Reinforced, Incremental and Cross-lingual Event Detection From Social Messages

Hao Peng, Ruitong Zhang, Shaoning Li, Yuwei Cao, Shirui Pan, Philip S. Yu, *Fellow, IEEE*

Abstract—Detecting hot social events (e.g. political scandal, momentous meetings, natural hazards, etc.) from social messages is crucial as it highlights significant happenings to help people understand the real world. On account of the streaming nature of social messages, incremental social event detection models in acquiring, preserving, and updating messages over time have attracted great attention. However, the challenge is that the existing event detection methods towards streaming social messages are generally confronted with ambiguous events features, dispersive text contents, and multiple languages, and hence result in low accuracy and generalization ability. In this paper, we present a novel rein**F**orced, **i**ncremental and cross-li**n**gual social **Event** detection architecture, namely **FinEvent**, from streaming social messages. Concretely, we first model social messages into heterogeneous graphs integrating both rich meta-semantics and diverse meta-relations, and convert them to weighted multi-relational message graphs. Secondly, we propose a new reinforced weighted multi-relational graph neural network framework by using a Multi-agent Reinforcement Learning algorithm to select optimal aggregation thresholds across different relations/edges to learn social message embeddings. To solve the long-tail problem in social event detection, a balanced sampling strategy guided Contrastive Learning mechanism is designed for incremental social message representation learning. Thirdly, a new Deep Reinforcement Learning guided density-based spatial clustering model is designed to select the optimal minimum number of samples required to form a cluster and optimal minimum distance between two clusters in social event detection tasks. Finally, we implement incremental social message representation learning based on knowledge preservation on the graph neural network and achieve the transferring cross-lingual social event detection. We conduct extensive experiments to evaluate the **FinEvent** on Twitter streams, demonstrating a significant and consistent improvement in model quality with 14%-118%, 8%-170%, and 2%-21% increases in performance on offline, online, and cross-lingual social event detection tasks.

Index Terms—Social event detection, graph neural network, reinforcement learning, contrastive learning, DBSCAN

1 Introduction

Social events are occurrences of real-world unusual happenings that involve specific time, location, person, content, etc. [1], while are widely spread and discussed in social networks and media. For instance, nine people including the retired professional basketball player Kobe Bryant, his 13-year-old daughter Gianna, baseball coach John Altobelli, and five other passengers were killed in the *2020 Calabasas helicopter crash*[1] happened on January 26, 2020. Social media platforms (Twitter, Weibo, Facebook, Tumblr, Telegram, etc.) have become the main source of official and personal social news. These platforms attract large numbers of users because they provide convenient ways for people to share and seek views regarding social events in real-time. Detecting social events from mass social messages is beneficial. On the one hand, storing daily news and social messages in the forms of events can make the information storage more organized [2] and enrich the information recommendation [3]. On the other hand, social event detection can be applied in abundant real applications, such as Public Opinion Analysis [4], Sentiment Analysis [5], Enterprise Risk Management [6], Political Election Forecast [7], etc. Technically, social event detection focuses on learning highly effective event-related clusters modeled from a large number of real social messages.

However, the social event detection task is more challenging than traditional text mining or social network mining, since a general social event is a meaningful and influential combination of social messages in an open domain. Social events often contain some event-related heterogeneous elements, such as location, person, organization, relations, date and time, keywords, etc. Although there is heterogeneous information network (HIN) [8] based social messages models [4], [9], [10], how to learn more discriminative embedding of social messages is still an intractable problem to be solved. Especially, the contents of social messages are always overlapping, redundant and discrete, where the noisy nature of messages stream makes traditional outlier detection technologies [11]–[14] inappropriate for the semantically rich event detection task. Hence, the first challenge is still how to model social messages and design a more discriminative and explanatory social message embedding framework. In addition to the above complex semantics, the event detection task has characteristics of Long Tail distribution [15]. Generally, it is limited by the cost of social data collection and social event annotation. The number of messages (samples) contained in each event is relatively imbalanced [9]. The long-tail problem contributes to the second

- *Hao Peng, Ruitong Zhang, and Shaoning Li are with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China. E-mail: {penghao, rtzhang, lishaoning}@buaa.edu.cn.*
- *Yuwei Cao and Philip S. Yu are with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA. E-mail:{ycao43, psyu}@uic.edu.*
- *Shirui Pan is with the Department of Data Science and AI, Faculty of IT, Monash University, Clayton, VIC 3800, Australia. E-mail: shirui.pan@monash.edu.*

Manuscript received July 2021, major revised November 2021, minor revised January 2022, accepted January 2022. (Corresponding author: Hao Peng)

1. https://en.wikipedia.org/wiki/2020_Calabasas_helicopter_crash

challenge, which is the degradation of the performance of detection approaches and poor generalization. More practical methods of social event detection are usually using streaming clustering detection technology rather than event classification technology. Besides, practical social event detection methods also need to achieve incremental detection on streaming messages [12], [16] and even cross-lingual detection [17], [18]. On the one hand, both social messages and social events have time attributes, and the number of social events will also increase in social message streams. Different from the existing online social event detection methods [1], [13], [14], [19]–[22] based on deterministic patterns, such as clique, dense graph, keywords, topic, and template, etc., a semantic incremental event detection framework with better generalization performance is worthy of the effort. On the other hand, cross-lingual messages lead to inconsistencies in the semantic embedding space of the underlying words or entities [23], which also brings direct usability difficulties for the event detection model (although there are third-party translation tools or translation alignment models to solve these problems, these difficulties are relatively objective). Therefore, the third challenge is how to implement cross-lingual social event detection, and even generalize to low-resource language messages data.

To tackle the above challenges, we present a novel rein**F**orced, **i**ncremental, and cross-li**n**gual social **Event** detection architecture, namely **FinEvent**, from streaming social messages. The architecture mainly contains 5 modules of preprocessing, message embedding, training, detection, and transferring. Firstly, we also leverage heterogeneous information networks (HINs) [8] to organize event-oriented elements and relations of various types into one unified graph structure. Different from the previous methods [4], [9], [10] converting heterogeneous graph to homogeneous graph by using meta-path instances, we propose a weighted multi-relational graph for the first time to model the association between social messages, reserving the number of meta-path instances as different weight of edge/relation. Secondly, we propose a novel **M**ulti-**a**gent **r**einforced weighted multi-relational **G**raph **N**eural **N**etwork framework, namely **MarGNN**, to learn the social message embedding with reinforcement learning (RL). Concretely, we harness the power of GNN to learn representations from the semantic and structural information contained in the social messages and use the multi-agent Actor-critic algorithm (AC) [24] to learn the optimal numbers/thresholds for each relation, in order to guide both intra-relation and inter-relation messages aggregations, respectively. Third, to solve the long-tail problem in social event detection, a **Ba**lanced **s**ampling strategy based **C**ontrastive **L**earning mechanism, namely **BasCL**, is designed to guide the training of the framework. Then, we periodically update messages to keep an up-to-date embedding space, and implement incremental social event detection based on a well-designed knowledge preservation technology on the MarGNN. Fourth, we also design a new **D**eep **R**einforcement **L**earning(DRL) guided DBSCAN model, namely **DRL-DBSCAN** based on the learned social messages embeddings to achieve social event clustering detection tasks automatically. Through DRL-DBSCAN, we use the Twin Delayed Deep Deterministic policy gradient algorithm (TD3) [25] to learn optimal parameters of $minPts$ - the minimum number of samples required to form a cluster, and ϵ - the minimum distance between two clusters, in the social streams. In the end, a new **Cr**oss-**l**ingual social **m**essage **e**mbedding method, namely **Crlme**, is presented, transferring the parameters of MarGNN to improve the performance of embedding on target-language messages (non-English). Based on the Crlme method, we also implement the cross-lingual social event detection.

Our preliminary work appeared in the proceedings of Web Conference 2021 [10]. The journal version in this paper has extended the original parameters-preserved incremental event detection model KPGNN to a reinforced, incremental and cross-lingual social event detection architecture. This full-version involves several improvements in upgrading the methodology and the frame structure of the proposed architecture. In terms of model upgrades, different from transforming the HINs into homogeneous graphs, we first introduced the weighted multi-relational graphs to preserve richer structural and statistical features from heterogeneous relations in graphs. Second, we proposed the novel MarGNN framework to learn more discriminative social message embedding. Specifically, by learning the optimal preserving thresholds with RL, MarGNN reasonably retains and integrates the most top-p valuable semantic and structural information from each relation. Third, we proposed the new DRL-DBSCAN- differing from the heuristic K-Means or DBSCAN methods, to realize social event clustering detection tasks without manual parameters. Fourth, we presented the new Crlme method to achieve the cross-lingual social event detection. In the experiments, more state-of-the-art performances are presented and analyzed. Thorough and deep analyses are presented to demonstrate the effectiveness and explanation of FinEvent[2].

In summary, the contributions of this paper are summarized as follows:

- A novel social event detection architecture, named as FinEvent, is presented. FinEvent can realize offline, online, and cross-lingual social event detection by utilizing both social messages representation learning framework MarGNN, cross-lingual social message representation learning method Crlme, and social event clustering detection model DRL-DBSCAN.
- A new multi-agent reinforced weighted multi-relational graph neural network framework MarGNN is proposed to learn more discriminative social message embedding. It provides an insightful explanation from the perspective of the importance of different relations in social message aggregation.
- A balanced sampling-based contrastive learning strategy BasCL is designed as the fundamental objective function of social message representation learning to solve the long-tail problem in social event detection.
- A new deep reinforcement learning guided density-based spatial clustering model DRL-DBSCAN is proposed to achieve automatical social event clustering detection. It is the first work to learn optimal parameters of DBSCAN by deep reinforcement learning without offline maintenance or manual experience.

2. The source code and data of this work is publicly available at https://github.com/RingBDStack/FinEvent.

• A new cross-lingual social message representation learning method Crlme is presented to improve the performance of low-resource language message embedding. It provides a low-cost transfer learning application from the perspectives of cross-lingual social event detection.

• Extensive experiments and analyses are implemented on Twitter streams, demonstrating that FinEvent outperforms the existing SOTA social event detection methods in terms of effectiveness and explanation. Even incremental and cross-lingual social event detection tasks are released in the convenient way of knowledge preservation and knowledge transferring the applicability of FinEvent.

2 Problem Formulation and Notations

In this section, we summarize the main notations in Table 1, and formalize the definition of *Social Stream*, *Social Event*, *Heterogeneous Information Network*, *Weighted Multi-Relational Message Graph*, *Social Event Detection*, *Incremental Social Event Detection*, and *Cross-lingual Social Event Detection* as follows.

Definition 2.1. The **social stream** $S = M_0, ..., M_{i-1}, M_i, ...$ is a continuous and temporal sequence of blocks of social messages, where M_i is a *message block* that contains all the messages which arrive during time period $[t_i, t_{i+1})$. We denote a *message block* M_i as $M_i = \{m_j | 1 \leq j \leq |M_i|\}$, where $|M_i|$ is the total number of messages contained in the M_i, and m_j is one message.

Definition 2.2. A **social event** $e = \{m_i | 1 \leq i \leq |e|\}$ is a set of correlated social messages that discuss the same real-world happening event. Note that we assume each social message belongs to at most one event.

For example, the catastrophic fire at Notre Dame Cathedral in Paris in April 2019 triggered widespread discussions on Twitter. Here, *Notre Dame Cathedral fire*[3] can be defined as an event, which contains a series of related tweets (similar to the raw messages $m1$, $m2$, $m3$ in Fig. 2). In addition to hot events, our work also pays attention to less influential social events, such as *the Spanish Tomato War*[4], *the poisoning of the Golden Eagle in Aberdeenshire*[5] and so on.

Definition 2.3. A heterogeneous information network **HIN** is a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ stands for collections of nodes with various types of event-related elements in social events, and $\mathcal{E}$ stands for the collections of edges between the messages and corresponding event-oriented elements.

Definition 2.4. We define a **weighted multi-relational message graph** as $\mathcal{G} = \{\mathcal{M}, \mathcal{X}, \mathcal{W}, \{\mathcal{E}_r^w\}|_{r=1}^R\}$, where $\mathcal{M}$ is a set of nodes $\mathcal{M} = \{m_1, ..., m_i, ..., m_n\}$ and n is the number of nodes. Each node refers to a message m_i and has a d-dimension feature vector denoted as $\mathbf{x}_{m_i} \in \mathbb{R}^d$. $\mathcal{X} = \{\mathbf{x}_{m_1}, ..., \mathbf{x}_{m_i}, ...\mathbf{x}_{m_n}\}$ is a set of all node features. $e_{i,j}^r = (m_i, m_j) \in \mathcal{E}_r^w$ is an edge/connect between message m_i and m_j with a relation $r \in \{1, ..., R\}$ and given edge weight $w_{i,j}^r \in \mathcal{W}$. Note that an edge can be associated with multiple relations and there are R different types of relations.

3. https://en.wikipedia.org/wiki/Notre-Dame_de_Paris_fire
4. https://en.wikipedia.org/wiki/La_Tomatina
5. https://www.bbc.co.uk/news/uk-scotland-north-east-orkney-shetland-57000780

TABLE 1: Glossary of Notations.

Notation	Description
$S; M; G$	Social stream; Message block; Event based heterogeneous information network
m	Message or message as a node type
$e; E$	Event; Set of events
w	The window size for maintaining the model
$o; e; u$	Word; Named entity; User (node types)
$\boldsymbol{A}_{mr}$	The adjacency matrix between node m and relation r
$\mathcal{G}$	Weighted Multi-relational Message graph
N	The total number of messages in $\mathcal{G}$
$\boldsymbol{X}$	The initial feature vectors of the messages in $\mathcal{G}$
$\mathcal{E}(\boldsymbol{X}, \boldsymbol{A})$	GNN that embeds the messages in $\mathcal{G}$
$l; L$	GNN layer number; Total number of layers
$\boldsymbol{h}_{m_i}^{(l)}$	The representation of m_i at the l-th layer
$\boldsymbol{h}_{m_i}$	The final representation of m_i
$A_{agg}, S_{agg}, R_{agg}$	Action space, state space and reward function of Multi-agent Reinforcement Learning
$a_{agg,r}^{(k)}, s_{agg,r}^{(k)}, r_{agg,r}^{(k)}$	Action, state and reward of Multi-agent Reinforcement Learning at epoch k
p_r	Preserving threshold under relation r;
$\mathcal{L}_{agg,r}$	Actor loss under relation r;
$b; B$	Mini-batch number; Total number of mini-batches
$\{m_b\}$	A set of messages in the b-th mini-batch
$\tilde{\boldsymbol{h}}_{m_i}$	The corrupted representation of m_i
m_i+	A message in the same class as m_i
m_i-	A message that is not in the same class as m_i
$\mathcal{L}_t$; $\mathcal{L}_p$	Triplet loss; Global-local pair loss
$\boldsymbol{s}$	The summary vector of $\mathcal{G}$
$A_{clu}, S_{clu}, R_{clu}$	Action space, state space and reward function of Deep Reinforcement Learning
$a_{clu,r}^{(\tau)}, s_{clu,r}^{(\tau)}, r_{clu,r}^{(\tau)}$	Action, state and reward of Deep Reinforcement Learning at episode τ
$\mathcal{L}_{clu}$	Value network loss;

Definition 2.5. Given a message block M_i, a **social event detection** algorithm learns a model $f(M_i; \theta) = E_i$, such that $E_i = \{e_k | 1 \leq k \leq |E_i|\}$ is a set of events contained in M_i. Here, θ denotes the parameter of f.

