ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/380538651

An Efficient Automatic Meta-Path Selection for Social Event Detection via Hyperbolic Space

Conference Paper · May 2024

DOI: 10.1145/3589334.3645526

CITATIONS
4

READS
20

4 authors:

Zitai Qiu
Macquarie University
7 PUBLICATIONS **7** CITATIONS
SEE PROFILE

Congbo Ma
Macquarie University
31 PUBLICATIONS **415** CITATIONS
SEE PROFILE

Jia Wu
Macquarie University
296 PUBLICATIONS **7,762** CITATIONS
SEE PROFILE

Jian Yang
Macquarie University
347 PUBLICATIONS **6,392** CITATIONS
SEE PROFILE

All content following this page was uploaded by Congbo Ma on 21 June 2024.

The user has requested enhancement of the downloaded file.

An Efficient Automatic Meta-Path Selection for Social Event Detection via Hyperbolic Space

Zitai Qiu
Macquarie University
School of Computing
Sydney, NSW, Australia
zitai.qiu@students.mq.edu.au

Congbo Ma
Macquarie University
School of Computing
Sydney, NSW, Australia
congbo.ma@mq.edu.au

Jia Wu
Macquarie University
School of Computing
Sydney, NSW, Australia
jia.wu@mq.edu.au

Jian Yang
Macquarie University
School of Computing
Sydney, NSW, Australia
jian.yang@mq.edu.au

ABSTRACT

Social events reflect changes in communities, such as natural disasters and emergencies. Detection of these situations can help residents and organizations in the community avoid danger and reduce losses. The complex nature of social messages makes social event detection on social media challenging. The challenges that have a greater impact on social media detection models are as follows: (1) the amount of social media data is huge but its availability is small; (2) social media data is a tree structure and traditional Euclidean space embedding will distort embedded features; and (3) the heterogeneity of social media networks makes existing models unable to capture rich information well. To solve the above challenges, we propose a Heterogeneous Information Graph representation via Hyperbolic space combined with an Automatic Meta-path selection (GraphHAM) model, an efficient framework that automatically selects the meta-path's weight and combines hyperbolic space to learn information on social media. In particular, we apply an efficient automatic meta-path selection technique and convert the selected meta-path into a vector, thereby reducing the requisite amount of labeled data for the model. We also design a novel Hyperbolic Multi-Layer Perceptron (HMLP) to further learn the semantic and structural information of social information. Extensive experiments show that GraphHAM can achieve outstanding performance on real-world data using only 20% of the whole dataset as the training set. Our code can be found on GitHub [1].

CCS CONCEPTS

• **Information systems** → **Social networks**; *Data mining*; • **Computing methodologies** → *Neural networks.*

KEYWORDS

Social Event Detection; Graph Neural Networks; Automatic Meta-Path; Hyperbolic Space

ACM Reference Format:
Zitai Qiu, Congbo Ma, Jia Wu, and Jian Yang. 2024. An Efficient Automatic Meta-Path Selection for Social Event Detection via Hyperbolic Space. In *Proceedings of the ACM Web Conference 2024 (WWW '24), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3589334.3645526

1 INTRODUCTION

Social events can be defined as hot topics that people discuss on social media [6]. It reflects a major impact on communities, such as natural disasters and emergencies, and also reflects people's attitudes and reactions to some events, such as presidential elections and institutional reforms. Social event detection extracts these influential events from the vast reservoir of online information, making it a vital and indispensable process in understanding contemporary societal dynamics. Today's ubiquity of mobile devices means social media platforms such as Twitter, Facebook, and Weibo are often the first to witness events as they occur. Therefore, social media often serves as an important source of information for relief organizations and governments on emergencies and natural disasters, allowing them to detect disasters at an early stage, monitor the development of disasters, and carry out subsequent rescue operations [26]. Based on the above reasons, social event detection in social media has gained significant attention from academia and industry [2, 6, 9, 11, 27, 32, 33, 45].

The difficulty of social event detection lies in text classification or clustering similar messages into the same topic [2, 20]. However, unlike traditional text classification, classifying text from social media has the following three challenges: **1) Limited data availability**: Social media generates volumes of unlabeled data, but the high costs associated with manual labeling make it impractical to annotate social media data in large quantities; **2) Feature embedding distortion**: The spread of social media is mainly based on mentions and forwardings between people, resulting in social media data having a tree structure [1]. Embedding tree-structured data into Euclidean space will lead to distortion of features [16, 42]; and **3) Heterogeneity**: Social media messages are not just text,

[1] https://github.com/ZITAIQIU/GraphHAM

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WWW '24, May 13–17, 2024, Singapore, Singapore

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0171-9/24/05…$15.00
https://doi.org/10.1145/3589334.3645526

usually containing some other heterogeneous information, such as time, places, people, and entities.

For challenge #**1**, most of the existing social event detection models are solved through unsupervised learning like HISEvnet [7] and HCRC [15]. However, for unsupervised models, their results are often not as good as supervised ones. Therefore, GraphMSE [21] adopts another approach to solve the challenge, which is to reduce the proportion of the training set. Normally, we divide more than 50% of the entire dataset into the training set, but GraphMSE can achieve good results by achieving a training set with a proportion of 20%. However, the disadvantage of GraphMSE is that when the proportion of the training set is less than 20%, the model will learn nothing. Therefore, how to deal with the small amount of available data is still a challenge. For challenge #**2**, tree-structured data grows exponentially. However, existing social event detection models are generally designed based on Euclidean space, and they cannot capture tree structure data well [16, 42]. Therefore, how to better learn features from tree-structured data is also a problem to be solved. For challenge #**3**, existing social event detection methods [6, 27, 28] roughly converts a heterogeneous information network (HIN) into a homogeneous information network for feature learning, which results in the loss of a large amount of semantic information and structural information. A common way to deal with HINs is to use meta-path [22, 36], which can intuitively exploit structural information in heterogeneous graphs. However, most of the existing models [40, 44] use fixed meta-path sets, hindering their capacity to adequately express real-world datasets. Thus, how to learn features from HIN is still a thorny problem.

In this paper, we propose a framework: Heterogeneous **Graph** representation via **H**yperbolic space combined with an **A**utomatic **M**eta-path selection (GraphHAM). It combines a text model and a graph neural network model with efficient automatic meta-path selection technology and hyperbolic space representation to solve the above challenges. In particular, we only enumerate all meta-paths from the training set instead of enumerating meta-paths from all nodes in the dataset, which greatly improves our efficiency. We also supplement it with a hyperbolic space embedding of all text features, allowing the framework to maintain good performance even when the training set is extremely scarce. Next, we address the heterogeneity issue by embedding meta-paths for each category through a hyperbolic space representation model. Finally, the attention mechanism is used to select meta-paths. The contributions of our work can be summarized as follows:

- We pioneer the application of automatic meta-path selection and hyperbolic space in social media detection, leveraging their advantages.
- We address the challenge of limited training set proportion, allowing our framework to maintain feature learning capabilities even with a very small training ratio (5%).
- Our algorithm can achieve excellent performance with only a small amount of computational resources (Apple M1 chip with 16GB RAM) and demonstrates a fast execution speed.
- We conduct extensive experiments and analyses to validate the effectiveness of the algorithm including comparison with the baseline models, model parameter analysis, model ablation experiments, and analysis of loss during model training.

2 RELATED WORK

2.1 Social Event Detection

The development of social event detection starts from the models based on topic detection, such as LDA [5] and TF-IDF [34]. This type of model mainly focuses on classifying similar texts into the same topic and calculates the similarity between them by counting the co-occurring words that appear in the text. However, this method is largely limited by the sparsity of keywords, especially short texts on social media such as Facebook and Twitter. The large number of short texts appearing on social media has greatly reduced the performance of this type of model. To solve this problem, the researchers try to let the model understand the meaning behind the words and let the model learn the semantic information of the words. The Word2Vec [24] is their method to let the model understand the semantics of the words. Based on Word2Vec, researchers proposed GPU-DMM [18] further to improve the model's performance in short text classification. However, the semantic improvement behind simply understanding words is limited, and the sparseness of words is still a key issue that hinders model improvement of model performance. It still needs some additional information to supplement the problems caused by word sparseness. Therefore, researchers proposed SGNS [35], which takes the relationship between words into model learning. This model solves the problem of sparse words in short texts by learning the semantics of the entire sentence.

