

Rumor Detection on Social Media with Temporal Propagation Structure Optimization

Xingyu Peng, Junran Wu*, Ruomei Liu, Ke Xu

State Key Laboratory of Complex & Critical Software Environment,

Beihang University, Beijing, China

{xypeng, wu_junran, rmliu, kexu}@buaa.edu.cn

Abstract

Traditional methods for detecting rumors on social media primarily focus on analyzing textual content, often struggling to capture the complexity of online interactions. Recent research has shifted towards leveraging graph neural networks to model the hierarchical conversation structure that emerges during rumor propagation. However, these methods tend to overlook the temporal aspect of rumor propagation and may disregard potential noise within the propagation structure. In this paper, we propose a novel approach that incorporates temporal information by constructing a weighted propagation tree, where the weight of each edge represents the time interval between connected posts. Drawing upon the theory of structural entropy, we transform this tree into a coding tree. This transformation aims to preserve the essential structure of rumor propagation while reducing noise. Finally, we introduce a recursive neural network to learn from the coding tree for rumor veracity prediction. Experimental results on two common datasets demonstrate the superiority of our approach.

1 Introduction

Social media has increasingly become a fertile ground for the generation and dissemination of rumors, which can have significant adverse impacts on society. Detecting rumors on social media is not only a pressing public concern but also a complex and multifaceted challenge. As the flood of false and misleading information continues, extensive efforts have been devoted to automating the process of rumor detection. Traditional methods for identifying rumors often rely on textual analysis (Castillo et al., 2011; Yang et al., 2012; Yu et al., 2017). Recent studies have highlighted the significance of utilizing rumor-spreading patterns, represented as propagation trees, in discerning rumors from non-rumors (Ma et al., 2018; Bian et al., 2020). As

*Corresponding author.

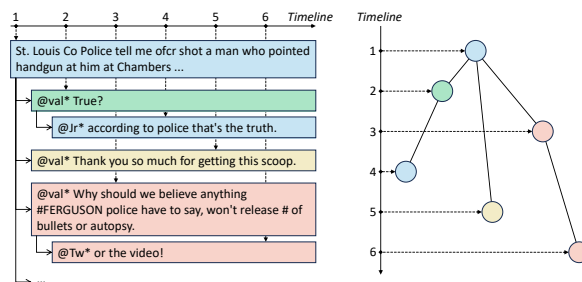


Figure 1: Example of a rumor propagation tree related to the *Ferguson* event.

illustrated in Figure 1, a propagation tree is formed by a central claim and relevant posts. This tree encapsulates the dynamics of rumor spreading on social media, where the original claim initiates a cascade of reactions and interactions as it disseminates through the network. Therefore, understanding these propagation trees is crucial for effective rumor detection.

Researchers have explored leveraging deep learning models to extract structural representations from propagation trees, employing methods such as Long Short-Term Memory (LSTM) networks (Kochkina et al., 2017), Recursive Neural Networks (RvNN) (Ma et al., 2018), and Transformer networks (Khoo et al., 2020). With the rapid proliferation of Graph Neural Networks (GNNs) in contemporary research, there is a growing trend towards adopting Graph Convolutional Networks (GCN) (Bian et al., 2020; Wei et al., 2021; Wu et al., 2024b) and Graph Attention Networks (GAT) (Lin et al., 2021; Chen et al., 2024). However, these methods tend to overlook the temporal structure inherent in information propagation, despite evidence suggesting its advantages for rumor detection (Cheng et al., 2021). Also, they often assume that the observed structure of the propagation tree is completely accurate, disregarding the presence of potential noise, such as inaccurate relations (Wei et al., 2021).

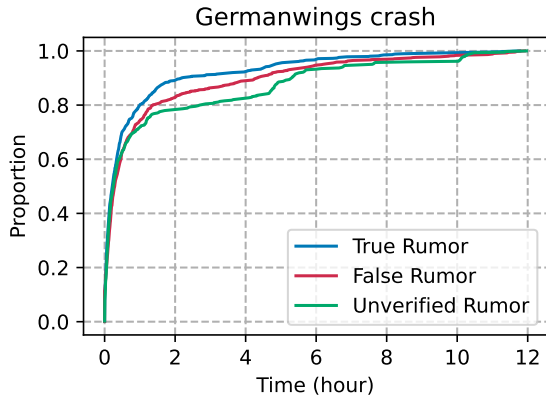


Figure 2: The Empirical Cumulative Distribution Function (ECDF) plots of the time delay distributions since the initial claim was posted for posts responding to true, false, and unverified rumors in the *Germanwings crash* event¹ from the PHEME (Zubiaga et al., 2016) dataset.

The temporal structure of rumor propagation involves the sequence and intervals of posts along the timeline originating from the central claim (Huang et al., 2020). Figure 2 shows statistically significant differences in the time delay distributions among the three categories of rumors ($p < 0.001$, Anderson-Darling test). Wei et al. (2019) observed temporal variations in stance distributions across posts discussing different categories of rumors. These findings underscore the critical need to incorporate temporal information when predicting the veracity of rumors. While previous research (Khoo et al., 2020) utilized the time delay of rumor propagation as the transformer’s position embedding, it overlooked the hierarchical structure that emerges during rumor propagation. To comprehensively consider both the structural and temporal aspects of rumor propagation, we propose constructing a weighted propagation tree, with each edge assigned a weight corresponding to the time interval between connected posts. This approach facilitates a deeper understanding of the temporal dynamics involved in rumor propagation, including the speed and efficiency of information flow.

Drawing upon the theory of structural entropy (Li and Pan, 2016), which quantifies the structural complexity and organization of graphs, we derive a coding tree from the propagation tree. The coding tree is a refined version of the original propagation tree that preserves the essential structure of propagation dynamics while reducing noise. In

pursuit of the optimal coding tree, we design a greedy algorithm aimed at minimizing structural entropy. To effectively leverage the enriched information within the coding tree, we introduce a recursive neural network for representation learning, referred to as CT-RvNN. CT-RvNN employs an efficient bottom-up message-passing scheme, iteratively propagating information from leaf nodes to the root node. Finally, we create a comprehensive representation of the coding tree through a hierarchical readout strategy, encapsulating critical information for rumor veracity prediction. In comparison to current state-of-the-art methods, CT-RvNN demonstrates superior performance while consuming fewer computational resources. Overall, our contributions can be summarized as follows:

- We demonstrate the significance of temporal information in predicting the veracity of rumors through statistical analysis. We incorporate temporal characteristics to construct time-weighted propagation trees, capturing the temporal dynamics of rumor propagation.
- We derive coding trees from the weighted propagation trees through structural entropy minimization. These coding trees preserve the essential structure of rumor propagation while reducing noise. We then employ recursive neural networks to effectively learn rumor representations from these coding trees.
- Experimental results on two widely used datasets demonstrate the effectiveness and efficiency of our approach. Additionally, in-depth analyses underscore the benefits of incorporating temporal information and the efficacy of the coding tree transformation.

2 Related Work

Rumor Detection. Rumor detection on social media has emerged as a prominent and rapidly evolving research field in recent years. Early studies primarily focused on detecting rumors through handcrafted features extracted from various sources, including post contents (Castillo et al., 2011), user profiles (Yang et al., 2012), and patterns of information propagation (Kwon et al., 2013). Ma et al. (2015) introduced a time series model designed to capture the temporal evolution of social context information. Wu et al. (2015) and Ma et al. (2017) employed Support Vector Machines (SVM) with various kernel functions for rumor detection.

