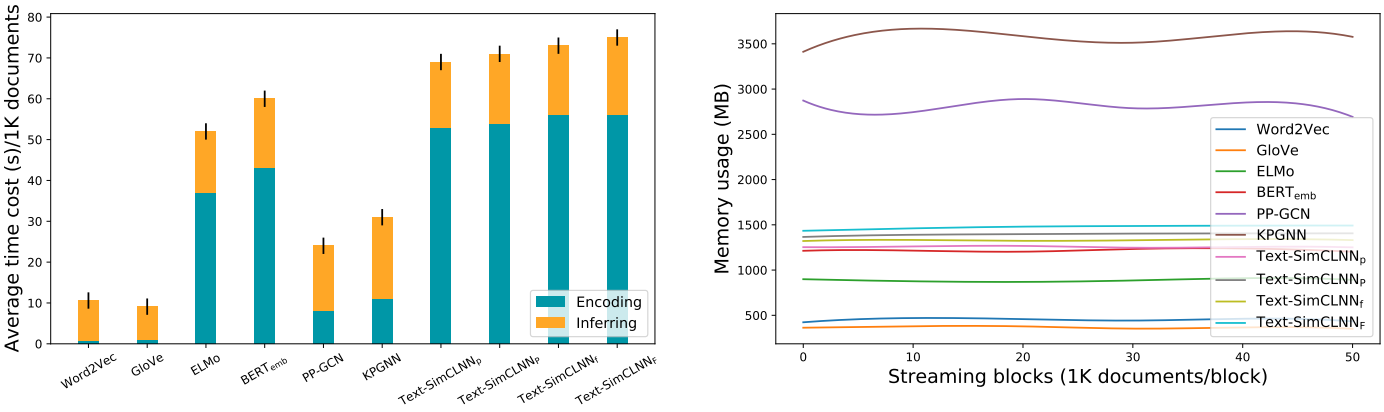


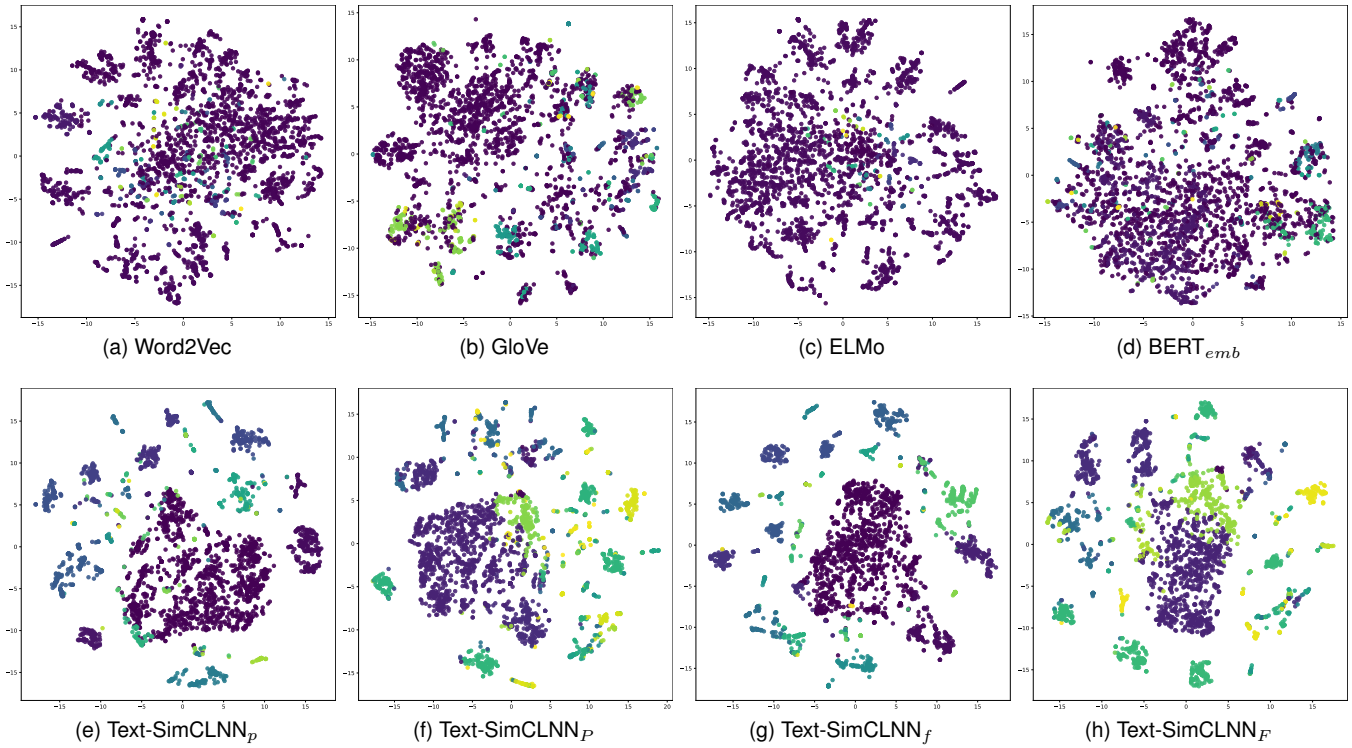
Fig. 5. Time and memory costs in streaming scenarios.

