

# Rumor Detection on Social Media with Reinforcement Learning-based Key Propagation Graph Generator

Yusong Zhang\*  
The Chinese University of Hong Kong  
Hong Kong  
1155107841@link.cuhk.edu.hk

Kun Xie\*  
The Chinese University of Hong Kong  
Hong Kong  
xiekun@se.cuhk.edu.hk

Xingyi Zhang  
Mohamed bin Zayed University of  
Artificial Intelligence  
United Arab Emirates  
xingyi.zhang@mbzuai.ac.ae

Xiangyu Dong  
The Chinese University of Hong Kong  
Hong Kong  
xydong@se.cuhk.edu.hk

Sibo Wang<sup>†</sup>  
The Chinese University of Hong Kong  
Hong Kong  
swang@se.cuhk.edu.hk

## Abstract

The spread of rumors on social media, particularly during significant events like the US elections and the COVID-19 pandemic, poses a serious threat to social stability and public health. Current rumor detection methods primarily rely on propagation graphs to improve the model performance. However, the effectiveness of these methods is often compromised by noisy and irrelevant structures in the propagation process. To tackle this issue, techniques such as weight adjustment and data augmentation have been proposed. However, they depend heavily on rich original propagation structures, limiting their effectiveness in handling rumors that lack sufficient propagation information, especially in the early stages of dissemination. In this work, we introduce *Key Propagation Graph Generator (KPG)*, a novel reinforcement learning-based framework, that generates contextually coherent and informative propagation patterns for events with insufficient topology information and identifies significant substructures in events with redundant and noisy propagation structures. KPG comprises two key components: the *Candidate Response Generator (CRG)* and the *Ending Node Selector (ENS)*. CRG learns latent variable distributions from refined propagation patterns to eliminate noise and generate new candidates for ENS, while ENS identifies the most influential substructures in propagation graphs and provides training data for CRG. Furthermore, we develop an end-to-end framework that utilizes rewards derived from a pre-trained graph neural network to guide the training process. The resulting key propagation graphs are then employed in downstream rumor detection tasks. Extensive experiments conducted on four datasets demonstrate that KPG outperforms current state-of-the-art methods.

## CCS Concepts

• **Information systems** → *Data mining*.

\*Yusong Zhang and Kun Xie are joint first authors.

<sup>†</sup>Sibo Wang is the corresponding author.



## Keywords

Rumor Detection, key propagation graph, reinforcement learning, graph neural network, response generator

### ACM Reference Format:

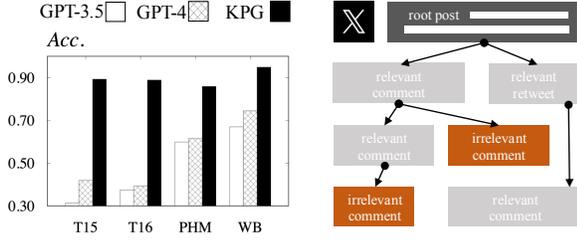
Yusong Zhang, Kun Xie, Xingyi Zhang, Xiangyu Dong, and Sibow Wang. 2025. Rumor Detection on Social Media with Reinforcement Learning-based Key Propagation Graph Generator. In *Proceedings of the ACM Web Conference 2025 (WWW '25)*, April 28-May 2, 2025, Sydney, NSW, Australia. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3696410.3714651>

## 1 Introduction

As a primary medium for information dissemination, social media eliminates temporal and spatial constraints on communication, facilitating the spread of information. However, the popularity of certain topics on social media often leads to the propagation of intentional or unintentional misinformation [87], negatively impacting individual health and social stability [72, 85]. Unfortunately, individuals who encounter unverified information often struggle to distinguish between truth and falsehood. This difficulty can inadvertently lead to the further spread of misinformation through social media platforms [46, 87] and word-of-mouth effect [17–19]. Additionally, this challenge extends to sophisticated tools like large language models, which also struggle to accurately identify rumors on social networks, as shown in the left subfigure of Fig. 1. The complexity of this issue underscores the need for more nuanced approaches to misinformation detection and management.

In recent years, rumor detection has gradually shifted from traditional machine learning approaches [40, 45, 69, 86], which require the manual definition of content, structural, or information source features, to deep learning approaches [28, 33, 38, 41, 63, 66, 79]. Notably, *graph neural networks (GNNs)* [12, 13, 31, 58, 80, 82] effectively combine node features and graph propagation topology, achieving superb detection performance [6, 21, 53, 65, 78]. As illustrated in the right subfigure of Fig. 1, root nodes in the propagation graphs represent the original posts, while other nodes represent comments or retweets received during the spread. Edges indicate the relationships of commenting and retweeting between nodes.

Although using propagation information improves rumor detection accuracy, irrelevant or untrustworthy comments on social networks significantly impair the reliability of propagation graphs



**Figure 1: Left: classification accuracy on four datasets. Right: an example propagation graph on the social network.**

in identifying the veracity of claims, as noted in [53, 65]. To address this challenge, techniques such as weight adjustment [64, 65] and data augmentation [21, 35, 53] have been introduced. However, these strategies rely on graphs with abundant structural information, which is often lacking in rumors, especially in the early stages of their spread on real-world social networks. Specifically, in the early stages of spread, interactions among social network users are limited due to the short duration of engagement. The original post has not yet garnered a significant number of comments or retweets, resulting in a relatively small propagation graph. This small size renders existing solution ineffective because they are based on edge reweighting or graph augmentation, which adjust the topology of the input propagation graph but cannot generate or expand the graph itself. Furthermore, as noted in recent studies [36, 83], existing graph neural networks used in rumor detection methods struggle to effectively address events with insufficient topological information. The absence of distinguishable structures in small graphs is the primary reason for unsatisfactory results [36]. Overall, designing an effective rumor detector that can handle both noisy propagation graphs with irrelevant comments and small propagation graphs in the early stage of spread remains an unresolved problem.

Motivated by these challenges, we propose KPG (Key Propagation Graph Generator) to generate enhanced key propagation graphs for rumor detection. Our KPG consists of two key components: the *Candidate Response Generator (CRG)* and the *Ending Node Selector (ENS)*. First, we propose the CRG module to address situations where propagation graphs are too small to provide sufficient candidates for selection. CRG generates new responses that are aligned with the structure and contextual information of the graph. These responses are derived from realistic yet refined noise propagation patterns, aiming to enhance the discrimination potential of the augmented graphs. Next, we introduce ENS to explore the candidate graph from both local and global perspectives. It identifies the key propagation graph by iteratively selecting critical nodes from the candidate set. Instead of choosing nodes based on the semantic relevance of comments, ENS evaluates each node based on the contribution of its feature and topology information to the classification performance of the event. By incorporating these two modules, our KPG can select distinguishable key patterns from the original noisy propagation graphs and expand small propagation graphs with informative new comments, thereby improving the overall model performance.

We further incorporate a *reinforcement learning (RL)* framework to integrate the candidate node generation and key node selection

processes with graph classification accuracy. To achieve this, we design rewards based on the classification accuracy obtained from a pre-trained *Graph Neural Network (GNN)* classifier. Specifically, we offer rewards for newly generated responsive features that show superior performance compared to the original features in terms of classification accuracy. Additionally, we encourage the selection of nodes that enhance the discriminative capability of the graph and maximize the expected cumulative performance improvement. Finally, we develop KPG as an end-to-end framework that trains the ENS and CRG alternately. This allows both components to generate intermediate results for each other and be optimized iteratively. On one hand, the CRG generates new responses to supplement the candidate set of the ENS, enabling ENS to overcome limitations imposed by insufficient input information. On the other hand, the key patterns identified by ENS serve as training samples for CRG, facilitating the extraction of latent variable distributions from realistic and distinguishable propagation patterns. The final key propagation graphs are then utilized to train a downstream GNN classifier for the rumor detection task.

In summary, our contributions are as follows:

- We propose KPG, a novel rumor detection model comprising two interdependent modules. It can augment small propagation graphs with contextually relevant, reliable responses and precisely select the indicative key nodes considering both feature and structural information.
- We incorporate the RL framework into propagation graph-based rumor detection. The carefully designed rewards improve discriminability during key graph generation process.
- Extensive experiments show that KPG achieves state-of-the-art performance in the rumor detection task.

## 2 Preliminaries

### 2.1 Notation

Let  $s = \{r, G = (V, E, X)\}$  denote an event, where  $r$  is the source post and  $G$  indicates the propagation graph of the source post.  $G$  is a directed acyclic tree rooted at the source post  $r$ . The node set  $V = \{r, v_1, v_2, \dots, v_{n_s-1}\}$  contains all  $n_s$  comments and retweets in the propagation process. The text stored in node  $u$  is denoted as  $\text{text}_u$ . The edge set  $E = \{(v_i, v_j) | v_i, v_j \in V\}$  represents the propagating relation between posts, i.e., there exists an directed edge  $(v_i, v_j) \in E$  if  $v_j$  is a comment or retweet of  $v_i$ .  $X \in \mathbb{R}^{n_s \times d}$  contains the initial features of posts in the event  $s$ . Each row of the feature matrix  $X[v_i]$  is a  $d$ -dimensional vector of the corresponding post  $v_i$ . Table 6 in Appendix A lists the frequently used notations in this paper.

**Rumor Detection.** Let  $y_s \in \mathcal{Y}$  be the label of the event  $s$ , indicating the veracity of the event. The label set  $\mathcal{Y}$  consists of non-rumors, false rumors, true rumors, and unverified rumors. Some datasets contain only two classes: rumors and non-rumors. Given a set of events  $\{s_0, s_1, \dots\}$  obtained from social networks, the goal of rumor detection is to predict the veracity of each event.

