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

Anonymous Author(s)
Submission Id: 1775*

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*.

KEYWORDS

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

ACM Reference Format:

Anonymous Author(s). 2018. Rumor Detection on Social Media with Reinforcement Learning-based Key Propagation Graph Generator. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

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 [68], negatively impacting individual health and social stability [56, 66]. 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 [37, 68]. Additionally, this challenge extends to sophisticated tools like large language models, which also struggle to accurately identify and curb the spread of 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 [31, 36, 53, 67], which require the manual definition of content, structural, or information source features, to deep learning approaches [6, 14, 19, 24, 29, 32, 44, 48, 50, 51, 62, 63]. Notably, *graph neural network (GNN)*-based rumor detectors have achieved superb performance. These models leverage both the textual content and the topology of propagation graphs that represent the spread of posts on social networks. 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 in identifying the veracity of claims, as noted in [44, 50]. To address this challenge, techniques such as weight adjustment [49, 50] and data augmentation [14, 26, 44] 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

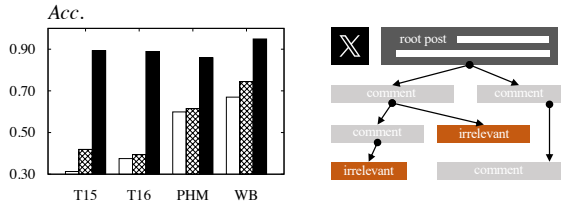


Figure 1: Left: classification accuracy on four datasets, where the white/crosshatched/black bars represent the results of GPT-3.5/GPT-4/KPG, respectively. Right: an example propagation graph of posts on social networks, including irrelevant comments and retweets.

[27, 64], 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 [27]. 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[1] 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[2] show that KPG achieves state-of-the-art performance in the rumor detection task.

2 PRELIMINARIES

2.1 Notation

Let $s = \{r, G = (V, E, \boldsymbol{X})\}$ 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 $V = \{r, v_1, v_2, \cdots, v_{n_s-1}\}$ contains all n_s comments and retweets in the propagation process. The text stored in node u is denoted as text_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. $\boldsymbol{X} \in \mathbb{R}^{n_s \times d}$ contains the initial features of posts in the event s. Each row of the feature matrix $\boldsymbol{X}[v_i]$ 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 $\{s_0, s_1, \cdots\}$ 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 $g_t = \{V_{g_t}, E_{g_t}, \boldsymbol{X}_{g_t}\}$, 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 $\boldsymbol{X}[r]$. 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 briefly review existing *graph-based rumor detection models*. Additional related work is discussed in Appendix B.

A line of research [15, 59, 60] 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 [28] uses a dual

[1] Key Propagation Graph Generator

[2] Code available at https://anonymous.4open.science/r/KPG-A279/README.md

co-attention mechanism to provide reasonable explanations by learning features from four aspects. Cui et al. [9] propose encoding metapaths from heterogeneous graphs. FinerFact [16] reasons for important evidence from a constructed claim-evidence graph using a mutual reinforcement mechanism [11]. SureFact [55] further leverages RL to measure the importance of nodes and conducts subgraph reasoning. 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.

Another line of research adopts GNN to enhance the exploitation of the propagation graphs. For example, BiGCN [6] employs GCN on both top-down and bottom-up propagation graphs. EBGCN [50] and FGCN [49] suggest adjusting the weight of each edge to reflect the intensity of interactions between comments. Models like RDEA [14], GACL [44], TrustRD [26] and others [29, 61] employ data augmentation strategies like edge perturbation and masking to improve the robustness of the model through contrastive learning. AARD [42] generates position-aware adversarial responses to improve the robustness of models.

However, current models heavily rely on the original topology information to combat untrustworthy comments in propagation graphs. Thus, these strategies are ineffective in the case of small graphs with insufficient structural information. In contrast, our KPG proposes two interdependent modules that generate key propagation patterns for propagation graphs of varied sizes, achieving superb performance, as shown in our experiments.