Definition 2.6. Given a social stream S, an **incremental social event detection** algorithm learns a sequence of event detection models $f_0, ..., f_{t-w}, f_t, f_{t+w}, ...$, such that $f_t(M_i; \theta_t, \theta_{t-w}) = E_i$ for all message blocks in $\{M_i | t+1 \leq i \leq t+w\}$. Here, $E_i = \{e_k | 1 \leq k \leq |E_i|\}$ is a set of events contained in the message block M_i, w is the window size for updating the model, θ_t and θ_{t-w} are the parameters of f_t and f_{t-w}, respectively. Note that f_t extends the knowledge of its predecessor f_{t-w} by depending on θ_{t-w}. In particular, we call f_0 which extends no previous model as the *initial model*.

Definition 2.7. Given a social event detection algorithm $f_E(M_i; \theta) = E_i$, $M_i \in S_E$ in English social stream, **cross-lingual social event detection** algorithm learn a upgraded model $f_{NoE}(M_j; \bar{\theta}) = E_j$ from $f_E(M_i; \theta)$, where $E_j = \{e_k | 1 \leq k \leq |E_j|\}$ is also a set of events contained in no-English *message block* $M_j \in S_{NoE}$. Here, $\bar{\theta}$ denotes the parameter of the upgraded model f_{NoE} from f_E.

3 FinEvent Architecture

FinEvent performs social event detection in an incremental life-cycle mechanism. The design of FinEvent architecture aims at being consistent with the streaming nature of social messages and real situations in event detection. Concretely, we require a flexible architecture to **i).** deal with the streaming nature of social messages in an online manner; **ii).** continuously acquire, preserve and extend the message semantics; **iii).** handle the actual problems, especially the cross-lingual event detection task. In addition, FinEvent is also capable of realizing cross-lingual transfer. This section encompasses the different stages in the life-cycle of FinEvent, and gives a picture of how FinEvent operates cross-lingual transferring detection.

3.1 Life-cycle Mechanism

To cope with the streaming message challenge, FinEvent adopts the life-cycle mechanism (shown in Fig. 1), which comprises four stages, i.e., pre-training, single-language detection, cross-lingual detection, and maintenance denoted as Stage I, Stage II, Stage III, and Stage IV respectively in Fig. 1. In the pre-training stage, we store a small portion of social messages in advance, upon which the initial weighted multi-relational message graph is constructed. Then we leverage the processed graph to train the initial model. In the detection stage, we update the weighted multi-relational message graph with the incoming message block by inserting the new message nodes, their linkages with the existing message nodes, and the linkages within themselves, and reconstruct the weighted relation. Simultaneously, after the updating process, the pre-trained model is **directly** used to detect social events from unseen messages without a new round of training. In the maintenance stage, we remove the obsolete social messages in the message graph, and the model is maintained by continuing training using the updated graph. The maintained model in this life-cycle can be then used for the detection stage in the next life-cycle. With such a designed life-cycle mechanism, FinEvent is able to well adapt the streaming nature of social messages, and achieve the online manner of event detection.

3.2 Incremental Learning Framework

Aiming at the generalization challenges in incremental social event detection, we adopt incremental learning combined with the above life-cycle mechanism in order to help FinEvent continuously acquire, preserve and extend the semantic space, and make use of its advantages to deal with the actual problems.

Preliminarily, we require our architecture to efficiently organize and process various elements in the social streams for full utilization and effectively interpret these elements to discover underlying knowledge that would help event detection. In order to obtain the message embedding, we first leverage the HIN to model the streaming event-oriented social messages. Then we proceed to extract different relations in the constructed heterogeneous graph which are further converted to a series of weighted multi-relational homogeneous graphs according to their meta-path instances.

Secondly, FinEvent needs to effectively and efficiently update its embedding space when new messages arrive.

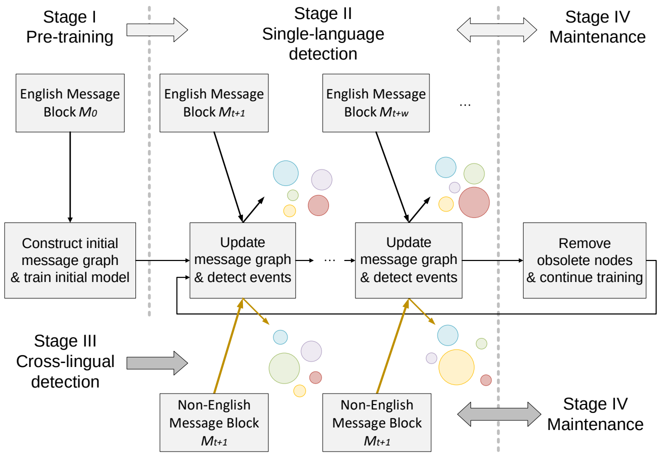

Stage I
Pre-training
Stage II
Single-language
detection
Stage IV
Maintenance
English Message Block M_0
English Message Block M_{t+1}
English Message Block M_{t+w}
...
Construct initial message graph & train initial model
Update message graph & detect events
Update message graph & detect events
Remove obsolete nodes & continue training
Stage III
Cross-lingual
detection
Non-English Message Block M_{t+1}
Non-English Message Block M_{t+1}
Stage IV
Maintenance

Fig. 1: Incremental life-cycle and cross-lingual transferring mechanisms in FinEvent architecture.

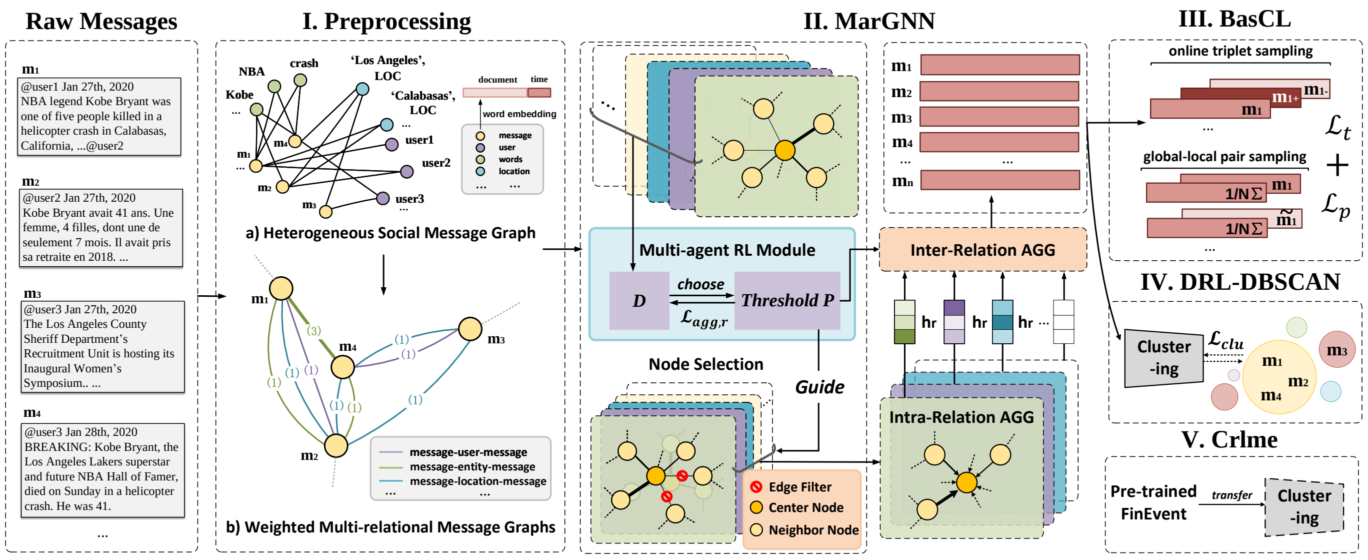
We accordingly design a reinforced graph neural network framework MarGNN in FinEvent based on the obtained multi-relational graph, which harnesses the great power of GNNs to tune parameters for social event detection purposes and preserve the helpful knowledge about social data. Combined with the life-cycle mechanism, our FinEvent can achieve continuous training in incremental learning mode instead of retaining from scratch.

3.3 Cross-lingual Transferring Mechanism

Inspired by the preserving and extending feature of FinEvent, which leverages the preserved parameters in MarGNN to infer events in the detection stage and extend MarGNN by resuming the training process using incoming data in the maintenance stage, we adopt such powerful inductive learning ability to solve the cross-lingual problem. As shown in the cross-lingual detection stage in Fig. 1, if MarGNN is trained using English messages, for instance, and the incoming messages are all in French or Arabic, we can still achieve the detection owing to the preserved parameters in MarGNN and continue to extend it for next life cycle. Hence, we obtain the dynamic FinEvent architecture and explore it for real-world application.

4 Reinforced Multi-relational Graph Neural Network Framework

Fig. 2 shows the proposed framework FinEvent with five important modules. For the original social messages, we first convert them into a weighted multi-relational graph (Sec. 4.1). Then we use multi-agent reinforcement learning to select neighbors for different relations, and obtain the aggregation of all messages through the weighted multi-relational graph neural network (Sec. 4.2). In order to balance sampling and cope with incremental detection scenarios, a strategy-based contrast learning mechanism is used to train the GNN (Sec. 4.3). Finally, we use the DBSCAN model guided by deep reinforcement learning for event clustering parameter debugging (Sec. 4.4). In addition, we also give cross-language detection in Sec. 4.5, overall algorithm procedures, and Algorithm 1 at the end of this section.


m₁
@user1 Jan 27th, 2020
NBA legend Kobe Bryant was one of five people killed in a helicopter crash in Calabasas, California, ...@user2

m₂
@user2 Jan 27th, 2020
Kobe Bryant avait 41 ans. Une femme, 4 filles, dont une de seulement 7 mois. Il avait pris sa retraite en 2018. ...

m₃
@user3 Jan 27th, 2020
The Los Angeles County Sheriff Department's Recruitment Unit is hosting its Inaugural Women's Symposium.. ...

m₄
@user3 Jan 28th, 2020
BREAKING: Kobe Bryant, the Los Angeles Lakers superstar and future NBA Hall of Famer, died on Sunday in a helicopter crash. He was 41.

Fig. 2: The architecture of the proposed FinEvent.

4.1 Weighted Multi-relational Social Message Graph

The heterogeneous social graph in Fig. 2 is a heterogeneous information network (HIN) extracted from messages and event-oriented elements in social streams. In order to prevent the loss of heterogeneous information between different types of event elements, FinEvent saves richer connection information by mapping HIN to a weighted multi-relational graph $\mathcal{G} = \{\mathcal{M}, \mathcal{X}, \mathcal{W}, \{\mathcal{E}_r^w\}|_{r=1}^R\}$ of message nodes. We define the nodes in the multi-relational graph as a series of message collections $\mathcal{M}$ with d-dimensional features $\mathcal{X}$ (contains the average value of pre-trained word embeddings of all words [26] and timestamp encoding). When messages share different types of event elements, edges belonging to different relations will be established respectively. The edges $e_{i,j}^r = (m_i, m_j) \in \mathcal{E}_r$ of message nodes m_i and m_j under relation r in the multi-relational graph follows:

$$e_{i,j}^r = min\left\{[A_{mr} \cdot A_{mr}^{\intercal}]_{i,j}, 1\right\}. \quad (1)$$

Here, A_{mr} is a submatrix of the adjacency matrix of the HIN heterogeneous graph, where the rows represent all information nodes, and the columns represent all event element nodes belonging to the relation r. ^{T} is the transpose of the matrix, and $min\{,\}$ takes the smaller of the two elements. In addition, in order to cope with the information loss between two message nodes with multiple common elements under the same relation, FinEvent introduces edge weights in the construction process of different relations. We define the weighted edges of the message nodes m_i and m_j under relation r in the graph $\mathcal{G}$ as $e_{i,j}^r = (m_i, m_j) \in \mathcal{E}_r^w$, where the edge weight $w_{i,j}^r \in \mathcal{W}$ is $\left[A_{mr} \cdot A_{mr}^{\intercal}\right]_{i,j}$.

4.2 Multi-agent Reinforced Aggregation in Multi-relational Graph Neural Network Framework (MarGNN)

4.2.1 Reinforced Neighbor Selection

Considering that there are some meaningless links between social information that affect message representation, we first sample each relation before aggregation to retain neighbors with high semantic and structural connections. Since different relations in the multi-relational graph have different degrees of impurities and collectively affect the embedding results, a collaborative learning method is needed to find the balance between different relations. Previous works use Bernoulli Multi-armed Bandit process [27] or attention mechanism [28] to select neighbors, but they are no longer applicable in increasing detection. For this reason, we introduce multi-agent reinforcement learning in framework MarGNN to guide each relation to perform Top-p neighbor sampling before aggregation (the node selection part of Fig. 2). Specifically, we first sort the neighbors of each under the relation r in ascending order according to the Euclidean distance, and then establish an agent for each relation as the selector that retains the preserving threshold $p_r \in [0, 1]$. When p_r is 1, all neighbors are reserved, and when p_r is 0, all neighbors are discarded.

Concretely, the agent of each relation will learn in the game how to find the balance between relations in the task of streaming social detection. This process is a Markov game of R agents, consisting of four elements $(N_{agg}, A_{agg}, S_{agg}, R_{agg})$, where N_{agg} is the number of agents, A_{agg} is the action space of the agents, S_{agg} is the state space of the agents, R_{agg} is the reward function of the agents. The specific details of the whole process are as follows:

- ***State:*** In view of the fact that the preserving thresholds of different relations jointly affect the final aggregation effect, we use the neighbor node representation $\boldsymbol{h}_m^{(k)}$ aggregating by the preserving thresholds of all relations to calculate the average weighted distance under one relation, so that each agent can take into account the influence of other relations. At epoch k, the state $s_{agg,r}^{(k)}$ which one agent observe under relation r is defined as:

$$s_{agg,r}^{(l)(k)} = \frac{1}{|N|} \sum_{i=1}^{N} \frac{\sum_{m_j \in \mathcal{N}_r^{(k)}(m_i)} w_{i,j}^{(r)} \cdot \mathcal{D}(\boldsymbol{h}_{m_i}^{(k)}, \boldsymbol{h}_{m_j}^{(k)})}{\sum_{m_j \in \mathcal{N}_r^{(k)}(m_i)} w_{i,j}^{(r)}}, \quad (2)$$

where $\mathcal{N}_r^{(k)}(m_i)$ is the set of all preserved neighbor nodes m_j of the central node m_i under relation r in the k-th epoch. $w_{i,j}^{(r)}$ is the weight of the edges of nodes m_i and m_j under the relation r.

• ***Action:*** The action $a_{agg,r}^{(l)(k)} \in A_{agg}$ of the agent is the preserving threshold $p_{agg,r}^{(k)}$ under the relation r in epoch k. Since $p_{agg,r}^{(k)} \in [0,1]$ has the highest accuracy, we use discrete actions to speed up the learning process.

• ***Reward:*** Since the goal of enhanced aggregation is to find the best aggregation scheme to obtain the best clustering performance of the message, we use Normalized Mutual Information (NMI) as the reward function to objectively measure the clustering effect. And define the reward $r_{agg,r}^{(k)}$ under relation r in epoch k as follows:

$$r_{agg,r}^{(k)} = NMI_k(|E_{true}|). \tag{3}$$

Here, $NMI_k(|E_{true}|)$ refers to the NMI score that uses the actual number of message categories $|E_{true}|$ to cluster the representations of the messages. The representation of message is aggregated based on action $a_{agg,r}^{(l)(k)}$, and we will expand this process in detail in Sec. 4.2.2. Note that in order to prevent disturbances caused by the DRL-DBSCAN training process, we use K-Means as the clustering method.