**Key Propagation Generation** We formulate the key propagation generation problem within the reinforcement learning framework. At step  $t$ , the *state* is the key propagation graph  $g_t = \{V_{g_t}, E_{g_t}, X_{g_t}\}$ , where  $|V_{g_t}| = n_t$  and  $|E_{g_t}| = m_t$ . The initial state  $g_0$  contains only the root node  $r$  and its feature  $X[r]$ . The *action*  $a_t = (v_t, e_t)$  represents the node and edge to be added to  $g_t$  for generating  $g_{t+1}$ .

## 2.2 Related Work

We review existing *propagation graph-based rumor detection models* in this section, deferring the discussion of other related work to Appendix B. Specifically, BiGCN [6] pioneered the application of GCN to enhance the exploitation of propagation graphs through both top-down and bottom-up directions. EBGCN [65] and FGCN [64] proposed edge weight adjustment mechanisms to capture the interaction intensity between comments. Subsequently, models including RDEA [21], GACL [53], TrustRD [35], and others [9, 38, 77] incorporated data augmentation strategies such as edge perturbation and masking to enhance model robustness through contrastive learning. Furthermore, researchers have introduced innovative techniques to improve the accuracy and reliability of rumor detection, including position-aware adversarial response generation [51], diverse counterfactual evidence exploitation [78], and semantic evolution extraction [55]. However, existing approaches heavily rely on original topology, limiting their effectiveness on small-scale graphs with insufficient structural information. In contrast, our KPG incorporates two interdependent modules that generate key patterns for propagation graphs of varying sizes, demonstrating superb performance in experimental evaluations.

## 3 Key Propagation Graph Generator

In this section, we present our KPG model, which comprises two key components: the *Candidate Response Generator (CRG)* and the *End Node Selector (ENS)*. To address the limitations of existing models in handling small input graphs with insufficient topological information, we propose the CRG module to generate realistic and reliable responses for expanding the propagation graph. Subsequently, the ENS module generates a probability distribution for each node, indicating the likelihood of the node being selected during the step-by-step construction of the key propagation graph. Specifically, we select propagation patterns that are critical and indicative for classification, while filtering out irrelevant and noisy comments. An RL framework is utilized to design rewards for both modules, further enhancing the model performance on the generated key propagation graphs. With the guidance of carefully designed rewards, the classification accuracy of the key propagation graph in the current state is encouraged to surpass that of its preceding state, ensuring continuous improvement throughout the generation process. The rest of this section is arranged as follows. First, we present the overall framework of KPG in Section 3.1. Then, we elaborate on CRG and ENS in Sections 3.2 and 3.3, respectively. Finally, we illustrate the reward functions used for guiding CRG and ENS, and present the learning algorithm in Section 3.4.

### 3.1 Overview

Figure 2 illustrates the KPG framework at step  $t$ , showing the generation process of the key propagation graph. At the bottom right of Figure 2, a toy example of a key propagation graph  $g_t$  with two nodes at step  $t$  is presented. Suppose the current candidate graph  $G_t$  is the graph at the middle left of Figure 2. If the number of candidates is less than  $\gamma$ , the CRG  $\pi_R(\cdot)$  generates new responses drawn from a learned latent distribution. In this example, node 5 is added into the new candidate graph  $G_{t+1}$ , aiding in generating the candidate set  $C_{t+1}$ . Next, the ENS  $\pi_S(\cdot)$  encodes the current

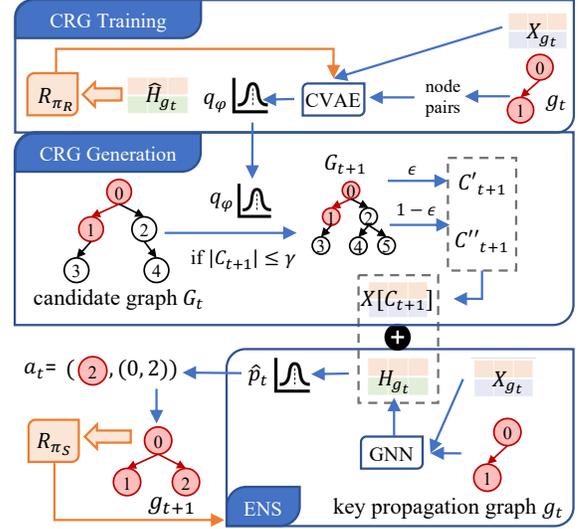


Figure 2: Framework of KPG.

graph  $g_t$  using a GNN and concatenates the node representations with candidate features  $X[C_{t+1}]$ . An MLP is then utilized to derive the probability of each available action  $a_t$ . For example, at the bottom left of Figure 2, graph  $g_{t+1}$  is updated based on action  $a_t = (2, (0, 2))$ . Finally, the rewards  $R_{\pi_S}$  and  $R_{\pi_R}$  are designed to increase the discriminative capacity of the generated key graph. Next, we elaborate on the CRG component in KPG.

### 3.2 Candidate Response Generator

In this section, we present the CRG component  $\pi_R(\cdot)$ . Existing data augmentation approaches primarily focus on perturbing the input structure. However, perturbation-based models struggle with small propagation graphs, which inherently lack sufficient topology and feature information. In contrast, CRG presents a novel approach by generating entirely new content as responses to the given nodes, thereby augmenting the graph in a contextually coherent manner. Specifically, CRG supports event classification from three aspects:

- (i) Contextually coherent: generating responses relevant to the original context in the propagation graph.
  - (ii) Realistic yet refined from noise: generating responses that reflect real-world social network propagation patterns while being robust to noisy and irrelevant comments.
  - (iii) Discriminative: generating distinct responses that contribute to rumor detection and downstream performance improvement.
- Among these objectives, we achieve (iii) through carefully designed rewards, discussed in Section 3.4. The mechanisms behind (i) and (ii) are detailed in this section.

To achieve objective (i), we construct a *conditional variational auto-encoder (CVAE)* [50]. This extension of the traditional VAE [30] incorporates condition variables to generate new responses following a latent distribution. By extracting this distribution from real-world propagation patterns, CVAE effectively captures the data distribution of existing responses, thereby facilitating the generation of responses that maintain contextual coherence.

To achieve objective (ii), we leverage the currently identified key propagation graph  $g_t$  rather than the original input graphs as training data for CRG, filtering out noisy and irrelevant comments. This approach allows CRG to learn from propagation patterns that significantly contribute to the classification task, alleviating the impact of noisy comments.

**CRG Training.** Let  $g_t$  be a key propagation graph at step  $t$ . We extract all pairs from the current key propagation graph  $g_t$  as the training data. Let  $(u, v) \in g_t$  represent a context-response node pair, where  $u$  denotes the *context* and  $v$  denotes the *response*. Initially, we employ a GRU [8] to encode the text of  $u$  and  $v$  into representations  $\mathbf{h}_u$  and  $\mathbf{h}_v$ , respectively. To address the one-to-many problem [50], we concatenate the latent representation of  $u$  with its source post  $r$  to strengthen the connection between  $\mathbf{h}_u$  and its central topic:

$$\begin{aligned} \mathbf{h}_u &= \text{CONCAT}(\text{GRU}(\text{text}_u), \text{GRU}(\text{text}_r)), \\ \mathbf{h}_v &= \text{GRU}(\text{text}_v). \end{aligned} \quad (1)$$

Next, representations  $\mathbf{h}_u$  and  $\mathbf{h}_v$  are fed into the CVAE model, which includes an encoder  $q_\varphi(\mathbf{z}|\mathbf{h}_u, \mathbf{h}_v)$  with latent space representation  $\mathbf{z}$ , an MLP decoder, and a GRU decoder. Assume that  $\mathbf{z}$  follows a Gaussian distribution  $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ . During the encoding phase,  $\mathbf{h}_u$  and  $\mathbf{h}_v$  are concatenated and fed into an MLP to learn distribution parameters of the encoder:

$$\boldsymbol{\mu} = \text{MLP}_\mu(\text{CONCAT}(\mathbf{h}_u, \mathbf{h}_v)), \quad \boldsymbol{\sigma}^2 = \text{MLP}_\sigma(\text{CONCAT}(\mathbf{h}_u, \mathbf{h}_v)).$$

A sample  $\mathbf{z}' \sim q_\varphi(\mathbf{z}|\mathbf{h}_u, \mathbf{h}_v)$  is then drawn from the learned distribution  $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$  using the reparameterization trick [50]. This sample is then combined with the context representation  $\mathbf{h}_u$  to reconstruct the response through the MLP decoder:

$$\hat{\mathbf{h}}_v = \text{MLP}_{dec}(\text{CONCAT}(\mathbf{z}', \mathbf{h}_u)). \quad (2)$$

The goal of CRG is to minimize the reconstruction error between representations of the generated response  $\hat{\mathbf{h}}_v$  and the original response  $\mathbf{h}_v$ , thereby extracting contextually coherent distributions. Detailed loss functions and rewards are presented in Section 3.4. Next, we elaborate on the candidate set generation process after the completion of the CRG training.

**CRG Generation.** In the generation phase of CRG, we set a threshold  $\gamma$  for the size of the candidate set, which is initialized using the original graph. If the number of candidate nodes exceeds  $\gamma$ , the candidate graph remains unchanged and is inherited in the next step. Otherwise, we uniformly sample nodes from the current candidate graph  $G_t$  as contexts and invoke the trained CRG to generate new response nodes for these selected context nodes until the candidate size requirement is satisfied.