However, traditional social event detection models based on topic detection only focus on text, and they ignore the essence of social media: heterogeneity. If the model only learns text information, it will lose a lot of structural information and semantic information between entities. Therefore, researchers try to construct social media data into a heterogeneous information network (HIN) to retain as much useful information as possible. Based on our knowledge, PP-GCN [27] applies HIN to social media detection for the first time. It learns the semantic information and structural information on HIN through meta-path [36]. However, the meta-path set on PP-GCN is designed manually, which has limitations and uncertainties. Even if the meta-paths are designed by experienced experts, they cannot perfectly describe real-world data. KPGNN [6] and FinEvent [28] are the latest HIN-based social event detection models proposed two years later, but they focus on incremental social event detection. Their methods for learning HIN are still similar to PP-GCN and have not been improved. Therefore, learning more correct features from HIN is still a thorny problem that needs to be solved.

2.2 Automatic Meta-path Selection

The problem with meta-path is that it requires manual design by experts based on prior knowledge. This design cannot perfectly describe real-world data. In addition, the importance of each meta-path in the dataset is also different, and it is difficult to manually define the weight of each meta-path. Therefore, how to automatically select a meta-path from HIN has become a research hot spot in many fields. Wei et al. [41] proposed a model in 2018 that selects the importance of meta-paths by maintaining the proximity between nodes. In the same year, Wang et al. [38] proposed an unsupervised meta-path selection model that uses a minimum spanning

tree to rank meta-path importance. However, both of these models only consider the problem of information retrieval and do not consider representation learning. GTN [43] is a GCN representation learning model combined with meta-path selection. GTN selects the meta-path based on the graph transformer layer. However, the meta-path selection of the GTN model is performed on all nodes, which means that all nodes need to be enumerated, which is very resource-consuming. In addition, all nodes are projected in the same feature space, which cannot distinguish the features of different nodes well.

2.3 Hyperbolic Space Representation

Another disadvantage of the meta-path is that its length is limited. For data with a tree structure like social media data [30], it is difficult to capture long-distance root-to-leaf relationships. On the other hand, although embedding learning in Euclidean space has achieved great success, embedding tree-structured data into Euclidean space will cause distortion [14]. Since the data growth of tree structure is exponential, but Euclidean space is not, this will cause the leaves far away from the root to be very close and unrecognizable. Unlike Euclidean spaces, the growth of hyperbolic spaces is also exponential. Therefore, hyperbolic space is ideal for embedding tree-structured data [12, 14]. However, there are no basic statistical algorithms in hyperbolic spaces, such as calculations of vectors and matrices [14]. Therefore, some basic algorithms cannot be applied in hyperbolic spaces. HNN [14] applies simple neural networks such as RNN to hyperbolic space, but it only focuses on the structural information of the graph and ignores node features. HGCN [8] applies GCN to hyperbolic space based on HNN and implements GCN's aggregation of nodes on hyperbolic space. Nevertheless, existing hyperbolic space learning is based on homogeneous information networks, and there are still some shortcomings in the application of heterogeneous information networks. Therefore, we need to design a framework to apply hyperbolic space in the heterogeneous social event detection environment.

3 PRELIMINARIES

In this section, we summarize the concepts related to the background of our work, including HIN, Meta-path, and hyperbolic space representation.

Definition 1 HIN. A HIN is defined a graph $\mathcal{G} = (V, E, \mathcal{N}, \mathcal{E}, \mathcal{X})$. Here V and E are the set of nodes and edges. $\mathcal{N}$ and $\mathcal{E}$ denote the sets of node and edge types where $|\mathcal{N}| > 1$ and $|\mathcal{E}| > 1$. $\mathcal{X}$ is the set of node features. Furthermore, there are two functions $\Phi : V \rightarrow \mathcal{N}$ and $\Psi : E \rightarrow \mathcal{E}$ mapping nodes and edges to their types.

Definition 2 Meta-path and Meta-path Instance. A meta-path $\mathcal{P} = n_1 \xrightarrow{e_1} n_2 \xrightarrow{e_2} \cdots \xrightarrow{e_l} n_{l+1}$, where $n_i \in \mathcal{N}$ and $e_j \in \mathcal{E}$ and l is the length of this meta-path. It describes the relation $e = (e_1, e_2, \cdots, e_l)$ between node types $n = (n_1, n_2, \cdots, n_{l+1})$. A meta-path instance p is a real path that follows a certain meta-path $\mathcal{P}$ in the heterogeneous graph $\mathcal{G}$.

Definition 3 Hyperbolic Representation. Hyperbolic representation aims to map the features from Euclidean space to hyperbolic space. We have two types of hyperbolic embedding models: $\mathbb{P}$ for the Poincare Ball model [14] and $\mathbb{H}$ for the Hyperboloid model [8]. For the Poincare Ball model, we denote $E_o\mathbb{P}^{d,c}$ as the Euclidean space and $\mathbb{P}^{d,c}$ as the hyperbolic representation via the Poincare Ball model, where o is the center of the space, d is the dimensions, and c is the curvature of this space. Thus, the mapping process from Euclidean space to hyperbolic space is $exp_o^c : E_o\mathbb{P}^{d,c} \rightarrow \mathbb{P}^{d,c}$, and the opposite mapping is $log_o^c : \mathbb{P}^{d,c} \rightarrow E_o\mathbb{P}^{d,c}$. Specifically, for a node $a \in E_o\mathbb{P}^{d,c}$ and $a' \in \mathbb{P}^{d,c}$, we have: $exp_o^c(a) = a'$ and $log_o^c(a') = a$, where

$$exp_o^c(a) = tanh(\sqrt{c}\|a\|)\frac{a}{\sqrt{c}\|a\|}, \tag{1}$$

$$log_o^c(a') = artanh(\sqrt{c}\|a'\|)\frac{a'}{\sqrt{c}\|a'\|}. \tag{2}$$

For the hyperboloid model, we denote $E_o\mathbb{H}^{d,c}$ as the Euclidean space and $\mathbb{H}^{d,c}$ as the hyperbolic representation via the hyperboloid model and the exp_o^c and log_o^c functions are defined as:

$$exp_o^c(x) = cosh\left(\frac{\|x\|}{\sqrt{c}}\right)y' + \sqrt{c} \cdot sinh\left(\frac{\|x\|}{\sqrt{c}}\right)\frac{x}{\|x\|}, \tag{3}$$

$$log_o^c(x') = d_{\mathbb{H}}^c(x', y')\frac{y' + \frac{1}{c}\langle x', y'\rangle_{\mathcal{M}}x'}{\|y' + \frac{1}{c}\langle x', y'\rangle_{\mathcal{M}}x'\|}, \tag{4}$$

where $x', y' \in \mathbb{H}^{d,c}, x \in E_o\mathbb{H}^{d,c}$ with $x' \neq y', x \neq 0$, and $d_{\mathbb{H}}^c(\cdot)$ is the function calculates the distance between two nodes in hyperbolic space and $\langle ., . \rangle_{\mathcal{M}}$ is the Minkowski inner product.

4 METHODOLOGY

In this section, we introduce our framework: Heterogeneous Information Graph representation via Hyperbolic space combined with an Automatic Meta-path selection, called GraphHAM, an overview of this framework shown in Figure 1. In general, our framework contains two main models: the text model for text feature extraction and the Heterogeneous Information Network (HIN) model for HIN structural and semantic information learning. The overall process of our framework is shown in Algorithm 1.

4.1 HIN Modeling

For data preprocessing, our main purposes are (1) to make full use of social media data by extracting different types of elements from messages, and (2) to construct relationships between different types of elements. To meet the above purposes, we adopt a HIN, which is a graph that contains multiple nodes and the relationships between nodes. It can express different types of data in social media and the relationships between them. Let's take Twitter data as an example. When given a tweet t_i, we will extract the text information in t_i, delete special characters, emoji, and URL, and extract the named entities e_j and the rest of the words after removing the stop words in the text. Here, the named entities extraction is implemented using the "*en_core_web_sm*" model from spaCy [2]. Next, we will put t_i and e_j into HIN as a type of nodes, and add a type of edge between them. To better describe social media, we add users u_k as a node to HIN and connect them with the tweets they have sent or the users mentioned by tweets.

