2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)

Robust Social Event Detection via Deep Clustering

Jiaofu Zhang*, Lianzhong Liu*, Zihang Huang*, Lihua Han†, Shuhai Wang†, Tongge Xu*, Jingyi Zhang*, Yangyang Li‡, Yifeng Liu‡, Md Zakirul Alam Bhuiyan§
* School of Cyber Science and Technology, Beihang University, Beijing 100191, China
† School of Continuing Education, Shijiazhuang Railway University, Shijiazhuang 050043, China
‡ National Engineering Laboratory for Risk Perception and Prevention (NEL-RPP),
China Academy of Electronics and Information Technology, Beijing 100041, China
§ Department of Computer and Information Sciences, Fordham University, NY 10458, USA
Email: zhangliangzjf@qq.com,{lz_liu,pppihf,xutg}@buaa.edu.cn,turuarua@163.com,
{hanlihua,wangshuhai}@stdu.edu.cn, {liyangyang,liuyifeng3}@cetc.com.cn,mbhuiyan3@fordham.edu

***Abstract*—Social networks are quickly becoming the primary medium for discussing what is happening around real-world events. However, it is still a challenge to detect events on social media due to its real-time nature, scale and amount of unstructured data generated. In this paper, we present a novel real-time system for detecting surrounding real-world events. Our proposed framework consists of four main components, including text filtering, text representation, deep clustering, and event merging. After filtering non-event messages, we use entities and words to represent messages. Based on text representation, we propose a novel density clustering algorithm for online event detection. The resulted sub-events are further merged based on time information and keyword similarity. Experiments on standard and real-world datasets demonstrated the effectiveness of our proposed method.**

***Index Terms*—event detection, cluster analysis, temporal information, social media**

I. Introduction

Nowadays, various social media sites, such as Twitter and Weibo, have become an indispensable part of in personal daily life. Consumers share their views and broadcast news and information about ongoing events. These messages posted can typically reflect these events as they happen. Social event detection is very important since it provides valuable insights for us to make timely responses, and therefore has many applications such as predictive analysis [1–6], crisis management [7] and public opinion analysis [8, 9]. Thus, how to effectively and efficiently detect events over social streams has become a hot spot in academia and industry.

Compared with traditional media (e.g., online news sites, blogs), social media has unique characteristics that make event detection particularly challenging [10]. Firstly, the noisy characteristics of social networks makes relevant events buried in large amount of noisy data. Secondly, the most social message is short and often contain abbreviations that are not in the dictionary, so understanding their semantics is difficult. In addition, due to the real-time nature of social networks, the system needs to recognize and track new or unforeseen events in real time to provide accurate results as quickly as possible. Moreover, event detection can't be static, and it needs to further analysis of the ongoing events from the massive event clusters generated by online clustering.

While previous work [11–19] has achieved good results in event detection, these approaches may be susceptible to noise, which can reduce the quality of social event detection. For example, tweets not directly related to a specific event may introduce noise (e.g., many tweets about his career in "Tiger Woods accident").

To address these challenges, we propose a real-time event detection framework, including text filtering, text representation, deep clustering, and event merging. Specifically, the text filtering component identifies event-related tweets from noisy and irrelevant messages. Then, the text representation component employs entities and words to model texts. After text representation, the deep clustering component groups similar tweets within a period in the same event cluster. Finally, we merge the events in different time slices. Figure 1 provides an overview of our proposed online event detection framework.

Compared to previously published work on event detection, the main contributions of this work are summarized as follows:

- We propose a general framework for real-time event detection, including text filtering, text representation, online clustering, and event merging. It can track event over time and shows the temporal analysis of sub-events. Moreover, the framework is flexible and easy to expand, so users can change the method of a certain module.
- Aiming at the problem that DBSCAN algorithm can' t tackle the various densities datasets correctly, we propose a novel density clustering algorithm, which can automatically detect noisy events and further process these events.
- Experiments on standard dataset shows that the system has strong anti-interference ability and improves the efficiency of online event detection.

II. Preliminaries and Problem Definition

In this section, we first summarize the main notations used in this paper in Table I. Then we formalize *Social Stream*, *Social Event* and *Event Detection* as follows.

