Information diffusion analysis in online social networks based on deep representation learning

Chen, X.

Chapter 7

Participant-level Rumor detection based on Information Diffusion Analysis

Users are the main contributor to the rumor spreading in Online social networks (OSNs) [26], which go through the whole life cycle of news diffusion. In this chapter, we propose two independent and complementary participant-level models for rumor detection, i.e., PLRD (Participant-Level Rumor Detection model) and UMLARD (User-aspect Multi-view Learning with Attention for Rumor Detection mlodel). And both PLRD and UMLARD are implemented according the following two design considerations.

First, we considered the life cycle of real and fake news on social media, which plays a crucial role in information diffusion. When the news is produced by the content creator, it starts its journey on the social media platform. Once people are exposed to the news, they become the content consumers. According to the confirmation bias theory, people tend to favor, interpret and share information in a way that confirms or strengthens their prior beliefs or ideologies [145]. As a result, if a news item confirms the consumers' prior ideology, they may share it within their social networks in the role of content distributor. Since fake news is intentionally written to mislead readers into believing and propagating false information (e.g., 5G networks trigger COVID-19), it is plausible that fake news is more easily distributed among its believers than real news, which is neutral in its beliefs and ideology. This idea is supported by prior studies, which have noted that false information tends to spread significantly faster, further, deeper, and more broadly than real information [43]. Therefore, considering all participants, including content creators and content distributors, in the news diffusion chain may improve the overall rumor detection performance of our approach.

Another design consideration is the lack of effective methods to represent all participants in the news diffusion chain. Prior predictive studies have simply used aggre-

gated statistics, such as the total number of content distributors (retweets) and the average time of information distribution, to quantify the diffusion process. This inevitably results in information loss and suboptimal performance. Other examples of such aggregated statistics include network-level attributes (e.g., density) to represent diffusion networks, the final hidden representation from recurrent neural networks to model the temporal spreading sequence, and overall descriptive statistics of user characteristics (e.g., mean user tenure) to describe users in the diffusion [23]. While such data may be helpful in modeling, they are not quite specific enough to provide a clear picture of by whom, when, why, and how news is diffused. Therefore, the key question motivating our study is how to design an effective predictive method that represents all-participant patterns throughout the whole diffusion process.

7.1 PLRD: A Participant-Level Rumor Detection Framework via Fine-grained User Representation Learning

7.1.1 Section Overview

In this section, we propose a novel framework based on deep representation learning for rumor detection, named PLRD (Participant-Level Rumor Detection). In view of theories on propagation and social influence, PLRD incorporates multi-scale features of all users[1] enrolled in the diffusion process to predict a given post's credibility (e.g., classify it as rumor or non-rumor). Specifically, PLRD first employs sparse matrix factorization to embed the global graph (i.e., user-interaction graph constructed on all propagation threads), which can efficiently learn the social homophily for users. Then, it uses a multi-hop graph convolutional layer, and a bi-directional GRU to learn fine-grained user representations (i.e., the user influence, user susceptibility, and user temporal information). To understand the different importance of users' multi-scale representations, a feature-level attention layer was designed to explain which types of features are essential in rumor propagation. Moreover, to capture the uncertainty in learned features, PLRD introduces a variational autoencoder. Finally, PLRD employs a user-level attention layer to allocate different importance to users and aggregate them to form a unique rumor representation. The rumor prediction is generated based on the learned unique representation.

Our design science work on rumor detection makes two main contributions to the literature in this field. First, our design is rooted in social influence and propagation theory, from which we derive various constructs in our model. PLRD detects rumors at a very fine-grained participant level. It is very different from prior works that also combine information from various sources (e.g., reply networks, diffusion sequence,

[1] In our work, the user influence, user susceptibility and user temporal information are collectively referred to as multi-scale information of users

and attributes of spreading users), but still heavily depend on the text features. Our approach comprehensively exploits user-profiles and propagation threads and shows a strong ability to detect rumors without using any text information. Second, we make a methodological contribution by proposing an approach to learn fine-grained user representation via deep representation learning that effectively captures all participant information in a diffusion chain. This information includes user influence, user susceptibility, and user temporal information. Experimental results using real-world datasets confirm the effectiveness of our approach over prior rumor detection methods. Our approach has direct implications for social media platforms that are vulnerable to rumor spreading since it can be deployed to identify original users who initiate rumors and those who spread rumors. Overall, the proposed rumor-detection model can help improve the user experience and benefit society by helping individuals obtain healthy and genuine information.