Suppose at step  $t$ , the size of the current candidate set is less than or equal to  $\gamma$ , and we select a context node  $i$  from the candidate graph  $G_t$ . We first generate the topic-aware representation  $\mathbf{h}_i$  based on Eq. (1). Next,  $\mathbf{z}'$  is sampled following the learned distribution  $q_\varphi$ . We then combine  $\mathbf{z}'$  and  $\mathbf{h}_i$  to generate the response  $\mathbf{h}_j$  according to Eq. (2). The text of node  $i$  can be generated using the GRU decoder based on the decoded representation  $\mathbf{h}_j$ . After that, we add the node  $j$  and edge  $(i, j)$  to the candidate graph  $G_t$ . This process will continue until the size of candidate set exceeds  $\gamma$ .

After obtaining the updated candidate graph  $G_{t+1}$  with sufficient nodes, we select candidates for the current identified key propagation graph  $g_t$  using a hybrid selection method. Specifically, we

consider two types of candidates. The first type, referred to as *local candidates*, consists of the nodes that are directly connected to nodes in the current key propagation graph  $g_t$ :

$$C'_{t+1} = \{v \in V_{G_{t+1}} | (u, v) \in E_{G_{t+1}}, u \in V_{g_t}\}.$$

The second type of nodes comprises those that exist in the candidate graph  $G_{t+1}$  but are not present in the current propagation graph  $g_t$ . These nodes are referred to as *global candidates*:

$$C''_{t+1} = \{v \in V_{G_{t+1}} | v \notin V_{g_t}\}.$$

**Example.** In the example illustrated in Figure 2,  $G_{t+1}$  consists of 6 nodes, and the current key propagation graph  $g_t$  contains node 0 and node 1. Therefore, the local candidate set  $C'_{t+1} = \{2, 3\}$  and the global candidate set  $C''_{t+1} = \{2, 3, 4, 5\}$ .

By introducing the second type of nodes, we incorporate a restart mechanism that allows all unselected nodes to be considered during the node selection process. This approach strikes a balance between localized exploration around the existing key nodes and a global search across the entire candidate graph. Besides, a trade-off parameter  $\epsilon \in [0, 1]$  is introduced to control the balance between these two types of candidates. At each step  $t$ , we either set  $C_{t+1} = C'_{t+1}$  with a probability of  $\epsilon$  to locally explore the boundary of the current key propagation graph  $g_t$ , or set  $C_{t+1} = C''_{t+1}$  with a probability of  $(1 - \epsilon)$  to restart the exploration from one of the other nodes that have not been selected before, enabling a global search across of the entire graph. In our experiments, we set  $\epsilon = 0.8$  by default.

### 3.3 Ending Node Selector

In this section, we introduce the Ending Node Selector  $\pi_S(\cdot)$ . At each step  $t$ , we first adopt CRG to update the candidate graph from  $G_t$  to  $G_{t+1}$ . Subsequently, based on the current key propagation graph  $g_t$  and the candidate graph  $G_{t+1}$ , ENS predicts the probabilities of candidates being selected for action  $a_t$ . These probabilities are determined by the potential improvement in event classification performance. ENS aims to select the node that contributes most significantly to enhancing classification accuracy while disregarding those that are irrelevant to the veracity of the event.

To achieve this, we assess each node in the candidate set  $C_{t+1}$  based on its own features and its connection with the current key graph  $g_t$ . Specifically, ENS first encodes the current key propagation graph  $g_t$  using a GCN. Next, it augments the features of candidate nodes through concatenation operations. The augmented features of candidates are then fed into an MLP, followed by a softmax function, to generate the probability distribution for the next action  $a_t$ . Finally, the key propagation graph is updated accordingly.

**ENS Component.** We aggregate neighbor features from the graph structure using a *Graph Convolutional Network (GCN)* [31], such that both textual and topological information in the current key graph can be incorporated. Let  $\hat{\mathbf{A}}_{g_t}$  is the normalized adjacency matrix of  $g_t$  with self-loops and  $\mathbf{X}_{g_t}$  be the feature matrix associated with nodes in  $g_t$ , we derive the representation for nodes in  $g_t$  through a two-layer GCN model:

$$\mathbf{H}_{g_t} = \delta(\hat{\mathbf{A}}_{g_t} \delta(\hat{\mathbf{A}}_{g_t} \mathbf{X}_{g_t} \mathbf{W}_1) \mathbf{W}_2),$$

where  $\delta(\cdot)$  is the activation function and  $\mathbf{W}_1$  and  $\mathbf{W}_2$  are learnable parameter matrices.

Next, we augment the candidate features by concatenating the input candidate features with the representations of their parents in  $H_{g_t}$ . Given a candidate node  $v \in C_{t+1}$  with input feature  $X[v]$ , if its parent node  $v_p \in G_{t+1}$  also exists in the current propagation graph  $g_t$ , ENS directly concatenates its feature  $X[v]$  with the representation of  $v_p$ :

$$X'[v] = \text{CONCAT}(X[v], H_{g_t}[v_p]).$$

Otherwise, if the parent node  $v_p \notin V_{g_t}$ , we pad its feature with a zero vector:

$$X'[v] = \text{CONCAT}(X[v], \mathbf{0}).$$

After that, we employ an MLP with a softmax function to predict the probability of each candidate being selected. Let  $X'[C_{t+1}]$  be the collection of augmented features of candidates in set  $C_{t+1}$ . Then, the probability distribution  $\mathbf{p}_t$  is computed as follows:

$$\mathbf{p}_t = \text{softmax}(\text{MLP}(X'[C_{t+1}])).$$

Finally, based on the probability distribution  $\mathbf{p}_t$ , action  $a_t$  selects a new node  $v_t$  from  $C_{t+1}$  to be added into  $g_t$ . The probability of node  $v_t$  being selected by action  $a_t$ , denoted as  $\Pr[a_t = (v_t, e(v_t))]$ , is determined by the distribution  $\mathbf{p}_t$ :

$$\Pr[a_t = (v_t, e(v_t))] = \mathbf{p}_t[v_t], \quad \forall v_t \in C_{t+1}.$$

Here,  $e(v_t)$  represents the edge corresponding to  $v_t$ . To explain, if the parent node  $v_p$  of  $v_t$  exists in  $g_t$ , we set  $e(v_t) = (v_p, v_t)$ ; otherwise, we set  $e(v_t) = (r, v_t)$  to emphasize the importance of the root node  $r$ . The key propagation graph  $g_t$  is then updated to  $g_{t+1}$  by adding the newly selected node and edge in action  $a_t$ :

$$g_{t+1} = (V_{g_t} \cup v_t, E_{g_t} \cup e(v_t), X_{g_t} \cup X[v_t]).$$

**Remark.** As discussed in Section 3.2, when generating the candidate set  $C_{t+1}$ , CRG considers two types of candidates and introduces a trade-off parameter  $\epsilon$ . This strategy is similar to the restart probability in the random walk [14, 22, 23, 56, 59–62, 81]. We either select the next nodes locally from the out-neighbors of the currently selected nodes with  $\epsilon$  probability, or restart from another node in the remaining part of the graph  $G_{t+1}$  with  $(1 - \epsilon)$  probability, enabling a global exploration. It is important to note that when restarting from other nodes, it is possible to select a node  $v_{t+1}$  that is not directly connected to the current key propagation graph  $g_t$ . In such cases, we add an edge between the root node  $r$  and  $v_{t+1}$ . This allows CRG to explore additional key propagation patterns without being limited by the structure of the current graph.

### 3.4 Model Integration

In this section, we introduce the rewards designed to guide the training of both the ENS and CRG modules. The objective is to select actions that maximize the expected improvement in classification accuracy during the key propagation graph generation process.

**CRG Reward.** Recap from Section 3.2 that the CRG aims to generate coherent responses that align with the contextual information in the graph, thereby fulfilling objectives (i) and (ii). Besides, CRG seeks to select informative responses in the propagation graph that improve the overall classification performance, which corresponds to objective (iii). To achieve this, we design the reward according to the improvement of classification accuracy obtained by the updated

features of responses. Let  $y_s$  denote the ground-truth label of event  $s$ . The reward for CRG is defined as follows:

$$R_{\pi_R}^{(t)} = e^{-(f(g'_t)[y_s] - f(g_t)[y_s])}, \quad (3)$$

where  $f(\cdot)$  is a BiGCN classifier with primary discrimination ability.  $g'_t$  is the current key propagation graph with original features, and  $g_t$  is the graph that shares the same topology as  $g'_t$  but is associated with updated features generated by CRG. If CRG successfully extracts informative features during training, i.e., the updated features lead to higher classification accuracy than the original features, we reward the model; otherwise, we impose a penalty. Specifically, if the generated responses contribute to the improved classification confidence, we have  $R_{\pi_R}^{(t)} < 1$ , encouraging the module to maintain its well-trained state. In contrast, if the responses are not informative, we obtain  $R_{\pi_R}^{(t)} \geq 1$ , pushing the module away from its current unsatisfactory state.

Besides, we adopt the stochastic gradient variational Bayes framework to optimize the reconstruction error and the KL divergence between the variational distribution and the prior distribution. The loss function of CRG is defined as follows:

$$\mathcal{L}_{\pi_R}^{(t)} = R_{\pi_R}^{(t)} \cdot \Sigma(\mathbb{E}_{q_\phi(z|\mathbf{h}_u, \mathbf{h}_v)}[\log p_\theta(\mathbf{h}_v|z, \mathbf{h}_u)] - \text{KL}(q_\phi(z|\mathbf{h}_u, \mathbf{h}_v) \parallel p_\theta(z|\mathbf{h}_u))). \quad (4)$$

**ENS Reward.** The ENS reward aims to generate a more discriminative key propagation graph after adding a new node. To achieve this, we design the reward of ENS from two aspects: (i) the improvement of the prediction score and (ii) the enhancement of future performance estimation. Specifically, we aim to achieve a higher prediction score for class  $y_s$  derived from the classifier  $f(\cdot)$  on the updated key graph  $g_{t+1}$  compared to  $g_t$ . Simultaneously, we aim to maximize the future performance of  $g_{t+1}$  in the subsequent steps.