¹Distributions for more events are available in Figure A4.

However, these approaches relied heavily on feature engineering, which is both time-consuming and labor-intensive.

In more recent times, researchers have explored the application of deep learning methods in the domain of rumor detection. Yu et al. (2017) leveraged Convolutional Neural Networks (CNN) to extract essential features from input sequences and capture high-level interactions among these features. Ma et al. (2016) adopted Recurrent Neural Networks (RNN) to acquire representations that encapsulate the evolving contextual information of relevant posts over time. Yu et al. (2020) divided lengthy threads into shorter subthreads and utilized a hierarchical Transformer framework to learn both local and global interactions among them.

In order to extract informative patterns from both textual content and propagation structures, Ma et al. (2018) introduced an RvNN-based model designed to uncover hidden patterns in tweets organized as propagation trees. Khoo et al. (2020) incorporated a self-attention mechanism to model long-distance interactions within propagation trees. Recent advancements in this field have also seen the incorporation of GNNs. Wei et al. (2019) combined structural characteristics and temporal dynamics in the context of rumor propagation using GCN and RNN. Lin et al. (2021) represented conversation threads as undirected interaction graphs and employed post- and event-level graph attention mechanisms to extract multi-level rumor-indicative features. Bian et al. (2020) focused on both top-down propagation and bottom-up dispersion dynamics among nodes in propagation trees for rumor detection. Wei et al. (2021) considered the inherent uncertainty in propagation structures, adaptively adjusting weights of unreliable relations to capture robust structural features. Liu et al. (2022) proposed utilizing two shared channels for extracting task-invariant textual and structural features, alongside two task-specific graph channels aimed at enhancing structural features. Chen et al. (2024) introduced an attention mechanism to capture interaction information among subthreads, incorporating a stance-rumor interaction network that integrates users' stance information with rumor verification. Luo et al. (2024) employed a graph transformer to concurrently acquire structural and semantic information. Additionally, they utilized a partition filter network to explicitly model rumor- and stance-specific features, as well as shared interactive features. Wu et al. (2024b) developed a

multi-view fusion framework that leverages GCN to encode each conversation view, complemented by a CNN to harness consistent and complementary information across all views. Despite the extensive research dedicated to rumor detection, there has been a notable lack of focus on integrating both the structural and temporal aspects of information propagation into the analysis. In addition, the potential influence of noise within the propagation structure has largely been overlooked.

Structural Entropy. Structural entropy (Li and Pan, 2016) is an extension of Shannon entropy for structured systems, providing a measure of their structural complexity. In recent years, structural entropy has been successfully applied to various domains, including community structure deception (Liu et al., 2019), graph classification (Wu et al., 2022b), graph pooling (Wu et al., 2022a), graph contrastive learning (Wu et al., 2023, 2024a), graph structure learning (Zou et al., 2023), and text classification (Zhang et al., 2022; Zhu et al., 2023, 2024).

3 Preliminaries

3.1 Problem Statement

Rumor detection aims to determine the veracity of a claim. Formally, we consider a set of conversation threads on social media platforms, denoted by $\mathcal{C} = \{c^1, \dots, c^{|\mathcal{C}|}\}$. Each thread consists of a central claim and a series of relevant posts sorted chronologically: $c^i = \{p_1^i, \dots, p_{n_i}^i\}$, where p_1^i is the original claim, and n_i is the number of posts in thread c^i . Each post in the thread is represented as $p_j^i = (x_j^i, t_j^i, r_j^i)$, where x_j^i is the textual content of the post, t_j^i is the publication time of the post, and r_j^i is the index of the post that p_j^i responds to. The propagation process, starting from the original claim, forms a tree-structured graph denoted as $\mathcal{G}^i = (\mathcal{V}^i, \mathcal{E}^i)$, where \mathcal{V}^i represents the set of posts in thread c^i and $\mathcal{E}^i = \left\{ \left(p_{r_j^i}^i, p_j^i \right) \mid j \in [2, n_i] \right\}$ denotes the edges between posts and their responses.

Each claim c^i is associated with a ground-truth label $y^i \in \mathcal{Y}$, which indicates its veracity. The label set $\mathcal{Y} = \{TR, FR, UR\}$ corresponds to the categories of true rumor, false rumor, and unverified rumor, respectively. The primary objective is to learn a classifier $f : \mathcal{C} \rightarrow \mathcal{Y}$ capable of accurately determining the veracity of a claim by analyzing its textual semantics and propagation structure.

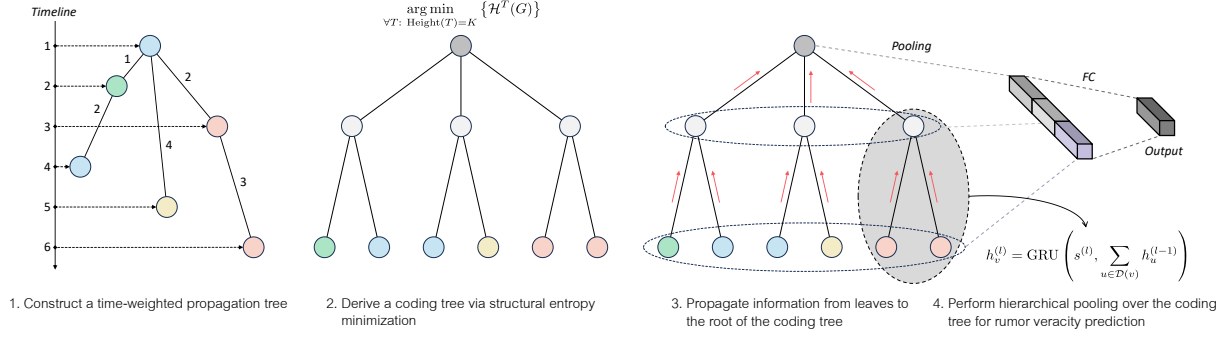


Figure 3: Overview of our approach.

3.2 Structural Entropy

Li and Pan (2016) introduced the concept of the coding tree of a graph, which serves as a lossless encoding of the graph, with its leaf nodes corresponding to nodes in the graph. They define the structural entropy of a graph $G = (V, E)$ with respect to a coding tree T as follows:

$$\mathcal{H}^T(G) = - \sum_{v_\alpha \in T} \frac{g_\alpha}{\text{vol}(G)} \log_2 \frac{\text{vol}(v_\alpha)}{\text{vol}(v_\alpha^-)}. \quad (1)$$

In the equation above, v_α represents a non-root node in the tree T , corresponding to a node subset $V_\alpha \subset V$, which comprises the leaf nodes in the subtree of T rooted at v_α . v_α^- refers to the ancestor of v_α in T and $g_\alpha = |E(V_\alpha^+, V_\alpha)|$ represents the number of edges from the complement of V_α (denoted as V_α^+) to V_α . In the case of weighted graphs, g_α is the sum of the weights of all the edges between V_α^+ and V_α . The terms $\text{vol}(G)$, $\text{vol}(v_\alpha)$, and $\text{vol}(v_\alpha^-)$ refer to the total (weighted) degrees of nodes in V , V_α , and V_α^- , respectively.