The current three types of nodes do not have feature vectors yet, so the next step is assigning them features. For the features of tweets, we use the pre-trained 300-d GloVe [31] vectors [2] to convert

[2]https://spacy.io/models/en

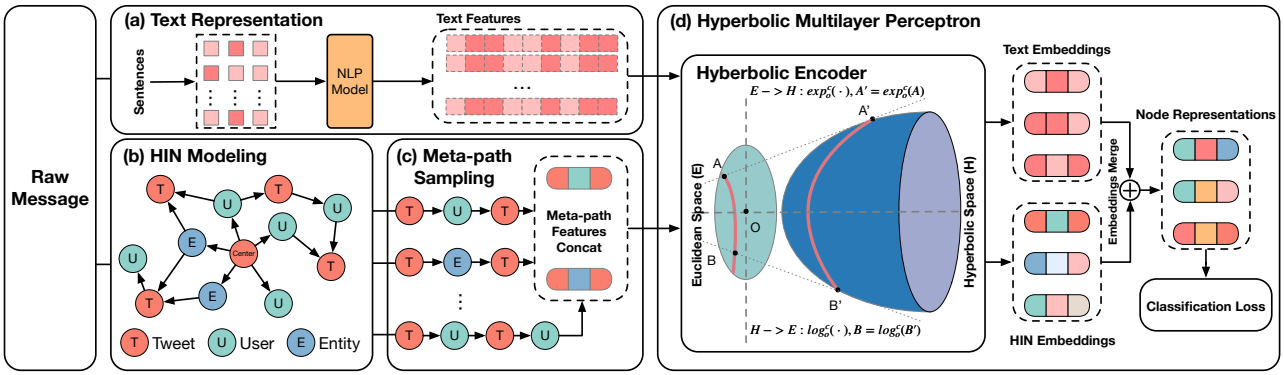


Figure 1: An overview of GraphHAM. GraphHAM contains four main parts: (a) Text Representation, (b) HIN Modeling, (c) Meta-path Sampling, and (d) Hyperbolic Nodes Embedding.

the words extracted from the tweets into feature vectors and assign them to the tweet node. In addition, as time is an indispensable feature, we also convert it into a feature vector and combine it with the word feature vector to assign it to the tweet node.

For entity nodes, we use similar operations as tweet nodes to convert entities into feature vectors using the pre-trained 300-d GloVe. The difference is the way it combines time vectors. Since there will be a large number of repeated entities in different tweets, and they are distributed at different time nodes, we convert all the time nodes where this entity appears into feature vectors and then add them to take the mean. This aims to better estimate the time when people are discussing the event a lot.

For user features, based on the protection of user information, we will not extract any information about the user, including but not limited to the user's name, gender, address, beliefs, followers, and other information related to the user. All we have is the user's ID. Therefore, we connect the tweets associated with each user ID, filter out users who are only connected to one tweet, and the features of the remaining users will be the average of the features of the tweets associated with them. In this way, we can also extract what events the user is more concerned about when there is little user information. After we operate on all tweets, HIN for social media data is constructed.

4.2 Meta-path Sampling

As mentioned before, GTN's automatic meta-path is enumerated from all nodes. Inspired by GraphMSE [21], we adopt the meta-path sampling part of GraphMSE. We only collect meta-path instances from the training set V_s, where $V_s \subset V$. Then, we intentionally discard some meta-path instances by limiting the number of neighbor nodes searched, shown in Figure 1. This alleviates over-fitting and over-smoothing caused by exploring all neighbor nodes [19].

4.3 Nodes Embedding

This section introduces how we embed meta-path instances into feature vectors. First, we introduce the Hyperbolic Multi-Layer Perceptron (HMLP), which contains the hyperbolic encoder inspired by HGCN [8]. We then introduce how to embed node features from the text model and the HIN model into hyperbolic space using HMLP in Section 4.3.2 – 4.3.4.

4.3.1 Hyperbolic Multi-Layer Perceptron (HMLP). To better learn tree-structured data, we use hyperbolic space as the low-dimensional space for embedding. However, as mentioned before, there is no vector calculation in hyperbolic space, so we cannot directly apply MLP in hyperbolic space. Therefore, for the implementation of HMLP, we need to project the hyperbolic embedding into Euclidean space for vector operations. For single-layer perceptron in Euclidean space, we have

$$\mathbb{E} = \sigma(\mathcal{W}X + b), \tag{5}$$

where $\mathcal{W}$ is a $V \times V$ weight matrix ($\mathcal{W} \in \mathbb{R}^{V*V}$), X is the node feature on Euclidean space, σ is the activate function and b is the bias. Based on the mapping relationship shown in the **Definition 3**, the vector operations on hyperbolic space like:

$$\mathcal{W} \otimes \mathcal{X} = exp_o^c(\mathcal{W} log_o^c(\mathcal{X})), \tag{6}$$

and

$$\mathcal{X} \oplus b = exp_o^c(log_o^c(\mathcal{X}) + b)), \tag{7}$$

where $\mathcal{X}$ is the node feature on hyperbolic space, o is the center of the space, and c is the curvature of the space. Thus, the single-layer perceptron on hyperbolic space is:

$$\mathbb{H}_{SLP} = exp_o^c(\sigma_1(log_o^c(\mathcal{W}_1 \otimes \mathcal{X} \oplus b_1))), \tag{8}$$

after that, we can deduce that the output of the two-layer perceptron is:

$$\mathbb{H}_{2LP} = exp_o^c(\sigma_2(log_o^c(\mathcal{W}_2 \otimes \mathbb{H}_{SLP} \oplus b_2))). \tag{9}$$

4.3.2 Meta-path Embedding. For a type of meta-path $\mathcal{P}_i$, there are several meta-path instances $p_j \in \mathcal{P}_i$ When we explore the neighboring nodes of point v. However, the features between different types of nodes are heterogeneous, so we cannot simply sum or average these features. Therefore, we concatenate these features according to order [13]:

$$\mathcal{X}_p = CONCAT(\mathcal{X}_{p_1}, \mathcal{X}_{p_2}, \cdots, \mathcal{X}_{p_n}). \tag{10}$$

Then, since there are meta-paths of different lengths and types, we use different HMLPs to correspond one-to-one with each type of

meta-path to align the output vectors. For a type of meta-path P_i from node v, we have:

$$\mathbb{H}_{\mathcal{P}_i}(v) = \sum HMLP_{\mathcal{P}_i}(CONCAT(\mathcal{X}_{p_1}, \mathcal{X}_{p_2}, \cdots, \mathcal{X}_{p_n})), \quad (11)$$

where, $\mathcal{P}_i = (p_1, p_2, \cdots, p_n)$. Then, we need to map this hyperbolic representation to Euclidean space for the downstream task:

$$H_{\mathcal{P}_i}(v) = \mathcal{D}(\mathbb{H}_{\mathcal{P}_i}(v)). \quad (12)$$

where $\mathcal{D}(\cdot)$ is a decoder for hyperbolic embedding.

4.3.3 Attention Layer. We adopt graph attention from GAT [37]. For a node v, we have:

$$h_v = \mathcal{W}_1 X_v + \sum_{\mathcal{P}_i \in \mathcal{P}} \delta_i H_{\mathcal{P}_i}(v), \quad (13)$$

$$\delta_i = \frac{1}{|V|} \sum_{v \in V} \delta_{vi}, \quad (14)$$

$$\delta_{vi} = \frac{exp(-e_{vi})}{\sum_i exp(-e_{vi})}, \quad (15)$$

$$e_{vi} = LeakyReLU(\delta^T tanh([h_v || H_{\mathcal{P}_i}(v)])). \quad (16)$$

Now, we have the HIN node representation $h_v, v \in V_s$ from $\mathcal{G}$ with attention weights of meta-paths $\alpha_i, \mathcal{P}_i \in \mathcal{P}$. The whole HIN representation is:

$$H_h = \sum_{v \in V_s} h_v. \quad (17)$$

4.3.4 Nodes Embedding Merge. Since we filter out users who are only related to one tweet during HIN modeling, we will delete some nodes, resulting in the loss of some information. So in this part, we use the text features of the tweet as the input of HMLP and learn the representation of all tweet nodes:

$$\mathbb{H}_t = HMLP(X_t), \quad (18)$$

$$H_t = \mathcal{D}(\mathbb{H}_t), \quad (19)$$

where H_t is the set of tweet representations, and X_t is the set of tweet features via Word2Vec. Finally, we generate the final node representations by combining the H_h and H_t:

$$H_f = H_h \oplus H_t. \quad (20)$$

here, $\oplus$ is the method for combining these representations, it could be sum, mean, or contact. The performance of these methods will be demonstrated in Section 5.3 ablation studies.