Optimization. Based on the above definition, each agent uses the Actor-critic algorithm [24] to select actions according to the state through the actor network and finally obtains the same reward to update the loss function. In this process, each agent strives to obtain the greatest overall benefit, multi-agents belong to a cooperative relation. The loss function of actor under relation r is defined as:

$$\mathcal{L}_{agg,r} = -\Bigg[\Big(r_{agg,r}^{(k)} + \gamma Q(s_{agg,r}^{(k+1)}, a_{agg,r}^{(k+1)}) - Q(s_{agg,r}^{(k)}, a_{agg,r}^{(k)})\Big) \cdot log\Big[\pi(a_{agg,r}^{(k)}|s_{agg,r}^{(k)})\Big]\Bigg], \tag{4}$$

where $Q(,)$ refers to the action value function and $\pi(|)$ represents the policy.

4.2.2 Weighted Relation-aware Neighbor Aggregation

In order to better guide the weighted multi-relational Graph Neural Network to learn the message embedding, we propose a weighted relation-aware neighbor aggregation. As shown in message embedding part of Fig. 2, the entire aggregation process in MarGNN is divided into two parts: intra-relation aggregation and inter-relation aggregation.

For intra-relation aggregation, participating neighbor messages are controlled by the preserving threshold. This process is formally expressed as the aggregation process of the message m_i belonging to the relation r at the l-th layer:

$$\mathbf{h}_{m_i,r}^{(l)} \leftarrow AGG_{intra,r}^{(l)}\Big(\overset{heads}{\|}\big(\oplus\{\mathbf{h}_{m_j}^{(l-1)} : m_j \in \mathcal{N}_r^l(m_i)\}\big)\Big), \tag{5}$$

where $\mathbf{h}_{m_j,r}^{(l-1)}$ represents the embedding of the neighbor message m_j of the message m_i in the $l-1$-th layer under relation r. $\mathcal{N}_r^l(m_i)$ means the set of a series of neighbors of the message m_i after the neighbor selection process with the preserving threshold value $p_r^{(l)}$. The embedding $\mathbf{h}_m^{(0)}$ of each messages is the input feature. We introduce the multi-head attention mechanism of Graph Attention Network [29] for the neighbor aggregator $AGG_{intra,r}^{(l)}$ within relation r considering the continuously increasing message stream. $\overset{heads}{\|}$ represents head-wise concatenation [30], we splice the outputs of multiple heads in the middle layer and average them in the last layer. $\oplus$ stands the summation aggregation operator.

Then in the inter-relation aggregation, the preserving threshold of the relation is used as the weight of the relation embedding during inter-relation aggregation (we will compare other aggregation to verify in Sec. 6.3.1). We define the inter-relation aggregation of message m_i in the l-th layer as:

$$\mathbf{h}_{m_i}^{(l)} \leftarrow \mathbf{h}_{m_i}^{(l-1)} \otimes AGG_{inter}^{(l)}\big(\otimes\{p_r^{(l)} \cdot \mathbf{h}_{m_i,r}^{(l)}\}|_{r=1}^{R}\big). \tag{6}$$

Here, $\mathbf{h}_{m_i,r}^{(l)}$ represents the inter-aggregation embedding of m_i belongs to the relation r. We propose an inter-relation aggregator $AGG_{inter}^{(l)}$ based on preserving threshold, where $\otimes$ is a splicing aggregation operator, e.g., concatenation, sum, or Multi-layer Perceptron (MLP). The result of the inter-relation aggregator and the embedding of the upper layer of message m_i are spliced as the final representation of m_i in the l-th layer. The inter-aggregation embedding $\mathbf{h}_{m_i}^{(L)}$ is regarded as the final embedding $\mathbf{h}_{m_i}$ of m_i.

4.3 Balanced Sampling Strategy based Contrastive Learning Mechanism (BasCL)

The number of event classes in the incremental event detection task is constantly changing, which makes the cross-entropy loss function widely used in GNNs no longer applicable. Contrastive learning [31] is not limited to a fixed number of classes because it focuses on learning the common features between similar instances and distinguishing differences between non-similar instances. Particularly, contrastive learning can tackle the long-tail problem of social events in the real world [15], in which some events are sparse and niche, rendering the model hard to detect. What's more, the representations based on contrastive learning contain more cluster-like structure information, which can benefit the downstream event clustering tasks [32]. Inspired by [33], [34] and [35], [36], we introduce triplet losses in BasCL to balance a large number of negative samples of other event classes and a small number of positive samples of the same event class, and add global-local pair loss on this basis to preserve the graph structure information in the process of detecting long-tail events incrementally.

For each message m_i, we first sample a positive sample m_i+ and a negative sample m_i- to construct triple loss and update the embedding of the message in the direction of the positive sample. The process is formalized as:

$$\mathcal{L}_t = \sum_{(m_i,m_i+,m_i-)\in T} max\big\{\mathcal{D}(\boldsymbol{h}_{m_i}, \boldsymbol{h}_{m_i+}) - \mathcal{D}(\boldsymbol{h}_{m_i}, \boldsymbol{h}_{m_i-}) + a, 0\big\}, \tag{7}$$

where (m_i, m_i+, m_i-) is a triple, and T is a series of triples sampled in an online manner [34]. $D(,)$ calculates the Euclidean distance between two messages. $max\{.\}$ is used to get the larger of the two elements. a is a hyper-parameter that determines how far away a message is from its negative sample compared to its positive sample.

We also construct a global-local pair loss to make better use of the influence of similar structural information by minimizing the cross-entropy of global summary and local message representation:

$$\mathcal{L}_p = \frac{1}{N}\sum_{i=1}^{N}\Big(\log\mathcal{S}(\boldsymbol{h}_{m_i},\boldsymbol{s}) + \log\big(1-\mathcal{S}(\tilde{\boldsymbol{h}}_{m_i},\boldsymbol{s})\big)\Big). \quad (8)$$

The global summary $\boldsymbol{s}$ is the average of all message representations, and $\mathcal{S}(,)$ is a bilinear scoring function used to get the probability that the two operands come from the joint distribution. We use the noise-contrastive method [37] to construct $\tilde{\boldsymbol{X}}$ by the row-wise shuffle of $\boldsymbol{X}$ to obtain the corrupted representation $\tilde{\boldsymbol{h}}_{m_i}$ of message representation $\boldsymbol{h}_{m_i}$.

Optimization. We consider positive-negative sample comparisons and global-local comparisons in a balanced manner, and define the overall loss function of BasCL as:

$$\mathcal{L}_{BasCL} = \mathcal{L}_t + \mathcal{L}_p. \quad (9)$$

4.4 Deep Reinforcement Learning guided DBSCAN Model (DRL-DBSCAN)

In the event detection stage, we cluster the messages according to the learned message representation (this process follows the Def. 2.6). The commonly used distance-based clustering algorithm K-Means is easily used to cluster representations, but there are limitations in the task of incremental detection due to the need to specify the number of categories. Density-based clustering algorithm DBSCAN [38] automatically adjust the number of classes, but it still has two parameters (the distance parameter ϵ and the minimum sample number parameter $minPts$) that need to be manually adjusted and cannot adapt to match message blocks with different data distributions in the constantly changing message input. To this end, we first propose a novel reinforcement learning in model DRL-DBSCAN to explore how to obtain a stable clustering effect of social events in the multi-round parameter interaction with DBSCAN. To the best of our knowledge, we are the first to combine DBSCAN with RL-methods. DRL-DBSCAN regards the parameter adjustment system as an agent, the incremental social data as the environment, and formally express the process as a Markov decision process MDP $(S_{clu}, A_{clu}, R_{clu})$, which S_{clu} is state space, A_{clu} is action space, R_{clu} is reward function. Specifically, we define three elements as follows:

• ***State:*** The state is the clustering situation of the message block events observed by the agent after each episode of parameter adjustment, that is, the description of the clustering result. Considering that the cluster label of the event cannot be used when observing the state, the state $s_{clu}^{(\tau)}$ in the episode τ is defined a quaternary set:

$$s_{clu}^{(\tau)} = \{\epsilon^{(\tau)}, |E|^{(\tau)}, coh^{(\tau)}, sep^{(\tau)}\}, \quad (10)$$

composed of the current minimum neighbor distance $\epsilon^{(\tau)}$, the number of clusters $|E|^{(\tau)}$, the average cohesion distance $coh^{(\tau)}$ and the average separation distance $sep^{(\tau)}$ [39]. These states are affected by the action $a_{clu}^{(\tau-1)}$ of the previous episode.

• ***Action:*** In order to prevent the curse of dimensionality and speed up the DBSCAN processing speed, we use t-Distributed Stochastic Neighbor Embedding [40] to reduce the dimensionality of the message representation to 2 dimensions. Therefore, based on the recommendation of the author of the DBSCAN algorithm for the minimum sample number parameter, $minPts$ is fixed to 2. Then, we define the action $a_{clu}^{(\tau)}$ in episode τ is the change of ϵ parameter value that should be selected for the current state $s_{clu}^{(\tau)}$. In addition, the action space is continuous data with upper and lower bounds.

• ***Reward:*** For the reward function, we introduce the external evaluation index Calinski-Harabasz [41] (also known as the Variance Ratio Criterion) to stimulate the agent. The setting of this kind of reward does not need to rely on the true label of the sample and the number of event classes, and it can find the cluster combination in the detection process faster than other external evaluation indexes. The reward function is defined as:

$$r_{clu}^{(\tau)} = \begin{cases} \dfrac{SS_B^{(\tau)}}{|E|^{(\tau)}-1} \Big/ \dfrac{SS_W^{(\tau)}}{N-|E|^{(\tau)}}, & |E|^{(\tau)}\ in\ bounds. \\ 0 & , \ |E|^{(\tau)} out\ of\ bounds. \end{cases} \quad (11)$$

Here, $SS_W^{(\tau)}$ is the overall within-cluster variance and $SS_B^{(\tau)}$ is the overall between-cluster variance in the episode τ. N is the total number of messages. In view of the fact that CH may be in a state of failure for extreme situations (e.g. the CH index with the cluster number of 2 is abnormally high), we heuristically define the acceptable number of clusters, and give a reward value of 0 for cases that exceed the range.

Optimization. In the process of parameter adjustment, there is a strong correlation between the states of adjacent episodes. In order to avoid the one-sided problem of neural network learning, we choose the Twin Delayed Deep Deterministic policy gradient algorithm [25] to optimize one policy network and two value networks. Among them, the loss function of value network is expressed as follows:

$$\mathcal{L}_{clu} = -\mathbb{E}\Big[\frac{1}{2}\Big(r_{clu}^{(\tau)}+\gamma\min\hat{Q}\big(s_{clu}^{(\tau+1)},a_{clu}^{(\tau+1)}\big)-Q\big(s_{clu}^{(\tau)},a_{clu}^{(\tau)}\big)\Big)^2\Big]. \quad (12)$$

Here, $\mathbb{E}$ represents the process of randomly sampling the transition tuple $(s_{clu}^{(\tau)}, a_{clu}^{(\tau)}, r_{clu}^{(\tau)}, s_{clu}^{(\tau)})$ from the experience replay memory and calculating the expectation. Q is the action value function in the current value network. $\min\hat{Q}$ indicates that the smallest action value in the two value networks is selected, which is to suppress overestimation.

4.5 Cross-lingual Message Embedding Method (Crlme)

In a wider social event detection scenario, in addition to resource-rich languages such as English, there are also some non-English languages with insufficient original information that cannot reuse the training process of the English model. In order to achieve more low-resource social event detection and reduce training costs, we directly inherit the parameters θ preserved in English model f_E training as parameters $\bar{\theta}$ of the non-English model f_{NoE} when detecting non-English events. We name this method as Crlme and formalize the process in Def. 2.7. Among them, the parameters

Algorithm 1: FinEvent: Reinforced, Incremental, and Cross-lingual Social Event Detection.

```
Input: A social stream $S = M_0, M_1, M_2, ...$, available
       labels*, window size $w$, the number of layers
       $L$, and the number of mini-batches $B$.
Output: Sets of events: $E_0, E_1, E_2, ....$
1  for $t = 0, 1, 2, ...$ do
2      if $t == 0$ then
3          $\mathcal{G} \leftarrow$ construct initial message graph (Sec. 4.1)
4      else
5          $\mathcal{G} \leftarrow$ update $M_t$ into message graph (Sec. 4.1)
6      if $t != 0$ then // Detect events from $M_t$
7          Execute Algorithm 3
8      if $t\%w == 0$ then // Pre-train or
        maintain model
9          if $t != 0$ then
10             $\mathcal{G} \leftarrow$ remove obsolete messages (Sec. 4.1)
11         Execute Algorithm 2
```

*Labels are used for pre-training and maintenance; full labeling is not required (see Section 4.3 for details).

θ include the cognition of the semantics and structure of the message retained through contrastive learning, the multi-relational balance method discovered through multi-agent reinforcement learning, and the DBSCAN clustering parameter adjustment ability obtained through deep reinforcement learning. In addition, we use the LNMAP model [42] to map a non-English message m to the English semantic space to enhance the adaptability of English parameters.

4.6 Maintenance Strategies

1) All message strategy, keeping all the messages. In the detection stage, we simply insert the newly arrived message block into $\mathcal{G}$. In the maintenance stage, we continue the training process using all the messages in $\mathcal{G}$. In other words, we let FinEvent memorize all the messages it ever received. This strategy is impractical (the messages accumulated in $\mathcal{G}$ can gradually slow down the model and will eventually exceed the embedding space capacity of the message encoder $\mathcal{E}$) and we implement it just for comparison purposes.
2) Relevant message strategy, keeping messages that are related to the newly arrived ones. In the detection stage, we insert the newly arrived message block into $\mathcal{G}$. In the maintenance stage, we first remove messages that are not connected to any messages that arrived during the last time window and then continue training using all the messages in $\mathcal{G}$. In other words, we let FinEvent forget the messages that are both old (i.e., arrived beyond the window) and unrelated (to the new messages that arrived within the window). Note that the knowledge acquired from these messages is preserved in the form of model parameters.
3) Latest message strategy, keeping the latest message block. In the detection stage, we use only the newly arrived message block to reconstruct $\mathcal{G}$. In the maintenance stage, we continue training with all the messages in $\mathcal{G}$, which only involves the latest message block. In other words, we let FinEvent forget all the messages except those in the latest message block. The knowledge learned from the removed messages is memorized in the form of model parameters.