Definition 2.1: A **social stream** $M = \{m_0, ..., m_i, ...\}$ is a sequence of messages arriving continuously, m_i is associated with a pair (m_i, t_i) , where m_i is user generated text segments and t_i is the publishing time in non-descending order.

978-0-7381-2646-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00116

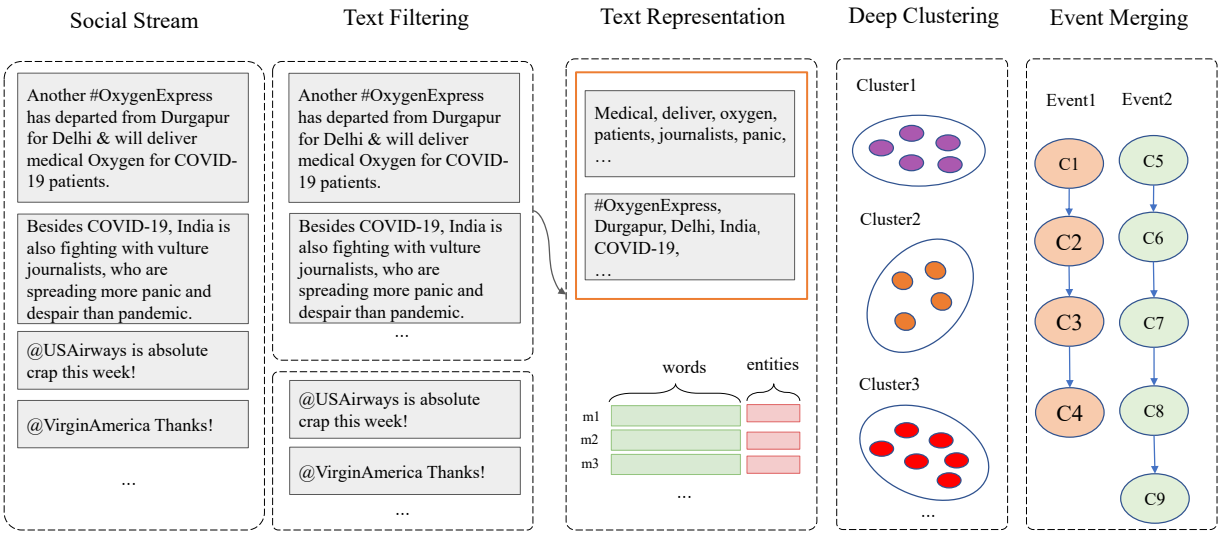

Fig. 1. Event detection framework for social stream

TABLE I
GLOSSARY OF NOTATIONS.

Notation	Description
M;m	Social stream;Social message
c_i;C	Cluster i;Set of clusters
e_i;E	Event i; Set of events
$\boldsymbol{R}(\boldsymbol{e})$; $\boldsymbol{R}(\boldsymbol{w})$	The representation of entities;The representation of words
$\boldsymbol{D}$	Euclidean distance
$N_{Eps}(p)$	The Eps-neighborhood of a point p
Δ	The threshold of event merging
ζ	The threshold of event linking

Definition 2.2: An **event** is a significant thing occurring at a specific place and time and involves people interacting with other people or objects [20].

An event is usually composed of a set of messages reporting on it. We consider an event $E = \{m_0, ..., m_i\}$ as a sequence of correlated social messages. For example, social media's tweet about "Texas freeze" is an influential event in the real world. It consists of sequential news describing different aspects of the event such as snowstorms, power outages, rescue efforts, and so on. Note that we assume each social message belongs to at most one event.

Definition 2.3: Given a social stream M, the task of **event detection** is to discover social messages reporting on the same event and divide them into event-centric clusters $\{c_1, c_2, ..., c_i, ...\}$.

III. MODEL

In this section, we describe the framework of our proposed online event detection method to automatically group tweets according to the events they report on.

A. Text Filtering

With the rapid development of the Mobile Internet, social networks have gradually become an indispensable part of people's lives. In addition to sharing their opinions about the ongoing events on social media, users also post tweets such as personal updates. These noisy data bring great difficulties to event detection, so it is necessary to separate event-related text and irrelevant text.