4.4 Objective Function

The objective function we use in this work is cross-entropy with labels L:

$$L' = softmax(H_f), \quad (21)$$

$$\mathcal{L}_{GraphHAM} = -\sum_{i=0}^{n} l_i log l'_i. \quad (22)$$

where, L' is the prediction labels from final nodes embedding after sofmax, and $l_i \in L$, $l'_i \in L'$.

Algorithm 1: GraphHAM

Input: A social message database $M = \{m_1, ..., m_n\}$; Text feature encoder: $t(\cdot)$; HIN encoder $h(\cdot)$; Hyperbolic decoder $\mathcal{D}(\cdot)$; Softmax layer $S(\cdot)$; Combine method $C(\cdot)$.

Output: Predicted social event label set $\{l'_0, l'_1, ..., l'_n\}$, where n is the number of classes.

```
for m_i ∈ M do
    Feature X, set of nodes N, set of edges E, train index I
      ← Data processing.
for e ∈ Epoch do
    HIN representations H_h ← D(h(N_i, E_i, X_i)), i ∈ I;
    Text representations H_t ← D(t(X));
    Final representations H_f ← C(H_h, H_t);
    Predict label l'_i = S(H_f);
    Cross-entropy loss L_GraphHAM = −Σ_{i=0}^{n} l_i log l'_i;
    Update parameters
```

5 EXPERIMENTS

In this section, we will present and discuss the experimental results. Our experiments focus on the node classification performance of our model under different settings. The experiments are designed to answer the following research questions:

- RQ 1: Is our model competitive compared to the baseline models?
- RQ 2: How does the training ratio impact our model?
- RQ 3: Can hyperbolic space improve model performance?
- RQ 4: What role do the text model and the HIN model play in the framework?
- RQ 5: How to choose the node embedding combination method?
- RQ 6: Why should meta-path be automatically selected?
- RQ 7: What impact do hyperparameters have on the model?

5.1 Experimental Settings

5.1.1 Datasets and Baselines. We use three published real-world datasets related to Twitter in our experiments: Kawarith [3], CrisisLexT6 [25] [3], and Twitter2012 [23] datasets. The statistics of these datasets are shown in Appendix A.1 (see Table 7).

It is worth noting that the performance of our model is closely related to the complexity of the dataset. We discuss it in Appendix A.2. Overall, the complexity of the datasets used in this study is arranged from complex to simple: Twitter2012 ≫ Kawarith > CrisisLexT6.

We compare our model with the following baselines: **TF-IDF** (2003) [34], **Word2Vec** (2013) [24], **FastText** (2016) [17], **Bert** (2018) [10], **GraphMSE** (2021) [21], and **FinEvent** (2022) [29]. All experiments are conducted on a Macbook Air with an M1 chip. The result reports the mean and standard deviations of 5 times experiments.

[3]For Kawarith and CrisisLexT6 datasets, the original datasets only published tweet ID, we retrieve the elements required by Section 4.1 through the Twitter API

Table 1: Comparison experiment results of node classification of all models.

Kawarith										
Tr. Ratio	5%		10%		20%		40%		70%	
Metrics	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1	Macro F1
TF-IDF	0.6285±0.0086	0.5217±0.1152	0.7791±0.0646	0.7193±0.0859	0.8382±0.0376	0.8125±0.0047	0.9105±0.0175	0.9015±0.0208	0.9287±0.0097	0.9249±0.0110
Word2Vec	0.5447±0.0252	0.4315±0.0223	0.5816±0.0048	0.4523±0.0055	0.5900±0.0045	0.4728±0.0130	0.6248±0.0052	0.5151±0.1260	0.6564±0.0080	0.5623±0.0091
FastText	0.2501±0.0000	0.0667±0.0000	0.2519±0.0001	0.0693±0.0004	0.5612±0.0027	0.3467±0.0011	0.7201±0.0000	0.5684±0.0023	0.8512±0.0019	0.8206±0.0032
BERT	0.6522±0.0122	0.6164±0.0181	0.6849±0.0097	0.6448±0.0104	0.7247±0.0043	0.7048±0.0067	0.7565±0.0072	0.7313±0.0075	0.7695±0.0092	0.7502±0.0101
GraphMSE	0.1720±0.0778	0.1232±0.0490	0.1382±0.0002	0.1216±0.0010	0.9105±0.0002	0.8966±0.0011	0.9330±0.0016	0.9218±0.0022	0.9470±0.0008	0.9400±0.0003
FinEvent	0.7986±0.0185	0.8001±0.0100	0.8632±0.0000	0.8566±0.0002	0.8898±0.0200	0.8813±0.0200	0.9066±0.0250	0.8964±0.0237	0.9259±0.0087	0.9136±0.0132
GraphHAM	**0.8512±0.0036**	**0.8262±0.0047**	**0.8818±0.0134**	**0.8632±0.0152**	**0.9151±0.0052**	**0.9026±0.0073**	**0.9340±0.0029**	**0.9224±0.0040**	**0.9510±0.0041**	**0.9457±0.0053**
CrisisLexT6										
TF-IDF	**0.9064±0.0126**	**0.9063±0.0126**	**0.9216±0.0056**	**0.9209±0.0057**	**0.9265±0.0036**	**0.9255±0.0037**	**0.9316±0.0025**	**0.9307±0.0026**	0.9365±0.0015	0.9356±0.0016
Word2Vec	0.7081±0.0062	0.7093±0.0068	0.7329±0.0057	0.7342±0.0047	0.7573±0.0027	0.7590±0.0030	0.7766±0.0053	0.7775±0.0053	0.7948±0.0043	0.7925±0.0034
FastText	0.6512±0.0072	0.6402±0.0120	0.8967±0.0000	0.8957±0.0000	0.9162±0.0002	0.9152±0.0001	0.9282±0.0004	0.9271±0.0004	**0.9387±0.0009**	**0.9380±0.0009**
BERT	0.7794±0.0054	0.7765±0.0058	0.7984±0.0053	0.7948±0.0055	0.8327±0.0018	0.8297±0.0017	0.8459±0.0028	0.8436±0.0029	0.8495±0.0038	0.8468±0.0036
GraphMSE	0.1475±0.0380	0.0997±0.0319	0.1538±0.0430	0.1136±0.0184	0.8911±0.0044	0.8656±0.0552	0.9065±0.0013	0.9061±0.0008	0.9100±0.0032	0.9094±0.0031
GraphHAM	0.8577±0.0220	0.8574±0.0219	0.8826±0.0094	0.8815±0.0096	0.9031±0.0010	0.9024±0.0017	0.9165±0.0017	0.9158±0.0017	0.9218±0.0027	0.9213±0.0023
Twitter2012										
TF-IDF	0.3059±0.0077	0.0721±0.0038	0.4384±0.0069	0.1527±0.0055	0.5610±0.0008	0.2295±0.0050	0.6393±0.0006	0.2935±0.0007	0.6789±0.0026	0.3405±0.0035
Word2Vec	0.4480±0.0063	0.1653±0.0053	0.5025±0.0019	0.2004±0.0023	0.5348±0.0040	0.2372±0.0050	0.5685±0.0039	0.2741±0.0051	0.5714±0.0054	0.2813±0.0047
FastText	0.0999±0.0000	0.0073±0.0000	0.0989±0.0004	0.0013±0.0000	0.1333±0.0008	0.0036±0.0000	0.1537±0.0001	0.0055±0.0000	0.1725±0.0000	0.0070±0.0000
BERT	0.4817±0.0031	0.2451±0.0056	0.5695±0.0056	0.3433±0.0067	0.6262±0.0041	0.4272±0.0055	0.6638±0.0062	0.4841±0.0176	0.6889±0.0032	0.5158±0.0052
GraphMSE	0.0015±0.0004	0.0006±0.0004	0.0017±0.0003	0.0009±0.0002	0.7494±0.0019	0.5633±0.0077	0.7856±0.0040	0.6343±0.0070	0.8057±0.0046	0.6716±0.0094
GraphHAM	**0.6887±0.0154**	**0.4492±0.0129**	**0.7538±0.0105**	**0.5457±0.0006**	**0.8028±0.0022**	**0.6392±0.0016**	**0.8290±0.0016**	**0.6776±0.0042**	**0.8414±0.0013**	**0.7100±0.0040**

5.2 GraphHAM Mode Performance Comparison (RQ 1 and RQ 2)

The experimental results of GraphHAM and baseline models are shown in Table 1. We set the training ratio at 5%, 10%, 20%, 40%, and 70% for node classification task. For each training ratio, we set the validation ratio to 10% and the rest to the testing ratio.