Algorithm 2: Pre-train or Maintain Model.

```
   // MarGNN
1  for $k = 1, 2, ..., K$ do
2      for $r = 1, 2, ..., R$ do
3          Obtain the current state $s^{(k)}_{agg,r}$ via Eq. (2)
4          Choose the action $a^{(k)}_{agg,r}$ for the current state
5      for $b = 1, 2, ..., B$ do
6          $\{m_b\} \leftarrow$ sample a mini-batch
7          for $l = 1, 2, ..., L$ do
8              for $r = 1, 2, ..., R$ do
9                  Top-p sampling by using $a^{(k)}_{agg,r}$
10                 Get intra-relation aggregation $\boldsymbol{h}^{(l)}_{m_i,r}$
                    via Eq. (5), $\forall m_i \in \{m_b\}$
11             Get inter-relation aggregation $\boldsymbol{h}^{(l)}_{m_i}$ via Eq.
                (6), $\forall m_i \in \{m_b\}$
12         $\boldsymbol{h}_{m_i} \leftarrow \boldsymbol{h}^{(L)}_{m_i}, \forall m_i \in \{m_b\}$
           // BasCL
13         Update $\mathcal{L}_{BasCL}$ via Eq. (9)
14     Observe the reward $r^{(k)}_{agg}$ via Eq. (3)
15     Update $\mathcal{L}_{agg}$ via Eq. (4)
```

Algorithm 3: Detecting Events from M_t.

```
   // MarGNN
1  for $r = 1, 2, ..., R$ do
2      Obtain the current state $s_{agg,r}$ via Eq. (2)
3      Choose the action $a_{agg,r}$ for the current state
4  for $l = 1, 2, ..., L$ do
5      for $r = 1, 2, ..., R$ do
6          Top-p sampling by using $a_{agg.r}$
7          Get intra-relation aggregation $\boldsymbol{h}^{(l)}_{m_i,r}$ via Eq.
            (5), $\forall m_i \in M_t$
8      Get inter-relation aggregation $\boldsymbol{h}^{(l)}_{m_i}$ via Eq. (6),
        $\forall m_i \in M_t$
9  $\boldsymbol{h}_{m_i} \leftarrow \boldsymbol{h}^{(L)}_{m_i}, \forall m_i \in M_t$
   // DRL-DBSCAN
10 for $\tau = 1, 2, ..., \mathcal{T}$ do
11     Obtain the current state $s^{(\tau)}_{clu}$ via Eq. (10)
12     Choose the action $a^{(\tau)}_{clu}$ for the current $s^{(\tau)}_{clu}$
13     Get message cluster $E^{(\tau)}$ by using $a_{clu}$
14     Observe the current reward $r^{(\tau)}_{clu}$ via Eq. (11)
15     Update $\mathcal{L}_{clu}$ via Eq. (12)
16 $E_t \leftarrow E^{(\mathcal{T})}$
```

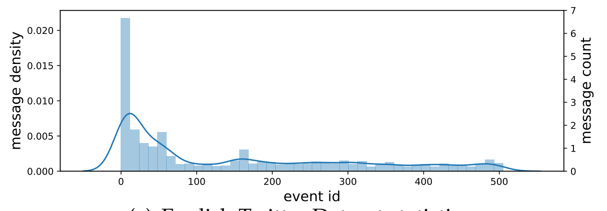
4.7 Proposed FinEvent

Algorithm 1 shows the life cycle of our proposed FinEvent. In the life cycle, the weighted multi-relational graph is constantly changing. Specifically, we initialize a weighted multi-relational graph $\mathcal{G}$ in the pre-training stage (Line 3 in Algorithm 1). When the new messages arrives, that is, the single-language detection or cross-language detection stage, we update the graph $\mathcal{G}$ by inserting the new messages node and establishing a connection (Line 5 in Algorithm 1). In addition, we regularly delete expired nodes and edges during the maintenance stage, and detailed three maintenance strategies in Sec. 4.6. The neighbor selector of MarGNN is initialized and trained in the pre-training stage, and continues to be trained regularly in the maintenance stage so that the neighbor selector is optimized as new messages arrive (Line 15 in Algorithm 2). When performing model single-language detection or cross-language detection, the selector is not updated and directly observes the newly arrived similarity to keep neighbors (Lines 1-3 in Algorithm 3). It is worth noting that the aggregation module will participate in the entire life cycle, whether it is pre-training or maintenance and detecting stages (Lines 5-12 in Algorithm 2, Lines 4-9 in Algorithm 3). After MarGNN, BasCL only appear in the training and maintenance stages (Line 13 in Algorithm 2). In addition, the DRL-DBSCAN part will only participate in the detection stage of the life cycle and restart the game in every block for optimization. In the game, DRL-DBSCAN will carry out a series of action changes until a stable event cluster is obtained (Lines 10-15 in Algorithm 3).

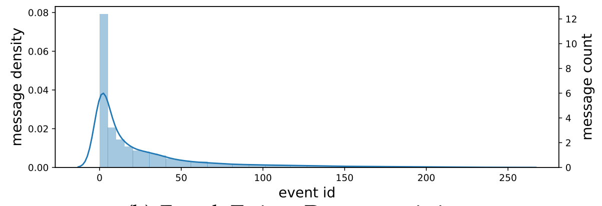
5 Experimental Setup

5.1 Datasets

We conduct experiments mainly on a large-scale, publicly available dataset, i.e., Twitter dataset collected by Twitter API to evaluate the effectiveness of streaming social event detection for it consists of multiple message blocks with continuous timestamps. After filtering out repeated and irretrievable tweets, the Twitter dataset entirely contains *68,841* manually labeled tweets related to *503* event classes, respectively, spread over about *4* weeks (a period of 29 days). To conduct the cross-lingual experiment, we additionally collect French Twitter dataset containing *64,516* labeled tweets related to *257* event classes and spread over about *3* weeks (a period of 23 days). Furthermore, we set three relations, involving ***M-U-M*** (message-user-message), ***M-L-M*** (message-location-message) and ***M-E-M*** (message-entity-message), as the meta-path schema for weighted multi-relational graph. For incremental detection, We split the Twitter dataset by date to construct a social stream. Specifically, we use the messages of the first week to form an initial message block M_0 and the messages of the remaining days to form the following message blocks $M_1, M_2, ..., M_{21}$. Appendix C shows the statistics of the resulting social stream, containing the total number of messages and event classes for each block. Fig. 3 visualizes the distribution of events for both datasets, which follow a long-tail trend, and the message count belonging to each event is extremely imbalanced. A few events are associated with a large quantity of messages while a large fraction of events is associated with a small quantity of messages. The possible reason is preferential attachment in the social network. In such a situation, learning event detectors for small samples is much more difficult due to the poor generalizability caused by insufficient training instances, especially in social streams.



(a) English Twitter Dataset statistics.



(b) French Twitter Dataset statistics.

Fig. 3: Long-tailed distribution and data imbalance challenge in social stream. The curve and bar indicate the density and count of messages in one event, respectively. The order of magnitude of message count is 10^3.

5.2 Baselines

We compare FinEvent to general message representation learning and similarity measuring methods, offline social event methods, incremental methods, and the original KPGNN. The baselines are: **Word2vec** [43], which uses the average of the pre-trained Word2vec embeddings of all the words in a message as its representation; **LDA** [44], a generative statistical model that learns message representations by modeling the underlying topic and word distributions; **WMD** [45], which measures the dissimilarity between two messages by calculating the minimum distance that the word embeddings in one need to travel to reach that of the other; **BERT** [46], which uses the average of BERT embeddings of all the words in a message as its representation; **BiLSTM** [47], which learns bidirectional long-term dependencies between words in a message; **PP-GCN** [9], an offline fine-grained social event detection method based on GCN [48]; **EventX** [14], a fine-grained event detection method based on community detection and is applicable to the online scenario. **KPGNN**, a knowledge-preserving incremental heterogeneous graph neural network model for streaming social event detection; **KPGNN**$_t$, in which the global-local pair loss term $\mathcal{L}_p$ is removed from the loss function and only the triplet loss term $\mathcal{L}_t$ is used. **FinEvent**$_k$, only implements K-Means for clustering the final embeddings.

5.3 Experiment Setting

For the proposed FinEvent, we set the number of attention heads for the intra-relation aggregation in MarGNN to 4, the embedding dimension to 64, the total number of layers to 2, the learning rate to 0.001, the optimizer to Adam, the training epochs to 100 with the patience of 5 for early stopping, and selecting the best model for inference. BiLSTM and other GNN-based baselines use basically the same configuration as above except for setting the dimension to 32. In addition, we set the total number of topics to 50 for LDA, and adopt the hyperparameters as suggested in the original paper [14] for EventX. For maintenance strategy and BasCL, we set the triple margin a to 3, use mini-batch sampling with the size 2000 to improve training efficiency, adopt the *latest message strategy* and maintenance window size 3 (KPGNN and KPGNN$_t$ also use these). For the multi-agent reinforcement learning algorithm Actor-critic of MarGNN, we use the discrete action space with the step size of 0.01 in the range of $[0, 1]$ to each preserving threshold, and initialize the optimal threshold of each relation to 1.0. For the deep reinforcement learning algorithm TD3 of DRL-DBSCAN, we set the continuous action spaces with the range of $[-5, 5]$, the value range of Eps to $(0, 10]$, the cluster boundary of $[15, 100]$, the batch size to 32, and the perplexity of dimensionality reduction to 40. Among them, the initial parameter is the midpoint of the action space. We repeat all experiments 5 times and report the mean and standard variance of the results. Note that KPGNN and FinEvent do not require pre-defining the total number of event classes, but some of the baselines (Word2vec, LDA, WMD, BERT, and BiLSTM) do. To get a fair comparison, after obtaining the message similarity matrix from WMD and message representations from the other models except for EventX (EventX does not pre-define its total number of detected classes), we leverage Spectral and K-Means clustering, respectively, and set the total number of classes to the number of grounded-truth classes. Otherwise, FinEvent is designed for a previously unknown number of classes in the case of incremental detection. Hence we additionally analyze the self-adaptive DRL-DBSCAN clustering module at length to demonstrate our advantage. The complete architecture in the experiment is named as FinEvent$_d$ to be easily distinguished.

5.4 Implementation.

For Word2vec, we use the pre-trained 300-d GloVe [49] vectors. For LDA, WMD, BERT, PP-GCN, and KPGNN (KPGNN$_t$), we use the open-source implementations [6,7,8]. We implement EventX with Python 3.7.3, BiLSTM and FinEvent with Pytorch 1.8.1. All experiments are conducted on a 64 core Intel Xeon CPU E5-2680 v4@2.40GHz with 512GB RAM and 1$\times$NVIDIA Tesla P100-PICE GPU.

5.5 Evaluation Metrics

To evaluate the performance of all models, we measure the similarities between their detected message clusters and the ground-truth clusters. We utilize normalized mutual information (NMI) [50], adjusted mutual information (AMI) [51], and adjusted rand index (ARI) [51]. NMI measures the amount of information one can extract from the distribution of the predictions regarding the distribution of the ground-truth labels and is broadly adopted in social event detection method evaluations [9], [14]. AMI, similar to NMI, also measures the mutual information between two clusters but is adjusted to account for chance [51]. ARI considers all prediction-label pairs and counts pairs that are assigned in the same or different clusters, and ARI also accounts for chance [51].

6. https://github.com/huggingface/transformers
7. https://github.com/RingBDStack/PPGCN
8. https://github.com/RingBDStack/KPGNN

6 Evaluation

In this section, we first compare FinEvent to different baselines including offline as well as incremental social event detection models. We further investigate the effects and stability of reinforcement learning guided multi-relational GNN. Then, we give the cross-lingual transferring evaluation. Lastly, we provide a time complexity analysis for the FinEvent.

6.1 Offline Evaluation

In this section, we compare the preliminary model KPGNN and improved version FinEvent to the baselines in an offline scenario. For both datasets, we randomly sample 70%, 20%, and 10% for training, test, and validation, as such partitions are commonly adopted by GNN studies [9]. To ensure consistency, we compare FinEvent$_k$ with other baselines which implement K-Means for clustering.

Table 2 summarizes the offline evaluation results. FinEvent$_k$ outperforms general message embedding methods (Word2vec, LDA, BERT, and BiLSTM), similarity measuring methods (WMD) by large margins in all metrics (8%-172%, 24%-$1,450\%$, and 20%-$2,300\%$ in NMI, AMI, and ARI on the Twitter dataset). The reason for this is that these methods rely either on measuring the distributions of messages' elements (LDA) or message embeddings (Word2vec, WMD, BERT, and BiLSTM), and they all ignore the underlying social graph structure. Note that although PP-GCN shows strong performance, it assumes a stationary graph structure and cannot adapt to dynamic social streams. FinEvent$_k$, on the contrary, is capable of continuously adapting to and extending its knowledge from the incoming messages (empirically verified in Section 6.2). KPGNN also outperforms both PP-GCN and KPGNN$_t$, which implies the advantage of $\mathcal{L}_p$. EventX shows the highest metric score on NMI among other baselines but much lower compared to KPGNN and FinEvent$_k$. This suggests that EventX tends to generate a larger number of clusters, regardless of whether there is actually more information captured, while FinEvent$_k$ are stronger in general. FinEvent$_k$ outperforms its original version KPGNN. The improvement is attributed to the multi-agent guided aggregation in multi-relational GNN. It is capable of adaptively filtering noise (i.e. irrelevant messages but accidentally linked for both mentioning the same location) and capturing the contributed neighbors. Lastly, FinEvent$_d$ outperforms FinEvent$_k$ by 10% in NMI, 11% in AMI and 100% in ARI. It demonstrates the great power of DRL-DBSCAN in offline detection tasks. To conclude, FinEvent significantly outperforms other baselines by 11% in NMI, 38% in AMI, and 140% in ARI.

6.2 Incremental Evaluation

This subsection evaluates FinEvent in an incremental detection scenario. Tables 3, 4 and 5 summarize the incremental social event detection results in NMI, AMI, and ARI, respectively. Note that PP-GCN, as an offline baseline, cannot be directly applied to the dynamic social streams and we mitigate this by retraining a new PP-GCN model from scratch for each message block (i.e., use the previous blocks as the training set and predict on the current block). For FinEvent,

TABLE 2: Offline evaluation results on the Twitter dataset. The best results are marked in **bold**.

Metrics	Word2vec	LDA	WMD	BERT	BiLSTM	PP-GCN	EventX	KPGNN$_t$	KPGNN	**FinEvent$_k$**	**FinEvent$_d$**	
NMI	.44±.00	.29±.00	.65±.00	.64±.00	.63±.00	.68±.02	.72±.00	.69±.01	.70±.01	.79±.01	**.80**±.01	↑ .08
AMI	.13±.00	.04±.00	.50±.00	.44±.00	.41±.00	.50±.02	.19±.00	.51±.00	.52±.01	.62±.01	**.69**±.01	↑ .19
ARI	.02±.00	.01±.00	.06±.00	.07±.00	.17±.00	.20±.01	.05±.00	.21±.01	.22±.01	.24±.01	**.48**±.01	↑ .28

TABLE 3: Incremental evaluation NMIs. The best results are marked in **bold** and second-best in *italic*.

Blocks	Word2vec	LDA	WMD	BERT	BiLSTM	PP-GCN	EventX	KPGNN$_t$	KPGNN	**FinEvent$_k$**	**FinEvent$_d$**	
M_1	.19±.00	.11±.00	.32±.00	.36±.00	.24±.00	.23±.00	.36±.00	.38±.01	*.39±.00*	.38±.01	**.84±.01**	↑ .49
M_2	.50±.00	.27±.01	.71±.00	.78±.00	.50±.00	.57±.02	.68±.00	.78±.01	.79±.01	*.81±.00*	**.84±.01**	↑ .06
M_3	.39±.00	.28±.00	.67±.00	.75±.00	.39±.00	.55±.01	.63±.00	.77±.00	.76±.00	*.83±.00*	**.89±.00**	↑ .14
M_4	.34±.00	.25±.00	.50±.00	.60±.00	.40±.00	.46±.01	.63±.00	.68±.01	.67±.00	*.71±.01*	**.71±.01**	↑ .08
M_5	.41±.00	.26±.00	.61±.00	.72±.00	.41±.00	.48±.01	.59±.00	.73±.01	.73±.01	*.76±.00*	**.83±.00**	↑ .11
M_6	.53±.00	.32±.00	.61±.00	.78±.00	.50±.00	.57±.01	.70±.00	.81±.00	.82±.01	**.84±.00**	*.83±.00*	↑ .06
M_7	.25±.00	.18±.01	.46±.00	.54±.00	.33±.00	.37±.00	.51±.00	.54±.01	.55±.01	*.56±.00*	**.73±.01**	↑ .02
M_8	.46±.00	.37±.01	.67±.00	.79±.00	.49±.00	.55±.02	.71±.00	.79±.01	.80±.00	*.87±.01*	**.87±.02**	↑ .08
M_9	.35±.00	.34±.00	.55±.00	.70±.00	.43±.00	.51±.02	.67±.00	.74±.01	.74±.02	*.78±.02*	**.79±.01**	↑ .09
M_{10}	.51±.00	.44±.01	.61±.00	.74±.00	.50±.00	.55±.02	.68±.00	.79±.01	.80±.01	*.81±.01*	**.82±.01**	↑ .08
M_{11}	.37±.00	.33±.01	.50±.00	.68±.00	.49±.00	.50±.01	.65±.00	.73±.00	.74±.01	**.76±.00**	*.75±.00*	↑ .08
M_{12}	.30±.00	.22±.01	.60±.00	.59±.00	.39±.00	.45±.01	.61±.00	*.69±.01*	.68±.01	**.76±.01**	.67±.01	↑ .15
M_{13}	.37±.00	.27±.00	.54±.00	.63±.00	.46±.00	.47±.01	.58±.00	.68±.01	*.69±.01*	.67±.00	**.79±.00**	↑ .18
M_{14}	.36±.00	.21±.00	.66±.00	.64±.00	.44±.00	.44±.01	.57±.00	.68±.01	.69±.00	*.74±.00*	**.82±.00**	↑ .16
M_{15}	.27±.00	.21±.00	.51±.00	.54±.00	.40±.00	.39±.01	.49±.00	.57±.01	.58±.00	*.64±.00*	**.69±.01**	↑ .15
M_{16}	.49±.00	.35±.01	.60±.00	.75±.00	.53±.00	.55±.01	.62±.00	.78±.01	.79±.01	*.80±.00*	**.90±.01**	↑ .15
M_{17}	.33±.00	.19±.00	.55±.00	.63±.00	.45±.00	.48±.00	.58±.00	.69±.01	.70±.01	*.73±.00*	**.83±.00**	↑ .20
M_{18}	.29±.00	.18±.00	.63±.00	.57±.00	.44±.00	.47±.01	.59±.00	.68±.01	.68±.02	*.72±.01*	**.74±.01**	↑ .11
M_{19}	.37±.00	.29±.01	.54±.00	.66±.00	.44±.00	.51±.02	.60±.00	.73±.00	*.73±.01*	**.76±.02**	.66±.01	↑ .10
M_{20}	.38±.00	.35±.00	.58±.00	.68±.00	.48±.00	.51±.01	.67±.00	*.73±.00*	.72±.02	*.73±.00*	**.80±.00**	↑ .12
M_{21}	.31±.00	.19±.00	.58±.00	.59±.00	.41±.00	.41±.02	.53±.00	.59±.01	.60±.00	*.65±.01*	**.74±.01**	↑ .15

TABLE 4: Incremental evaluation AMIs. The best results are marked in **bold** and second-best in *italic*.