According to Definition 2.2, events are representable by a group of entities (e.g., person, organization, location and other). For example, the 92nd Academy Awards show can be represented by the nominated actors, actresses, and films that are being discussed. Hence, we can employ entities to filter irrelevant messages. In addition, tweets with many URL links or user mentions lack enough information, which can be filtered out. Table II shows a strict filtering paradigm, which is defined to remove the meaningless tweets.

Firstly, we employ the regular expression to match user mentions, hashtags or URL links, and we discard tweets which meet the filtering paradigm a. Then, we use Stanza [21] for named entity recognition to filter out messages without organization, person, or location. After that, we segment the text and remove stop words, and filter out the text with the number of words less than 6.

B. Text Representation

Unlike long texts, one piece of short text only contains few sentences or even just a few words. Sparseness and brevity are two inherent characteristics of such short texts. Our solution to this problem is to create two vector representations $\boldsymbol{R}(\boldsymbol{e})$, $\boldsymbol{R}(\boldsymbol{w})$ for each tweet. The $\boldsymbol{R}(\boldsymbol{e})$ consists of the named entities(person, organization and location) and hashtags in tweets, the $\boldsymbol{R}(\boldsymbol{w})$ consists of all terms in the tweets (with stop words removed).

It is common for many named entities in tweets to represent the same real-world entity. For example, "Donald Trump", "Donald John Trump" and "#Trump" all refer to the 45th President of the United States. For interested events, we build

TABLE II
PARADIGM IN THE TEXT FILTERING

Filtering Paradigm	Examples
a.The number of hashtag words, mentioned users or url is greater than 3.	@mranti @bitinn @MJTVHoPin https://t.co/Jwn2tXmyzF.
b.The number of named entities(person,organization, location) is less than 1.	First and foremost is #BustTheFilibuster.It's time to go.
c.The number of text segmentation is less than 6.	Go Jack! You are awesome.

a database to store the relationship between entities. Then we replace the entities pointing to the same thing with the same terms (e.g., "Academy Awards" is replaced by "Oscars"). Finally, we take an entity as a whole word and learn the embedding using TF-IDF.

After vectorizing the incoming tweets within the same period, we calculate the Euclidean distance matrix between their entities $\boldsymbol{D}(\boldsymbol{e})$ and the Euclidean distance matrix between words $\boldsymbol{D}(\boldsymbol{w})$ respectively. Then the two matrices are unified into one matrix according to $Eq.(1)$. Here α is the weight coefficient, and the general value is greater than 1.

$$\boldsymbol{D} = \alpha * \boldsymbol{D}(\boldsymbol{e}) + \boldsymbol{D}(\boldsymbol{w}) \tag{1}$$

C. Deep Clustering

Aiming at the problem that DBSCAN algorithm [22] can' t tackle the various density datasets correctly, we propose a deep clustering method, which can automatically detect noisy events and further process these events. For incoming tweets, we employ deep clustering to detect events as shown in Algorithm 1.

After employing DBSCAN, in order to measure the internal noise of an event cluster, we define the representative point here. If there are the most unvisited core points in an Eps-neighborhood of a core point p, p is the representative point, and p and the core points in its Eps-neighborhood are marked as visited. In a cluster, the more representative points, the noisier the cluster.

Algorithm 1: Deep Clustering

Input: $M = \{m_0, m_1, m_2, ...\}$: Social stream,
ϵ:Radius, $minPts$:Density threshold,
δ:Representative point threshold.
Output: Event clusters: $\{c_0, c_1, c_2, ...\}$.