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

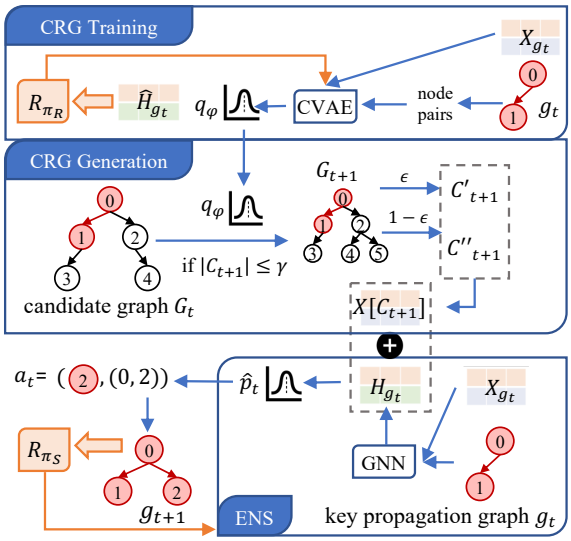


Figure 2: Framework of KPG.

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 γ, 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 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_{π_S} and R_{π_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)* [41]. This extension of the traditional VAE [21] 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 $\boldsymbol{h}_u$ and $\boldsymbol{h}_v$, respectively. To address the one-to-many problem [41], we concatenate the latent representation of u with its source post r to strengthen the connection between $\boldsymbol{h}_u$ and its central topic:

$$\begin{aligned} \boldsymbol{h_u} &= \text{CONCAT}\left(\text{GRU}(\text{text}_u), \text{GRU}(\text{text}_r)\right), \\ \boldsymbol{h_v} &= \text{GRU}\left(\text{text}_v\right). \end{aligned} \tag{1}$$

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

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

A sample $\boldsymbol{z}' \sim q_\varphi(\boldsymbol{z}|\boldsymbol{h}_u, \boldsymbol{h}_v)$ is then drawn from the learned distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ using the reparameterization trick [41]. This sample is then combined with the context representation $\boldsymbol{h_u}$ to reconstruct the response through the MLP decoder:

$$\hat{\boldsymbol{h}_v} = \text{MLP}_{dec}(\text{CONCAT}(\boldsymbol{z}', \boldsymbol{h_u})). \tag{2}$$

The goal of CRG is to minimize the reconstruction error between representations of the generated response $\hat{\boldsymbol{h}}_v$ and the original response $\boldsymbol{h}_v$, 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 γ for the size of the candidate set, which is initialized using the original graph. If the number of candidate nodes exceeds γ, 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 γ, and we select a context node i from the candidate graph G_t. We first generate the topic-aware representation $\boldsymbol{h}_i$ based on Eq. (1). Next, $\boldsymbol{z}'$ is sampled following the learned distribution q_φ. We then combine $\boldsymbol{z}'$ and $\boldsymbol{h}_i$ to generate the response $\boldsymbol{h}_j$ according to Eq. (2). The text of node i can be generated using the GRU decoder based on the decoded representation $\boldsymbol{h}_j$. 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 γ.

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 ϵ 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. Specifically, we aggregate neighbor features through the graph structure using a *Graph Convolutional Network (GCN)* [22], such that both textual and topological information in the current key graph can be incorporated. Let $\boldsymbol{A}_{g_t}$ be the adjacency

matrix of g_t and $\boldsymbol{X}_{g_t}$ 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:

$$\boldsymbol{H}_{g_t} = \delta(\hat{\boldsymbol{A}}_{g_t}\delta(\hat{\boldsymbol{A}}_{g_t}\boldsymbol{X}_{g_t}W_1)W_2),$$

where $\hat{\boldsymbol{A}}_{g_t}$ is the normalized adjacency matrix of g_t with self-loops and $\delta(\cdot)$ is the activation function.

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

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

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

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

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

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

Finally, based on the probability distribution $\boldsymbol{p}_t$, 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 $\boldsymbol{p}_t$:

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

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:

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

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 ϵ. This strategy is similar to the restart probability in the random walk [46]. We either select the next nodes locally from the out-neighbors of the currently selected nodes with ϵ 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:

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

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:

$$\begin{aligned}\mathcal{L}_{\pi_R}^{(t)} = R_{\pi_R}^{(t)} \cdot \Sigma(&\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{h_u},\boldsymbol{h_v})}[\log p_\theta(\boldsymbol{h_v}|\boldsymbol{z},\boldsymbol{h_u})] \\ &- KL(q_\phi(\boldsymbol{z}|\boldsymbol{h_u},\boldsymbol{h_v})\|p_\theta(\boldsymbol{z}|\boldsymbol{h_u}))).\end{aligned} \tag{4}$$