Blocks	Word2vec	LDA	WMD	BERT	BiLSTM	PP-GCN	EventX	KPGNN$_t$	KPGNN	FinEvent$_k$	**FinEvent$_d$**	
M_1	.08±.00	.08±.00	.30±.00	.34±.00	.12±.00	.21±.00	.06±.00	.36±.01	*.37±.00*	.36±.01	**.84±.01**	↑ .50
M_2	.41±.00	.20±.01	.69±.00	.76±.00	.41±.00	.55±.02	.29±.02	.77±.01	*.78±.01*	.77±.00	**.84±.01**	↑ .08
M_3	.31±.00	.22±.01	.63±.00	.73±.00	.31±.00	.52±.01	.18±.01	.75±.00	.74±.00	*.82±.01*	**.89±.01**	↑ .16
M_4	.24±.00	.17±.00	.45±.00	.55±.00	.30±.00	.42±.01	.19±.01	.65±.01	.64±.01	*.67±.02*	**.69±.00**	↑ .14
M_5	.33±.00	.21±.00	.57±.00	*.71±.00*	.33±.00	.46±.01	.14±.00	.71±.01	.71±.01	*.74±.00*	**.82±.00**	↑ .11
M_6	.40±.00	.20±.00	.57±.00	.74±.00	.36±.00	.52±.02	.27±.00	.78±.00	.79±.01	*.81±.00*	**.82±.02**	↑ .08
M_7	.13±.00	.12±.01	.46±.00	.50±.00	.20±.00	.34±.00	.13±.00	.50±.01	.51±.01	*.53±.00*	**.72±.00**	↑ .22
M_8	.33±.00	.24±.01	.63±.00	.75±.00	.35±.00	.49±.02	.21±.00	.75±.01	.76±.01	*.84±.01*	**.87±.01**	↑ .12
M_9	.24±.00	.24±.00	.46±.00	.66±.00	.32±.00	.46±.02	.19±.00	.70±.01	.71±.02	*.75±.00*	**.78±.01**	↑ .12
M_{10}	.39±.00	.36±.01	.57±.00	.70±.00	.39±.00	.51±.02	.24±.00	.76±.01	*.78±.01*	*.78±.00*	**.81±.00**	↑ .11
M_{11}	.26±.00	.25±.01	.42±.00	.65±.00	.37±.00	.46±.01	.24±.00	.70±.00	.71±.01	*.73±.00*	**.74±.00**	↑ .09
M_{12}	.23±.00	.16±.01	.58±.00	.56±.00	.32±.00	.42±.01	.16±.00	.66±.01	.66±.01	**.75±.01**	*.67±.02*	↑ .17
M_{13}	.23±.00	.19±.00	.50±.00	.59±.00	.31±.00	.43±.01	.16±.00	.65±.01	*.67±.01*	.64±.00	**.79±.00**	↑ .20
M_{14}	.26±.00	.15±.00	.64±.00	.61±.00	.34±.00	.41±.01	.14±.00	.65±.01	.65±.00	*.72±.00*	**.82±.01**	↑ .18
M_{15}	.15±.00	.13±.00	.47±.00	.50±.00	.26±.00	.35±.01	.07±.00	.53±.01	.54±.00	*.61±.00*	**.67±.01**	↑ .17
M_{16}	.36±.00	.27±.01	.59±.00	.72±.00	.41±.00	.52±.01	.19±.00	.75±.01	*.77±.01*	.75±.01	**.90±.01**	↑ .20
M_{17}	.24±.00	.13±.00	.57±.00	.60±.00	.35±.00	.45±.00	.18±.00	.67±.01	.68±.01	*.71±.02*	**.82±.01**	↑ .22
M_{18}	.21±.00	.12±.00	.60±.00	.53±.00	.35±.00	.45±.01	.16±.00	.66±.01	.66±.02	*.70±.00*	**.74±.00**	↑ .14
M_{19}	.28±.00	.22±.01	.49±.00	.63±.00	.35±.00	.48±.02	.16±.00	.70±.00	*.71±.01*	**.75±.01**	.66±.00	↑ .12
M_{20}	.24±.00	.23±.00	.55±.00	.62±.00	.34±.00	.45±.02	.18±.00	*.68±.00*	*.68±.02*	*.68±.00*	**.78±.00**	↑ .16
M_{21}	.21±.00	.13±.00	.52±.00	*.57±.00*	.31±.00	.38±.02	.10±.00	.57±.01	.57±.00	*.63±.01*	**.64±.01**	↑ .07

preserving thresholds will be initialized at every beginning of the maintenance stage. To ensure consistency, we compare FinEvent$_k$ with other baselines which implement K-Means for clustering.

Both KPGNN and FinEvent$_k$ significantly and consistently outperform the baselines for all message blocks. They gain over EventX by 8%-32% (21% on average) in NMI, 169%-771% (345% on average) in AMI, and 69%-1782% (560% on average) in ARI. The reason is, EventX relies solely on community detection, while FinEvent$_k$ incorporates the semantics of the social messages. They outperform WMD by 12%-52% (28% on average) in NMI, 11%-63% (29% on average) in AMI, and 0%-266% (110% on average) in ARI and over BERT by up to 3%-28% (11% on average) in NMI, up to 1%-33% (13% on average) in AMI, and up to 0%-83% (39% on average) in ARI. This is because our designs are capable of leveraging the structural information of the social stream, which is ignored by WMD and BERT. They also outperform PP-GCN by 42%-65% (53% on average) in NMI, 48%-117% (62% on average) in AMI, and 0%-51% (15% on average) in ARI. This suggests that our designs effectively preserve up-to-date knowledge, while PP-GCN can be distracted by obsolete information as the messages accumulate.

FinEvent$_k$ outperforms KPGNN and KPGNN$_t$ for the overwhelming majority of message blocks in NMI and AMI. The improvements testify to the impression of weighted multi-relational graph structure and positive effects of

TABLE 5: Incremental evaluation ARIs. The best results are marked in **bold** and second-best in *italic*.

Blocks	Word2vec	LDA	WMD	BERT	BiLSTM	PP-GCN	EventX	KPGNN$_t$	KPGNN	FinEvent$_k$	**FinEvent$_d$**	
M_1	.01±.00	.01±.00	.04±.00	.03±.00	.03±.00	.05±.00	.01±.00	.06±.01	*.07±.01*	.05±.00	**.90±.00**	↑ .85
M_2	.49±.00	.08±.00	.48±.00	.64±.00	.49±.00	.67±.03	.45±.02	*.76±.01*	*.76±.02*	.67±.01	**.90±.01**	↑ .23
M_3	.16±.00	.02±.01	.28±.00	.43±.00	.17±.00	.47±.01	.09±.01	*.60±.02*	.58±.01	.58±.00	**.89±.01**	↑ .42
M_4	.07±.00	.07±.00	.11±.00	.19±.00	.11±.00	.24±.01	.07±.01	*.30±.01*	*.29±.01*	.27±.02	.27±.01	↑ .06
M_5	.17±.00	.06±.00	.26±.00	.44±.00	.19±.00	.34±.00	.04±.00	*.48±.01*	.47±.03	.43±.01	**.63±.02**	↑ .19
M_6	.25±.00	.07±.01	.16±.00	.44±.00	.18±.00	.55±.03	.14±.00	.67±.05	*.72±.03*	.65±.00	**.74±.00**	↑ .19
M_7	.02±.00	.01±.00	.08±.00	.07±.00	.08±.00	.11±.02	.02±.00	.11±.01	*.12±.00*	.09±.01	**.45±.01**	↑ .34
M_8	.17±.00	.03±.00	.22±.00	.50±.00	.08±.00	.43±.04	.09±.00	.59±.02	.60±.01	*.65±.02*	**.72±.01**	↑ .22
M_9	.08±.00	.03±.01	.12±.00	.33±.00	.27±.00	.31±.02	.07±.00	.45±.02	*.46±.02*	.43±.00	**.68±.00**	↑ .35
M_{10}	.23±.00	.09±.02	.20±.00	.44±.00	.22±.00	.50±.07	.13±.00	.64±.01	*.70±.06*	.62±.02	**.74±.01**	↑ .24
M_{11}	.09±.00	.03±.01	.12±.00	.27±.00	.17±.00	.38±.02	.16±.00	.48±.01	*.49±.03*	.42±.01	**.60±.01**	↑ .22
M_{12}	.09±.00	.02±.01	.27±.00	.31±.00	.13±.00	.34±.03	.07±.00	**.50±.03**	*.48±.01*	.44±.00	.26±.00	↑ .16
M_{13}	.06±.00	.01±.00	.13±.00	.14±.00	.13±.00	.19±.01	.04±.00	.28±.01	*.29±.03*	.21±.02	**.75±.02**	↑ .56
M_{14}	.10±.00	.02±.00	.33±.00	.30±.00	.16±.00	.29±.01	.10±.00	*.43±.02*	.42±.02	*.43±.01*	**.81±.01**	↑ .48
M_{15}	.09±.00	.01±.00	.16±.00	.10±.00	.14±.00	.15±.00	.01±.00	.16±.02	**.17±.00**	.16±.00	**.46±.00**	↑ .31
M_{16}	.10±.00	.11±.01	.32±.00	.41±.00	.10±.00	.51±.03	.08±.00	.62±.03	*.66±.05*	.56±.01	**.88±.01**	↑ .37
M_{17}	.06±.00	.02±.00	.26±.00	.24±.00	.17±.00	.35±.03	.12±.00	.41±.03	*.43±.05*	.36±.01	**.81±.01**	↑ .46
M_{18}	.21±.00	.02±.00	.35±.00	.24±.00	.19±.00	.39±.03	.08±.00	.46±.02	*.47±.04*	.44±.01	**.52±.01**	↑ .13
M_{19}	.28±.00	.03±.00	.12±.00	.32±.00	.16±.00	.41±.02	.07±.00	*.50±.01*	**.51±.03**	.44±.00	.35±.01	↑ .10
M_{20}	.24±.00	.02±.01	.19±.00	.33±.00	.20±.00	.41±.01	.11±.00	*.51±.01*	*.51±.04*	.43±.02	**.71±.01**	↑ .30
M_{21}	.21±.00	.01±.01	.19±.00	.18±.00	.16±.00	*.20±.03*	.01±.00	*.23±.02*	.20±.01	*.23±.00*	**.48±.00**	↑ .27

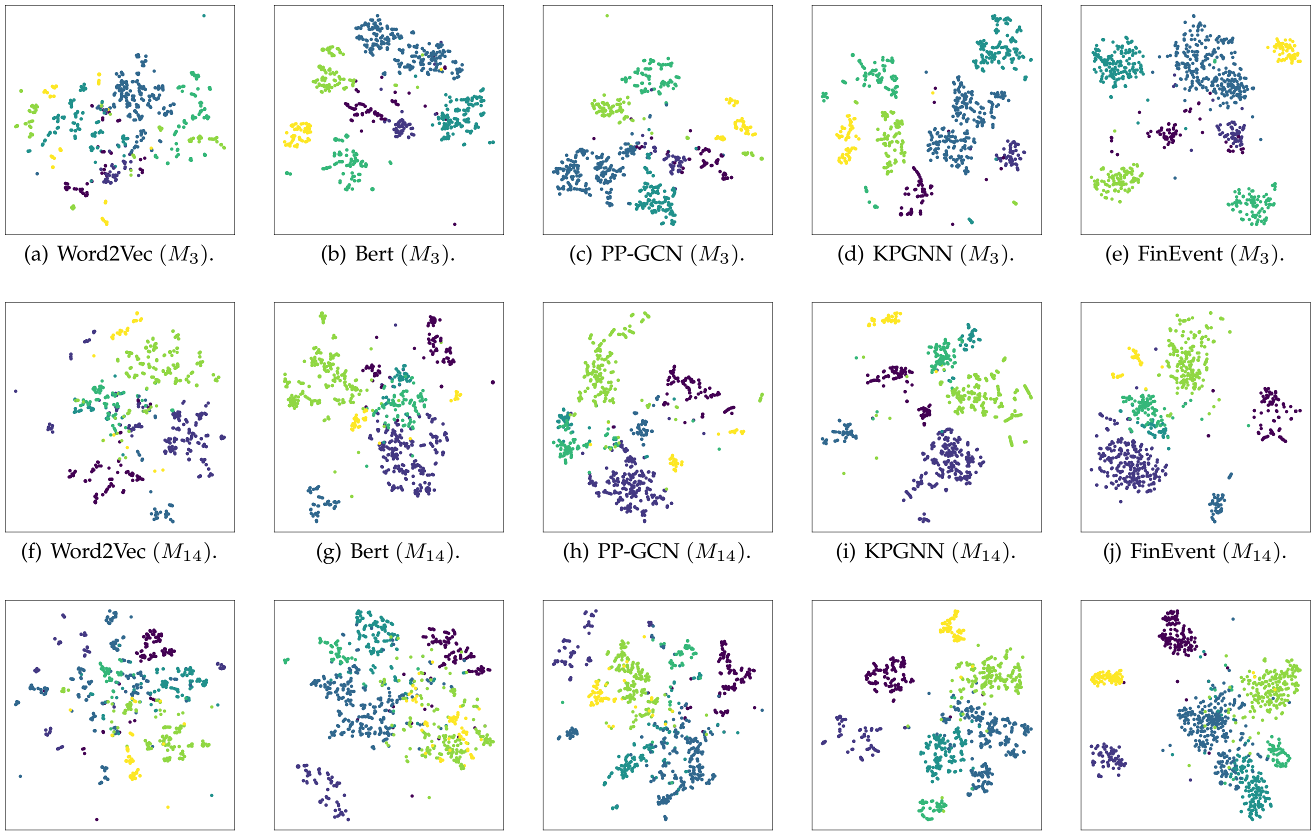
TABLE 6: Ablation study for neighbor sampler strategy, Intra-relation Aggregation AGG_{intra} and Inter-relation Aggregation AGG_{inter}. The best results are marked in **bold** and second-best in *italic*.