The structural entropy of the graph G is then defined as the minimum value of $\mathcal{H}^T(G)$ over all possible coding trees:

$$\mathcal{H}(G) = \min_T \{\mathcal{H}^T(G)\}. \quad (2)$$

While identifying the optimal coding tree is important, there are scenarios where decoding a predefined hierarchy for the original graph is more advantageous. In such cases, a coding tree with a predetermined height becomes preferable. To this end, the K -dimensional structural entropy is employed to decode the optimal coding tree with a specific height K :

$$\mathcal{H}^K(G) = \min_{\forall T: \text{Height}(T)=K} \{\mathcal{H}^T(G)\}. \quad (3)$$

4 Methodology

Figure 3 provides an overview of our approach. In Section 4.1, we leverage the structural and temporal characteristics of rumor propagation to construct time-weighted propagation trees. Next, we devise an algorithm to derive an optimal coding tree from a propagation tree through structural entropy minimization (Section 4.2). Following this, we introduce our representation learning model, which is designed to acquire meaningful representations of rumors from the coding trees (Section 4.3). Finally, we classify the veracity of rumors (Section 4.4).

4.1 Time-weighted Propagation Tree Construction

In the context of rumor detection, comprehending the propagation structure of a rumor is critical for assessing its veracity. Following contemporary research practices, our initial step involves constructing a propagation tree. This tree plays a pivotal role in capturing how a claim spreads within a social media platform, with nodes representing the original claim, its subsequent responses, and responses to those responses, collectively forming a hierarchical structure.

In reality, individuals require time to process and evaluate incoming information before deciding whether to share or endorse a rumor. Therefore, a temporal dimension exists that should be factored into our analysis. To achieve this, we assign weights to the edges of the propagation tree. These weights indicate the time difference between the publication of the connected posts, reflecting the duration it took for the information to propagate from one post to another. Specifically, the weight w_{jk} of an edge connecting a post p_j and its response p_k is calculated as $w_{jk} = t_k - t_j$. As a result, the adjacency matrix of the weighted propa-

gation tree is defined as $\mathcal{A}^i = \left\{ w_{jk}^i \right\}^{n_i \times n_i}$, where

$$w_{jk}^i = \begin{cases} t_k^i - t_j^i, & \text{if post } p_k^i \text{ responds to post } p_j^i, \\ 0, & \text{otherwise.} \end{cases}$$

4.2 Coding Tree Construction

After constructing the time-weighted propagation tree, the next step involves transforming it into a coding tree for structure optimization. To achieve this, we propose a greedy algorithm for minimizing the K -dimensional structural entropy, thereby generating the optimal coding tree with a specified height of K . To start, we define essential member functions for the tree data structure.

Definition 1. Given the root node v_λ and two of its descendants v_j and v_k in a tree, the function $join(v_j, v_k)$ inserts a new node v_β between the node v_λ and the nodes v_j and v_k :

$$\begin{aligned} v_\beta.descendants &= \{v_j, v_k\}, \\ v_\lambda.descendants &+= v_\beta, \\ v_\lambda.descendants &-= \{v_j, v_k\}. \end{aligned}$$

Definition 2. Given an internal node (i.e., a node that is neither the root nor a leaf) v_β in a tree, the function $trim(v_\beta)$ removes the node v_β from the tree and integrates its descendants into its ancestor's descendants:

$$\begin{aligned} v_\beta.ancestor.descendants &+= v_\beta.descendants, \\ v_\beta.ancestor.descendants &-= v_\beta. \end{aligned}$$

Definition 3. Given a non-root node v_α in a tree, the function $pad(v_\alpha)$ inserts a new node v_β between the node v_α and its ancestor:

$$\begin{aligned} v_\beta.descendants &= \{v_\alpha\}, \\ v_\alpha.ancestor.descendants &+= v_\beta, \\ v_\alpha.ancestor.descendants &-= v_\alpha. \end{aligned}$$

In Algorithm 1, we outline the coding tree construction process based on the three functions defined above. Initially, we start with a tree structure that comprises only root and leaf nodes, with each leaf node associated with a node in the propagation tree. We then expand this initial structure into a full-height binary tree using a greedy strategy. This strategy involves iteratively merging two descendant nodes of the root node to form a new division, aiming to minimize structural entropy at each step. Once we have constructed the full-height binary tree, we proceed to compress it to the prescribed height. This compression is achieved using

Algorithm 1 Coding Tree Construction via Structural Entropy Minimization

Input: Graph $G = (V, E)$; coding tree height K

Output: Coding tree $T = (V_T, E_T)$ of height K that minimizes the structural entropy of the graph G

- 1: Create a tree T of height 1 with a root node v_λ and leaf nodes, where each leaf node corresponds to a node in V ;
{Expand the tree into a full-height binary tree}
 - 2: **while** $|v_\lambda.descendants| > 2$ **do**
 - 3: Select v_j and v_k from $v_\lambda.descendants$ by $\arg \max_{(v_j, v_k)} \{ \mathcal{H}^T(G) - \mathcal{H}^{T.join(v_j, v_k)}(G) \}$;
 - 4: $T.join(v_j, v_k)$;
 - 5: **end while**
{Compress the tree to the specific height K }
 - 6: **while** $Height(T) > K$ **do**
 - 7: Select an internal node v_β from the tree T by $\arg \min_{v_\beta} \{ \mathcal{H}^{T.trim(v_\beta)}(G) - \mathcal{H}^T(G) \}$;
 - 8: $T.trim(v_\beta)$;
 - 9: **end while**
{Align the depths of all leaf nodes in the tree to K }
 - 10: **for** $v_\alpha \in T$ **do**
 - 11: **if** $Height(v_\alpha^-) - Height(v_\alpha) > 1$ **then**
 - 12: $T.pad(v_\alpha)$;
 - 13: **end if**
 - 14: **end for**
 - 15: **return** T ;
-

a similar greedy strategy, where internal nodes are iteratively removed while minimizing the increase in structural entropy, until the desired height limit is reached. After the compression process, the resulting coding tree may feature leaf nodes with varying depths. To ensure all leaf nodes attain a consistent depth K for effective representation learning, the function pad introduces internal nodes as needed. Proposition 1 demonstrates that the pad operation does not impact structural entropy. Finally, our algorithm yields the optimal coding tree at the specified height, denoted as $T = (V_T, E_T)$. Here, $V_T^{(0)} = V$, where $V_T^{(0)}$ denotes the nodes of height 0 in the coding tree T (i.e., leaf nodes). To facilitate understanding of the coding tree construction algorithm, illustrations of the $join$, $trim$, and pad operations, along with the expansion and compression processes, as well as the complexity

analysis, are provided in Appendix A.

Proposition 1. For two nodes v_j and v_k in a coding tree T such that v_j is the ancestor of v_k , we have $\mathcal{H}^T(G) = \mathcal{H}^{T.pad(v_k)}(G)$.

Proof. See Appendix B. \square

4.3 Rumor Representation Learning

In this section, based on the temporal information incorporation and the coding tree transformation, we present our model for rumor representation learning. The coding tree serves as a condensed representation of the original propagation structure, preserving essential elements of the rumor-spreading process while reducing data redundancy and noise. Our model is designed to progressively acquire node representations in the coding tree, layer by layer, ultimately culminating in a comprehensive representation of the entire coding tree through a hierarchical readout strategy.