Initialize $Results = \{\}$
Initialize $NoisyCluster = [1]$
while $NoisyCluster \neq \emptyset$ **do**
  if $NoisyCluster[0] \neq 1$ **then**
    $M = NoisyCluster[0]$
  for $m_0, m_1, m_2, ...$ **do**
    calculate the vector representation $\boldsymbol{R}(\boldsymbol{e})$ and $\boldsymbol{R}(\boldsymbol{w})$
  calculate the distance matrix $\boldsymbol{D}$
  $\{c_0, c_1, c_2, ...\} = DBSCAN(\boldsymbol{D}, \epsilon, minPts)$
  $K = len(\{c_0, c_1, c_2, ...\})$
  Initialize $Counter = []$
  for $i = 0, 1, ...K-1$ **do**
    Initialize the unvisited core point set of c_i: Γ
    Initialize the representative point set: $\Omega = \emptyset$
    while $\Gamma \neq \emptyset$ **do**
      calculate $|N_{Eps}(q)|, \forall p \in N_{Eps}(q), p \in \Gamma$
      select point $q, q = max(|N_{Eps}(q_j)|)$
      $\Omega.add(q)$
      $\Gamma.remove(N_{Eps}(q))$
    $Couter.append(|\Omega|)$
  for $i = 0, 1, ...K-1$ **do**
    if $Counter[i] \geq \delta$ **then**
      $NoisyCluster.append(c_i)$
    else
      $Results.add(c_i)$
  del $NoisyCluster[0]$
return $Results$

D. Event Merging

After an event happens, a large number of relevant tweets usually come out in a long time. However, due to the real-time nature of social networks, our clustering method detects events in different time slice, and these individual sub-events can't effectively describe the event itself. Therefore, it is necessary to merge clusters that are sub-events of an event after deep clustering.

Specifically, each cluster c_i has a set of additional information $< ID_i, T_i, Key_i, L_i >$. The ID_i is the unique identifier of c_i. The T_i is the temporal information of c_i and we use the earliest published time of the tweet in c_i. We use the textrank algorithm [23] to extract keywords from c_i as Key_i. The L_i is the ID of the cluster linked to c_i and we initialize it to the ID_i. After getting the additional information, we incrementally merge each time-slice sub-events according to the given rules.

Given a new cluster c_{new} after deep clustering, we first use keywords to query clusters on the same day from the database, and get clusters $\{c_1, c_2, ...\}$ that contain at least one same keyword. Then we calculate the similarity between c_{new} and $\{c_1, c_2, ...\}$ as $Eq.(2)$. If the similarity between the most similar cluster c_{old} and c_{new} is larger than a merging threshold Δ, we merge the two clusters into c_M and update the additional information $< ID_M, T_M, Key_M >$. The ID_M is ID_{old} and T_M is the earlier one between T_{old} and T_{new}. To get Key_M, we use textrank again to extract keywords from the merged cluster c_M. Otherwise, we save the cluster c_{new} in the database.