To simulate future actions, we employ a modified Rollout [5] to select up to  $(l - 1)$  additional nodes from the current candidate set. These  $l$  nodes, including the selected one, are incrementally added to the current key propagation graph  $g_t$ , thereby simulating potential actions over the next  $l$  steps. We then combine the current and the future prediction scores to calculate an overall score for the updated key propagation graph  $g_{t+1}$ :

$$r_{t+1} = \frac{1}{2} \left( f(g_{t+1})[y_s] + \frac{1}{l} \left( \sum_{i=1}^l f(\hat{g}_{t+i})[y_s] \right) \right),$$

where  $\hat{g}_{t+i}$  is the estimated future graph after  $i$  steps. The reward for ENS is defined as the margin between two scores:

$$R_{\pi_S}^{(t)} = e^{-(r_{t+1} - r_t)}.$$

Similar to the rewards in CRG, if the performance improves after action  $a_t$ , we obtain  $R_{\pi_S}^{(t)} < 1$ . Additionally, we introduce another reward to penalize the module if the classification performance on the current key propagation graph is unsatisfactory:

$$\bar{R}_{\pi_S}^{(t)} = 1.5 - f(g_{t+1})[y_s].$$

If the prediction score on class  $y_s$  falls below 0.5, then  $\bar{R}_{\pi_S}^{(t)} \geq 1$ , which penalizes the ENS module. The final loss of ENS is derived by combining this penalty with the cross-entropy loss of the GNN:

$$\mathcal{L}_{\pi_S}^{(t)} = \bar{R}_{\pi_S}^{(t)} \cdot R_{\pi_S}^{(t)} \cdot L_{CE}.$$

**Training Pipeline.** ENS and CRG modules are interdependent within the KPG framework. On one hand, ENS relies on CRG to generate new responses, ensuring that there are sufficient nodes in the candidate set for selection. On the other hand, CRG requires ENS to provide effective context-response pairs to accurately capture the latent variable distribution of responses. Therefore, we employ an alternative training approach that enables end-to-end learning of both modules. Specifically, we first fix CRG  $\pi_R$  and update ENS  $\pi_S$  by minimizing  $\mathcal{L}_{\pi_S}$  using Eq. 3.4 through policy gradient [54]. Next, we fix ENS  $\pi_S$  and update CRG  $\pi_R$  by minimizing  $\mathcal{L}_{\pi_R}$  using Eq. 4. This iterative process continues until either the early stop condition is met or the maximum number of steps is reached. Algorithm 1 summarizes the learning pipeline of KPG.

To enhance generalization and mitigate over-fitting, we adopt batch training along with an early stop strategy. The final key propagation graphs are used to train a new BiGCN classifier for downstream rumor detection tasks. Following GACL [53], we concatenate the texts of the source post and comments in the propagation graph as the input text for a BERT classifier [11]. The results from two classifiers are fused using the mean operation to derive the final classification results. Additionally, we feed the training events to the model in descending order based on the sizes of their propagation graphs. This strategy ensures that both ENS and CRG can learn from informative propagation patterns at the beginning stage and better handle limited-spread events with small graphs.

**Time Complexity.** The time complexity of KPG per epoch is  $O(L(Mh + Nh^2))$ , where  $N$  and  $M$  denote the total nodes and edges across all graphs, respectively,  $L$  is the maximum number of generation steps, and  $h$  is the dimension of hidden layers. The overall cost is primarily determined by the step number  $L$  and the cost of the GNN model. In addition, the step number  $L$  is a tunable parameter that can be adjusted to balance processing time and model accuracy. Experimental results on the Twitter16 dataset in Table 5 show that KPG achieves superior classification accuracy compared to other baseline methods, even with small  $L$  values.

## 4 Experiment

In this section, we compare our KPG against 12 state-of-the-art competitors on 4 datasets<sup>1</sup>. We also conduct early-stage rumor detection and the ablation study to demonstrate the effectiveness of each component of KPG. Finally, we perform the parameter analysis to examine the impact of the parameters on the model performance.

### 4.1 Datasets

We evaluate KPG on three real-world benchmark datasets: Twitter15 [40], Twitter16 [40], and PHEME [89]. Twitter15 and Twitter16 are collected from the Twitter platform and divided into four rumor classes: *non-rumor (NR)*, *true rumor (TR)*, *false rumor (FR)*, and *unverified rumor (UR)*. The PHEME dataset, also derived from Twitter, is related to five events and is annotated with two labels: *rumor (R)* and *non-rumor (NR)*. These datasets are commonly used in existing research studies [6, 41, 53] on rumor detection.

The Weibo dataset used in [6, 39] lacks critical information required for several baselines, including original comment text and

author details. However, these details cannot be fully recovered due to the presence of many deleted user accounts and posts on Weibo. In addition, the propagation patterns and topics of rumor in real-world social networks have evolved rapidly. To address these issues, we have constructed a new dataset **Weibo22**, which is also available in our code repository. Events in Weibo22 are collected from Sina Weibo, one of the largest social media platforms in China. The dataset covers events from November 2019 to March 2022, with more than half of the events related to the COVID-19 pandemic. Specifically, the events in Weibo22 are divided into two categories, rumor and non-rumor, based on information provided by Weibo Community Management Center [4] and China Internet Joint Rumor Debunking Platform [2]. By using the Weibo API [3], we collected 4,174 source posts with 960,000 microblogs, including reposts and comments in their propagation graphs. For each microblog, we also collected user profile information, including the number of followers, number of friends, verification status, verification type, and verification reason. Detailed statistics of these four datasets are shown in Table 7.

### 4.2 Experimental Settings

**Baselines.** We compare KPG with twelve state-of-the-art baselines, including BERT, RvNN, BiGCN, EBGCN, RDEA, GACL, TrustRD, SMG, AdaSNN, DCE-RD, GLAN, SMAN, and SBAG. Detailed descriptions of these baselines can be found in Appendix D.

**Metrics and evaluation protocol.** We adopt Accuracy (Acc.) and micro  $F_1$  score ( $F_1$ ) for each class as our evaluation metrics. Notice that several baselines [6, 21, 35, 53, 65] utilize a batch-wise averaging approach. However, this may introduce performance variance, particularly when the final batch is smaller than the others. For example, consider a dataset with 110 records and a batch size of 100. If the accuracy for the first batch is 0.8 and for the second batch is 0.9, the batch-wise averaging yields a result of 0.85, obviously skewed by the smaller second batch. To ensure a reliable evaluation, we calculate the average accuracy across all batches in each epoch, mitigating any impact from variations in the final batch size.

### 4.3 Experimental Results

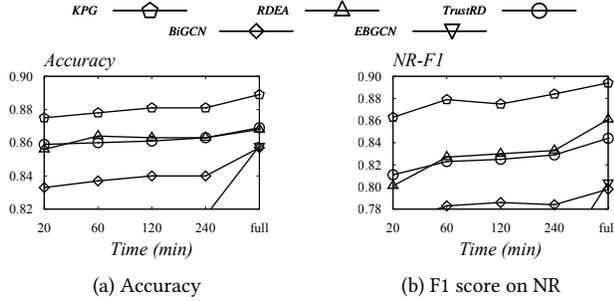
Table 1 shows the performance of each model on four real-world datasets. Note that we exclude any baseline that cannot complete detection within 7 days. As observed, our KPG consistently achieves the best accuracy across all datasets, demonstrating the effectiveness and generalization capability of our proposed RL-based key propagation graph generation method. On the Twitter15 and Twitter16 datasets, our KPG outperforms the second-best results by achieving an increase of 2.7% in the  $F_1$  score for the non-rumor class. Additionally, on the PHEME and Weibo22 datasets, KPG leads by 1.4% and 2.9% in the  $F_1$  score for the rumor class compared to the second-best performances.

Moreover, KPG achieves the highest average ranking, highlighting its superior ability to generalize across diverse datasets and metrics. Specifically, compared to TrustRD, the model with the second highest average rank, KPG takes the lead by 2.7% and 2% in terms of accuracy on Twitter15 and Twitter16 datasets, respectively. Furthermore, KPG outperforms models that use additional user information, including GLAN, SMAN, and SBAG. Compared

<sup>1</sup>Our code is available at <https://github.com/kkkkk001/KPG>

**Table 1: Results on four datasets. NR, FR, TR, UR, and R represents for non-rumor, false rumor, true rumor, unverified rumor, and rumor, respectively. Best and second-best results are highlighted with bold and underlined text. We exclude methods that cannot finish detection within 7 days. The last column reports the average rank over all datasets and metrics of each method.**