Leaf Node Encoding. The leaf nodes in the coding tree correspond to posts in the respective conversation thread. We obtain post representations using a sentence encoder:

$$\{g_1, \dots, g_n\} = \text{SentenceEncoder}(\{x_1, \dots, x_n\}), \quad (4)$$

where n is the number of posts in the thread.

Next, we initialize the leaf node representations with the corresponding post representations:

$$h_j^{(0)} = g_j, \quad \forall j \in [1, n]. \quad (5)$$

Tree Positional Encoding. To capture the hierarchical structure of nodes in the coding tree, we introduce a positional encoding mechanism to enable the model to distinguish nodes at different depths. We define the positional embedding $s^{(l)}$ for nodes at height l as follows:

$$s^{(l)} = \text{PositionEncoder}(l), \quad (6)$$

where $\text{PositionEncoder}(\cdot)$ generates unique embeddings for each layer of the coding tree.

Bottom-Up Message Passing. To facilitate effective learning from the coding tree, we introduce a recursive neural network. This network employs an efficient bottom-up message passing scheme, iteratively propagating information from leaf nodes to the root node. As iterations proceed, the model progressively learns representations for each non-leaf node by aggregating the representations of its

descendants, eventually deriving the representation for the root node. In this context, we employ the Gated Recurrent Unit (GRU) (Chung et al., 2014) as the aggregate function. Consequently, the representation of a non-leaf node at height l in the coding tree is computed as:

$$\begin{aligned} \bar{h}_v^{(l)} &= \sum_{u \in \mathcal{D}(v)} h_u^{(l-1)}, \\ r^{(l)} &= \sigma(\mathbf{W}_r s^{(l)} + \mathbf{U}_r \bar{h}_v^{(l)}), \\ z^{(l)} &= \sigma(\mathbf{W}_z s^{(l)} + \mathbf{U}_z \bar{h}_v^{(l)}), \\ \tilde{h}_v^{(l)} &= \tanh(\mathbf{W}_h s^{(l)} + \mathbf{U}_h (r^{(l)} \odot \bar{h}_v^{(l)})), \\ h_v^{(l)} &= (1 - z^{(l)}) \odot \bar{h}_v^{(l)} + z^{(l)} \odot \tilde{h}_v^{(l)}, \end{aligned} \quad (7)$$

where $\mathcal{D}(v)$ represents the set of descendants of node v in the coding tree, $h_u^{(l-1)}$ denotes the representation of a descendant node u at the previous layer, and \odot denotes element-wise multiplication.

Tree Representation Readout. The representation h_T of the entire coding tree is generated by combining the representations from each layer in the tree. This is achieved by pooling the representations of nodes at each layer and then concatenating the resulting representations across all layers:

$$h_T = \left\| \left\| \text{Pool} \left(\left\{ h_v^{(l)} \mid v \in V_T^{(l)} \right\} \right) \right\|_{l=0}^K, \quad (8)$$

where $\|$ denotes concatenation, $V_T^{(l)}$ refers to the nodes at height l in the coding tree T , and $\text{Pool}(\cdot)$ represents a pooling operation such as summation, averaging, or maximization.

4.4 Classification and Model Training

The entire coding tree's representation is fed into a fully connected layer, followed by a softmax function to compute the predicted label probabilities:

$$\hat{y} = \text{softmax}(\mathbf{W}h_T + \mathbf{b}). \quad (9)$$

For model training, we adopt the cross-entropy loss as the objective function:

$$\mathcal{L}_C = -\frac{1}{|\mathcal{C}|} \sum_{i=1}^{|\mathcal{C}|} \sum_{j=1}^{|\mathcal{Y}|} y_j^i \log \hat{y}_j^i + \eta \|\theta\|_2, \quad (10)$$

where y_j^i denotes the ground truth probability distribution for the i -th sample in \mathcal{C} .

5 Experiments

5.1 Experimental Setup

Datasets. We conduct experiments using two publicly available datasets: **PHEME** (Zubiaga et al., 2016) and **Rumoreval** (Derczynski et al., 2017). These datasets consist of real-world data collected from Twitter and are widely used in rumor detection. We adopt a leave-one-event-out cross-validation approach for the PHEME dataset, as established in previous studies (Wei et al., 2019, 2021; Liu et al., 2022; Luo et al., 2024; Chen et al., 2024; Wu et al., 2024b). The Rumoreval dataset is evaluated using its official division. Both datasets exhibit imbalances in class distribution. To ensure a comprehensive evaluation of model performance, we prioritize Macro-F1 as the primary evaluation metric and supplement it with Accuracy. Key statistics for both datasets are summarized in Table A1.

Implementation Details. To align with prior research (Bian et al., 2020; Wei et al., 2021; Wu et al., 2024b), we use TF-IDF as the sentence encoder to represent posts as 5000-dimensional vectors based on their textual content. The pooling operation is performed using a summation function. For positional encoding, we employ randomly initialized embeddings, which are further adapted during the model training process. In our experiments on the PHEME and Rumoreval datasets, we set the coding tree height to 7 and 5, respectively.

Baselines. In our evaluation, we compare our model against several state-of-the-art rumor detection methods. These methods can be broadly categorized into two groups: *sequence-based methods* and *structure-based methods*. For sequence-based methods, we consider **BranchLSTM** (Kochkina et al., 2017), **HiTPLAN** (Khoo et al., 2020), and **Hierarchical Transformer** (Yu et al., 2020). For structure-based methods, we consider **TD-RvNN** (Ma et al., 2018), **Hierarchical GCN-RNN** (Wei et al., 2019), **Bi-GCN** (Bian et al., 2020), **ClaHi-GAT** (Lin et al., 2021), **EBGCN** (Wei et al., 2021), **STL-GT** (Liu et al., 2022), **MTL-SMI** (Liu et al., 2022), **SSRI-Net** (Chen et al., 2024), Luo et al. (2024), and **GMVCN** (Wu et al., 2024b).

5.2 Main Results

Table 1 provides an extensive performance comparison of various methods on the PHEME and Rumoreval datasets. Our model consistently outperforms all baseline methods in terms of both

Method	PHEME		Rumoreval	
	Macro-F1	Accuracy	Macro-F1	Accuracy
BranchLSTM	0.259	0.314	0.491	0.500
TD-RvNN	0.264	0.341	0.509	0.536
Hierarchical GCN-RNN	0.317	0.356	0.540	0.536
HiTPLAN	0.361	0.438	0.581	0.571
Hierarchical Transformer	0.372	0.441	0.592	0.607
Bi-GCN*	0.316	0.442	0.607	0.617
ClaHi-GAT*	0.369	0.556	0.539	0.536
EBGCN*	0.375	0.521	0.639	0.643
STL-GT	0.359	0.430	0.618	0.607
MTL-SMI†	0.409	0.468	0.685	0.679
SSRI-Net†	<u>0.483</u>	0.568	0.743	0.750
Luo et al. (2024)†	0.448	0.479	<u>0.754</u>	<u>0.767</u>
GMVCN	0.441	<u>0.647</u>	0.721	0.721
CT-RvNN (Ours)	0.486	0.794	0.792	0.786

Table 1: Performance comparison of various methods on the PHEME and Rumoreval datasets. The best result for each metric is highlighted in bold, with the second-best result underlined. Results marked with * are reported in Wu et al. (2024b). Methods marked with † are multi-task frameworks that additionally utilize stance information.