	Neighbor Sampling Strategy			Cluster Type	Intra-relation Aggregator		Threshold	Inter-relation Aggregator			Avg. Metrics		
	Random	Constant	Reinforced		Shared-GNN	RNN		Cat.	Sum	MLP	NMI	AMI	ARI
1	-	-	✓	DBSCAN	-	-	✓	✓	-	-	**0.788**	**0.777**	**0.645**
2	-	-	✓	K-Means	-	-	✓	✓	-	-	*0.727*	*0.702*	0.443
3	-	✓	-	K-Means	-	-	-	-	-	-	0.698	0.671	*0.448*
4	-	top 50%	-	K-Means	-	-	-	✓	-	-	0.719	0.699	0.441
5	-	bottom 50%	-	K-Means	-	-	-	✓	-	-	0.698	0.670	0.427
6	✓	-	-	K-Means	-	-	-	✓	-	-	0.718	0.694	0.437
7	-	✓	-	K-Means	-	-	-	✓	-	-	0.722	0.696	0.442
8	-	-	✓	K-Means	✓	-	✓	✓	-	-	0.700	0.673	0.416
9	-	-	-	K-Means	-	✓	-	-	-	-	0.449	0.324	0.168
10	-	-	✓	K-Means	-	-	-	✓	-	-	0.723	0.697	0.438
11	-	-	✓	K-Means	-	-	✓	-	✓	-	0.702	0.674	0.425
12	-	-	✓	K-Means	-	-	-	-	✓	-	0.700	0.672	0.422
13	-	-	✓	K-Means	-	-	✓	✓	-	✓	0.653	0.620	0.383
14	-	-	✓	K-Means	-	-	-	✓	-	✓	0.645	0.610	0.368

multi-agent guided aggregation, which incorporates more relational information and greatly enhances both intra- and inter-aggregation. The reason for the decrease of ARI is that the ARI metric prefers to measure the performance when events have equal-sized clustering, but the ground truth events clustering in our experiment are unbalanced (as shown in Fig. 3). And this also explains the increase of FinEvent$_k$ in AMI for it prefers unbalanced clusters instead. The use of MarGNN and DRL-DBSCAN improve the clustering accuracy but might intensify the effect of event imbalance brought by a social stream at the same time. To conclude, the performance of FinEvent$_k$ is superior to the baselines. Lastly, FinEvent$_d$ significantly outperforms FinEvent$_k$ by $0\% - 1210\%$ in NMI, $0\% - 1330\%$ in AMI, and $0\% - 1700\%$ in ARI. The improvements are attributed to the well-designed DRL-DBSCAN, testifying DRL-DBSCAN not only breaks through the limitation of K-Means in stream clustering, but achieves greater performance with deep reinforcement learning agents as well. What's more, FinEvent$_d$ has significant performance on ARI, which makes up the limitation of K-Means on clustering unbalanced events.

Overall, FinEvent significantly outperforms other baselines by $8\% - 136\%$ in NMI, $11\% - 147\%$ in AMI, and $24\% - 170\%$ in ARI on the incremental detection scenario.

Ablation Study. To evaluate the effectiveness of MarGNN, we conduct an ablation study on the process of AGG_{intra} and AGG_{inter} respectively, denoted as the intra-relation aggregator and intra-relation aggregator. For the intra-relation aggregator, we design (a) shared-GNN, which only leverages one sole GNN to learn information from all relational graphs; and (b) RNN, which substitutes GNNs with RNNs. To avoid the impacts of parameters selection in DBSCAN, we implement K-Means for all variants. It is shown in Table 6 that compared with shared-GNN, MarGNN outperforms by 3.9% in NMI, 4.6% in AMI. Moreover, MarGNN and shared-GNN outperform KPGNN by 6.5% in ARI, 4.2% in NMI and 4.6% in AMI, 6.5% in ARI, 0.3% in NMI and 0.2% in AMI, respectively. It demonstrates that integrating the distinct information from multi-relational graphs is crucial in social event detection because the types and schema of events should be fully considered. MarGNN and KPGNN outperform the RNN family by 61.9% in NMI, 116.7% in AMI, and 163.7% in ARI, 55.5% in NMI, 107.1% in AMI, and 116.7% in ARI, respectively. The advantages can



(a) Word2Vec (M_3). (b) Bert (M_3). (c) PP-GCN (M_3). (d) KPGNN (M_3). (e) FinEvent (M_3).

(f) Word2Vec (M_{14}). (g) Bert (M_{14}). (h) PP-GCN (M_{14}). (i) KPGNN (M_{14}). (j) FinEvent (M_{14}).

(k) Word2Vec (M_{18}). (l) Bert (M_{18}). (m) PP-GCN (M_{18}). (n) KPGNN (M_{18}). (o) FinEvent (M_{18}).

Fig. 4: Cluster visualization of message representations in the detection stage.

be attributed to the graph modeling of social messages. Compared with the GNN-based model, RNNs neglect the structural information in social streams.

For the inter-relation aggregator, we conduct different combination strategies: concatenation (Cat.), Sum and MLP mentioned in Section 4.2.2. The output dimension of GNNs and MLPs is set to 64. It is shown in Table 6 that the concatenation operation could boost the performance of MarGNN over sum by 3.6% in NMI, 4.2% in AMI and 4.2% in ARI and MLP by 11.3% in NMI, 13.2% in AMI and 15.7% in ARI. It is attributed to that the direct concatenation operation can best preserve the knowledge and structural information among relations from social streams.

Visualization Analysis. To better measure the effect of FinEvent, we report the dimensionality reduction visualization of the information representation of specific blocks, i.e., M_3, M_{14} and M_{18} in detection stages as Fig. 4. In these figures, we focus on the top 7 events by volume due to the long-tail challenge of social data (Fig. 3). The representations of the messages belonging to the same event are marked with the same color. Comparing Column $1, 2$ to Column $3, 4$ and 5 in Fig. 4, it is observed that clustering results of GNN-based methods are more compact than those merely depending on representations. It demonstrates the advantages of introducing GNNs to event detection. Furthermore, for internal comparison among PP-GCN, KPGNN, and FinEvent, obviously FinEvent achieves better performance. This shows that the architecture of FinEvent combined with different RL modules can be well adapted to the incremental event detection.

6.3 Study on RL process

In this section, we focus on the reinforcement learning process in FinEvent, including multi-agent RL guided optimal preserving threshold selection and aggregation, as well as DRL guided DBSCAN, respectively.

6.3.1 *Preserving thresholds*

As shown in Fig. 5, we concatenate all the epochs in the training process, containing 1 pre-training stage (from epoch 0 to epoch 50) and 7 maintenance stages (from epoch 50 to epoch 340), and visualize the changes of preserving thresholds under each relation. It can be observed that each preserving threshold gradually converges to a fixed value after oscillating in several epochs. The multi-agents steadily learn from graph structures until all preserving thresholds reach the terminal conditions.

We further conduct an ablation study on reinforced neighbor selection. Firstly, we verify the effectiveness of neighbor selection. Concretely, we select the top 50% and bottom 50% number of neighbors for every target node in terms of their representation distance (Section 4.2.1). We take the average of the NMI, AMI, and ARI results of 21 blocks. As shown in Table 6, the selection of top 50% neighbors achieves better results compared with the selection of bottom 50% neighbors by 3.0% in NMI, 4.3% in AMI, and 3.3% in ARI. It testifies that some messages or users are indeed

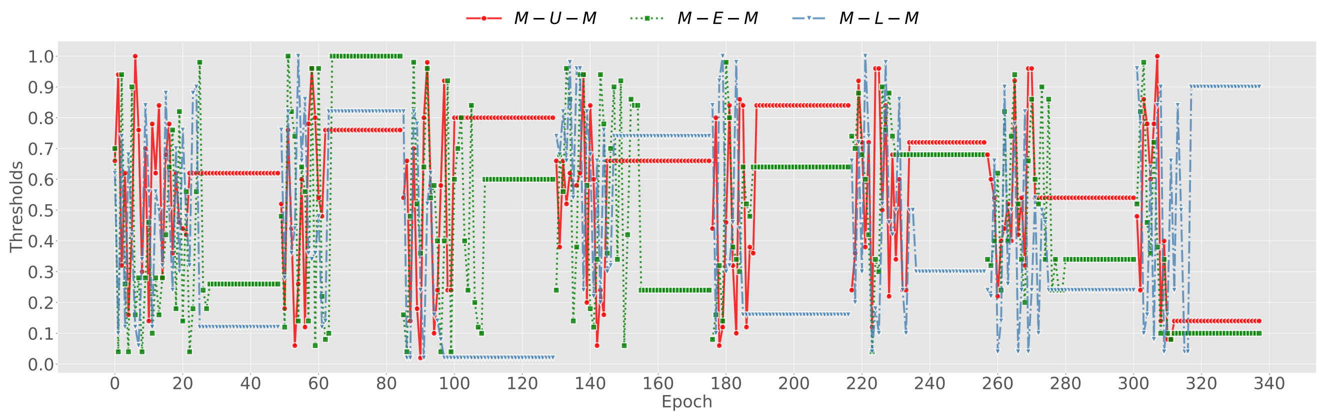
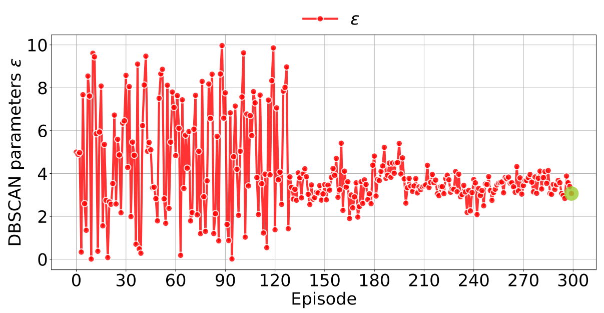
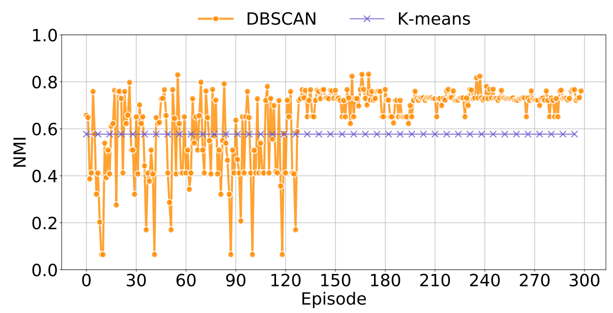


Fig. 5: Multi-agent reinforcement learning process in the online maintenance stage. We summarize the epochs of all time periods. Note that each process from fluctuation to stability is a pre-training or maintenance stage. The figure contains a total of one pre-training process and seven maintenance processes.



(a) DBSCAN parameters.



(b) NMI scores.

Fig. 6: Deep reinforcement learning process in the online detection stage. We show the DRL-DBSCAN parameter adjustment and NMI change process of block M_7 as an example of DRL-DBSCAN, where the green marked points represents the final convergence parameter.

noise in the social network, which would harm the performance of GNNs and are urgent to be filtered. Secondly, we discuss different neighbor sampling strategies to verify the effectiveness of Top-p neighbor sampling guided by multi-agent. In addition to the proposed reinforced strategy, we set **i).** a constant number of neighbors for every target node, as we leverage 2-layer GNNs, the sizes are set to 25 for the first hop and 15 for the second hop recommended by [52]; **ii).** random number of neighbors, the number of neighbors for the first hop is randomly selected in the range of 10 to 100 and 10 to 50 for the second hop. It is shown in Table 6 that the reinforced sampling strategy performs better than random sampling strategy by 1.3% in NMI, 1.2% in AMI, and 1.4% in ARI, and constant sampling strategy by 0.7% in NMI, 0.9% in AMI, and 0.2% in ARI. It verifies the effectiveness of the optimal preserving thresholds. The values of elaborate optimal preserving thresholds guided by multi-agents in the detection stage are shown in Table 7. Instead of manually fixing the thresholds, multi-agents are capable of using their experience learning from block features, e.g., information and relation structure, to select the optimal preserving thresholds in the detection stage adaptively. We further evaluate the advantages of optimal preserving thresholds in the inter-relation aggregation process. As shown in Table 6, the proposed optimal preserving thresholds can improve the performance on inter-aggregator by 0.6% in NMI, 0.7% in AMI and 1.1% in ARI for concatenation, 0.3% in NMI, 0.3% in AMI and 0.7% in ARI for sum, and 1.2% in NMI, 1.6% in AMI and 4.0% in ARI for MLP. This provides a significant explanation for our assumption of the importance of noise and relation described in Section 4.2.2, that is, the relation with too much noise contributes less to aggregation.

TABLE 7: Preserving thresholds in the detection stage.

Blocks	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
M-U-M	–	.88	.24	.98	.24	.50	.40	.40	.12	.06	.96
M-E-M	–	.82	.56	.22	.20	.88	.90	.74	.20	.28	.50
M-L-M	–	.96	.08	.80	.54	.42	.78	.80	.56	.70	.86
Blocks	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}	M_{17}	M_{18}	M_{19}	M_{20}	M_{21}
M-U-M	.22	.94	.22	.14	.10	.46	.44	.24	.10	.32	.38
M-E-M	.66	.72	.24	.60	.76	.82	.90	.90	.20	.10	.34
M-L-M	.74	.64	.20	.98	.54	.18	.50	.46	.24	.16	.92

TABLE 8: DBSCAN parameters in the detection stage.

Blocks	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
ϵ	–	3.87	3.29	2.57	3.25	3.24	3.70	2.35	2.95	3.39	3.57
Blocks	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}	M_{17}	M_{18}	M_{19}	M_{20}	M_{21}
ϵ	3.19	3.28	3.76	3.93	2.00	3.37	3.54	3.23	4.00	2.23	3.18

6.3.2 *DRL-DBSCAN*

Stationary analysis. In the detection stage, each block creates a brand new clustering scene for DRL-DBSCAN, so the deep reinforcement learning agent restarts the game process every block. Fig. 6 shows the DBSCAN parameters adjustment learning process in one block taking the reinforcement learning process during block M_7 as an example. We observe that parameter ϵ in Fig. 6(a) fluctuate continuously throughout the action space and stabilize in the range 3-4 starting from episode 200. This process embodies the process of the reinforcement learning agent gaining parameter adjustment experience in the interaction with the environment. Furthermore, every time period of detection, DRL-DBSCAN is used to detect a completely new block of messages. We give the parameter combinations predicted by DRL-DBSCAN in all detection stages in Table 8. It can be seen that the final parameter result fluctuates with different message blocks.

Effectiveness analysis. In order to evaluate the incentive ability of the reward function in Eq. 11, we measure the revenue of all reinforcement learning parameter search games (only block M_7 is shown as an example). Fig. 6(b) shows the DBSCAN NMI changes using our DRL-DBSCAN and K-Means NMI changes using the known number of events. It is worth noting that after the interaction of multiple episode parameters in Fig. 6(a), the final convergence NMI of DRL-DBSCAN exceeds K-Means, which shows that the heuristic external indicator reward function can effectively stimulate the parameters to higher levels. Note that the reward of K-Means remains unchanged because the event distribution and message representation of the same epoch are fixed. In addition, the complete measurement results corresponding to the final parameters of the different message blocks in Table 8 are shown in Table 3, Table 4, and Table 5. By comparing with K-Means, it is found that our model has a stable advantage in different blocks.