	Twitter15					Twitter16					Pheme			Weibo22			Rank
	Acc.	NR- $F_1$	FR- $F_1$	TR- $F_1$	UR- $F_1$	Acc.	NR- $F_1$	FR- $F_1$	TR- $F_1$	UR- $F_1$	Acc.	R- $F_1$	NR- $F_1$	Acc.	R- $F_1$	NR- $F_1$	
BERT	0.784	0.841	0.771	0.811	0.714	0.753	0.804	0.660	0.837	0.699	0.816	0.809	0.815	0.887	0.888	0.886	12.1
RvNN	0.793	0.817	0.791	0.822	0.745	0.783	0.755	0.737	0.849	0.783	0.768	0.774	0.762	0.858	0.858	0.857	12.4
BiGCN	0.837	0.813	0.846	0.892	0.798	0.857	0.798	<b>0.836</b>	0.923	0.873	0.841	0.842	0.841	0.905	0.907	0.904	6.1
EBGCN	0.851	0.827	0.864	0.891	<u>0.825</u>	0.858	0.804	0.835	0.920	<u>0.874</u>	0.839	0.840	0.837	0.870	0.876	0.864	6.6
RDEA	0.860	0.872	0.866	0.888	0.815	0.868	0.861	0.817	0.920	0.871	0.838	0.840	0.837	<u>0.918</u>	<u>0.920</u>	<u>0.917</u>	5.1
GACL	0.785	0.877	0.740	0.780	0.738	0.758	0.814	0.686	0.801	0.719	0.836	0.839	0.833	0.885	0.884	0.886	10.8
TrustRD	<u>0.866</u>	0.875	<u>0.871</u>	<u>0.896</u>	0.821	0.869	0.844	0.827	<u>0.929</u>	0.874	<u>0.846</u>	0.846	<u>0.846</u>	<u>0.918</u>	<u>0.920</u>	<u>0.917</u>	3.1
SMG	0.828	0.861	0.840	0.869	0.736	0.841	0.812	0.812	0.898	0.839	0.830	0.830	0.830	-	-	-	9.4
AdaSNN	0.798	0.763	0.767	0.844	0.772	0.792	0.711	0.783	0.871	0.800	0.820	0.811	0.827	-	-	-	11.8
DCE-RD	0.827	0.850	0.817	0.855	0.776	0.826	0.808	0.785	0.884	0.821	0.834	0.833	0.835	-	-	-	12.5
GLAN	0.827	0.820	0.839	0.871	0.779	0.831	0.756	0.805	0.923	0.843	0.845	<u>0.849</u>	0.840	0.902	0.903	0.902	7.6
SMAN	0.853	<u>0.894</u>	0.851	0.860	0.806	0.853	<u>0.867</u>	0.788	0.914	0.842	0.836	<u>0.837</u>	0.835	0.911	0.912	0.909	6.6
SBAG	0.862	0.876	0.862	0.887	0.821	<u>0.870</u>	<u>0.863</u>	<b>0.836</b>	0.913	0.866	0.841	0.842	0.841	0.912	0.912	0.911	4.4
KPG	<b>0.893</b>	<b>0.921</b>	<b>0.898</b>	<b>0.903</b>	<b>0.847</b>	<b>0.889</b>	<b>0.894</b>	<b>0.836</b>	<b>0.930</b>	<b>0.896</b>	<b>0.859</b>	<b>0.863</b>	<b>0.854</b>	<b>0.949</b>	<b>0.949</b>	<b>0.948</b>	1.0



**Figure 3: Early stage rumor detection on Twitter16.**

to these three models, KPG leads by up to 6.6% on Twitter15 and 5.8% on Twitter16 in terms of accuracy, further showcasing the effectiveness of KPG.

On our newly collected dataset, Weibo22, KPG outperforms all competitors across all metrics. Compared to the second-best baseline, KPG takes a lead by 3.1%, 2.9%, and 3.1% in terms of accuracy,  $F_1$  score for rumor, and  $F_1$  score for non-rumor, respectively. This again demonstrates that our key graph generation method effectively reduces noise while extracting more useful information.

#### 4.4 Early Stage Rumor Detection

We also conduct experiments to validate the performance of KPG in detecting rumors during the early stage of their spread. To achieve this, we set a temporal threshold  $\Delta$  to filter nodes within each propagation graph. Specifically, for every node in a propagation graph, we compute the time difference between the publication time of the comment (or retweet) and the creation time of the root claim. Nodes with a time difference less than  $\Delta$  are retained, while those exceeding  $\Delta$  are excluded. We conduct experiments with varying  $\Delta \in \{20, 60, 120, 240\}$  minutes. We compare KPG with the top four baselines in terms of average rank, excluding methods that require additional author information. The results for TrustRD, RDEA, BiGCN, EBGCN, and KPG in terms of accuracy and  $F_1$  score on non-rumor events are presented in Figure 3. The results for  $F_1$  scores on the other three classes are similar to those of the NR class and thus are omitted for brevity.

**Table 2: Ablation study on Twitter15.**

	Acc.	NR- $F_1$	FR- $F_1$	TR- $F_1$	UR- $F_1$
KPG	<b>0.893</b>	<b>0.921</b>	<b>0.898</b>	<b>0.903</b>	<b>0.847</b>
KPG\ens	0.886	0.920	0.877	0.900	0.847
KPG\crg	0.880	0.916	0.869	0.892	0.843
KPG\rreward	0.883	0.915	0.875	0.896	0.845

**Table 3: Ablation study on Twitter16.**

	Acc.	NR- $F_1$	FR- $F_1$	TR- $F_1$	UR- $F_1$
KPG	<b>0.889</b>	<b>0.894</b>	<b>0.836</b>	<b>0.930</b>	<b>0.896</b>
KPG\ens	0.865	0.882	0.810	0.910	0.855
KPG\crg	0.861	0.878	0.806	0.908	0.849
KPG\rreward	0.865	0.880	0.808	0.914	0.855

As we can see, during the early stage of spread, the textual features and topology information contained in the propagation graphs diminish, leading to a general decrease performance. However, the performance of our KPG consistently surpasses that of other baselines across varying  $\Delta$  values. When  $\Delta = 20$ , KPG outperforms TrustRD, the second-best baseline, in both accuracy and  $F_1$  score for the non-rumor class, even though TrustRD utilizes full propagation graphs. This superior performance further demonstrates the effectiveness of our KPG, particularly the CRG module, which generates realistic and informative responses and plays a crucial role in early-stage rumor detection.

#### 4.5 Ablation Study and Parameter Analysis

In this section, we first examine the effectiveness of each submodule in KPG, and then analyze the effects of hyperparameters.

**Ablation Study.** We implement three variants of KPG:

- KPG\ens: it randomly selects out-neighbors of existing nodes;
- KPG\crg: it removes the CRG module and stops generating extra candidate nodes for small propagation graphs;
- KPG\rreward: it eliminates rewards during training.

Tables 2-3 report the results of KPG and its three variants on the Twitter15 and Twitter16 datasets, respectively. As we can observe, the absence of any component leads to a decrease in rumor detection

**Table 4: Results with varying  $\epsilon$  on Twitter16.**

$\epsilon$	1.0	0.8	0.6	0.4	0.2	0.0
Acc.	0.880	0.889	0.880	0.878	0.880	0.880
NR- $F_1$	0.884	0.894	0.881	0.884	0.880	0.888
FR- $F_1$	0.831	0.836	0.827	0.824	0.831	0.827
TR- $F_1$	0.922	0.930	0.927	0.922	0.924	0.923
UR- $F_1$	0.884	0.896	0.885	0.882	0.884	0.883

**Table 5: Results with varying  $\tau$  and  $l$  on Twitter16.**

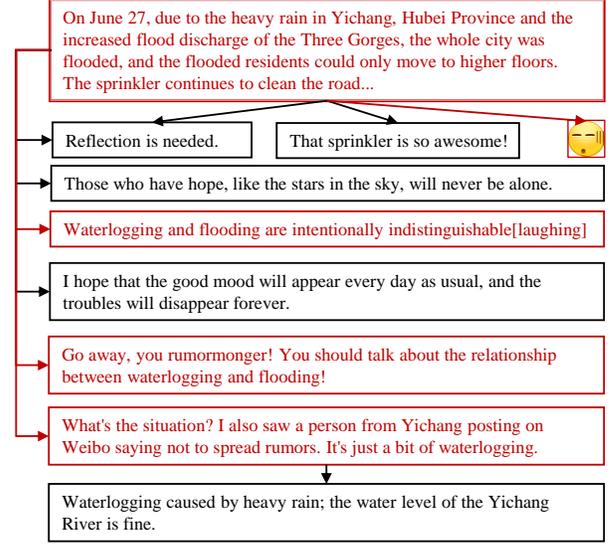
	$\tau$				$l$		
	0	$2^1$	$2^2$	$2^3$	0	5	10
Acc.	0.832	0.875	0.883	0.889	0.879	0.872	0.889
NR- $F_1$	0.763	0.878	0.896	0.896	0.891	0.870	0.894
FR- $F_1$	0.819	0.825	0.824	0.836	0.822	0.823	0.836
TR- $F_1$	0.887	0.923	0.932	0.930	0.919	0.918	0.930
UR- $F_1$	0.859	0.872	0.880	0.896	0.882	0.875	0.896

performance. This decline not only demonstrates the effectiveness of each module but also underscores the importance of synergy of all three components.

**The Effect of Parameter  $\epsilon$  in Candidate Selection.** The parameter  $\epsilon \in [0, 1]$  serves as a trade-off between local and global candidates. A larger  $\epsilon$  indicates a higher probability for ENS to select the next node from the local neighbors of currently selected nodes, while a smaller  $\epsilon$  increases the probability of selecting from nodes that have not yet been selected. We vary the parameter  $\epsilon$  from 0 to 1 and report the corresponding results on Twitter16 in Table 4. The results on other datasets are similar to those on Twitter16 and thus are omitted. In general, the model performance is influenced by the balance between local and global exploration. Specifically, on Twitter16, optimal performance is achieved when  $\epsilon = 0.8$ . A similar performance trend is observed across other datasets. Therefore, we set  $\epsilon = 0.8$  for KPG in our experiments.

**The Effect of the Maximum Generation Steps.** Table 5 reports the experimental results on Twitter16 with varying  $\tau$ , which controls the maximum size of the generated key propagation graph. We set the maximum generation steps to  $\tau$  times the median size of original graphs, where  $\tau$  is chosen from  $\{0, 2^1, 2^2, 2^3\}$ . When  $\tau = 0$ , only the root node is used for classification, resulting in the worst performance across all metrics. This highlights the importance of leveraging propagation graphs for effective rumor detection. As  $\tau$  increases, we observe a general performance improvement across all metrics. When  $\tau = 2^3$ , KPG achieves peak performance on Twitter16. This can be attributed to the relatively small average graph size in the Twitter16 dataset. The results of the ablation study further support this observation. Without the CRG component, reaching the maximum generation step becomes challenging, resulting in the most significant performance decline among three variants. Compared to other baselines, KPG achieves superior accuracy across various  $\tau$  values, even when  $\tau$  is relatively small.