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 π_R and update ENS π_S by minimizing $\mathcal{L}_{\pi_S}$ using Eq. 3.4 through policy gradient [45]. Next, we fix ENS π_S and update CRG π_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 [44], we concatenate the texts of the source post and comments in the propagation graph as the input text for a BERT classifier [10]. 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 11 state-of-the-art competitors on 4 datasets. 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 [31], Twitter16 [31], and Pheme [69]. 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, 32, 44] on rumor detection.

The Weibo dataset used in [6, 30] 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**. 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 eleven state-of-the-art baselines, including BERT, RvNN, BiGCN, EBGCN, RDEA, GACL, TrustRD, SMG, AdaSNN, GLAN, SMAN, and SBAG. Detailed descriptions of these baselines can be found in Appendix D.

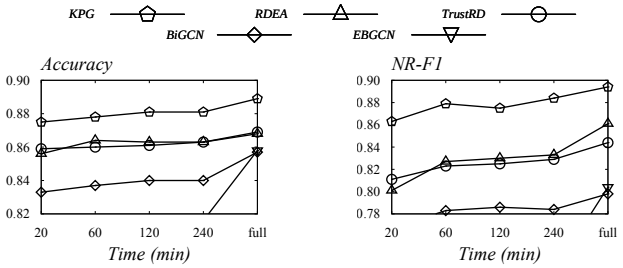
Metrics and evaluation protocol. We adopt Accuracy (Acc.) and micro F_1 score (F_1) for each class as our evaluation metrics. Notice that several baselines [6, 14, 26, 44, 50] utilize a batch-wise averaging approach. However, this may introduce performance variance, particularly when the final batch 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.

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	11.3
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	11.6
BiGCN	0.837	0.813	0.846	0.892	0.798	0.857	0.798	**0.836**	0.923	0.873	0.841	0.842	0.841	0.905	0.907	0.904	5.9
EBGCN	0.851	0.827	0.864	0.891	0.825	0.858	0.804	0.835	0.920	0.874	0.839	0.840	0.837	0.870	0.876	0.864	6.4
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	0.918	0.920	0.917	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	10.2
TrustRD	0.866	0.875	0.871	0.896	0.821	0.869	0.844	0.827	0.929	0.874	0.846	0.846	0.846	0.918	0.920	0.917	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	-	-	-	9.1
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	-	-	-	10.8
GLAN	0.827	0.820	0.839	0.871	0.779	0.831	0.756	0.805	0.923	0.843	0.845	0.849	0.840	0.902	0.903	0.902	7.4
SMAN	0.853	0.894	0.851	0.860	0.806	0.853	0.867	0.788	0.914	0.842	0.836	0.837	0.835	0.911	0.912	0.909	6.6
SBAG	0.862	0.876	0.862	0.887	0.821	0.870	0.863	**0.836**	0.913	0.866	0.841	0.842	0.841	0.912	0.912	0.911	4.4
KPG	**0.893**	**0.921**	**0.898**	**0.903**	**0.847**	**0.889**	**0.894**	**0.836**	**0.930**	**0.896**	**0.859**	**0.863**	**0.854**	**0.949**	**0.949**	**0.948**	1.0



(a) Accuracy (b) F1 score on NR

Figure 3: Early stage rumor detection on Twitter16.

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 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 Δ 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 Δ are retained, while those exceeding Δ 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.

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 Δ 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.

Table 2: Ablation study on Twitter15.

	Acc.	NR-F_1	FR-F_1	TR-F_1	UR-F_1
KPG	**0.893**	**0.921**	**0.898**	**0.903**	**0.847**
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\reward	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	**0.889**	**0.894**	**0.836**	**0.930**	**0.896**
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\reward	0.865	0.880	0.808	0.914	0.855

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;

Table 4: Results with varying ϵ on Twitter16.

ϵ	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 τ and l on Twitter16.