6.4 Cross-lingual Transferring Evaluation

As described in Section 3.3, due to the advantage of FinEvent in preserving and extending knowledge, we leverage such inductive learning ability to resolve the cross-lingual transferring problem. Generally, a well-trained model on high-resource English dataset is transferred to low-resource non-English datasets, which are fine-tuned with little training samples, and directly used to detect the incoming events. We treat the English Twitter dataset as a high-resource dataset to pre-train FinEvent, and apply it to achieve cross-lingual detection on French Twitter dataset, which plays a role in low-resource Twitter dataset. It is noted that these two datasets are thoroughly different, which own respective timestamps and event types (though part of the event is plotted in Fig. 3 are repetitive, there are totally different events). In our experiment, we directly calculate the document features of French messages, and Google Translate to firstly translate French messages and then obtain the node features.

To verify our assumption on cross-lingual detection, we compare the performance of FinEvent in two different situations: (a) we train a new FinEvent detector on both processed French dataset, named FinEvent$_r$(Raw) and FinEvent$_g$(Google Translated). Settings and maintenance strategy remain the same; (b) we fine-tune the well-trained FinEvent from English dataset with the initial M_0 in French datasets, named FinEvent$_{cr}$ and FinEvent$_{cg}$ respectively, and keep them in the detection stage so as to continuously detect events from the transferred French dataset. We report NMIs results as the performances of them in Table 9. All clustering types of FinEvent are set to K-Means to avoid the impacts of distinct parameters selection in DBSCAN. It is observed that the performances of FinEvent$_{cr}$ and FinEvent$_{cg}$ are better than FinEvent$_r$ and FinEvent$_g$. Concretely, among 15 blocks, FinEvent$_{cr}$ takes the lead in 12 blocks compared with FinEvent$_r$ and achieves improvement in the range of 1.7% to 20.6%. FinEvent$_{cg}$ also outperforms FinEvent$_g$ in 12 blocks and achieves improvement in the range of 1.7% to 20.8%. The block-wise average NMIs of FinEvent$_r$ and FinEvent$_{cr}$ are 0.554 and 0.597, respectively. The block-wise average NMIs of FinEvent$_g$ and FinEvent$_{cg}$ are 0.554 and 0.595, respectively. The feasibility of FinEvent$_{cr}$ and FinEvent$_{cg}$ demonstrates that the knowledge preserved in FinEvent trained on high-resource English dataset can be extended and guide the event detection on low-resource non-English datasets. In addition, the similar performances of FinEvent$_r$ and FinEvent$_g$ also give us insights that despite different language expressions of messages, the semantic and structure of message graphs are more crucial in event detection.

TABLE 9: Cross-lingual transferring evaluation on French dataset. The best results are marked in **bold**.

Blocks	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
FinEvent$_r$	–	.575	**.620**	.577	.512	.556	.488	.584
FinEvent$_{cr}$	–	**.576**	.578	**.580**	**.534**	**.633**	**.656**	**.615**
FinEvent$_g$	–	**.592**	.574	.591	.474	.568	.511	.580
FinEvent$_{cg}$	–	.578	**.575**	**.604**	**.542**	**.641**	**.659**	**.600**

Blocks	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
FinEvent$_r$	**.640**	.484	**.627**	.529	.545	.472	.519	.586
FinEvent$_{cr}$	.612	**.523**	.597	**.610**	**.616**	**.569**	**.622**	**.630**
FinEvent$_g$	**.625**	.484	**.623**	.546	.548	.474	.530	.591
FinEvent$_{cg}$	.598	**.512**	.599	**.584**	**.610**	**.563**	**.640**	**.626**

6.5 Time Analysis

In this section, we will analyze the time complexity and consumption of the online maintenance stage to explore the practicality of FinEvent. The overall running time of FinEvent is $O(N_e)$, where $O(N_e)$ is the total number of edges in the messsage graph. Among them, the time complexity of constructing or maintaining the multi-relational graph with N messages is $O(N + N_e) = O(N_e)$. The aggregation of MarGNN framework takes $O(N_e+NR+N+N_e) = O(N_e)$, where the first $O(N_e)$ is the complexity of intra-relation aggregation, $O(NR + N)$ is the complexity of inter-relation aggregation, the second $O(N_e)$ is the complexity of reinforced neighbor selection. Among them, R is the number of relations. For the BasCL mechanism, loss calculation takes $O(\sum_{b=1}^{B})|m_b^2| + m_b)$, where m_b is the number of messages in the b-th batch. In the Crlme method, we need $O(N)$ to map non-English messages to English semantic space. In addition, we count the number of epochs consumed and the average time consumed in each epoch of the online maintenance stage (more details in Fig. 3 in APPENDIX C.). From the statistical information of each time period, it can be seen that the average time consumed by each epoch is proportional to the number of messages that need to be maintained, and the number of epochs consumed in the time period gradually decreases as the number of maintenance increases. Note that the newly added preserving threshold selection mechanism does not bring a significant increase in complexity, so the time complexity $O(N_e)$ of our architecture is the same as the original work KPGNN. From the perspective of overall time and epoch consumption, although **FinEvent** brings higher epoch consumption in the early maintenance stages, as maintenance progresses, epoch consumption in the subsequent moments gradually decreases and shows a stable trend.

7 Related Work

Social Event Detection. Based on their objectives, social event detection methods can be broadly categorized into *document-pivot* (DP) methods [9], [14], [53]–[56]: aim at clustering social messages based on their correlations, and *feature-pivot* (FP) ones [13], [57]: aim at clustering social messages elements (such as words and named entities) based on their distributions. The presented FinEvent is a DP method. Based on their application scenarios, social event detection methods can be divided into offline [9] and online [4], [13], [14], [54], [56], [58] methods. Though offline methods are essential in analyzing retrospective, domain-specific events such as disasters, stock market trading, major sports events and political campaigns, online methods that continuously work on the dynamic social streams are desirable [4], [13], [58]. For different techniques and mechanisms, social event detection methods can be separated into popular classes such as methods rely on incremental clustering [53], [54], [56], community detection [13], [14], [18], [22] and topic models [55]. These methods, however, are limited by latent knowledge as they ignore the rich semantics and structural information contained in the social streams to some extent. Furthermore, these models have too few parameters to preserve the learned knowledge. Though the Text-SimCLNN [58] presents a graph attention network and contrastive learning based text semantic encoding method, it does not fully consider the structural features and relations among social messages, and even judges the cluster by single document without considering the common contributions of adjacent documents that appear in the same period, which makes it easy for nodes that should belong to the existing cluster to be misjudged as a new cluster. Both [9] and [59] belong to GNN-based social event detection methods, but these models only implement static event classification and can only work offline. FinEventis different from the existing methods as it effectively acquires, extends, preserves, and transfers knowledge by continuously adapting to the incoming social messages.

Inductive Learning with Graph Neural Networks. The Graph Neural Networks (GNNs) [29], [48], [60]–[63] have been widely used in graph data representation learning and applications. In general, a GNN extracts, aggregates, and updates node contextual representations from source neighborhoods, which realizes data diffusion across the graph structure. Depending on their extraction and aggregation strategies, some GNNs [48] only conduct transductive learning [64] as they require pre-known, fixed graph structures. Others [29], [60], [65] can be used in inductive learning [64], which means that they can be applied to novel test patterns without repetitive training. Though often discussed, inductive learning using GNNs is rarely evaluated or utilized in real application scenarios [64]. The proposed FinEvent is the first to leverage GNNs' inductive learning ability for incremental social event detection.

Combination GNNs and Reinforcement Learning. There are a few attempts to marry GNNs and RL to boost the representation learning ability of graph data. DeepPath [66] is a knowledge graph embedding and reasoning framework based on RL policy, and the RL agent is trained to ascertain the reasoning paths in the knowledge base. RL-HGNN [67] devises different meta-paths to learn its effective representations, and uses a DRL-based policy network for adaptive meta-paths selection in downstream tasks. As opposed to MarGNN, the RL-HGNN model pays more attention to revealing meaningful meta-paths or relations in heterogeneous graph analysis. Similar to our MarGNN, both CARE-GNN [27], using the Bernoulli Multi-armed Bandit process to select neighbors, and Rio-GNN [68], using recursive reinforcement learning to speed up the process of balancing filter thresholds when offline, are designed to improve the effectiveness of GNNs, but these models are no longer applicable in weighted graph and incremental scenarios. GraphNAS [69] employs RL to search the optimal graph neural architectures. Policy-GNN [70] formulates the GNN training problem as a Markov Decision Process, and can adaptively learn an aggregation policy to sample diverse iterations of aggregations for different nodes. However, neither GraphNAS nor Policy-GNN models consider heterogeneous neighborhoods in aggregation although they pay more attention to neural architecture searching.

Density-based Stream Clustering. Density-based stream data clustering methods [4], [71]–[74] are fundamental technologies for wide social data mining and analysis. However, compared to these clustering methods, the DRL-DBSCAN firstly explores how to obtain a stable clustering effect of social events by combining DBSCAN with a deep reinforcement learning method without an offline maintenance process.

8 Conclusion

This paper studies FinEvent a reinforced, incremental and cross-lingual social event detection architecture from streaming social messages. To learn social message embedding, a multi-agent reinforcement learning guided multi-relational graph neural network framework MarGNN is presented. A deep reinforcement learning guided density-based spatial clustering model DRL-DBSCAN is designed to select optimal parameters in event detection tasks. The conducted experiments on Twitter streams suggest that FinEvent achieves significant and consistent improvements in model quality in performance on offline, online, and cross-lingual social event detection tasks. In the future, we aim to extend the multi-agent RL guided GNNs to learn complex graph data representation and their applications, such as event correlation, event evolution, etc. In addition, it is also interesting to study how to extend our models to other clustering tasks.

ACKNOWLEDGMENTS

The authors of this paper were supported by the National Key R&D Program of China through grant 2021YFB1714800, NSFC through grant 62002007, S&T Program of Hebei through grant 20310101D, and the Fundamental Research Funds for the Central Universities. Philip S. Yu was supported by NSF under grants III-1763325, III-1909323, III-2106758, and SaTC-1930941. Thanks for computing infrastructure provided by Huawei MindSpore platform.

REFERENCES

[1] 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.

[2] J. Allan, *Topic detection and tracking: event-based information organization*. Springer Science & Business Media, 2012, vol. 12.

[3] Z. Wang, Y. Zhang, Y. Li, Q. Wang, and F. Xia, “Exploiting social influence for context-aware event recommendation in event-based social networks,” in *Proceedings of the IEEE INFOCOM*, 2017, pp. 1–9.

[4] H. Peng, J. Li, Y. Song, R. Yang, R. Ranjan, P. Yu, and L. He, “Streaming social event detection and evolution discovery in heterogeneous information networks,” *ACM TKDD*, vol. 15, no. 5, pp. 1–33, 2021.

[5] R. Gaspar, C. Pedro, P. Panagiotopoulos, and B. Seibt, “Beyond positive or negative: Qualitative sentiment analysis of social media reactions to unexpected stressful events,” *Computers in Human Behavior*, vol. 56, pp. 179–191, 2016.

[6] T. M. Nisar and M. Yeung, “Twitter as a tool for forecasting stock market movements: A short-window event study,” *The journal of finance and data science*, vol. 4, no. 2, pp. 101–119, 2018.

[7] F. Marozzo and A. Bessi, “Analyzing polarization of social media users and news sites during political campaigns,” *Social Network Analysis and Mining*, vol. 8, no. 1, pp. 1–13, 2018.

[8] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, “A survey of heterogeneous information network analysis,” *IEEE TKDE*, vol. 29, no. 1, pp. 17–37, 2016.

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

[10] 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 Web Conference*, 2021, pp. 3383–3395.

[11] W. Yu, C. C. Aggarwal, S. Ma, and H. Wang, “On anomalous hotspot discovery in graph streams,” in *Proceedings of the IEEE ICDM*, 2013, pp. 1271–1276.

[12] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra, “Spotlight: Detecting anomalies in streaming graphs,” in *Proceedings of the ACM SIGKDD*, 2018, pp. 1378–1386.

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

[14] 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 TKDD*, vol. 14, no. 3, pp. 1–28, 2020.

[15] Q. Li, H. Peng, J. Li, J. Wu, Y. Ning, L. Wang, S. Y. Philip, and Z. Wang, “Reinforcement learning-based dialogue guided event extraction to exploit argument relations,” *IEEE/ACM TASLP*, pp. 1–1, 2021.

[16] S. Zhao, Y. Gao, G. Ding, and T.-S. Chua, “Real-time multimedia social event detection in microblog,” *IEEE transactions on cybernetics*, vol. 48, no. 11, pp. 3218–3231, 2017.

[17] A. Subburathinam, D. Lu, H. Ji, J. May, S.-F. Chang, A. Sil, and C. Voss, “Cross-lingual structure transfer for relation and event extraction,” in *Proceedings of the EMNLP-IJCNLP*, 2019, pp. 313–325.

[18] J. Liu, Y. Chen, K. Liu, and J. Zhao, “Event detection via gated multilingual attention mechanism,” in *Proceedings of the AAAI*, 2018, pp. 4865–4872.

[19] A. Goswami and A. Kumar, “A survey of event detection techniques in online social networks,” *Social Network Analysis and Mining*, vol. 6, no. 1, pp. 1–25, 2016.

[20] C. Zhang, L. Liu, D. Lei, Q. Yuan, H. Zhuang, T. Hanratty, and J. Han, “Triovecevent: Embedding-based online local event detection in geo-tagged tweet streams,” in *Proceedings of the ACM SIGKDD*, 2017, pp. 595–604.

[21] D. T. Nguyen and J. E. Jung, “Real-time event detection for online behavioral analysis of big social data,” *FGCS*, vol. 66, pp. 137–145, 2017.

[22] 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.

[23] X. Tan, J. Chen, D. He, Y. Xia, Q. Tao, and T.-Y. Liu, “Multilingual neural machine translation with language clustering,” in *Proceedings of the EMNLP-IJCNLP*, 2019, pp. 962–972.

[24] V. R. Konda and J. N. Tsitsiklis, “Actor-critic algorithms,” in *Proceedings of the NIPS*, 2000, pp. 1008–1014.

[25] S. Fujimoto, H. van Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in *Proceedings of the ICML*, vol. 80. PMLR, 2018, pp. 1587–1596.

[26] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Proceedings of the NIPS*, 2013, p. 3111–3119.

[27] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, “Enhancing graph neural network-based fraud detectors against camouflaged fraudsters,” in *Proceedings of the ACM CIKM*, 2020, pp. 315–324.

[28] L. Yang, Z. Liu, Y. Dou, J. Ma, and P. S. Yu, “Consisrec: Enhancing gnn for social recommendation via consistent neighbor aggregation,” in *Proceedings of the ACM SIGIR*, 2021, pp. 2141–2145.

[29] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” in *Proceedings of the ICLR*, 2018, pp. 1–12.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proceedings of the NeurIPS*, 2017, pp. 5998–6008.

[31] K. Hassani and A. H. Khasahmadi, “Contrastive multi-view representation learning on graphs,” in *Proceedings of the ICML*, vol. 119. PMLR, 2020, pp. 4116–4126.

[32] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola, “What makes for good views for contrastive learning?” in *Proceedings of the NeurIPS*, vol. 33, 2020, pp. 6827–6839.

[33] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proceedings of the CVPR*. IEEE, 2015, pp. 815–823.

[34] A. Hermans, L. Beyer, and B. Leibe, “In defense of the triplet loss for person re-identification,” *arXiv preprint arXiv:1703.07737*, 2017.

[35] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, “Learning deep representations by mutual information estimation and maximization,” in *Proceedings of the ICLR*, 2019, pp. 1–14.

[36] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, “Mutual information neural estimation,” in *Proceedings of the ICML*, vol. 80. PMLR, 2018, pp. 531–540.