are maintained at a relatively stable value interval which is at around 0.7 and 0.8. For ARI, Text-SimCLNN$_F$ performs best in the beginning and falls to the last at the final step. For NMI and AMI, the four Text-SimCLNN variants have comparable performance. In general, KR brings no performance improvement. It is reasonable because topics are constantly changing in the streaming scenario, and the old ones are gradually no longer discussed, so the existing knowledge can not guide the inference stage as time flows. Text-SimCLNN$_f$ maintains high-level metrics at the final step, proving the effectiveness of our proposed model architecture. After four steps, the metrics have not been continuously reduced; that is, no apparent semantic drift problem has been shown, which shows the effectiveness of the update mechanism. Metrics of existing text representation learning methods decline over time and drop to a low level, showing their weaknesses against such streaming event detection tasks. There is a 20-40% performance gap between models specially designed for social event detection such as PP-GCN and KPGNN and our method, showing the superiority of the proposed Text-SimCLNN.

We further investigate the time costs and memory usages of our method in streaming scenarios. We use 1000 documents to count the average time consumption and total memory usages of GPU of our method and baselines. As shown in Fig. 5, larger models and higher-dimensional embeddings will cause the corresponding methods to consume more time and space when processing the same number of documents. Specifically, Word2Vec and GloVe own minimal consumptions of time and space, while methods involved with BERT usually take higher costs. Text-SimCLNNs consume the most time, but they are still not an order of magnitude higher than the simple methods, i.e., Word2Vec and GloVe. Our method does not specifically optimize for par-

TABLE 9
Transfer evaluation results on the TREC 2015 MicroBlog Track dataset. The best results are marked in bold and metrics after the slashes are the best results with offline training on the dataset.

Metircs	Word2Vec	GloVe	ELMo	BERT$_{emb}$	Text-SimCLNN$_p$	Text-SimCLNN$_P$	Text-SimCLNN$_f$	Text-SimCLNN$_F$
ARI	0.349	0.285	0.035	0.088	0.788/0.944	**0.810**/0.970	0.740/0.942	0.755/0.968
NMI	0.574	0.501	0.062	0.277	0.790/0.929	**0.809**/0.956	0.788/0.930	0.788/0.958
AMI	0.443	0.379	0.047	0.134	0.682/0.907	**0.698**/0.939	0.673/0.908	0.662/0.942



(a) Word2Vec (b) GloVe (c) ELMo (d) BERT$_{emb}$

(e) Text-SimCLNN$_p$ (f) Text-SimCLNN$_P$ (g) Text-SimCLNN$_f$ (h) Text-SimCLNN$_F$

Fig. 6. Visualization of the documents embeddings on F_5 of the Twitter Event-2012 dataset. Each node denotes one document, and each color denotes one predicted cluster category. These categories mainly consist of forty-seven political or sports events like the 2012 U.S. election, the 2012 UEFA champions league, and some international terrorist attacks.

allelization. When predicting, only one sample is processed at a time. In fact, our method is quite suitable for parallel processing, but that is another systemic problem, and we will leave it for future research. The space consumptions of Text-SimCLNNs mainly come from the model occupations and in-memory pre-computed event core points matrix and are less than PP-GCN and KPGNN.