**The Effect of the Maximum Step  $l$  in Modified Rollout.** Table 5 also reports the performance on Twitter16 with varying  $l$ , which is the maximum steps of the Rollout used to derive rewards for ENS. We conduct experiments with  $l \in \{0, 5, 10\}$ . As we can observe,

**Figure 4: An example of the generated propagation graph.**

the model performance peaks at  $l = 10$ , which demonstrates the effectiveness of evaluating each node from a long-term perspective.

## 4.6 Case Study

To illustrate the functionality of our KPG, we present the key propagation graph of a widespread rumor [1] from the Weibo22 dataset in Figure 4. In the original propagation graph, the event cannot be correctly classified by BiGCN. However, with the key propagation graph generated by KPG, the rumor can be correctly identified. Our observations indicate that KPG selects nodes (highlighted in red) that enhance the discriminative capability of rumor detection while disregarding irrelevant and noisy nodes. In particular, strong emotional responses are selected to refute the rumor.

## 5 Conclusion

In this paper, we propose KPG, a reinforcement learning-based model for rumor detection, comprised of two components: ENS and CRG. With the cooperation of two modules, KPG effectively identifies indicative substructures for events with noisy propagation information and generates realistic, reliable, and informative responses for events with insufficient propagation. The two components are trained alternately in an end-to-end framework, guided by our carefully designed rewards, to improve the discriminative ability of the generated key propagation graphs. Extensive experiments conducted on four real-world datasets demonstrate that KPG achieves state-of-the-art performance. Additionally, we construct a new dataset, Weibo22, which contains posts with more recent dates and topics, potentially contributing to the development of rumor detection-related research.

## Acknowledgments

This work is supported by the RGC GRF grant (No. 14217322), Hong Kong ITC ITF grant (No. MRP/071/20X), and Tencent Rhino-Bird Focused Research Grant.

## References

- [1] A rumor. [https://m.thepaper.cn/newsDetail\\_forward\\_8568258](https://m.thepaper.cn/newsDetail_forward_8568258).
- [2] China Internet Joint Rumor Debunking Platform. <https://www.piyao.org.cn/>.
- [3] Weibo API. <https://open.weibo.com/wiki/API>.
- [4] Weibo Community Management Center. <https://service.account.weibo.com/>.
- [5] D.P. Bertsekas. 1999. Rollout algorithms: an overview. In *Proceedings of the 38th IEEE Conference on Decision and Control*, Vol. 1. 448–449 vol.1.
- [6] Tian Bian, Xi Xiao, Tingyang Xu, Peilin Zhao, Wenbing Huang, Yu Rong, and Junzhou Huang. 2020. Rumor Detection on Social Media with Bi-Directional Graph Convolutional Networks. In *AAAI*. 549–556.
- [7] Zefeng Cai and Zerui Cai. 2022. PCVAE: Generating Prior Context for Dialogue Response Generation. In *IJCAI*. 4065–4071.
- [8] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.
- [9] Chaoqun Cui and Caiyan Jia. 2024. Propagation Tree Is Not Deep: Adaptive Graph Contrastive Learning Approach for Rumor Detection. In *AAAI*. 73–81.
- [10] Jian Cui, Kwanwoo Kim, Seung Ho Na, and Seungwon Shin. 2022. Meta-Path-based Fake News Detection Leveraging Multi-level Social Context Information. In *CIKM*. 325–334.
- [11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.
- [12] Xiangyu Dong, Xingyi Zhang, Yanni Sun, Lei Chen, Mingxuan Yuan, and Sibow Wang. 2024. SmoothGNN: Smoothing-based GNN for Unsupervised Node Anomaly Detection. *CoRR* abs/2405.17525 (2024).
- [13] Xiangyu Dong, Xingyi Zhang, and Sibow Wang. 2024. Rayleigh Quotient Graph Neural Networks for Graph-level Anomaly Detection. In *ICLR*.
- [14] Xinyu Du, Xingyi Zhang, Sibow Wang, and Zengfeng Huang. 2023. Efficient Tree-SVD for Subset Node Embedding over Large Dynamic Graphs. *Proc. ACM Manag. Data* 1, 1 (2023), 96:1–96:26.
- [15] Yajuan Duan, Zhimin Chen, Furu Wei, Ming Zhou, and Heung-Yeung Shum. 2012. Twitter Topic Summarization by Ranking Tweets using Social Influence and Content Quality. In *COLING*. 763–780.
- [16] Faezeh Faez, Yassaman Omimi, Mahdih Soleymani Baghshah, and Hamid R. Rabiee. 2021. Deep Graph Generators: A Survey. *IEEE Access* 9 (2021), 106675–106702.
- [17] Chen Feng, Xingguang Chen, Qintian Guo, Fangyuan Zhang, and Sibow Wang. 2024. Efficient Approximation Algorithms for Minimum Cost Seed Selection with Probabilistic Coverage Guarantee. *Proceedings of the ACM on Management of Data* 2, 4 (2024), 1–26.
- [18] Qintian Guo, Chen Feng, Fangyuan Zhang, and Sibow Wang. 2023. Efficient Algorithm for Budgeted Adaptive Influence Maximization: An Incremental RR-set Update Approach. *Proceedings of the ACM on Management of Data* 1, 3 (2023), 1–26.
- [19] Qintian Guo, Sibow Wang, Zhewei Wei, Wenqing Lin, and Jing Tang. 2022. Influence maximization revisited: efficient sampling with bound tightened. *TODS* 47, 3 (2022), 1–45.
- [20] Xiaojie Guo and Liang Zhao. 2023. A Systematic Survey on Deep Generative Models for Graph Generation. *TPAMI* 45, 5 (2023), 5370–5390.
- [21] Zhenyu He, Ce Li, Fan Zhou, and Yi Yang. 2021. Rumor Detection on Social Media with Event Augmentations. In *SIGIR*. 2020–2024.
- [22] Guanhao Hou, Xingguang Chen, Sibow Wang, and Zhewei Wei. 2021. Massively parallel algorithms for personalized pagerank. *PVLDB* 14, 9 (2021), 1668–1680.
- [23] Guanhao Hou, Qintian Guo, Fangyuan Zhang, Sibow Wang, and Zhewei Wei. 2023. Personalized pagerank on evolving graphs with an incremental index-update scheme. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–26.
- [24] Zhen Huang, Zhilong Lv, Xiaoyun Han, Binyang Li, Menglong Lu, and Dongsheng Li. 2022. Social Bot-Aware Graph Neural Network for Early Rumor Detection. In *COLING*. 6680–6690.
- [25] Yiqiao Jin, Xiting Wang, Ruichao Yang, Yizhou Sun, Wei Wang, Hao Liao, and Xing Xie. 2022. Towards Fine-Grained Reasoning for Fake News Detection. In *AAAI*. 5746–5754.
- [26] Mostafa Karimi, Arman Hasanzadeh, and Yang Shen. 2020. Network-principled deep generative models for designing drug combinations as graph sets. *Bioinformatics* 36 (2020), i445–i454.
- [27] Yash Khemchandani, Stephen O’Hagan, Soumitra Samanta, Neil Swainston, Timothy J. Roberts, Danushka Bollegala, and Douglas B. Kell. 2020. DeepGraphMolGen, a multi-objective, computational strategy for generating molecules with desirable properties: a graph convolution and reinforcement learning approach. *J. Cheminformatics* 12, 1 (2020), 53.
- [28] Ling Min Serena Khoo, Hai Leong Chiew, Zhong Qian, and Jing Jiang. 2020. Interpretable Rumor Detection in Microblogs by Attending to User Interactions. In *AAAI*. 8783–8790.
- [29] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
- [30] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *ICLR*.
- [31] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *ICLR*.
- [32] Jianxin Li, Qingyun Sun, Hao Peng, Beining Yang, Jia Wu, and Philip S. Yu. 2023. Adaptive Subgraph Neural Network With Reinforced Critical Structure Mining. *IEEE Trans. Pattern Anal. Mach. Intell.* 45, 7 (2023), 8063–8080.
- [33] Quanzhi Li, Qiong Zhang, and Luo Si. 2019. Rumor Detection by Exploiting User Credibility Information, Attention and Multi-task Learning. In *ACL*. 1173–1179.
- [34] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter W. Battaglia. 2018. Learning Deep Generative Models of Graphs. *CoRR* abs/1803.03324 (2018).
- [35] Leyuan Liu, Junyi Chen, Zhangtao Cheng, Wenxin Tai, and Fan Zhou. 2023. Towards Trustworthy Rumor Detection with Interpretable Graph Structural Learning. In *CIKM*. 4089–4093.
- [36] Zemin Liu, Qiheng Mao, Chenghao Liu, Yuan Fang, and Jianling Sun. 2022. On Size-Oriented Long-Tailed Graph Classification of Graph Neural Networks. In *WWW*. 1506–1516.
- [37] Yi-Ju Lu and Cheng-Te Li. 2020. GCAN: Graph-aware Co-Attention Networks for Explainable Fake News Detection on Social Media. In *ACL*. 505–514.
- [38] Guanghui Ma, Chunming Hu, Ling Ge, Junfan Chen, Hong Zhang, and Richong Zhang. 2022. Towards Robust False Information Detection on Social Networks with Contrastive Learning. In *CIKM*. 1441–1450.
- [39] Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting Rumors from Microblogs with Recurrent Neural Networks. In *IJCAI*. 3818–3824.
- [40] Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect Rumors in Microblog Posts Using Propagation Structure via Kernel Learning. In *ACL*. 708–717.
- [41] Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Rumor Detection on Twitter with Tree-structured Recursive Neural Networks. In *ACL*. 1980–1989.
- [42] Mariya Popova, Mykhailo Shvets, Junier Oliva, and Olexandr Isayev. 2019. MolecularRNN: Generating realistic molecular graphs with optimized properties. *arXiv preprint arXiv:1905.13372* (2019).
- [43] Feng Qian, Chengyue Gong, Karishma Sharma, and Yan Liu. 2018. Neural User Response Generator: Fake News Detection with Collective User Intelligence. In *IJCAI*. 3834–3840.
- [44] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *ICML*. 1278–1286.
- [45] Nir Rosenfeld, Aron Szanto, and David C. Parkes. 2020. A Kernel of Truth: Determining Rumor Veracity on Twitter by Diffusion Pattern Alone. In *WWW*. 1018–1028.
- [46] Victoria L. Rubin. 2010. On deception and deception detection: Content analysis of computer-mediated stated beliefs. In *ASIST*, Vol. 47. 1–10.
- [47] Iulian Vlad Serban, Alessandro Sordani, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues. In *AAAI*. 3295–3301.
- [48] Xiaoyu Shen, Hui Su, Shuzi Niu, and Vera Demberg. 2018. Improving Variational Encoder-Decoders in Dialogue Generation. In *AAAI*. 5456–5463.
- [49] Fangzhou Shi, Shan You, and Chang Xu. 2019. Reinforced Molecule Generation with Heterogeneous States. In *ICDM*. 548–557.
- [50] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning Structured Output Representation using Deep Conditional Generative Models. In *NeurIPS*. 3483–3491.
- [51] Yun-Zhu Song, Yi-Syuan Chen, Yi-Ting Chang, Shao-Yu Weng, and Hong-Han Shuai. 2021. Adversary-Aware Rumor Detection. In *ACL/IJCNLP*. 1371–1382.
- [52] Bin Sun, Shaoxiong Feng, Yiwei Li, Jiamou Liu, and Kan Li. 2021. Generating Relevant and Coherent Dialogue Responses using Self-Separated Conditional Variational AutoEncoders. In *ACL/IJCNLP*. 5624–5637.
- [53] Tiensun Sun, Zhong Qian, Sujun Dong, Peifeng Li, and Qiaoming Zhu. 2022. Rumor Detection on Social Media with Graph Adversarial Contrastive Learning. In *WWW*. 2789–2797.
- [54] Richard S. Sutton, David A. McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *NeurIPS*. 1057–1063.
- [55] Xiang Tao, Liang Wang, Qiang Liu, Shu Wu, and Liang Wang. 2024. Semantic Evolution Enhanced Graph Autoencoder for Rumor Detection. In *WWW*. 4150–4159.
- [56] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast Random Walk with Restart and Its Applications. In *ICDM*. 613–622.
- [57] Rakshit Trivedi, Jiachen Yang, and Hongyuan Zha. 2020. GraphOpt: Learning Optimization Models of Graph Formation. In *ICML*. 9603–9613.
- [58] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
- [59] Runhui Wang, Sibow Wang, and Xiaofang Zhou. 2019. Parallelizing approximate single-source personalized PageRank queries on shared memory. *Vldb J.* 28, 6 (2019), 923–940.