In general, we can see that GraphHAM performs better than all baseline models on the Kawarith and Twitter2012 datasets. Only on the CrisisLexT6 dataset, its performance is worse than the TF-IDF and FastText, but it still achieves the best performance among complex models. The reason is that the TF-IDF and FastText models are more suitable for datasets with simple data structures and fewer event types. Especially on the Twitter2012 dataset, FastText is basically unable to learn any information. In addition, compared to the strong baseline model GraphMSE, our model improves on all training ratios, especially at training ratios below 20%. At a training ratio of greater than or equal to 20%, it can also be 0.32%, 1.13%, and 4.42% higher than GraphMSE on the Kawarith, CrisislexT6 and Twitter2012 datasets respectively.

5.3 Ablation Studies

We conduct ablation experiments to verify the role of each component in the model, and we use a training ratio of 20% for experiments and report the mean and standard deviations of 5 times experiment results on the validation set.

5.3.1 GraphHAM Performance in Different Space (RQ 3). We run GraphHAM-PoincareBall, GraphHAM-Hyperboloid, and GraphHAM-Euclidean and test their node classification performance. GraphHAM-PoincareBall means the model embeds nodes into hyperbolic space via the Poincare Ball model. Similarly, the GraphHAM-Hyperboloid means model embeds nodes into the hyperbolic space through the Hyperboloid model. GraphHAM-Euclidean means the model embeds nodes into the Euclidean space. The performance is shown in Table 2. The hyperbolic space performs better than the Euclidean space, and, in general, the Poincare Ball model is more suitable for GraphHAM. Furthermore, the more complex the data, the more hyperbolic space improves the model. For example, on the CrisisLexT6 and Kawarith datasets, hyperbolic space improves by 3% and 4% compared to Euclidean space. For the more complex Twitter2012 dataset, the hyperbolic space improves the model's performance by 7%. Therefore, we believe that for complex data, hyperbolic space can help the model reduce the distortion caused by the feature embedding of tree-structured data.

Table 2: GraphHAM performance in different space.

Kawarith	Micro-F1	Macro-F1
GraphHAM-PincareBall	**0.9174±0.0015**	**0.9035±0.0008**
GraphHAM-Hyperboloid	0.9142±0.0182	0.8977±0.0136
GraphHAM-Euclidean	0.8734±0.0121	0.8628±0.0264
CrisisLexT6	Micro-F1	Macro-F1
GraphHAM-PincareBall	0.9035±0.0137	0.8924±0.0135
GraphHAM-Hyperboloid	**0.9193±0.0008**	**0.9069±0.0004**
GraphHAM-Euclidean	0.8882±0.0165	0.8735±0.0209
Twitter2012	Micro-F1	Macro-F1
GraphHAM-PincareBall	**0.8067±0.0034**	**0.6347±0.0040**
GraphHAM-Hyperboloid	0.7602±0.0004	0.6052±0.0016
GraphHAM-Euclidean	0.7333±0.0118	0.5787±0.0107

5.3.2 The Impact of the Text Model and the HIN Model (RQ 4). As mentioned before, GraphHAM has a text model as an auxiliary for heterogeneous information network (HIN) models. We train the

Table 3: The impact of different models in GraphHAM framework at 10%, 20% and 40% training ratio.

Training Ratio	10%		20%		40%	
Kawarith	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
GraphHAM	**0.9002±0.0015**	**0.8765±0.0028**	**0.9248±0.0030**	**0.9062±0.0070**	**0.9313±0.0000**	**0.9236±0.0003**
GraphHAM w/o Text Model	0.2478±0.0531	0.1704±0.0696	0.9109±0.0015	0.8913±0.0094	0.9217±0.0015	0.9045±0.0010
GraphHAM w/o HIN Model	0.8487±0.0015	0.8088±0.0016	0.8723±0.0075	0.8430±0.008	0.8798±0.0060	0.8592±0.0067
CrisisLexT6	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
GraphHAM	**0.9099±0.0049**	**0.8861±0.0010**	**0.9116±0.0016**	**0.9021±0.0015**	**0.924±0.0021**	**0.9206±0.0013**
GraphHAM w/o Text Model	0.1545±0.1108	0.1019±0.0792	0.9016±0.0024	0.8353±0.0870	0.9119±0.0020	0.9113±0.0005
GraphHAM w/o HIN Model	0.8332±0.0008	0.8293±0.0008	0.8692±0.0037	0.8665±0.0039	0.8841±0.0066	0.8844±0.0068
Twitter2012	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
GraphHAM	**0.7478±0.0045**	**0.5918±0.0540**	**0.8081±0.0011**	**0.6406±0.0016**	**0.8354±0.0015**	**0.6786±0.0021**
GraphHAM w/o Text Model	0.0031±0.0015	0.0003±0.0002	0.7663±0.0041	0.6028±0.0013	0.8133±0.0026	0.7091±0.0074
GraphHAM w/o HIN Model	0.4677±0.0032	0.2147±0.0015	0.5003±0.0002	0.2131±0.0020	0.4988±0.0059	0.2159±0.0028

Table 4: Nodes embedding combination methods.

Kawarith	Micro-F1	Macro-F1
GraphHAM-SUM	0.9013±0.0030	0.8706±0.0011
GraphHAM-MEAN	**0.9163±0.0060**	**0.9069±0.0020**
GraphHAM-CONCAT	0.8994±0.0242	0.8756±0.0353
CrisisLexT6	Micro-F1	Macro-F1
GraphHAM-SUM	0.9066±0.0062	0.8991±0.0050
GraphHAM-MEAN	0.9122±0.0016	**0.9021±0.0008**
GraphHAM-CONCAT	**0.9131±0.0012**	0.8388±0.0739
Twitter2012	Micro-F1	Macro-F1
GraphHAM-SUM	**0.8325±0.0064**	**0.6784±0.0001**
GraphHAM-MEAN	0.8086±0.0016	0.6388±0.0007
GraphHAM-CONCAT	0.8274±0.0026	0.6803±0.0015

text model, the HIN model, and the entire framework separately to compare their node classification performance. The results are shown in Table 3. We can find that when we train one of the models alone, the performance is not as good as when they are trained together. The HIN model accounts for a larger proportion than the text model at 20% training ratio. However, the role of these two models cannot be clearly seen by looking at the 20% training ratio, so we add the ratios near 20%: 10% and 40%. We can find that 20% exists as an interval point. When the training ratio is less than 20%, the text model plays a key role, and the HIN model is basically unable to learn features. When the training ratio reaches or even exceeds 20%, the HIN model begins to play a role and takes a dominant position, and the text model plays an auxiliary role in helping the entire framework learn information that the HIN model cannot capture. We further discuss the performance of the text model and the HIN model during training in the Appendix D. The results indicate that relying solely on either the HIN module or the text module is not viable. It is imperative to integrate both modules for effective performance.

5.3.3 The Impact of "SUM", "MEAN" and "CONCAT" Embedding Combination Methods on the Framework (RQ 5). We run GraphHAM-SUM, GraphHAM-MEAN, and GraphHAM-CONCAT to see the performance of node classifications on the three datasets. Table 4 shows the result of the experiments. It can be seen that for the Kawarith and CrisisLexT6 datasets, the "MEAN" method performs better, but for the Twitter2012 dataset, the "SUM" is the best-performing node representation combination method.