5.5 Generalization Ability

We further evaluate the generalization ability of our method. Offline trained models with the Twitter Event-2012 dataset are used to perform event clustering on the TREC 2015 MicroBlog Track dataset. Table 9 shows the results. The best scores of ARI, NMI, and AMI are $0.810, 0.809, 0.698$, against $0.924, 0.924, 0.896$, which are gained with an offline trained model. To our surprise, the best results are even better than the results of Word2Vec with offline training. The experimental results suggest that the text representations learned by our method can generalize to other domains and different categories. We guess that Text-SimCLNN$_P$ performs better than Text-SimCLNN$_F$ because the embedding weights in GNN do not gain adequate pre-training. Other baselines can hardly perform like this.

5.6 Text representation Visualization

We visualize the text vectors obtained on F_5 by t-SNE as shown in Fig. 6. Text representation learning methods, including Word2Vec, GloVe, ELMo, and BERT, can only find out a small number of event categories, and the learned text representations visualized in t-SNE do not show obvious distinguishability. Compared with them, Text-SimCLNNs discover more event categories, and the learned text representations of Text-SimCLNNs are sparsely distributed after visualization. Besides, Text-SimCLNNs with KR (i.e., providing prior knowledge) show a more obvious distinction. Such a phenomenon is most evident in Text-SimCLNN$_F$, which proves that it learns the most discriminative text representations for streaming social event detection.

6 Conclusion

In this study, we mainly address social event detection from the perspective of text representation learning techniques. We propose an end-to-end text representation learning method named Text-SimCLNN which bridges the gap, i.e., not wholly matching between text representation learning and similarity measures in social event detection. In addition, we have made efforts to accelerate and optimize the

inferring process for event detection in streaming scenarios. These efforts include practical but straightforward ideas, such as representing an event as a small core point set based on CLT. We apply the proposed method to streaming event detection scenarios through a graph-based update mechanism. Experimental results prove the effectiveness of our method. We further investigate the generalization ability of Text-SimCLNN, and it suggests that well-trained Text-SimCLNN can perform zero-shot induction with acceptable accuracy.

A direction worth studying in the future would be extending the semi-supervised contrastive learning in this article into an unsupervised manner.

ACKNOWLEDGMENTS

This work is supported by the Strategic Priority Research Program of the Chinese Academy of Sciences with No. XDC02030000 and S&T Program of Hebei through grant 21340301D.

REFERENCES

[1] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes twitter users: real-time event detection by social sensors," in *Proceedings of the WWW*, 2010, pp. 851–860.

[2] S. Zhao, Y. Gao, G. Ding, and T.-S. Chua, "Real-time multimedia social event detection in microblog," *IEEE Trans Cybern*, vol. 48, no. 11, pp. 3218–3231, 2017.

[3] T. Brants, F. Chen, and A. Farahat, "A system for new event detection," in *Proceedings of the SIGIR*, 2003, pp. 330–337.

[4] L. Hu, B. Zhang, L. Hou, and J. Li, "Adaptive online event detection in news streams," *Knowl Based Syst*, vol. 138, no. C, pp. 105–112, 2017.

[5] X. Feng, B. Qin, and T. Liu, "A language-independent neural network for event detection," *Sci. China Inf. Sci.*, vol. 61, no. 9, pp. 1–12, 2018.

[6] A. Dhiman and D. Toshniwal, "An approximate model for event detection from twitter data," *IEEE Access*, vol. 8, pp. 122 168–122 184, 2020.

[7] S. Unankard, X. Li, and M. A. Sharaf, "Emerging event detection in social networks with location sensitivity," *Proceedings of the WWW*, pp. 1393–1417, 2015.

[8] H. Peng, J. Li, Q. Gong, Y. Song, Y. Ning, K. Lai, and P. S. Yu, "Fine-grained event categorization with heterogeneous graph convolutional networks," in *Proceedings of the IJCAI*, 2019, pp. 3238–3245.

[9] Y. Cao, H. Peng, J. Wu, Y. Dou, J. Li, and P. S. Yu, "Knowledge-preserving incremental social event detection via heterogeneous gnns," in *Proceedings of the WWW*, 2021, p. 3383–3395.

[10] H. Becker, M. Naaman, and L. Gravano, "Learning similarity metrics for event identification in social media," in *Proceedings of the WSDM*, 2010, pp. 291–300.