- [60] Sibow Wang, Youze Tang, Xiaokui Xiao, Yin Yang, and Zengxiang Li. 2016. HubPPR: Effective Indexing for Approximate Personalized PageRank. *Proc. VLDB Endow.* 10, 3 (2016), 205–216.
- [61] Sibow Wang, Renchi Yang, Runhui Wang, Xiaokui Xiao, Zhewei Wei, Wenqing Lin, Yin Yang, and Nan Tang. 2019. Efficient Algorithms for Approximate Single-Source Personalized PageRank Queries. *ACM Trans. Database Syst.* 44, 4 (2019), 18:1–18:37.
- [62] Sibow Wang, Renchi Yang, Xiaokui Xiao, Zhewei Wei, and Yin Yang. 2017. FORA: Simple and Effective Approximate Single-Source Personalized PageRank. In *SIGKDD*. 505–514.
- [63] Yaqing Wang, Fenglong Ma, Zhiwei Jin, Ye Yuan, Guangxu Xun, Kishlay Jha, Lu Su, and Jing Gao. 2018. EANN: Event Adversarial Neural Networks for Multimodal Fake News Detection. In *SIGKDD*. 849–857.
- [64] Lingwei Wei, Dou Hu, Wei Zhou, Xin Wang, and Songlin Hu. 2024. Modeling the Uncertainty of Information Propagation for Rumor Detection: A Neuro-Fuzzy Approach. *IEEE Trans. Neural Networks Learn. Syst.* 35, 2 (2024), 2522–2533.
- [65] Lingwei Wei, Dou Hu, Wei Zhou, Zhaojun Yue, and Songlin Hu. 2021. Towards Propagation Uncertainty: Edge-enhanced Bayesian Graph Convolutional Networks for Rumor Detection. In *ACL/IJCNLP (1)*. 3845–3854.
- [66] Penghui Wei, Nan Xu, and Wenji Mao. 2019. Modeling Conversation Structure and Temporal Dynamics for Jointly Predicting Rumor Stance and Veracity. In *EMNLP-IJCNLP*. 4786–4797.
- [67] Bowen Wu, Mengyuan Li, Zongsheng Wang, Yifu Chen, Derek F. Wong, Qihang Feng, Junhong Huang, and Baoxun Wang. 2020. Guiding Variational Response Generator to Exploit Persona. In *ACL*. 53–65.
- [68] Jiaying Wu and Bryan Hooi. 2023. Decor: Degree-corrected social graph refinement for fake news detection. In *SIGKDD*. 2582–2593.
- [69] Ke Wu, Song Yang, and Kenny Q. Zhu. 2015. False rumors detection on Sina Weibo by propagation structures. In *ICDE*. 651–662.
- [70] Mingqi Yang, Yanming Shen, Heng Qi, and Baocai Yin. 2021. Soft-mask: Adaptive Substructure Extractions for Graph Neural Networks. In *WWW*. 2058–2068.
- [71] Ruichao Yang, Xiting Wang, Yiqiao Jin, Chaozhao Li, Jianxun Lian, and Xing Xie. 2022. Reinforcement Subgraph Reasoning for Fake News Detection. In *SIGKDD*. 2253–2262.
- [72] Shuo Yang, Kai Shu, Suhang Wang, Renjie Gu, Fan Wu, and Huan Liu. 2019. Unsupervised Fake News Detection on Social Media: A Generative Approach. In *AAAI*. 5644–5651.
- [73] Jiakuan You, Bowen Liu, Zhitao Ying, Vijay S. Pande, and Jure Leskovec. 2018. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. In *NeurIPS*. 6412–6422.
- [74] Jiakuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models. In *ICML*. 5694–5703.
- [75] Chunyuan Yuan, Qianwen Ma, Wei Zhou, Jizhong Han, and Songlin Hu. 2019. Jointly Embedding the Local and Global Relations of Heterogeneous Graph for Rumor Detection. In *ICDM*. 796–805.
- [76] Chunyuan Yuan, Qianwen Ma, Wei Zhou, Jizhong Han, and Songlin Hu. 2020. Early Detection of Fake News by Utilizing the Credibility of News, Publishers, and Users based on Weakly Supervised Learning. In *COLING*. 5444–5454.
- [77] Guixian Zhang, Rongjiao Liang, Zhongyi Yu, and Shichao Zhang. 2022. Rumour Detection on Social Media with Long-Tail Strategy. In *IJCNN*. 1–8.
- [78] Kaiwei Zhang, Junchi Yu, Haichao Shi, Jian Liang, and Xiaoyu Zhang. 2023. Rumor Detection with Diverse Counterfactual Evidence. In *SIGKDD*. 3321–3331.
- [79] Qiang Zhang, Aldo Lipani, Shangsong Liang, and Emine Yilmaz. 2019. Reply-Aided Detection of Misinformation via Bayesian Deep Learning. In *WWW*. 2333–2343.
- [80] Xingyi Zhang, Jinchao Huang, Fangyuan Zhang, and Sibow Wang. 2024. FiCOM: an effective and scalable active learning framework for GNNs on semi-supervised node classification. *The VLDB Journal* 33, 5 (2024), 1723–1742.
- [81] Xingyi Zhang, Zixuan Weng, and Sibow Wang. 2024. Towards Deeper Understanding of PPR-based Embedding Approaches: A Topological Perspective. In *WWW*. 969–979.
- [82] Xingyi Zhang, Shuliang Xu, Wenqing Lin, and Sibow Wang. 2023. Constrained Social Community Recommendation. In *SIGKDD*. 5586–5596.
- [83] Tianxiang Zhao, Dongsheng Luo, Xiang Zhang, and Suhang Wang. 2022. TopoImb: Toward Topology-Level Imbalance in Learning From Graphs. In *LoG (PMLR, Vol. 198)*. 37.
- [84] Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders. In *ACL*. 654–664.
- [85] Xinyi Zhou, Apurva Mulay, Emilio Ferrara, and Reza Zafarani. 2020. ReCOVeRY: A Multimodal Repository for COVID-19 News Credibility Research. In *CIKM*. 3205–3212.
- [86] Xinyi Zhou and Reza Zafarani. 2019. Network-based Fake News Detection: A Pattern-driven Approach. *SIGKDD Explor.* 21, 2 (2019), 48–60.
- [87] Xinyi Zhou and Reza Zafarani. 2021. A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities. *ACM Comput. Surv.* 53, 5 (2021), 109:1–109:40.
- [88] Junyou Zhu, Chao Gao, Ze Yin, Xianghua Li, and Jürgen Kurths. 2024. Propagation Structure-Aware Graph Transformer for Robust and Interpretable Fake News Detection. In *SIGKDD*. 4652–4663.
- [89] Arkaitz Zubiega, Maria Liakata, and Rob Procter. 2016. Learning Reporting Dynamics during Breaking News for Rumour Detection in Social Media. *CoRR* abs/1610.07363 (2016).