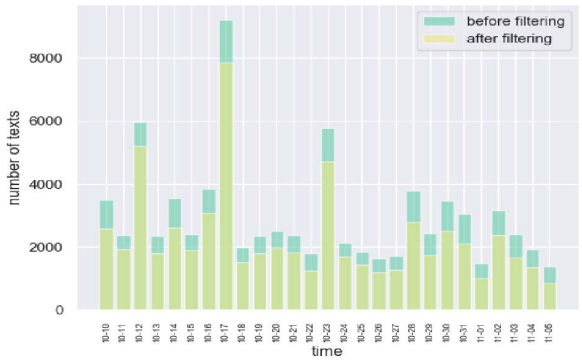

Fig. 2. Daily tweet number distribution

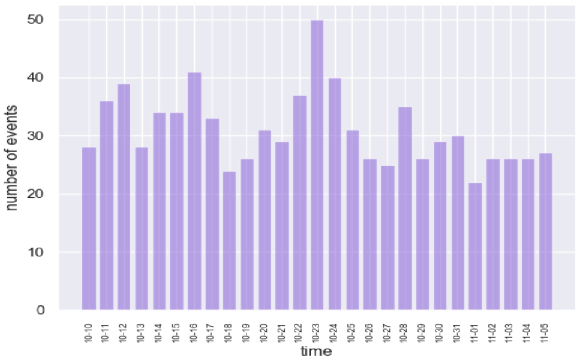

Fig. 3. Daily event number distribution

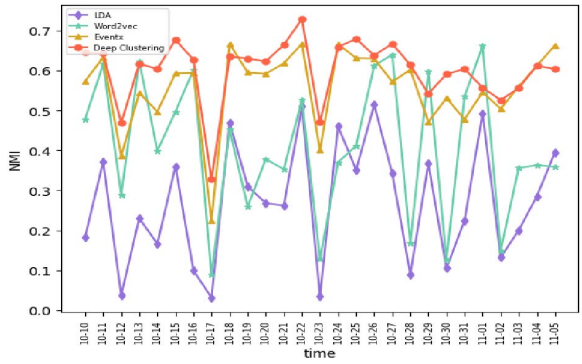

Fig. 4. NMI

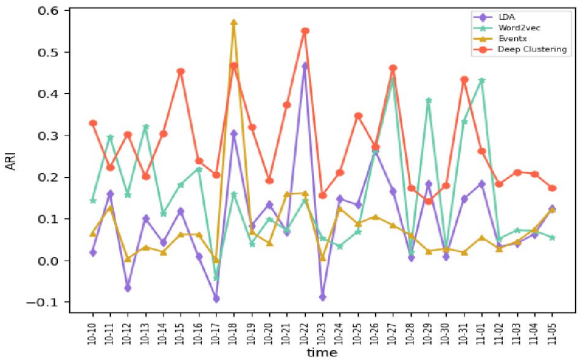

Fig. 5. ARI

$$Sim(c_i, c_j) = \frac{\boldsymbol{R}(Key_i) \cdot \boldsymbol{R}(Key_j)}{|\boldsymbol{R}(Key_i)| \cdot |\boldsymbol{R}(Key_j)|} \tag{2}$$

In addition, we query the database for clusters that contain at least one keyword of c_{new} within a certain time window (e.g., a week). After calculating the similarity, if the similarity between the most similar cluster c_p and c_{new} is larger than a linking threshold ζ, we link c_{new} with the previous cluster c_p. When c_{new} is successfully linked, we replace L_{new} with ID_p. By linking clusters where appropriate, we form cluster chains. Finally, a cluster chain is an event, where there are no events referring to the same event.

IV. Experiment

In this section, we first introduce the experimental setups, including the dataset, baselines, and the evaluation metrics. To evaluate the effectiveness of our proposed event detection method, we then compare our method with various baselines.

A. Datasets and evaluation metrics

We conduct experiments on a large-scale, publicly available Twitter dataset [24]. They released relevance judgements containing over 150,000 tweets, covering more than 500 events. We select 68,200 manually labeled tweets, which related to 401 event classes and spread over a period of four weeks. In addition, we add 14,600 event- irrelevant tweets to form the final experimental dataset.

According to the given rules as shown in Table II, 61.1% of tweets not related to events are filtered, 17.4% of related tweets are filtered, and the remaining 61996 tweets contain 394 events. The detailed statistics of the datasets are shown in Fig2 and Fig3.

We use two quality metrics for this evaluation: Normalized Mutual Information(NMI) [25] and Adjusted Rand Index(ARI) [26]. NMI measures the amount of information one can extract from the distribution of the predictions regarding the distribution of the ground-truth labels. ARI considers all prediction-label pairs and counts pairs that are assigned in the same or different clusters, and ARI also accounts for chance.

Specifically, for a set of clusters $C = \{C_1, C_2, .., C_i\}$ and events $E = \{E_1, E_2, \ldots, E_j\}$, where each C_i and E_k is a set of tweets, and n is the total number of tweets.

NMI is defined as follows:

$$NMI(C, E) = \frac{I(C, E)}{(H(E) + H(C))/2} \tag{3}$$

Where

$$I(C, E) = \sum_i \sum_j \frac{|c_i \cap e_j|}{n} \log \frac{n * |c_i \cap e_j|}{|c_i| * |e_j|} \tag{4}$$

$$H(C) = -\sum_i \frac{|c_i|}{n} \log \frac{|c_i|}{n} \tag{5}$$

$$H(E) = -\sum_{j} \frac{|e_j|}{n} \log \frac{|e_j|}{n} \tag{6}$$

ARI is defined as follows:

$$ARI = \frac{RI - Expected(RI)}{max(RI) - Expected(RI)} \tag{7}$$

Where

$$RI = \frac{TP + TN}{TP + FP + FN + TN} \tag{8}$$

B. Baselines

- **LDA [27]:** It is a generative statistical model for text documents that learns representations for documents as distributions over word topics, and allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar.
- **Word2vec [28]:** It uses the average of the pre-trained Word2vec embeddings of all the words in a message as its representation.
- **EventX [29]:** It's a fine-grained event detection method based on community detection and is applicable to the online scenario.