Macro-F1 and Accuracy metrics. Specifically, our model achieves a Macro-F1 of 48.6% and an Accuracy of 79.4% on the PHEME dataset. For the Rumoreval dataset, it attains a Macro-F1 of 79.2% and an Accuracy of 78.6%. A deeper analysis reveals that the notable improvement in Accuracy on the PHEME dataset is primarily driven by enhanced performance in identifying false rumors, which plays a key role in the overall boost.

In comparing various methods, those leveraging GNNs to model conversation structures generally demonstrate superior performance compared to methods relying on sequential models. This superiority stems from GNNs’ ability to capture complex dependencies and interactions among nodes in the propagation tree. Notably, Bi-GCN, EBGCN, and GMVCN leverage multiple directed views of conversation threads, integrating both top-down and bottom-up information flows to better capture the underlying patterns of rumor propagation and dispersion. Despite the prowess of these GNN-based methods, our model surpasses them in performance, demonstrating its superior ability to effectively leverage the specific characteristics of rumor propagation. Even when compared to methods that use rumor stance classification as an auxiliary task to enhance rumor veracity prediction, our model still achieves better performance.

Unlike TD-RvNN, which applies the recursive neural network directly to the original propagation tree, our proposed CT-RvNN employs the recursive neural network on the coding tree derived from the time-weighted propagation tree. This adaptation

Method	PHEME		Rumoreval	
	Macro-F1	Accuracy	Macro-F1	Accuracy
CT-RvNN	0.486	0.794	0.792	0.786
CT-RvNN (Unweighted)	0.465	0.772	0.751	0.750
CT-RvNN (Random)	0.421	0.703	0.687	0.714
CT-RvNN (Linear)	0.436	0.727	0.734	0.750

Table 2: Ablation study results.

leads to significant performance improvements, underscoring the advantages of incorporating temporal information and the efficacy of the coding tree transformation. We further analyze the specific performance enhancements provided by these two components in the subsequent analysis.

5.3 Ablation Study

In this section, we conduct an ablation study to analyze the impact of various modifications to our model, with the results presented in Table 2.

To assess the impact of incorporating temporal weights into the propagation tree, we construct unweighted propagation trees before the coding tree construction and perform experiments, denoted as **CT-RvNN (Unweighted)**. This modification results in a performance decrease on both datasets. Specifically, the Macro-F1 score decreases to 46.5% on PHEME and 75.1% on Rumoreval, while accuracy drops to 77.2% on PHEME and 75.0% on Rumoreval. This underscores the significance of considering temporal dynamics in the modeling process.

Next, we investigate the effects of our coding tree construction algorithm. The **CT-RvNN (Random)** variant randomly generates a coding tree of height K . The results show a noticeable reduction in performance, attributed to the random partitioning of nodes during coding tree construction, which disrupts the inherent structural information. In contrast, the coding tree constructed by minimizing structural entropy retains the essential structure of the original propagation tree, thereby enhancing the effectiveness of the learning process.

In the **CT-RvNN (Linear)** variant, we employ a linear layer to aggregate information from descendants instead of using the GRU. This adjustment aims to investigate the effects of incorporating the GRU for feature aggregation. The Macro-F1 score for this variant is 47.0% on PHEME and 74.4% on Rumoreval, with corresponding accuracies of 70.0% and 75.0%. Compared to this variant, the GRU-based recursive neural network not only demonstrates superior performance but also

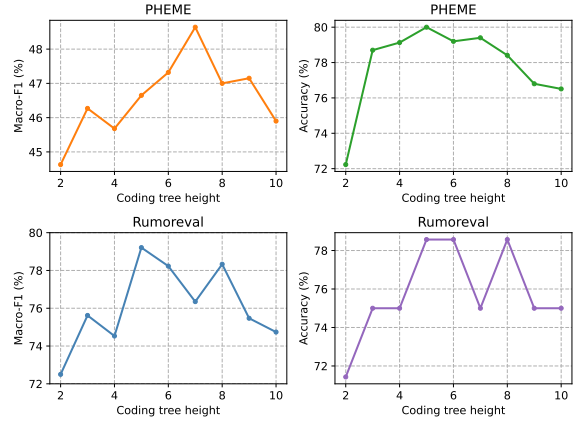


Figure 4: Performance analysis results on the PHEME and Rumoreval datasets in terms of coding tree height.

exhibits remarkable efficiency advantages through cross-layer weight sharing, especially when the height of the coding tree is high.

5.4 Performance Analysis

Impact of Coding Tree Height. Coding trees of varying heights capture distinct hierarchical information, thereby influencing the extraction and utilization of information from the leaf nodes. In Figure 4, we present an analysis of our model’s performance across different coding tree heights on the two datasets. It becomes evident that setting the coding tree height to 2 results in poor performance. This decline in performance can be attributed to the compression of the intricate rumor propagation structure into a coding tree of height 2, which results in the loss of valuable information. To preserve critical details within the propagation structure, a higher coding tree height is essential. For the PHEME dataset, we find that setting the coding tree height to 7 yields the best macro-F1 score of 48.6%. However, in terms of accuracy, the optimal coding tree height is 5. Conversely, for the Rumoreval dataset, our model demonstrates optimal performance when the coding tree height is set to 5, achieving a macro-F1 score of 79.2% and an accuracy of 78.6%. In summary, selecting an appropriate coding tree height significantly enhances model performance and ensures the effective utilization of information derived from the original claim and the propagation structure.

Early Rumor Detection. Early rumor detection aims to identify rumors at their inception, well before they gain widespread traction on social media platforms, enabling timely and appropriate re-

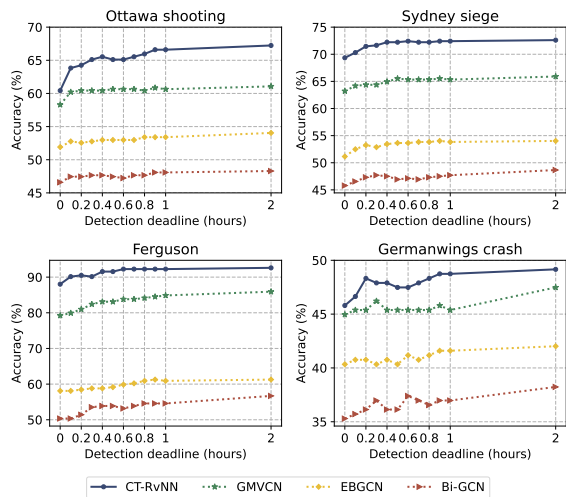


Figure 5: Early rumor detection performance analysis results for four distinct events from the PHEME dataset.

sponses. To assess our model’s effectiveness in early rumor detection, we follow the approach outlined in [Bian et al. \(2020\)](#). Specifically, we establish various detection deadlines relative to the publication time of the original claim, considering only posts preceding these deadlines to evaluate model performance. Our evaluation is conducted on the PHEME dataset, focusing on folds that use conversation threads related to *Ottawa Shooting*, *Sydney siege*, *Ferguson*, and *Germanwings crash* events for testing. We compare our model against three baseline models: Bi-GCN, EBGCN, and GMVCN. The results of our evaluation are presented in Figure 5.

It is worth noting that the performance of each model generally improves as the detection deadline is extended. This trend is expected, as a longer observation window allows for the accumulation of more available information. Notably, at each deadline, our model, CT-RvNN, consistently outperforms the baseline models. In summary, our model excels not only in long-term rumor detection but also significantly enhances the ability to identify rumors in their early stages.