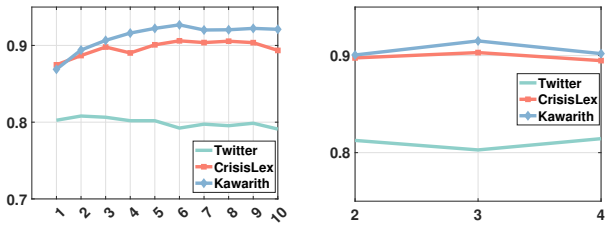
For such results, we believe that the performance of embedding combination methods is related to the complexity of the dataset. For data of simple complexity, such as the Kawarith and CrisisLexT6 datasets, the "CONCAT" or "MEAN" method is more suitable for our framework. Let's review Table 3 again. At a training ratio of 20%, the text and HIN models extract sufficient features. This is why the "CONCAT" or "MEAN" method can perform well in these two datasets: they can better extract the common features in the two models. However, for the complex dataset Twitter2012, we can see from the table that the performance of the text model and HIN model is not as good as the above two datasets. This also means that they extract a few common features. If we use "CONCAT" or "MEAN" to combine this set of features, some features will be lost for the entire framework. Therefore, the "SUM" method can better extract features for complex data.

Table 5: The impact of automatically selecting meta-path weights on model performance.

Kawarith	Micro-F1	Macro-F1
GraphHAM	**0.9157±0.0045**	**0.9021±0.0037**
GraphHAM w/o selection	0.8838±0.0002	0.8677±0.0004
CrisisLexT6	Micro-F1	Macro-F1
GraphHAM	**0.9025±0.0006**	**0.9022±0.0009**
GraphHAM w/o selection	0.8720±0.0005	0.8725±0.0006
Twitter2012	Micro-F1	Macro-F1
GraphHAM	**0.8041±0.0006**	**0.6387±0.0037**
GraphHAM w/o selection	0.7382±0.0006	0.5119±0.0005

5.3.4 The Impact of Automatically Selecting Meta-path (RQ 6). In this study, "automatic" indicates automatically optimizing the meta-path's weight. We conduct a control experiment by comparing the model without automatic meta-path selection. Our experiments primarily focus on a 20% training set across three datasets, and

the results are presented in Table 5. The results show that automatic meta-path selection contributes to the overall performance improvement of the model across all datasets. Micro-F1 scores have shown improvements of 3.6%, 3.5%, and 8.9% respectively.



(a) **Neiberhood limited τ** (b) **Meta-path length γ**

Figure 2: Performances with different parameters in all datasets at 20% training ratio.

5.4 Hyperparameter Analysis (RQ 7)

We analyze the hyperparameters of the model, which are the number of neighbor explorations around a node τ and the length of the meta-path γ. We set $\tau = 1, 2, 3, 4, \cdots, 9, 10$, the result shown in Figure 2(a). In general, the model with τ around 5 performs satisfactorily in all the datasets. Specifically, for the Kawarith and CrisisLexT6 datasets, when $\tau \leq 6$, the model's performance shows an increasing trend, and when $\tau > 6$, the model's performance fluctuates. For the Twitter2012 dataset, the best performance is when $\tau = 2$, but the performance when $\tau \leq 5$ is also satisfactory, and when $\tau > 5$, there is a downward trend.

In addition, we analyze the length of the meta-path γ. The time complexity of the model will increase exponentially as γ increases. What we want is to explore how long γ is required for our model to be satisfactory. The result is shown in the Figure 2(b). On the Kawarith and CrisislexT6 datasets, $\gamma = 3$ enables the model to achieve satisfactory performance. On the Twitter2012 dataset, $\gamma = 2$ can make the model perform well. Please refer to Appendix C for all parameter settings.

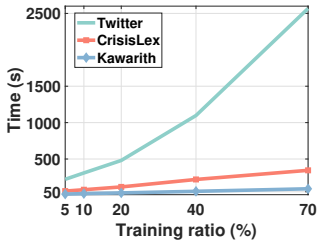


Figure 3: The relationship between training ratio and training time.

5.5 Time Efficiency

We analyze the efficiency of our model from two aspects. The first is the time required for our model training to reach convergence for different training ratios. As shown in Figure 3, we can see that in the smaller datasets, the Kawarith (4,860 tweets) and the CrisisLexT6 (18,157 tweets), our model's training time increases slowly as the training ratio increases. However, in the Twitter2012 (68,841 tweets) dataset, when the training ratio exceeds 20%, the training time will show a trend close to exponential growth. Therefore, for our model, the larger the dataset, the higher the time cost caused by increasing the training ratio.

Table 6: Time efficiency analysis for all models at 20% training ratio.

Models	Kawarith	CrisisLexT6	Twitter2012
TF-IDF	0.0719s	0.2065s	11.491s
Word2Vec	1.4112s	5.3205s	27.335s
FastText	0.1445s	2.0201s	32.543s
BERT	78.417s	312.61s	1303.1s
GraphMSE	0.2062s	1.0107s	2.5696s
FinEvent	3.0181s	–	–
GraphHAM	0.7929s	2.1031s	4.0710s

Then, we run all models on all datasets with a training ratio of 20%, and the test time is recorded. The results are shown in Table 6. Our model is only slightly slower than GraphMSE on all datasets. A significant improvement in performance (0.1720 to 0.8512 on the Kawarith dataset with 5% data as a training set) in exchange for an acceptable testing duration when compared to the GraphMSE model. Overall, the efficiency of our model is competitive among baseline models.

6 CONCLUSION

In this study, we propose a social media detection framework that reduces training data dependence: GraphHAM. Specifically, we ensure the model's learning ability in a small training ratio (5%) through a framework combining text and HIN models. In addition, since the model requires less training data and we use efficient sampling techniques, this greatly improves the overall efficiency of our model. Finally, we apply hyperbolic space representation to reduce the distortion caused by embedding tree-structured data in Euclidean space to improve model performance. Experimental results show that our model is highly competitive with existing social event detection models in terms of model performance and model efficiency. Compared to the baseline models, our model performs well with lower training ratios on all datasets we used, and performance improves steadily as the training ratio increases. Our model can achieve satisfactory performance when the training ratio reaches 20%. In the future, we plan to extend the model to online social media detection environments, including how to face incremental data streams and imbalanced datasets.

ACKNOWLEDGMENTS

This work was supported by the Australian Research Council Projects Nos. LP210301259, and DP230100899. J. Wu is the corresponding author.

REFERENCES

[1] Aaron B Adcock, Blair D Sullivan, and Michael W Mahoney. 2013. Tree-like structure in large social and information networks. In *2013 IEEE 13th international conference on data mining*. IEEE, Dallas, TX, USA, 1–10.

[2] Imad Afyouni, Zaher Al Aghbari, and Reshma Abdul Razack. 2022. Multi-feature, multi-modal, and multi-source social event detection: A comprehensive survey. *Information Fusion* 79 (2022), 279–308.

[3] Alaa Alharbi and Mark Lee. 2021. Kawarith: an Arabic Twitter corpus for crisis events. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*. Association for Computational Linguistics, Kyiv, Ukraine (Virtual), 42–52.

[4] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. 2016. Paper recommender systems: a literature survey. *International Journal on Digital Libraries* 17 (2016), 305–338.

[5] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[6] Yuwei Cao, Hao Peng, Jia Wu, Yingtong Dou, Jianxin Li, and Philip S Yu. 2021. Knowledge-preserving incremental social event detection via heterogeneous gnns. In *Proceedings of the Web Conference 2021*. ACM, Ljubljana, Slovenia, 3383–3395.

[7] Yuwei Cao, Hao Peng, Zhengtao Yu, and Philip S Yu. 2023. Hierarchical and Incremental Structural Entropy Minimization for Unsupervised Social Event Detection. *CoRR* abs/2312.11891 (2023), 13 pages. arXiv:2312.11891

[8] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. In *Advances in neural information processing systems*, Vol. 32. NeurIPS, Vancouver, BC, Canada, 4869–4880.

[9] Wanqiu Cui, Junping Du, Dawei Wang, Feifei Kou, and Zhe Xue. 2021. MVGAN: Multi-view graph attention network for social event detection. *ACM Transactions on Intelligent Systems and Technology (TIST)* 12, 3 (2021), 1–24.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Minneapolis, MN, USA, 4171–4186.