[37] P. Velickovic, W. Fedus, W. L. Hamilton, P. Lio, Y. Bengio, and R. D. Hjelm, “Deep graph infomax,” in *Proceedings of the ICLR*, 2019, pp. 1–13.

[38] 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 ACM SIGKDD*, vol. 96, no. 34, 1996, pp. 226–231.

[39] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[40] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne.” *JMLR*, vol. 9, no. 11, pp. 2579–2605, 2008.

[41] T. Caliński and J. Harabasz, “A dendrite method for cluster analysis,” *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

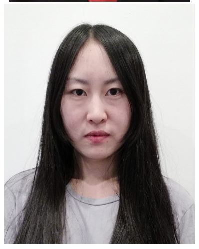
[42] T. Mohiuddin, M. S. Bari, and S. Joty, “Lnmap: Departures from isomorphic assumption in bilingual lexicon induction through non-linear mapping in latent space,” in *Proceedings of the EMNLP*, 2020, pp. 2712–2723.

[43] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of ICLR*, 2013, pp. 1–12.
[44] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *JMLR*, vol. 3, no. Jan, pp. 993–1022, 2003.
[45] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *Proceedings of the ICML*, 2015, pp. 957–966.
[46] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the NAACL*, 2019, pp. 4171–4186.
[47] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005.
[48] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of ICLR*, 2017, pp. 1–10.
[49] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the EMNLP*, 2014, pp. 1532–1543.
[50] 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.
[51] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *JMLR*, vol. 11, pp. 2837–2854, 2010.
[52] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *arXiv preprint arXiv:2005.00687*, 2020.
[53] C. C. Aggarwal and K. Subbian, "Event detection in social streams," in *Proceedings of the SDM*. SIAM, 2012, pp. 624–635.
[54] L. Hu, B. Zhang, L. Hou, and J. Li, "Adaptive online event detection in news streams," *Knowledge-Based Systems*, vol. 138, pp. 105–112, 2017.
[55] X. Zhou and L. Chen, "Event detection over twitter social media streams," *VLDB Journal*, vol. 23, no. 3, pp. 381–400, 2014.
[56] K. Zhang, J. Zi, and L. G. Wu, "New event detection based on indexing-tree and named entity," in *Proceedings of the ACM SIGIR*, 2007, pp. 215–222.
[57] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu, "Parameter free bursty events detection in text streams," in *Proceedings of the VLDB*. Citeseer, 2005, pp. 181–192.
[58] C. Tong, H. Peng, X. Bai, Q. Dai, R. Zhang, Y. Li, H. Xu, and X. Gu, "Learning discriminative text representation for streaming social event detection," *IEEE TKDE*, pp. 1–1, 2021.
[59] W. Cui, J. Du, D. Wang, F. Kou, and Z. Xue, "Mvgan: Multi-view graph attention network for social event detection," *ACM TIST*, vol. 12, no. 3, pp. 1–24, 2021.
[60] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the NeurIPS*, 2017, pp. 1024–1034.
[61] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, "Graph neural networks with convolutional arma filters," *IEEE TPAMI*, pp. 1–12, 2021.
[62] Z. Chen, X.-S. Wei, P. Wang, and Y. Guo, "Learning graph convolutional networks for multi-label recognition and applications," *IEEE TPAMI*, pp. 1–16, 2021.
[63] T. Schnake, O. Eberle, J. Lederer, S. Nakajima, K. T. Schutt, K.-R. Mueller, and G. Montavon, "Higher-order explanations of graph neural networks via relevant walks," *IEEE TPAMI*, pp. 1–1, 2021.
[64] L. Galke, I. Vagliano, and A. Scherp, "Can graph neural networks go" online"? an analysis of pretraining and inference," in *Proceedings of the ICLR*, 2019, pp. 1–5.
[65] G. Ciano, A. Rossi, M. Bianchini, and F. Scarselli, "On inductive-transductive learning with graph neural networks," *IEEE TPAMI*, pp. 1–1, 2021.
[66] W. Xiong, T. Hoang, and W. Y. Wang, "Deeppath: A reinforcement learning method for knowledge graph reasoning," in *Proceedings of the EMNLP*. ACL, 2017, pp. 564–573.
[67] Z. Zhong, C.-T. Li, and J. Pang, "Reinforcement learning enhanced heterogeneous graph neural network," *arXiv preprint arXiv:2010.13735*, 2020.
[68] H. Peng, R. Zhang, Y. Dou, R. Yang, J. Zhang, and P. S. Yu, "Reinforced neighborhood selection guided multi-relational graph neural networks," *ACM TOIS*, pp. 1–46, 2021.
[69] Y. Gao, H. Yang, P. Zhang, C. Zhou, and Y. Hu, "Graph neural architecture search," in *Proceedings of the IJCAI*, 2020, pp. 1403–1409.
[70] K.-H. Lai, D. Zha, K. Zhou, and X. Hu, "Policy-gnn: Aggregation optimization for graph neural networks," in *Proceedings of the ACM SIGKDD*, 2020, p. 461–471.
[71] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *Proceedings of the ACM SIGKDD*, 2007, pp. 133–142.
[72] G. Petkos, S. Papadopoulos, and Y. Kompatsiaris, "Social event detection using multimodal clustering and integrating supervisory signals," in *Proceedings of the ACM ICMR*, 2012, pp. 1–8.
[73] C. C. Aggarwal, "A survey of stream clustering algorithms," in *Data Clustering*. Chapman and Hall/CRC, 2018, pp. 231–258.
[74] C. Comito, C. Pizzuti, and N. Procopio, "Online clustering for topic detection in social data streams," in *Proceedings of the ICTAI*. IEEE, 2016, pp. 362–369.

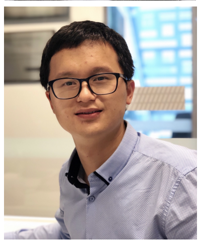
Hao Peng is currently an Assistant Professor at the School of Cyber Science and Technology in Beihang University. His research interests include representation learning, social network mining and reinforcement learning. To date, Dr Peng has published over 80+ research papers in top-tier journals and conferences, including the IEEE TKDE, TPDS, TNNLS, TASLP, JAIR, ACM TOIS, TKDD, and Web Conference.

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

Shaoning Li is currently a undergraduate in the School of Cyber Science and Technology in Beihang University. His research interests include deep learning and reinforcement learning.



Yuwei Cao is currently a Ph.D. candidate in the Department of Computer Science at University of Illinois at Chicago. Her research interests include representation learning, social event mining.



Shirui Pan is currently a Senior Lecturer with the Faculty of Information Technology, Monash University, Australia. Prior to this, he was a Lecturer with the School of Software, University of Technology Sydney. His research interests include data mining and machine learning. To date, Dr Pan has published over 80 research papers in top-tier journals and conferences, including the IEEE TPAMI, IEEE TKDE, and KDD.

Philip S. Yu is a Distinguished Professor and the Wexler Chair in Information Technology at the Department of Computer Science, University of Illinois at Chicago. Before joining UIC, he was at the IBM Watson Research Center, where he built a world-renowned data mining and database department. He was the Editor-in-Chiefs of ACM TKDD (2011-2017) and IEEE TKDE (2001-2004). He is a Fellow of the ACM and IEEE.

1 Appendix A. Optimal Preserving Thresholds Evaluation

Deep layers analysis. Preserving threshold also represents an explicable approach to overcome over-fitting and over-smoothing problems for deep layers in GNN. To verify our inference, we design three varients of FinEvent with 4/8/32 layers and their corresponding model without preserving thresholds. We still adopt the latest-message strategy and set window size to 1, and report the performance of the models on M_9 to M_{14} in Fig. 1. It is observed that FinEvent with preserving threshold improve the detection effects for all the cases, and the improvement is greater as the layer grows. Different from [1], which drops edges in graph randomly, multi-agents in FinEvent drops edges based on certain rules and their learned experiment. To conclude, FinEvent provides a reasonable edge-drop instruction for the improvement of deep GNNs.

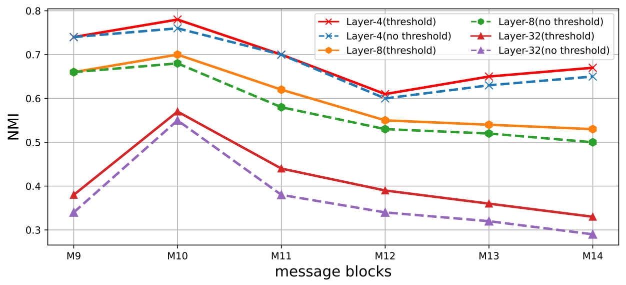


Fig. 1: Inference results for different depths of layers.

Interpretability analysis. In addition, while in training or detection stage, different agents would eventually choose different preserving threshold values. The various thresholds indicate different contributions made by agents, which can be abstracted as macroscopic attention of GNN towards different relations. As depicted in M_3, for instance, relation *M-E-M* gets higher threshold values than others probably for occupying higher importance in this block. We also display the preserving thresholds in a week as radar maps (shown in Fig. 2) for the convenience of observing the flow of relations' contributions. It demonstrates the change of block and event structures with time. The enclosed area can be approximately regarded as the overall contributions one relation makes to events of this week, and hence we can intuitively obtain the global relation importance. All of these give necessary interpretable explanation to GNN for event detection.

2 Appendix B. Hyper-parameter Sensitivity

This subsection studies the effects of hyperparameters in the incremental social event detection experiments. We set the hidden embedding dimension to 128 for Twitter dataset to minimize the graph entropy as much as possible, and only change the output embedding dimension d and window size w. Fig. 4 compares the performance of FinEvent when adopting different output dimension as well as window size for each message blocks. The NMI, AMI and ARI results have average deviations in the range from 0.01-0.03. This suggests that the metrics of FinEvent change with d and w, but rather significantly. The output embedding

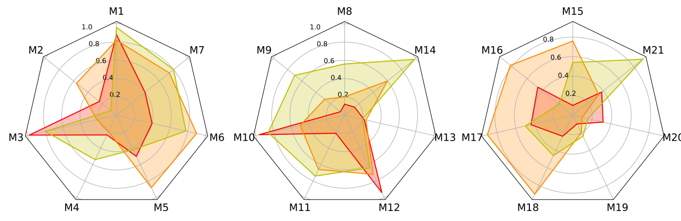


Fig. 2: Comparison on the performance of varients of FinEvent on English dataset in 3 weeks. The colors indicate different relations: *M-U-M* is colored in red, *M-E-M* is colored in orange and *M-L-M* is colored in green.

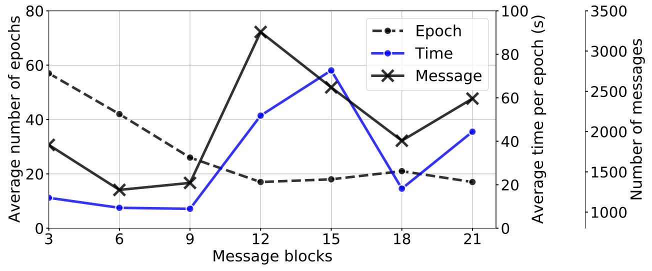


Fig. 3: Time consumption in the online maintenance stage.

dimension has little influence on the performance of FinEvent. For example, the block-wise average NMIs of output embedding dimension are 0.701, 0.719, 0.716, respectively. FinEvent reaches the best performance when d is set to 64. Adopting a smaller window size (2 or 3) in general gives a slightly better performance. For example, the block-wise average NMIs of different window sizes are 0.719, 0.729, 0.723, 0.714, respectively and average AMIs are 0.693, 0.702, 0.698, 0.687, respectively. When window size is set to 3, FinEvent achieves the best performance. A possible reason is that for Twitter dataset, $w = 3$ has best adaptability to the continuation of events. In a word, FinEvent is sensitive to the changes in hyperparameters.

3 Appendix C. Statistics of Social Streams

This section depicts the number of messages and the number of events composed in each block from English and French dataset, respectively. The details are shown in Table 1 and Table 2. In addition, the time consumption of the social stream is given in Figure 3

References

[1] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *Proceedings of the ICLR*, 2020.

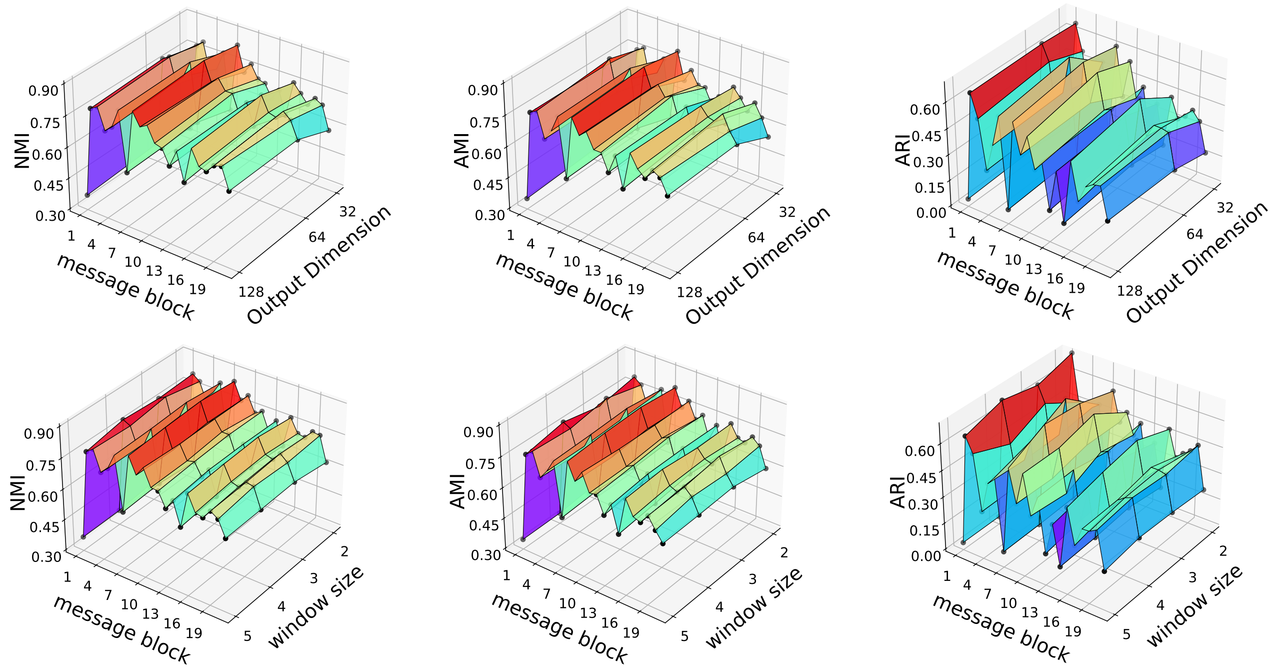


Fig. 4: FinEvent with different hyperparameters.

TABLE 1: The statistics of the social stream from English Twitter Dataset.

Blocks	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
# of messages	20,254	8,722	1,491	1,835	2,010	1,834	1,276	5,278	1,560	1,363	1,096
# of events	155	41	30	33	38	30	44	57	53	38	33
Blocks	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}	M_{17}	M_{18}	M_{19}	M_{20}	M_{21}
# of messages	1,232	3,237	1,972	2,956	2,549	910	2,676	1,887	1,399	893	2,410
# of events	30	42	40	43	42	27	35	32	28	34	32

TABLE 2: The statistics of the social stream from French Twitter Dataset.

Blocks	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
# of messages	14,328	5,356	3,186	2,644	3,179	2,662	4,200	3,454
# of events	79	22	19	15	19	27	26	23
Blocks	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
# of messages	2,257	3,669	2,385	2,802	2,927	4,884	3,065	2,411
# of events	25	31	32	31	29	28	26	25