Efficiency Analysis. In this section, we conduct an efficiency analysis by comparing the parameter counts of various models, including our proposed CT-RvNN and three baseline models: Bi-GCN, EBGCN, and GMVCN. As illustrated in Figure 6, CT-RvNN exhibits remarkable efficiency in terms of parameter count compared to the baseline models. This showcases its ability to operate with fewer computational resources while still achieving significant performance improvements. Notably, as a

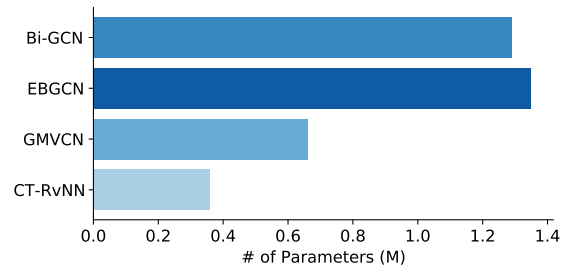


Figure 6: Comparison of the parameter counts of various models.

recursive neural network founded upon the GRU architecture, CT-RvNN maintains a consistent parameter count regardless of the coding tree height.

In comparison, Bi-GCN, EBGCN, and GMVCN follow the GCN paradigm. EBGCN and GMVCN, in particular, extend the bidirectional GCN framework introduced by Bi-GCN by constructing two directed graphs to represent rumor propagation: one top-down and the other bottom-up. This dual graph construction results in a doubling of the parameter count. Additionally, Bi-GCN and EBGCN augment the node features in the propagation tree with those of the root node. While this augmentation enhances the utilization of information from the original claim and improves model performance, it also introduces additional parameters.

6 Conclusion

In this paper, we use statistical analysis to underscore the significance of temporal information in predicting the veracity of rumors. By leveraging the structural and temporal characteristics of rumor propagation, we construct a time-weighted propagation tree. This tree is then refined into a coding tree through structural entropy minimization, effectively preserving the essential structure of rumor propagation while reducing noise. Finally, we introduce a recursive neural network to learn rumor representation from the coding tree. Experimental results demonstrate that our proposed approach outperforms current state-of-the-art methods while consuming fewer computational resources. Furthermore, in-depth analyses highlight the advantages of incorporating temporal information and the efficacy of the coding tree transformation.

Limitations

In line with prior research, we utilize TF-IDF as the sentence encoder. However, the performance

of CT-RvNN with alternative sentence encoders, such as static word embeddings, pre-trained language models, or large language models, remains unexplored.

Recent studies have shown that Transformers can effectively handle tree-structured data. Despite this, we chose to use RvNN over Transformers in this work. While Transformers may offer performance improvements, they introduce significantly more parameters, which conflicts with our primary goal—validating the effectiveness of our method design rather than optimizing performance through a more complex architecture. Additionally, using Transformers would complicate comparisons with prior research that uses RvNN. Nevertheless, integrating Transformers remains a promising direction for future research.

Moreover, previous studies have highlighted the benefits of incorporating rumor stance classification as an auxiliary task within multi-task learning frameworks to enhance rumor detection. These multi-task frameworks have demonstrated significant improvements over their corresponding single-task ablations (Wei et al., 2019; Yu et al., 2020; Liu et al., 2022; Luo et al., 2024). Despite consistently outperforming these multi-task models, our CT-RvNN holds untapped potential for further development in multi-task learning. In future work, we plan to integrate stance information into our model to enhance its performance.

Acknowledgements

This work has been supported by the Guangxi Science and Technology Major Project, China (No. AA22067070), NSFC (Grant No. 61932002), and CCSE project (CCSE-2024ZX-09).

References

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 *Proceedings of the AAAI conference on artificial intelligence*.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*.

Zhendong Chen, Siu Cheung Hui, Lejian Liao, and Heyan Huang. 2024. Ssri-net: Subthreads stance-rumor interaction network for rumor verification. *Neurocomputing*.

Yingying Cheng, Laijun Zhao, et al. 2021. Dynamical behaviors and control measures of rumor-spreading model in consideration of the infected media and time delay. *Information Sciences*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

L Derczynski, K Bontcheva, M Liakata, R Procter, GWS Hoi, and A Zubiaga. 2017. Semeval-2017 task 8: Rumoureal: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

Qi Huang, Chuan Zhou, Jia Wu, Luchen Liu, and Bin Wang. 2020. Deep spatial-temporal structure learning for rumor detection on twitter. *Neural Computing and Applications*.

Ling Min Serena Khoo, Hai Leong Chieu, Zhong Qian, and Jing Jiang. 2020. Interpretable rumor detection in microblogs by attending to user interactions. In *Proceedings of the AAAI conference on artificial intelligence*.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-1stm. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. 2013. Prominent features of rumor propagation in online social media. In *2013 IEEE 13th international conference on data mining*.

Angsheng Li and Yicheng Pan. 2016. Structural information and dynamical complexity of networks. *IEEE Transactions on Information Theory*.

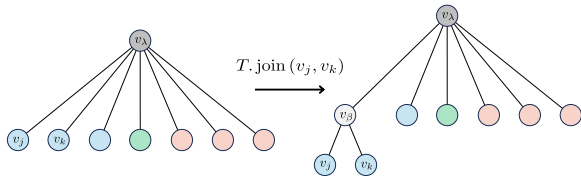
Hongzhan Lin, Jing Ma, Mingfei Cheng, Zhiwei Yang, Liangliang Chen, and Guang Chen. 2021. Rumor detection on twitter with claim-guided hierarchical graph attention networks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Yiwei Liu, Jiamou Liu, Zijian Zhang, Liehuang Zhu, and Angsheng Li. 2019. Rem: From structural entropy to community structure deception. *Advances in Neural Information Processing Systems*.

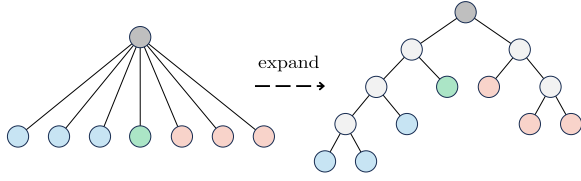
Yudong Liu, Xiaoyu Yang, Xi Zhang, Zhihao Tang, Zongyi Chen, and Zheng Liwen. 2022. Predicting rumor veracity on social media with graph structured multi-task learning. In *International Conference on Database Systems for Advanced Applications*.

Nanhang Luo, Dongdong Xie, Yiwen Mo, Fei Li, Chong Teng, and Donghong Ji. 2024. Joint rumour and stance identification based on semantic and structural information in social networks. *Applied Intelligence*.