## A Notations

We summarize the frequently used notations in Table 6.

## B Additional Related Work

**Heterogeneous Graph-based Fake News Detection.** Instead of propagation graphs reflecting the rumor-spreading patterns, another line of research [24, 75, 76] constructs heterogeneous graphs of publishers, tweets, and users to integrate both local and global relationships. Based on these heterogeneous graphs, a series of models have been proposed. For instance, GCAN [37] uses a dual co-attention mechanism to provide reasonable explanations by learning features from four aspects. Cui et al. [10] propose encoding metapaths from heterogeneous graphs. FinerFact [25] reasons for important evidence from a constructed claim-evidence graph using a mutual reinforcement mechanism [15]. SureFact [71] further leverages RL to measure the importance of nodes and conducts subgraph reasoning. Additionally, DECOR [68] and PSGT [88] incorporate auxiliary information to enhance the detection of fake news, such as degrees of news articles and malicious users during the user-article engagement. Note that additional information used to construct the heterogeneous graphs, such as news-user discussion graphs, user-user interaction graphs, news-post similarity graphs, is not included in the datasets we used. To ensure a fair comparison, we exclude these methods from our experiments.

**Graph Generation.** Graph generation is the task of learning the distribution of the observed graphs to generate new realistic graphs. One of the criteria used for classifying graph generative models is the generation process, including one-shot and sequential [20]. GraphRNN [74] lays the foundation for subsequent node-by-node deep auto-regressive models. After that, MolecularRNN [42] extends GraphRNN to generate realistic molecular graphs with optimized properties. DeepGMG [34] defines a sequential graph generation process as a sequence of decisions. Faez et al. summarize in [16] that current reinforcement learning-based graph generators typically use a sequential generation strategy. GCPN [73] and its related methods [26, 27, 49, 57] employ a step-by-step approach in generating molecular graphs by formulating the problem as a Markov decision process. GraphOpt [57] simulates the decision process for graph construction and searches for the reward function by optimizing the objective of assigning scores to the observed graphs. These graph generators mainly focus on the generation of chemical molecules, while we design a new dynamically updated candidate set and effectively combine node features with structural information for generating propagation graphs on social media.

**Response Generation.** The generation of responses for dialogue services is an important task in natural language processing. Earlier attempts [47], integrate the concept of variational auto-encoders (VAE) [30, 44] into response generation. After that, advanced studies [7, 48, 52, 67, 84] are proposed to enhance response diversity

**Table 6: Frequently used notations.**

Notations	Descriptions
$r, s, y_s$	Source post, event, and its ground-truth label.
$G, V, E, X$	Input propagation graph, node set, edge set, and feature matrix.
$g_t, G_t$	Key propagation graph and candidate graph at step $t$ .
$v_t, e_t$	The newly added node and edge at step $t$ .
$C_{t+1}$	Candidate set for ENS to select $v_t$ .
$\epsilon, \gamma$	Trade-off parameter and threshold of candidate sets.
$l$	The max step in modified Rollout.
$\tau$	Controlling the max generation step.

**Algorithm 1: Training Pipeline of KPG**


---

**Input** : Set of events  $\{s_0, s_1, \dots\}$ , pre-trained BiGCN classifier  $f(\cdot)$ , maximum step  $L$

**Output** : Key propagation graph for each event,  $\pi_S, \pi_R$

- 1 Initialize  $\pi_S, \pi_R$
- 2 **for each epoch do**
- 3     **for each batch do**
- 4         **for step  $t = 0$  to  $L$  do**
- 5             Train  $\pi_R$  using Eq. 4 with  $\pi_S$  fixed
- 6              $G_{t+1} \leftarrow \pi_R(G_t)$
- 7              $p_t \leftarrow \pi_S(G_{t+1}, g_t)$
- 8             Sample action  $a_t = (v_t, e(v_t))$  based on  $p_t$
- 9             Transfer to state
- 10              $g_{t+1} = (V_{g_t} \cup v_t, E_{g_t} \cup e(v_t), X_{g_t} \cup X[v_t])$
- Train  $\pi_S$  using Eq. 3.4 with  $\pi_R$  fixed

---

based on CVAE [50]. For instance, (kg)CVAE [84] integrates linguistic prior knowledge. SepaCVAE [52] introduces contrastive learning to leverage group information. PAGenerator [67] adds regularization terms to guide the generation of personal-aware and relevant responses. Instead of relying solely on news content [43], we exploit propagation graphs to learn the latent variable distribution of responses that are more helpful to identify rumors in social media.

**C Algorithm**

The complete learning pipeline is presented in Algorithm 1.

**D Datasets and Baselines**

**Datasets.** In our experiment, we utilize three commonly used datasets, Twitter15, Twitter16, and PHEME, to evaluate the effectiveness of our KPG. Besides, to explore the rumors with more recent dates and topics, we have collected a new dataset, Weibo22. The statistics of these four datasets are presented in Table 7. Due to deleted tweets and deactivated accounts, our dataset statistics slightly deviate from those originally reported in [40].

**Baselines.** Based on code availability and task consistency, we selected twelve state-of-the-art baseline models. The included competitors are introduced below.

**Table 7: Statistic of datasets.**

Statistic	Twitter15	Twitter16	PHEME	Weibo22
# posts	41,194	18,618	67,238	961,962
# users	34,668	16,805	34,287	705,831
# events	1,490	818	3,720	4,174
# Non-rumors	374	205	1,860	2,087
# False rumors	370	205	1,860	2,087
# True rumors	372	205	0	0
# Unverified rumors	374	203	0	0
Avg. # posts / event	28	23	18	230
Med. # posts / event	16	13	15	7
Max # posts / event	304	250	346	30,791
Min # posts / event	1	1	2	1

- BERT [11]: a powerful pre-trained language model based on bidirectional transformers;
- RvNN [41]: it employs tree-structured RNNs with GRUs to extract representations from the propagation structure in bottom-up and top-down manners;
- BiGCN [6]: a GCN-based model that learns features from both propagation and dispersion structures;
- EBGCN [65]: it uses a Bayesian approach to adjust the weights of uncertain relations and enforces consistency on latent relations using an edge-wise training framework;
- RDEA [21]: a contrastive self-supervised learning-based method with event augmentation;
- GACL [53]: it designs graph perturbation methods based on adversarial and supervised contrastive learning;
- TrustRD [35]: it incorporates self-supervised learning and a Bayesian network to derive trustworthy predictions;
- SMG [70]: it learns graph representations from a sequence of subgraphs to better capture task-relevant substructures and skip noisy parts;
- AdaSNN [32]: it generates critical subgraphs with a bi-level mutual information enhancement mechanism optimized in the reinforcement learning framework;
- DCE-RD [78]: it exploits the diverse counterfactual evidence to serve as multi-view interpretations on event graphs;
- GLAN [75]: it learns local semantic and global structural information via attention mechanisms on the heterogeneous graph;
- SMAN [76]: it adopts a structure-aware multi-head attention module on the heterogeneous graph to optimize the user credibility prediction and rumor detection task jointly;
- SBAG [24], it adopts a pre-trained MLP to capture social bot features in the propagation graph and uses it as a scorer to train a social bot-aware GNN for rumor detection.

**Remark.** SBAG requires additional datasets to train the social bot detection model while such additional information is unavailable in our experiments. Thus, the second-best performing variant reported in the original paper, SBAG-s, which scores the possibility of bots randomly, is compared in our experiment. For the BERT baseline, following previous settings [53], we concatenate the source post and all comment posts in the same event as the input texts to fine-tune a BERT-based classifier.

Among the baseline models, BERT is the only one that utilizes text features without propagation structures. RvNN, BiGCN, EBGCN, RDEA, GACL, and TrustRD are rumor detection methods utilizing the propagation structures of social network posts. Additionally, we also include SMG and AdaSNN, which are not specifically tailored for rumor propagation graphs, but for a general spectrum of graphs. These two graphs are proposed with a focus on handling graphs containing noisy or unreliable information. Moreover, GLAN, SMAN, and SBAG employ additional user information to construct heterogeneous graphs, while other competitors, along with our KPG, do not use any user information, but only the propagation structures in rumor detection.

## E Parameter Setting

Following [6, 21, 53, 65], we randomly split the datasets into five parts and conduct 5-fold cross-validation to obtain the final results. For each dataset, we use the same data split on all methods for

fair comparison. We set the maximum step of the key propagation graph generation equal to  $\tau$  times the median size of the original graphs in each dataset, and we utilize grid search to set  $\tau$  from  $\{2^1, 2^2, 2^3\}$ . For Weibo22, considering that the median size of the original graphs is much smaller than the average size, we add an additional choice,  $s_{avg}$ , to the search list, where  $s_{avg}$  is the average size of the original graphs. We set the threshold of candidates  $\gamma = 5$  in CRG. We set the maximum step of rollout in ENS to 10, and the trade-off parameter between two candidate sets  $\epsilon = 0.8$ . We pre-train the BiGCN classifier for 30 epochs to derive rewards.

After the generation of key propagation graphs, we train another BiGCN classifier on the key graphs with 200 epochs to obtain the final classification results. Our KPG is optimized by Adam algorithm [29]. The learning rate is initialized to  $5 \times 10^{-4}$  and gradually decreases during training with a decay rate of  $10^{-4}$ . The dimension of hidden feature vectors in all modules is set to 64, and the batch size is set to 128.