	τ				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

- KPG\crg: it removes the CRG module and stops generating extra candidate nodes for small propagation graphs;
- KPG\reward: 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 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 ϵ in Candidate Selection. The parameter $\epsilon \in [0, 1]$ serves as a trade-off between local and global candidates. A larger ϵ indicates a higher probability for ENS to select the next node from the local neighbors of currently selected nodes, while a smaller ϵ increases the probability of selecting from nodes that have not yet been selected. We vary the parameter ϵ 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 τ, which controls the maximum size of the generated key propagation graph. We set the maximum generation steps to τ times the median size of original graphs, where τ 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 τ 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 τ values, even when τ is relatively small.

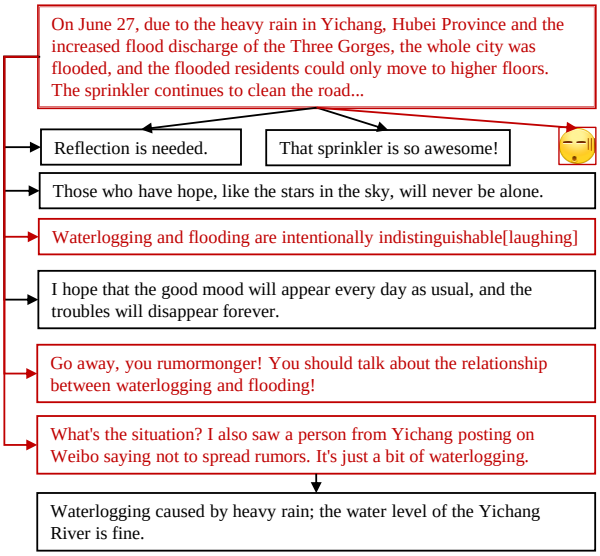


Figure 4: An example of the generated propagation graph.

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, 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.