- 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 *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*.
- Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM international on conference on information and knowledge management*.
- Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
- Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Rumor detection on twitter with tree-structured recursive neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
- 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 *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.
- Penghui Wei, Nan Xu, and Wenji Mao. 2019. Modeling conversation structure and temporal dynamics for jointly predicting rumor stance and veracity. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.
- Junran Wu, Xueyuan Chen, and Shangzhe Li. 2024a. Uncovering capabilities of model pruning in graph contrastive learning. In *Proceedings of the 32nd ACM International Conference on Multimedia*.
- Junran Wu, Xueyuan Chen, Bowen Shi, Shangzhe Li, and Ke Xu. 2023. Sega: Structural entropy guided anchor view for graph contrastive learning. In *International Conference on Machine Learning*.
- Junran Wu, Xueyuan Chen, Ke Xu, and Shangzhe Li. 2022a. Structural entropy guided graph hierarchical pooling. In *International conference on machine learning*.
- Junran Wu, Shangzhe Li, Jianhao Li, Yicheng Pan, and Ke Xu. 2022b. A simple yet effective method for graph classification. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*.
- Ke Wu, Song Yang, and Kenny Q Zhu. 2015. False rumors detection on sina weibo by propagation structures. In *2015 IEEE 31st international conference on data engineering*.
- Yang Wu, Jing Yang, Liming Wang, and Zhen Xu. 2024b. Graph-aware multi-view fusion for rumor detection on social media. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.
- Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. 2012. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD workshop on mining data semantics*.
- Feng Yu, Qiang Liu, Shu Wu, Liang Wang, Tieniu Tan, et al. 2017. A convolutional approach for misinformation identification. In *IJCAI*.
- Jianfei Yu, Jing Jiang, Ling Min Serena Khoo, Hai Leong Chieu, and Rui Xia. 2020. Coupled hierarchical transformer for stance-aware rumor verification in social media conversations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Chong Zhang, He Zhu, Xingyu Peng, Junran Wu, and Ke Xu. 2022. Hierarchical information matters: Text classification via tree based graph neural network. In *Proceedings of the 29th International Conference on Computational Linguistics*.
- He Zhu, Junran Wu, Ruomei Liu, Yue Hou, Ze Yuan, Shangzhe Li, Yicheng Pan, and Ke Xu. 2024. Hill: Hierarchy-aware information lossless contrastive learning for hierarchical text classification. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*.
- He Zhu, Chong Zhang, Junjie Huang, Junran Wu, and Ke Xu. 2023. Hitin: Hierarchy-aware tree isomorphism network for hierarchical text classification. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
- Dongcheng Zou, Hao Peng, Xiang Huang, Renyu Yang, Jianxin Li, Jia Wu, Chunyang Liu, and Philip S Yu. 2023. Se-gsl: A general and effective graph structure learning framework through structural entropy optimization. In *Proceedings of the ACM Web Conference 2023*.
- Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PLoS one*.



(a) Joining two descendants v_j and v_k of the root node v_λ .



(b) Expanding the initial tree into a full-height binary tree.

Figure A1: Illustration of the *join* operation and the expansion process.

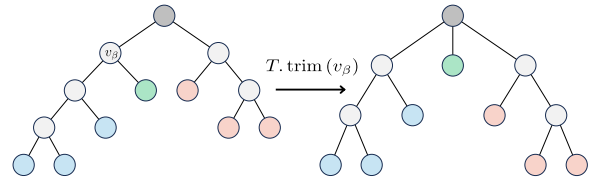
A Supplement to Algorithm 1

Coding Tree. A coding tree of a graph $G = (V, E)$ is defined as a rooted tree T that satisfies the following properties:

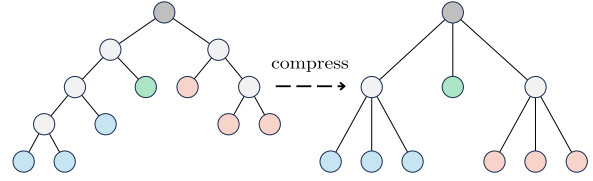
- Each non-leaf node $v_\gamma \in T$ corresponds to a non-empty subset of V , denoted as V_γ , which comprises the leaf nodes in the subtree of T rooted at v_γ . Specifically, the root node v_λ of T corresponds to the entire set V .
- Each leaf node $v_\zeta \in T$ corresponds to a unique node in V . That is, the subset V_ζ corresponding to a leaf node v_ζ is a singleton, containing exactly one node from V .
- If $v_{\beta_1}, v_{\beta_2}, \dots, v_{\beta_k}$ are descendants of a non-leaf node $v_\gamma \in T$, then the sets $\{V_{\beta_1}, V_{\beta_2}, \dots, V_{\beta_k}\}$ form a partition of V_γ . This means the sets are disjoint and collectively cover all elements of V_γ .

Illustrations. The *join* operation is illustrated in Figure A1(a). In this operation, a new node v_β is inserted between the root node v_λ and its two descendants, v_j and v_k . As a result, v_j and v_k become the descendants of v_β , while v_β itself becomes a descendant of v_λ . The selection of v_j and v_k is aimed at minimizing structural entropy. By iteratively selecting nodes v_j and v_k and performing the *join* operation, the initial tree (of height 1) is expanded into a full-height binary tree. This expansion process is illustrated in Figure A1(b), corresponding to Lines 2-5 in Algorithm 1.

The *trim* operation, as illustrated in Figure A2(a), involves removing an internal node v_α from the



(a) Trimming an internal node v_β .



(b) Compressing the tree to a specific height.

Figure A2: Illustration of the *trim* operation and the compression process.

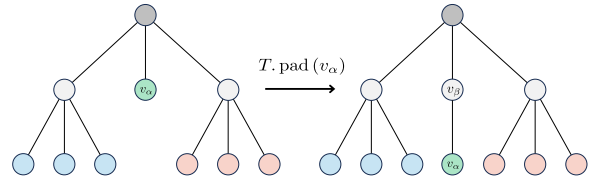


Figure A3: Padding a non-root node v_α .

coding tree and adopting its descendants to its ancestor. The selection of v_α aims to minimize the increase in structural entropy. Through an iterative process of selecting node v_α and executing the *trim* operation, the tree achieves the prescribed height. This compression process is illustrated in Figure A2(b), corresponding to Lines 6-9 in Algorithm 1.

The *pad* operation, depicted in Figure A3, entails inserting a new node v_β between a non-root node v_α and its ancestor. We iteratively select v_α such that $Height(v_\alpha^-) - Height(v_\alpha) > 1$ and apply the *pad* operation until all leaf nodes attain a consistent depth, facilitating the subsequent coding tree representation learning.

Complexity Analysis. The time complexity of the coding tree construction algorithm is $O(2|V| + h_{\max}(|E| \log |V| + |V|))$, where h_{\max} is the height of the full-height binary tree. It is worth noting that the algorithm tends to construct balanced coding trees, ensuring that h_{\max} is at most $\log(|V|)$. Additionally, in typical scenarios, graphs tend to have more edges than nodes, i.e., $|E| \gg |V|$, which implies that the runtime of the algorithm scales almost linearly with the number of edges.