[11] Mateusz Fedoryszak, Brent Frederick, Vijay Rajaram, and Changtao Zhong. 2019. Real-time event detection on social data streams. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. ACM, Anchorage, AK, USA, 2774–2782.

[12] Xingcheng Fu, Jianxin Li, Jia Wu, Qingyun Sun, Cheng Ji, Senzhang Wang, Jiajun Tan, Hao Peng, and S Yu Philip. 2021. ACE-HGNN: adaptive curvature exploration hyperbolic graph neural network. In *2021 IEEE international conference on data mining (ICDM)*. IEEE, Auckland, New Zealand, 111–120.

[13] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*. ACM, Taipei, Taiwan, 2331–2341.

[14] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic entailment cones for learning hierarchical embeddings. In *International Conference on Machine Learning*. PMLR, Stockholm, Sweden, 1646–1655.

[15] Yuanyuan Guo, Zehua Zang, Hang Gao, Xiao Xu, Rui Wang, Lixiang Liu, and Jiangmeng Li. 2024. Unsupervised social event detection via hybrid graph contrastive learning and reinforced incremental clustering. *Knowledge-Based Systems* 284 (2024), 16 pages.

[16] Birger Iversen. 1992. *Hyperbolic geometry*. Cambridge University Press.

[17] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. FastText.zip: Compressing text classification models. *CoRR* abs/1612.03651 (2016), 13 pages.

[18] Chenliang Li, Haoran Wang, Zhiqian Zhang, Aixin Sun, and Zongyang Ma. 2016. Topic modeling for short texts with auxiliary word embeddings. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, Pisa, Italy, 165–174.

[19] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*. AAAI, New Orleans, Louisiana, USA, 3538–3545.

[20] Qian Li, Hao Peng, Jianxin Li, Jia Wu, Yuanxing Ning, Lihong Wang, S Yu Philip, and Zheng Wang. 2021. Reinforcement learning-based dialogue guided event extraction to exploit argument relations. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 30 (2021), 520–533.

[21] Yi Li, Yilun Jin, Guojie Song, Zihao Zhu, Chuan Shi, and Yiming Wang. 2021. GraphMSE: efficient meta-path selection in semantically aligned feature space for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI, Virtual Event, 4206–4214.

[22] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z Sheng, Hui Xiong, and Leman Akoglu. 2021. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering* 35, 12 (2021), 12012–12038.

[23] Andrew J McMinn, Yashar Moshfeghi, and Joemon M Jose. 2013. Building a large-scale corpus for evaluating event detection on twitter. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, San Francisco, CA, USA, 409–418.

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations*. ICLR, Scottsdale, Arizona, USA, 12 pages.

[25] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. 2014. Crisislex: A lexicon for collecting and filtering microblogged communications in crises. In *Proceedings of the international AAAI conference on web and social media*. AAAI, Ann Arbor, Michigan, USA, 376–385.

[26] Viktor Pekar, Jane Binner, Hossein Najafi, Chris Hale, and Vincent Schmidt. 2020. Early detection of heterogeneous disaster events using social media. *Journal of the Association for Information Science and Technology* 71, 1 (2020), 43–54.

[27] Hao Peng, Jianxin Li, Qiran Gong, Yangqiu Song, Yuanxing Ning, Kunfeng Lai, and Philip S Yu. 2019. Fine-grained event categorization with heterogeneous graph convolutional networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. IJCAI, Macao, China, 3238–3245.

[28] Hao Peng, Ruitong Zhang, Shaoning Li, Yuwei Cao, Shirui Pan, and S Yu Philip. 2022. Reinforced, incremental and cross-lingual event detection from social messages. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 1 (2022), 980–998.

[29] Hao Peng, Ruitong Zhang, Shaoning Li, Yuwei Cao, Shirui Pan, and S Yu Philip. 2022. Reinforced, incremental and cross-lingual event detection from social messages. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 1 (2022), 980–998.

[30] Wei Peng, Tuomas Varanka, Abdelrahman Mostafa, Henglin Shi, and Guoying Zhao. 2021. Hyperbolic deep neural networks: A survey. *IEEE Transactions on pattern analysis and machine intelligence* 44, 12 (2021), 10023–10044.

[31] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. ACL, Doha, Qatar, 1532–1543.

[32] Shengsheng Qian, Hong Chen, Dizhan Xue, Quan Fang, and Changsheng Xu. 2023. Open-World Social Event Classification. In *Proceedings of the ACM Web Conference 2023*. ACM, Austin, TX, USA, 1562–1571.

[33] Zitai Qiu, Jia Wu, Jian Yang, Xing Su, and Charu C Aggarwal. 2023. Heterogeneous Social Event Detection via Hyperbolic Graph Representations. *CoRR* abs/2302.10362 (2023), 14 pages. arXiv:2302.10362

[34] Juan Ramos. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*. Citeseer, 29–48.

[35] Tian Shi, Kyeongpil Kang, Jaegul Choo, and Chandan K Reddy. 2018. Short-text topic modeling via non-negative matrix factorization enriched with local word-context correlations. In *Proceedings of the 2018 World Wide Web Conference*. ACM, Lyon, France, 1105–1114.

[36] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment* 4, 11 (2011), 992–1003.

[37] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *6th International Conference on Learning Representations*. ICLR, Vancouver, BC, Canada, 12 pages.

[38] Chenguang Wang, Yangqiu Song, Haoran Li, Ming Zhang, and Jiawei Han. 2018. Unsupervised meta-path selection for text similarity measure based on heterogeneous information networks. *Data Mining and Knowledge Discovery* 32 (2018), 1735–1767.

[39] Hu Wang, Guansong Pang, Chunhua Shen, and Congbo Ma. 2019. Unsupervised representation learning by predicting random distances. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. IJCAI, Yokohama, Japan, 2950–2956.

[40] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The world wide web conference*. ACM, San Francisco, CA, USA, 2022–2032.

[41] Xiaokai Wei, Zhiwei Liu, Lichao Sun, and Philip S Yu. 2018. Unsupervised meta-path reduction on heterogeneous information networks. *CoRR* abs/1810.12503 (2018), 8 pages. arXiv:arXiv:1810.12503

[42] Haoran Yang, Hongxu Chen, Shirui Pan, Lin Li, Philip S Yu, and Guandong Xu. 2022. Dual space graph contrastive learning. In *Proceedings of the ACM Web Conference 2022*. ACM, Lyon, France, 1238–1247.

[43] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. In *Advances in Neural Information Processing Systems*. NeurIPS, Vancouver, BC, Canada, 11960–11970.

[44] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. ACM, Anchorage, AK, USA, 793–803.

[45] Han Zhou, Hongpeng Yin, Hengyi Zheng, and Yanxia Li. 2020. A survey on multi-modal social event detection. *Knowledge-Based Systems* 195 (2020), 105695.

A DATASET

A.1 Dataset Statistics

The statistical information of the datasets is shown in Table 7.

Table 7: The statistical information of datasets.

Dataset	Nodes	Features	classes
Kawarith	4,860	302	6
CrisisLexT6	18,157	302	6
Twitter2012	68,841	302	503

We list the node type of these datasets, and after HIN modeling, the statistics of HINs of these three datasets are shown in Table 8:

- **Kawarith**: Tweet (T), Entity (E), User (U).
- **CrisisLexT6**: Tweet (T), Entity (E), User (U).
- **Twitter2012**: Tweet (T), Entity (E), User (U).

Table 8: The statistical information of HINs.

Dataset	Nodes	Edges	Meta-path
Kawarith-HIN	9,743	14,181	TE, TU, TUT, TET
CrisisLexT6-HIN	31.625	41,415	TE, TU, TUT, TET
Twitter2012-HIN	98,070	148,979	TE, TU, TUT, TET

A.2 Dataset Complexity

Table 9: Class balancing in the datasets.