REFERENCES

[1] A rumor. 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 SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics.
[9] 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.
[10] 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.
[11] 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.
[12] Faezeh Faez, Yassaman Ommi, Mahdieh Soleymani Baghshah, and Hamid R. Rabiee. 2021. Deep Graph Generators: A Survey. *IEEE Access* 9 (2021), 106675–106702.
[13] Xiaojie Guo and Liang Zhao. 2023. A Systematic Survey on Deep Generative Models for Graph Generation. *TPAMI* 45, 5 (2023), 5370–5390.
[14] Zhenyu He, Ce Li, Fan Zhou, and Yi Yang. 2021. Rumor Detection on Social Media with Event Augmentations. In *SIGIR*. 2020–2024.
[15] 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.
[16] 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*. AAAI Press, 5746–5754.
[17] 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.
[18] 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.
[19] Ling Min Serena Khoo, Hai Leong Chieu, Zhong Qian, and Jing Jiang. 2020. Interpretable Rumor Detection in Microblogs by Attending to User Interactions. In *AAAI*. 8783–8790.
[20] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
[21] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *ICLR*.
[22] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[23] 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.
[24] Quanzhi Li, Qiong Zhang, and Luo Si. 2019. Rumor Detection by Exploiting User Credibility Information, Attention and Multi-task Learning. In *ACL*. 1173–1179.
[25] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter W. Battaglia. 2018. Learning Deep Generative Models of Graphs. *CoRR* abs/1803.03324 (2018).
[26] Leyuan Liu, Junyi Chen, Zhangtao Cheng, Wenxin Tai, and Fan Zhou. 2023. Towards Trustworthy Rumor Detection with Interpretable Graph Structural Learning. In *CIKM 2023*. 4089–4093.
[27] Zemin Liu, Qiheng Mao, Chenghao Liu, Yuan Fang, and Jianling Sun. [n.d.]. On Size-Oriented Long-Tailed Graph Classification of Graph Neural Networks. In *WWW 2022*. ACM, 1506–1516.
[28] 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.
[29] 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.
[30] 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 2016*. IJCAI/AAAI Press, 3818–3824.
[31] Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect Rumors in Microblog Posts Using Propagation Structure via Kernel Learning. In *ACL*. 708–717.
[32] Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Rumor Detection on Twitter with Tree-structured Recursive Neural Networks. In *ACL*. 1980–1989.
[33] Mariya Popova, Mykhailo Shvets, Junier Oliva, and Olexandr Isayev. 2019. MolecularRNN: Generating realistic molecular graphs with optimized properties. *arXiv preprint arXiv:1905.13372* (2019).
[34] 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.
[35] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *ICML*, Vol. 32. 1278–1286.
[36] 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.
[37] Victoria L. Rubin. 2010. On deception and deception detection: Content analysis of computer-mediated stated beliefs. In *ASIST*, Vol. 47. 1–10.
[38] Iulian Vlad Serban, Alessandro Sordoni, 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.
[39] Xiaoyu Shen, Hui Su, Shuzi Niu, and Vera Demberg. 2018. Improving Variational Encoder-Decoders in Dialogue Generation. In *AAAI*. 5456–5463.
[40] Fangzhou Shi, Shan You, and Chang Xu. 2019. Reinforced Molecule Generation with Heterogeneous States. In *ICDM*. 548–557.
[41] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning Structured Output Representation using Deep Conditional Generative Models. In *NeurIPS*, Vol. 28.
[42] Yun-Zhu Song, Yi-Syuan Chen, Yi-Ting Chang, Shao-Yu Weng, and Hong-Han Shuai. 2021. Adversary-Aware Rumor Detection. In *ACL/IJCNLP*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.), Vol. ACL/IJCNLP 2021. 1371–1382.
[43] 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.
[44] Tiening 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.
[45] 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.
[46] Hanghang Tong, Christos Faloutsos, and Jia-yu Pan. 2006. Fast Random Walk with Restart and Its Applications. In *Sixth International Conference on Data Mining (ICDM'06)*.
[47] Rakshit Trivedi, Jiachen Yang, and Hongyuan Zha. 2020. GraphOpt: Learning Optimization Models of Graph Formation. In *ICML*, Vol. 119. 9603–9613.
[48] Yaqing Wang, Fenglong Ma, Zhiwei Jin, Ye Yuan, Guangxu Xun, Kishlay Jha, Lu Su, and Jing Gao. 2018. EANN: Event Adversarial Neural Networks for Multi-Modal Fake News Detection. In *KDD*. 849–857.
[49] Lingwei Wei, Dou hu, Wei Zhou, Xin Wang, and Songlin hu. 2022. Modeling the uncertainty of information propagation for rumor detection: A neuro-fuzzy approach. *IEEE Transactions on Neural Networks and Learning Systems* (2022).
[50] Lingwei Wei, Dou Hu, Wei Zhou, Zhaojuan Yue, and Songlin Hu. 2021. Towards Propagation Uncertainty: Edge-enhanced Bayesian Graph Convolutional Networks for Rumor Detection. In *ACL/IJCNLP (1)*. 3845–3854.
[51] 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.
[52] 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.
[53] Ke Wu, Song Yang, and Kenny Q. Zhu. 2015. False rumors detection on Sina Weibo by propagation structures. In *ICDE*, Johannes Gehrke, Wolfgang Lehner, Kyuseok Shim, Sang Kyun Cha, and Guy M. Lohman (Eds.). 651–662.
[54] Mingqi Yang, Yanming Shen, Heng Qi, and Baocai Yin. 2021. Soft-mask: Adaptive Substructure Extractions for Graph Neural Networks. In *The Web Conference 2021*. 2058–2068.
[55] Ruichao Yang, Xiting Wang, Yiqiao Jin, Chaozhuo Li, Jianxun Lian, and Xing Xie. 2022. Reinforcement Subgraph Reasoning for Fake News Detection. In *KDD*. 2253–2262.
[56] 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.
[57] Jiaxuan 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.
[58] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models. In *ICML*, Vol. 80. 5694–5703.

Table 6: Frequently used notations.

Notations	Descriptions
r, s, y_s	Source post, event, and its ground-truth label.
$G, V, E, \boldsymbol{X}$	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.
ϵ, γ	Trade-off parameter and threshold of candidate sets.
l	The max step in modified Rollout.
τ	Controlling the max generation step.

Algorithm 1: Training Pipeline of KPG

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

Output: Key propagation graph for each event, π_S, π_R

Initialize π_S, π_R
for *each epoch* **do**
  for *each batch* **do**
    for *step* $t = 0$ *to* L **do**
      Train π_R using Eq. 4 with π_S fixed
      $G_{t+1} \leftarrow \pi_R(G_t)$
      $\boldsymbol{p}_t \leftarrow \pi_S(G_{t+1}, g_t)$
      Sample action $a_t = (v_t, e(v_t))$ based on $\boldsymbol{p}_t$
      Transfer to state $g_{t+1} = (V_{g_t} \cup v_t, E_{g_t} \cup e(v_t), \boldsymbol{X}_{g_t} \cup \boldsymbol{X}[v_t])$
      Train π_S using Eq. 3.4 with π_R fixed

[59] 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.