B Proof for Proposition 1

Proof. According to Equation (1), $\mathcal{H}^T(G)$ is the summation of $\mathcal{H}_{v_\alpha}^T = -\frac{g_\alpha}{\text{vol}(G)} \log_2 \frac{\text{vol}(v_\alpha)}{\text{vol}(v_\alpha^-)}$ for all non-root node v_α in T . That is, $\mathcal{H}^T(G) = \mathcal{H}_{v_j}^T + \mathcal{H}_{v_k}^T + \dots$. Denote the structural entropy after padding v_k as $\mathcal{H}^{T,\text{pad}(v_k)}(G)$. Likewise, $\mathcal{H}^{T,\text{pad}(v_k)}(G) = \mathcal{H}_{v'_j}^{T,\text{pad}(v_k)} + \mathcal{H}_{v'_k}^{T,\text{pad}(v_k)} + \mathcal{H}_{v'_k}^{T,\text{pad}(v_k)} + \dots$. Here, v_β is the inserted internal node between v_j and v_k , while v'_j and v'_k correspond to v_j and v_k after the *pad* operation. The following equations hold:

$$\begin{aligned} \mathcal{H}_{v'_k}^{T,\text{pad}(v_k)} &= -\frac{g'_k}{\text{vol}(G)} \log \frac{\text{vol}(v'_k)}{\text{vol}(v'_k^-)} \\ &= -\frac{g'_k}{\text{vol}(G)} \log \frac{\text{vol}(v'_k)}{\text{vol}(v_\beta)} \\ &= -\frac{g'_k}{\text{vol}(G)} \log \frac{\text{vol}(v'_k)}{\text{vol}(v'_k)} \\ &= 0, \end{aligned} \quad (\text{A1})$$

$$\begin{aligned} \mathcal{H}_{v'_\beta}^{T,\text{pad}(v_k)} &= -\frac{g_\beta}{\text{vol}(G)} \log \frac{\text{vol}(v_\beta)}{\text{vol}(v_\beta^-)} \\ &= -\frac{g_k}{\text{vol}(G)} \log \frac{\text{vol}(v_k)}{\text{vol}(v_j)} \\ &= -\frac{g_k}{\text{vol}(G)} \log \frac{\text{vol}(v_k)}{\text{vol}(v_k^-)} \\ &= \mathcal{H}_{v_k}^T, \end{aligned} \quad (\text{A2})$$

$$\begin{aligned} \mathcal{H}_{v'_j}^{T,\text{pad}(v_j)} &= -\frac{g'_j}{\text{vol}(G)} \log \frac{\text{vol}(v'_j)}{\text{vol}(v'_j^-)} \\ &= -\frac{g_j}{\text{vol}(G)} \log \frac{\text{vol}(v_j)}{\text{vol}(v_j^-)} \\ &= \mathcal{H}_{v_j}^T. \end{aligned} \quad (\text{A3})$$

Thus, we have $\mathcal{H}^T(G) = \mathcal{H}^{T,\text{pad}(v_k)}(G)$. \square

C Detailed Experimental Setup

Datasets. The PHEME dataset comprises 2402 conversation threads related to nine events. To ensure robust results, we adopt a leave-one-event-out cross-validation approach following established practices. In each fold of the cross-validation, we use the conversation threads associated with one event for testing, while the conversation threads related to the remaining eight events are used for training.

Statistic	PHEME	Rumoreval
# of users	18813	3859
# of posts	32925	5568
# of authors	1024	203
# of claims	2402	325
# of true rumors	1067	145
# of false rumors	638	74
# of unverified rumors	697	106
Avg. # of posts	13.7	17.1
Avg. depth of the propagation tree	2.8	3.4
Avg. rumor lifespan (hours)	8.7	23.3

Table A1: Statistics of datasets.

The Rumoreval dataset contains 325 conversation threads, officially divided into training, validation, and test sets. These conversation threads are related to ten events, and the test set covers two events that are not present in the training and validation sets.

Each claim in both datasets is categorized into one of three classes: true rumor, false rumor, or unverified rumor.

Implementation Details. For model training, we use the AdamW optimizer and implement a linear learning rate scheduler with a 6% warmup and a maximum learning rate of 0.001. Additionally, we apply an L2 regularization weight penalty of 0.0005 to mitigate potential overfitting.

Baselines. In our evaluation, we compare our model against the following competitive baselines:

- **BranchLSTM** (Kochkina et al., 2017) is an LSTM-based sequential model designed to capture the conversation structure of a claim and its associated posts.
- **TD-RvNN** (Ma et al., 2018) is a top-down tree-structured model that utilizes recursive neural networks to model the propagation layout of rumors.
- **Hierarchical GCN-RNN** (Wei et al., 2019) utilizes a GCN to encode conversation structures for stance classification and an RNN to capture the temporal dynamics of stance evolution for veracity prediction.
- **HiTPLAN** (Khoo et al., 2020) is a structure-aware self-attention network that incorporates propagation structural information into the Transformer model.

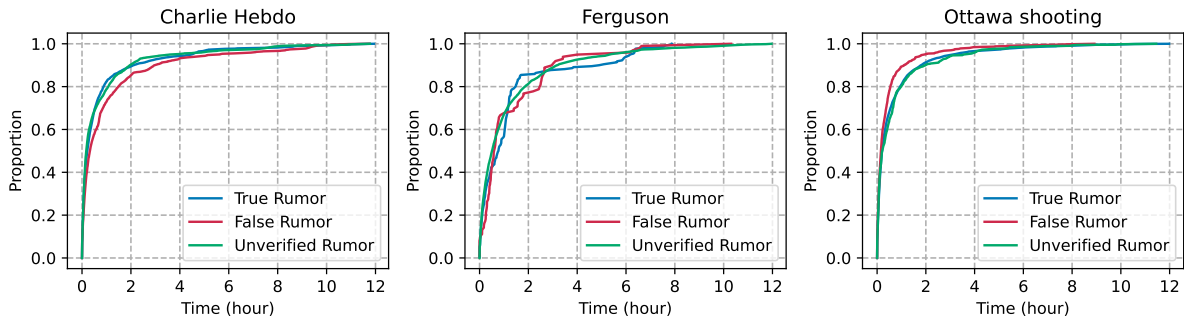


Figure A4: The ECDF plots of the time delay distributions since the initial claim was posted for posts responding to true, false, and unverified rumors for *Charlie Hebdo*, *Ferguson*, and *Ottawa shooting* events from the PHEME dataset.

- **Hierarchical Transformer** (Yu et al., 2020) generates conversation thread representations by dividing each long thread into shorter subthreads and employing BERT to separately represent each subthread.
- **Bi-GCN** (Bian et al., 2020) applies two GCNs on the top-down directed graph and the bottom-up directed graph to learn patterns of rumor propagation and dispersion.
- **ClaHi-GAT** (Lin et al., 2021) employs a claim-guided hierarchical attention mechanism at both post- and event-level to attend to informative posts.
- **EBGCN** (Wei et al., 2021) extends Bi-GCN by handling uncertainty in the propagation structure with a Bayesian method, adaptively adjusting weights of unreliable relations.
- **STL-GT** (Liu et al., 2022) utilizes two shared channels to extract task-invariant textual and structural features.
- **MTL-SMI** (Liu et al., 2022) extends STL-GT by incorporating two task-specific graph channels for multi-task learning.
- **SSRI-Net** (Chen et al., 2024) utilizes an attention mechanism to capture interaction details among subthreads and a stance-rumor interaction network to integrate users' stance information with rumor verification.
- **Luo et al. (2024)** employs a graph transformer to concurrently gather structural and semantic information and a partition filter network to explicitly model rumor- and stance-specific features.
- **GMVCN** (Wu et al., 2024b) is a multi-view fusion framework that utilizes a GCN and a CNN to encode and capture complementary information from various conversation views.

D Explanations for Figures 2 and A4

In Figures 2 and A4, statistically significant differences consistently appear in the time delay distributions among the three categories of rumors. Specifically, noteworthy disparities are evident in the distributions of true and false rumors across all four events. During the *Charlie Hebdo* and *Ottawa shooting* events, the distribution of unverified rumors closely resembles that of true rumors. However, in the remaining two events, the distribution of unverified rumors diverges significantly from the other two distributions. This observation highlights the critical role of temporal information in predicting the veracity of rumors.