[11] G. Chen, Q. Kong, and W. Mao, "Online event detection and tracking in social media based on neural similarity metric learning," in *Proceedings of the ISI*, 2017, pp. 182–184.

[12] N. Alsaedi, P. Burnap, and O. Rana, "Can we predict a riot? disruptive event detection using twitter," *ACM Trans. Internet Technol.*, vol. 17, no. 2, pp. 1–26, 2017.

[13] J. Ansah, L. Liu, W. Kang, J. Liu, and J. Li, "Leveraging burst in twitter network communities for event detection," 2020, pp. 2851–2876.

[14] M. Fedoryszak, B. Frederick, V. Rajaram, and C. Zhong, "Real-time event detection on social data streams," in *Proceedings of the SIGKDD*, 2019, pp. 2774–2782.

[15] H. Peng, J. Li, Y. Song, R. Yang, R. Ranjan, P. S. Yu, and L. He, "Streaming social event detection and evolution discovery in heterogeneous information networks," *ACM Trans. Knowl. Discov. Data*, vol. 15, no. 5, pp. 89:1–89:33, 2021.

[16] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[17] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the NAACL-HLT*, 2019, pp. 4171–4186.

[18] J. Allan, Ed., *Topic Detection and Tracking: Event-Based Information Organization*. Norwell, MA, USA: KAP, 2002.

[19] F. Atefeh and W. Khreich, "A survey of techniques for event detection in twitter," *Comput. Intell.*, vol. 31, no. 1, pp. 132–164, 2013.

[20] J. Weng and B.-S. Lee, "Event detection in twitter," in *Proceedings of the ICWSM*, 2011, pp. 401–408.

[21] M. Cordeiro and J. Gama, *Online social networks event detection: a survey*. Cham: Springer International Publishing, 2016, vol. 9580, pp. 1–41.

[22] X. Zhou and L. Chen, "Event detection over twitter social media streams," *VLDB J*, vol. 23, no. 3, pp. 381–400, 2013.

[23] X. Dong, D. Mavroeidis, F. Calabrese, and P. Frossard, "Multiscale event detection in social media," *Data Min Knowl Discov*, vol. 29, no. 5, pp. 1374–1405, 2015.

[24] J. Chae, D. Thom, H. Bosch, Y. Jang, R. Maciejewski, D. S. Ebert, and T. Ertl, "Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition," in *Proceedings of the VAST*, 2012, pp. 330–337.

[25] F. Chen and D. B. Neill, "Non-parametric scan statistics for event detection and forecasting in heterogeneous social media graphs," in *Proceedings of the SIGKDD*, 2014, pp. 1166–1175.

[26] J. Allan, R. Papka, and V. Lavrenko, "On-line new event detection and tracking," in *Proceedings of the SIGIR*, 1998, pp. 37–45.

[27] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, and Q. Yang, "Large-scale hierarchical text classification with recursively regularized deep graph-cnn," in *Proceedings of the WWW*, 2018, pp. 1063–1072.

[28] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the NAACL*, 2018, pp. 2227–2237.

[29] K. Shi, C. Gong, H. Lu, Y. Zhu, and Z. Niu, "Wide-grained capsule network with sentence-level feature to detect meteorological event in social network," *Future Gener Comput Syst*, vol. 102, pp. 323–332, 2020.

[30] Q. Li, A. Nourbakhsh, S. Shah, and X. Liu, "Real-time novel event detection from social media," in *Proceedings of the ICDE*, 2017, pp. 1129–1139.

[31] Y. Liu, H. Peng, J. Li, Y. Song, and X. Li, "Event detection and evolution in multi-lingual social streams," *Front. Comput. Sci.*, vol. 14, no. 5, pp. 1–15, 2020.

[32] C. Li, A. Sun, and A. Datta, "Twevent: segment-based event detection from tweets," in *Proceedings of the CIKM*, 2012, pp. 155–164.

[33] D. Zhang, Y. Han, and X. Li, "Dynamic detection method of microblog topic based on time series," in *Proceedings of the ICPCSEE*, 2018, pp. 192–200.

[34] S. Katragadda, R. G. Benton, and V. V. Raghavan, "Framework for real-time event detection using multiple social media sources," in *Proceedings of the HICSS*, 2017, pp. 1–10.