[60] 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.

[61] Guixian Zhang, Rongjiao Liang, Zhongyi Yu, and Shichao Zhang. 2022. Rumour Detection on Social Media with Long-Tail Strategy. In *IJCNN*. 1–8.

[62] Kaiwei Zhang, Junchi Yu, Haichao Shi, Jian Liang, and Xiaoyu Zhang. 2023. Rumor Detection with Diverse Counterfactual Evidence. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023*. ACM, 3321–3331.

[63] Qiang Zhang, Aldo Lipani, Shangsong Liang, and Emine Yilmaz. 2019. Reply-Aided Detection of Misinformation via Bayesian Deep Learning. In *WWW*. 2333–2343.

[64] Tianxiang Zhao, Dongsheng Luo, Xiang Zhang, and Suhang Wang. 2022. TopoImb: Toward Topology-Level Imbalance in Learning From Graphs. In *Learning on Graphs Conference, LoG 2022 (Proceedings of Machine Learning Research, Vol. 198)*. PMLR, 37.

[65] Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders. In *ACL*. 654–664.

[66] Xinyi Zhou, Apurva Mulay, Emilio Ferrara, and Reza Zafarani. 2020. ReCOVery: A Multimodal Repository for COVID-19 News Credibility Research. In *CIKM*. 3205–3212.

[67] Xinyi Zhou and Reza Zafarani. 2019. Network-based Fake News Detection: A Pattern-driven Approach. *SIGKDD Explor.* 21, 2 (2019), 48–60.

[68] 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.

[69] Arkaitz Zubiaga, 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

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 [13]. GraphRNN [58] lays the foundation for subsequent node-by-node deep auto-regressive models. After that, MolecularRNN [33] extends GraphRNN to generate realistic molecular graphs with optimized properties. DeepGMG [25] defines a sequential graph generation process as a sequence of decisions. Faez et al. summarize in [12] that current reinforcement learning-based graph generators typically use a sequential generation strategy. GCPN [57] and its related methods [17, 18, 40, 47] employ a step-by-step approach in generating molecular graphs by formulating the problem as a Markov decision process. GraphOpt [47] 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 [38], integrate the concept of variational auto-encoders (VAE) [21, 35] into response generation. After that, advanced studies [7, 39, 43, 52, 65] are proposed to enhance response diversity based on CVAE [41]. For instance, (kg)CVAE [65] integrates linguistic prior knowledge. SepaCVAE [43] introduces contrastive learning to leverage group information. PAGenerator [52] adds regularization terms to guide the generation of personal-aware and relevant responses. Instead of relying solely on news content [34], 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.

Baselines. We introduce the eleven state-of-the-art baseline models used in our experiments as follows.

- BERT [10]: a powerful pre-trained language model based on bidirectional transformers;
- RvNN [32]: it employs tree-structured RNNs with GRUs to extract representations from the propagation structure in bottom-up and top-down manners;

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

- BiGCN [6]: a GCN-based model that learns features from both propagation and dispersion structures;
- EBGCN [50]: 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 [14]: a contrastive self-supervised learning based method with event augmentation;
- GACL [44]: it designs graph perturbation methods based on adversarial and supervised contrastive learning;
- TrustRD [26]: it incorporates self-supervised learning and a Bayesian network to derive trustworthy predictions;
- SMG [54]: it learns graph representations from a sequence of subgraphs to better capture task-relevant substructures and skip noisy parts;
- AdaSNN[23]: it generates critical subgraphs with a bi-level mutual information enhancement mechanism optimized in the reinforcement learning framework;
- GLAN [59]: it learns local semantic and global structural information via attention mechanisms on the heterogeneous graph;
- SMAN [60]: 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 [15], 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 [44], 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, 14, 44, 50], 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 τ times the median size of the original graphs in each dataset, and we utilize grid search to set τ 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 [20]. The learning rate is initialized to 5×10^{-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.