Dataset	"0"	"1"	"2"	"3"	"4"	"5"	"tweets/event"
Kawarith	626	516	1207	522	657	1332	810
CrisisLexT6	2734	3302	2864	2700	3389	3268	3026
Twitter2012	292	3358	496	752	390	528	137

The complexity of the dataset will affect the performance of the model to a certain extent. We will analyze the complexity of the dataset we use from the following two perspectives:

Based on class balance in dataset: The balance of data classes is a critical factor in classification tasks. However, imbalanced datasets are very common. To better illustrate the class balance in the datasets we used, we show the number of occurrences of different events in each dataset, as shown in Table 9 ("0", "1",..."5" represent events category). It is noteworthy that the Twitter2012 dataset comprises 503 event categories; however, in the table, we only display the quantities for the first six categories. From the table, we calculate an intermediate value, "tweets/event", which corresponds to a state of class balance, assuming each class occurs equally.

It is evident that CrisisLext is relatively more balanced in terms of class distribution compared to the other two datasets: the occurrences of each class are closer to the intermediate value. In contrast, Twitter2012 exhibits the least class balance among the three datasets. In general, when the categories in a dataset are more imbalanced, the challenges tend to become more complex.

(a) **Kawarith** (b) **CrisisLexT6** (c) **Twitter2012**

Figure 4: Datasets' word cloud.

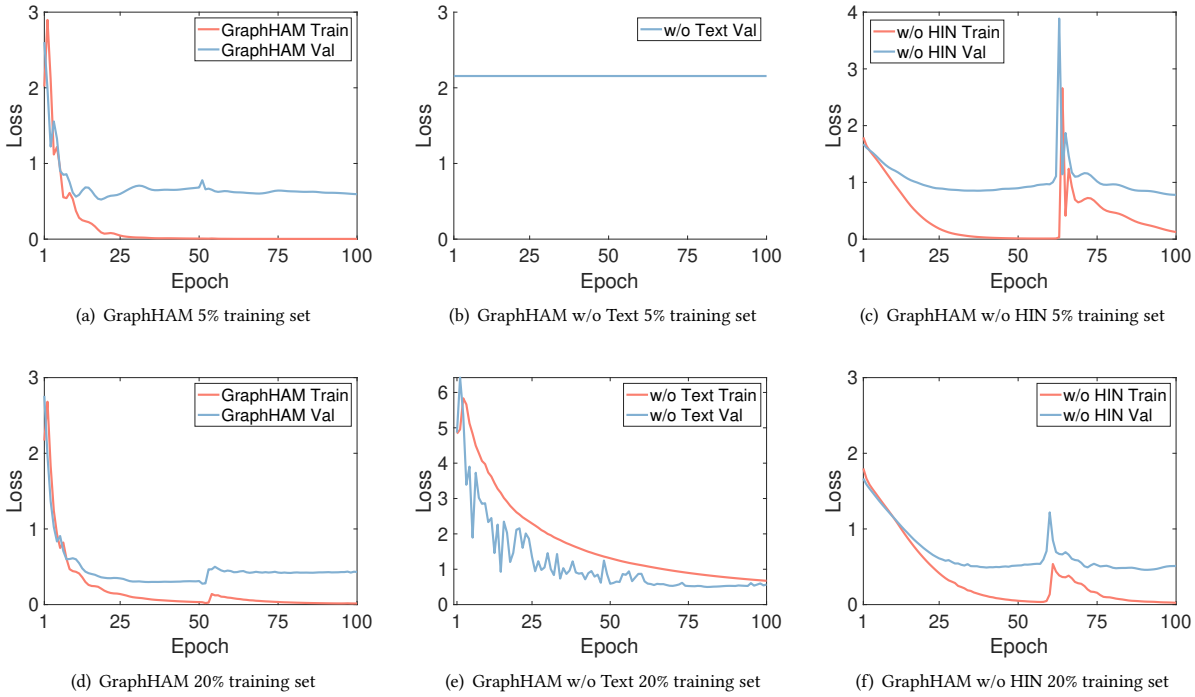
Based on word frequency: To visualize the complexity of the datasets, we employ word clouds to showcase the word frequencies. Word clouds depict the frequency of words through variations in font size, indicating the occurrence frequency, while the density of words per unit area reflects the overall quantity of words. As depicted in Figure 4, CrisisLext exhibits a relatively uniform and dense distribution of word frequencies. In contrast, both the Kawarith and Twitter2012 datasets display prominently high-frequency words, with Kawarith having a less dense distribution compared to Twitter2012. This indirectly suggests that the CrisisLext dataset is less complex than the Kawarith and Twitter2012 datasets. Because obvious high-frequency words may represent that some categories of events occur much more often than other events.

B BASELINES

We compare our model with the following baselines:

- **TF-IDF** [4] (2003) [34]: is based on the importance of word frequency and is widely used in data mining and text data recommendation [4];
- **Word2Vec** [5] (2013) [24]: is a method used by many social media detection models to understand the meaning behind the text;
- **FastText** [6] (2016) [17]: is a word vector calculation and text classification tool open-sourced by Facebook in 2016. It is not very innovative in academic terms, but its advantages are also very obvious. In text classification tasks, FastText can often achieve accuracy comparable to that of deep networks, but its training time is many orders of magnitude faster than deep networks.
- **Bert** [7] (2018) [10]: is the basis of many language learning models. It is widely used in many tasks that require text representation. Of course, this includes social event detection;
- **GraphMSE** [8] (2021) [21]: is the start-of-the-art heterogeneous graph representation learning model with automatically selects meta-path;
- **FinEvent** [9] (2022) [29]: is the start-of-the-art HIN social event detection model.

[4] https://scikit-learn.org/stable/modules/feature_extraction.html
[5] https://radimrehurek.com/gensim/models/word2vec.html
[6] https://fasttext.cc/docs/en/python-module.html
[7] https://huggingface.co/docs/transformers/model_doc/bert
[8] https://github.com/pkuliyi2015/GraphMSE
[9] https://github.com/RingBDStack/FinEvent



(a) GraphHAM 5% training set (b) GraphHAM w/o Text 5% training set (c) GraphHAM w/o HIN 5% training set

(d) GraphHAM 20% training set (e) GraphHAM w/o Text 20% training set (f) GraphHAM w/o HIN 20% training set

Figure 5: Training and validation losses for GraphHAM, GraphHAM w/o Text and GraphHAM w/o HIN models on 5% and 20% training sets of the Kawarith dataset.

C PARAMETER AND MODEL SETTINGS

We use 256-dimensional embeddings for the text model and 120-dimensional embeddings for the HIN model. For GraphHAM, we set the learning rate at 0.01, the neighborhood exploration limit number τ at 6 on the Kawarith and CrisisLexT6, 2 on the Twitter2012, and the meta-path length γ at 3 on the Kawarith and CrisisLexT6, 2 on the Twitter2012. For TD-IDF, Word2Vec, FastText, Bert, GraphMSE, and FinEvent we use the open-source implementations (see Appendix B). In particular, for the representation learning models TD-IDF, Word2Vec, and Bert, we only use pre-trained models to extract text representation in the three datasets and directly divide the extract representation into the same training ratio as we set before. Finally, enter the logistic regression classifier to classify the nodes without adding any additional operations. Furthermore, since FinEvent clusters events, our test metrics are for classification tasks. Therefore, we apply the method from [39] to map FinEvent clustering results to virtual labels and then convert them into a multi-classification problem. Note that we only ran the FinEvent model on the Kawarith dataset because out-of-memory issues due to graph message passing occurred on the other two datasets.

D LOSS DURING TRAINING

To explore how our model performs in training, we record the training and validation losses of the model on the Kawarith dataset at training ratios of 5% and 20%. We plot three plots illustrating the changes in training and validation losses for GraphHAM, GraphHAM w/o the text model, and w/o the HIN model shown in Figure 5. Through these plots, we observe that with a small training ratio (5%), the training loss for GraphHAM without the text model becomes NaN, and the validation loss remains unchanged. This suggests that the HIN module faces challenges in learning when confronted with extremely limited training data. When the training ratio reaches 20%, both GraphHAM without the text model and GraphHAM without the HIN model show the ability to learn features; however, compared with GraphHAM, their loss curves show a relatively slow decline. In contrast, the GraphHAM model achieved convergence in fewer epochs (25). Overall, the results show that when the training ratio is lower than 20%, the HIN model does not have enough meta-paths for training, and the learning ability of the framework comes from the text model. When the training ratio is greater than or equal to 20%, the HIN model plays a dominant role in the framework, while the text model plays a supporting role, providing the entire framework with features ignored by the HIN model.

View publication stats