C. Experimental Setting and Implementation

For LDA, we assign each topic is considered as an event and we set the total number of topics to 40. For Word2vec, we use the pre-trained 200-d GloVe [30] vectors. For EventX, we adopt the hyperparameters as suggested in the original paper [29]. For a fair comparison, after we obtain message representations from the other models and the message similarity matrix except EventX, we leverage DBSCAN clustering.

D. Experimental results

Here, in order to measure the effect of the model in the streaming environment, we split the dataset by dates to construct a social stream. Fig4 and Fig5 show the statistics of the resulting social stream. None of these models incorporate event merging. Table III summarizes the average social event detection results in NMI and ARI respectively. We can observe that the performance of our approach is better than the baseline methods in terms of NMI and ARI. In addition, the performance of our proposed model is more stable.

TABLE III
THE EVALUATION RESULTS ON THE DATASET

Methods	NMI	ARI
LDA	.27±.02	.10±.01
Word2vec	.41±.01	.15±.02
EventX	.55±.01	.08±.01
Deep Clustering	**.60±.00**	**.28±.01**

We also investigated the influence of the similarity threshold Δ. Clustering performance of our proposed method with different values of Δ on the dataset are illustrated in Fig6. As we can see, NMI and ARI are improved to different degrees

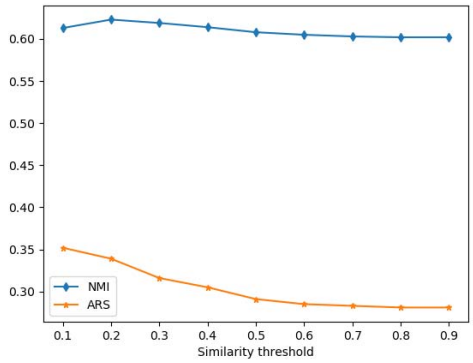

Fig. 6. Performance of our method with different values of Δ

compared with those without event merging when $\Delta < 0.5$. The NMI increases by two percentage points, and the ARI increases by two to seven percentage points. Besides, the number of daily event clusters is effectively reduced, which is closer to the actual number of events. For example, the number of events on 2012/10/10 is 28. When $\Delta = 0.2$, the number of clusters is 60 while the number of clusters is 155 without event merging.

V. Conclusion

In this study, we study the problem of event detection from the noisy content in social media. To tackle this problem, we propose a general framework for real-time event detection. In our method, we propose a novel density clustering algorithm, which can automatically detect noisy events and further process these events. Experimental results show that our proposed deep clustering algorithm achieves significant improvement over baselines.

In future work, we will leverage heterogeneous information network framework for modeling short texts to better incorporate the rich semantics and structural information contained in the social data.

Acknowledgment

The authors of this paper were supported by the National Key Research and Development Program of China under the Grant No.2018YFC0830804, and S&T Program of Hebei through grant 21340301D.

References

[1] H. Peng, M. Bao, J. Li, M. Z. A. Bhuiyan, Y. Liu, Y. He, and E. Yang, "Incremental term representation learning for social network analysis," *Future Generation Computer Systems*, vol. 86, pp. 1503–1512, 2018.

[2] Q. Mao, X. Li, H. Peng, J. Li, D. He, S. Guo, M. He, and L. Wang, "Event prediction based on evolutionary event ontology knowledge," *Future Generation Computer Systems*, vol. 115, pp. 76–89, 2021.

[3] H. Peng, R. Yang, Z. Wang, J. Li, L. He, P. Yu, A. Zomaya, and R. Ranjan, "Lime: Low-cost incremental learning for dynamic heterogeneous information networks," *IEEE Transactions on Computers*, 2021.