[35] H. Sayyadi, M. Hurst, and A. Maykov, "Event detection and tracking in social streams," in *Proceedings of the ICWSM*, 2009, pp. 311–314.

[36] Z. Saeed, R. A. Abbasi, O. Maqbool, A. Sadaf, I. Razzak, A. Daud, N. R. Aljohani, and G. Xu, "What's happening around the world? a survey and framework on event detection techniques on twitter," *J. Grid Comput.*, vol. 17, no. 2, pp. 279–312, 2019.

[37] N. Ding, Z. Li, Z. Liu, H. Zheng, and Z. Lin, "Event detection with trigger-aware lattice neural network," in *Proceedings of the EMNLP-IJCNLP*, 2019, pp. 347–356.

[38] A. J. McMinn, Y. Moshfeghi, and J. M. Jose, "Building a large-scale corpus for evaluating event detection on twitter," in *Proceedings of the CIKM*, 2013, pp. 409–418.

[39] H. Becker, M. Naaman, and L. Gravano, "Beyond trending topics: Real-world event identification on twitter," in *Proceedings of the ICWSM*, 2011, pp. 438–441.

[40] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the ICML*, 2020, pp. 1597–1607.

[41] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the CVPR*, 2020, pp. 9729–9738.

[42] P. Hensman and D. Masko, "The impact of imbalanced training data for convolutional neural networks," *Degree Project in Computer Science, KTH Royal Institute of Technology*, 2015.

[43] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, p. 150, 2019.

[44] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," in *Proceedings of the ACL*, 2019, pp. 4487–4496.

[45] G. Jawahar, B. Sagot, and D. Seddah, "What does bert learn about the structure of language?" in *Proceedings of the ACL*, 2019, pp. 3651–3657.

[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J Mach Learn Res*, vol. 15, no. 56, pp. 1929–1958, 2014.

[47] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[48] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the NeurIPS*, 2017, pp. 1025–1035.

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the NeurIPS*, 2017, pp. 6000–6010.

[50] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proceedings of the ICLR*, 2019.

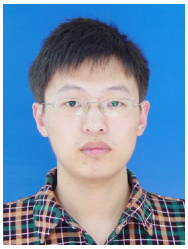
[51] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Proceedings of the KDD*, 1996, pp. 226–231.

[52] H. Peng, J. Li, S. Wang, L. Wang, Q. Gong, R. Yang, B. Li, P. S. Yu, and L. He, "Hierarchical taxonomy-aware and attentional graph capsule rcnns for large-scale multi-label text classification," *IEEE Trans Knowl Data Eng*, vol. 33, no. 6, pp. 2505–2519, 2021.

[53] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of Workshop at ICLR*, 2013.

Chaodong Tong received the Master degree in cyber security from the Institute of Information Engineering, Chinese Academy of Sciences, in 2020. He is currently a junior engineer in the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. His research interests include data mining and natural language processing.

Huailiang Peng received the Master degree in information security from the Institute of Information Engineering, Chinese Academy of Sciences, in 2016. He is currently an engineer in the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. His research interests include social network analysis and graph representation learning.



Xu Bai received the PhD degree in computer systems organization from the Institute of Information Engineering, Chinese Academy of Sciences, in 2019. He is currently an engineer in the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. His research interests include cryptographic algorithms and edge computing.

Qiong Dai received the Master degree in computer software and theory from Xiangtan University, in 2000. She is a Ph.D. candidate in computer science at Beijing University of Posts and Telecommunications. She is currently an associate professor in the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. Her current research interests include data mining, knowledge graph and collaborative computing.

Ruitong Zhang is currently a Ms.D. candidate in the School of Cyber Science and Technology in Beihang University, Beijing, China. Her research interests include deep learning, graph mining and reinforcement learning.

Yangyang Li is currently a senior engineer in the National Engineering Laboratory for Risk Perception and Prevention (NEL-RPP), CAEIT, Beijing, China. His research interests include data science, social network, and network security.

Hanjie Xu is currently an undergraduate student in McMaster University, Hamilton, Canada. His research interests include natural language processing and spatio-temporal analysis.

XianMing Gu is currently an associate professor with the School of Economic Mathematics/Institute of Mathematics, Southwestern University of Finance and Economics, Chengdu, China. His research interests include machine learning, parallel-in-time algorithms, and numerical solutions of (fractional) PDEs.