[4] H. Peng, J. Li, Q. Gong, Y. Ning, S. Wang, and L. He, "Motif-matching based subgraph-level attentional convolutional network for graph classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5387–5394.

[5] H. Peng, J. Li, Y. Song, and Y. Liu, "Incrementally learning the hierarchical softmax function for neural language models," in *Proceedings of the AAAI*. AAAI Press, 2017, p. 3267–3273.

[6] H. Peng, H. Wang, B. Du, M. Z. A. Bhuiyan, H. Ma, J. Liu, L. Wang, Z. Yang, L. Du, S. Wang *et al.*, "Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting," *Information Sciences*, vol. 521, pp. 277–290, 2020.

[7] D. Pohl, A. Bouchachia, and H. Hellwagner, "Social media for crisis management: clustering approaches for sub-event detection," *Multimedia tools and applications*, vol. 74, no. 11, pp. 3901–3932, 2015.

[8] 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 Transactions on Knowledge Discovery from Data*, vol. 15, no. 5, pp. 1–33, 2021.

[9] H. Peng, J. Li, S. Wang, L. Wang, Q. Gong, R. Yang, B. Li, P. Yu, and L. He, "Hierarchical taxonomy-aware and attentional graph capsule rcnns for large-scale multi-label text classification," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[10] M. Cordeiro and J. Gama, "Online social networks event detection: a survey," in *Solving Large Scale Learning Tasks. Challenges and Algorithms*. Springer, 2016, pp. 1–41.

[11] 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," *arXiv preprint arXiv:1906.04580*, 2019.

[12] Y. Cao, H. Peng, J. Wu, Y. Dou, J. Li, and P. S. Yu, "Knowledge-preserving incremental social event detection via heterogeneous gnns," *arXiv preprint arXiv:2101.08747*, 2021.

[13] C. C. Aggarwal and K. Subbian, "Event detection in social streams," in *Proceedings of the 2012 SIAM international conference on data mining*. SIAM, 2012, pp. 624–635.

[14] Y. Liu, H. Peng, J. Guo, T. He, X. Li, Y. Song, J. Li *et al.*, "Event detection and evolution based on knowledge base," *Proc. KBCOM*, 2018.

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

[16] M. Z. A. Bhuiyan and J. Wu, "Event detection through differential pattern mining in internet of things," in *2016 IEEE 13th international conference on mobile ad hoc and sensor systems (MASS)*. IEEE, 2016, pp. 109–117.

[17] Y. He, J. Li, Y. Song, M. He, H. Peng *et al.*, "Time-evolving text classification with deep neural networks." in *IJCAI*, 2018, pp. 2241–2247.

[18] W. Yu, J. Li, M. Z. A. Bhuiyan, R. Zhang, and J. Huai, "Ring: Real-time emerging anomaly monitoring system over text streams," *IEEE Transactions on Big Data*, vol. 5, no. 4, pp. 506–519, 2017.

[19] H. Peng, J. Li, H. Yan, Q. Gong, S. Wang, L. Liu, L. Wang, and X. Ren, "Dynamic network embedding via incremental skip-gram with negative sampling," *Science China Information Sciences*, vol. 63, no. 10, pp. 1–19, 2020.

[20] P. Over, G. Awad, M. Michel, J. Fiscus, G. Sanders, B. Shaw, W. Kraaij, A. F. Smeaton, and G. Quéot, "Trecvid 2012-an overview of the goals, tasks, data, evaluation mechanisms and metrics," 2013.

[21] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, "Stanza: A python natural language processing toolkit for many human languages," *arXiv preprint arXiv:2003.07082*, 2020.

[22] 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 *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[23] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text." *Emnlp*, pp. 404–411, 2004.

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

[25] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Transactions on neural networks*, vol. 20, no. 2, pp. 189–201, 2009.

[26] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *The Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, 2010.

[27] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[28] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[29] B. Liu, F. X. Han, D. Niu, L. Kong, K. Lai, and Y. Xu, "Story forest: Extracting events and telling stories from breaking news," *ACM Transactions on Knowledge Discovery from Data*, vol. 14, no. 3, pp. 1–28, 2020.

[30] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the EMNLP*, 2014, pp. 1532–1543.