This section is based on the following publication [45]:

- **Chen, X.**, Zhou, F., Zhang, F., Bonsangue, M.: Catch me if you can: A participant-level rumor detection framework via fine-grained user representation learning. Information Processing & Management, 58(2021) 102678

7.1.2 Birds of a feather flock together: the perspective of all participants

Rumor detection has long been a subject of interdisciplinary research. Various theories have been proposed and validated. In this section, we discuss several major theories that guide us to derive relevant constructs in our model for better rumor detection.

7.1.2.1 Theory

Users play major roles in the dissemination of rumors or fake news. A set of user-based and propagation-based theories have been developed to study how a rumor spreads, how users engage with a rumor, and the role users play in rumor creation, propagation, or intervention. For example, in the echo chamber effect, individuals tend to believe information is correct after repeated exposures [146]. Confirmation bias theory tells us that individuals tend to trust information that confirms their preexisting beliefs or hypotheses, which they perceive to surpass that of others [147]. People choose to interact with those who share similar opinions and avoid those with whom they profoundly disagree. Both indicate that people may react to and process information differently based on information type (e.g., rumor vs. non-rumor). On the other hand, homophily theory says that individuals in homophilic relationships share common characteristics (beliefs, demographics, etc.) that make communication easier [148]. Meanwhile, social identity theory shows that individuals do something primarily because others are doing it and to conform in order to be

liked and accepted by others. Such social influence and homophilic atmospheres also exist in and are commonly seen in online social networks [149].

In sum, all the above considerations suggest that user attributes likely have an impact on rumor detection, such as user influence, susceptibility, and temporal, etc. This assumption is also empirically confirmed by our data exploration (see below) and computational experiments (see the Evaluation section below). We therefore hypothesize that:

Hypothesis 1. *Combining various information at a fine-grained all-participant level in a diffusion chain will improve rumor discovery performance.*

Hypothesis 2. *Deep representation learning-based methods will improve rumor discovery performance compared to using shallow machine learning methods.*

7.1.2.2 Data

In our work, we conduct experiments on four standard real-word testbeds: Twitter15, Twitter16, Science and RumourEval19, all of which were collected from Twitter[1], one of the most popular social media platforms in the U.S. The descriptive statistics of all datasets are shown in Table 7.1.

Twitter15/16[2] were released by [84]. More details can be found in Section 6.4.1. In Twitter15 and Twitter16, we keep the tweets labeled as "non-rumor" or "false rumor" (relabeled as "rumor" in our work), since the others were beyond our research interest.

Science[3] is collected and studied by [43]. It includes complete retweet cascades linked to rumors that were verified and published by fact-checking websites. In the original data, each tweet cascade is related to a specific label, i.e., "TRUE", "FALSE" or "MIXED". In our work, we keep only the tweets labeled as "TRUE" or "FALSE" (relabeled as "non-rumor" and "rumor" in our work, respectively). To the best of our knowledge, the Science dataset is the most credible dataset among all the existing Twitter-based rumor detection datasets as it overcomes issues of partiality or bias because of the sampling restriction characteristic. In this chapter, we use the Science dataset to provide model-free evidence to support the **Hypothesis 1**.

RumourEval19[4] [150] is an extensive dataset from RumourEval17 [151], which is augmented with new Twitter test data and Reddit material. Here, we keep Twitter data but ignore the Reddit material, to finally obtain 381 Twitter conversation threads. Each thread consists of a claim and a tree of comments, and is related to a specific label, i.e., "true", "false" or "unverified". We filter out unverified tweets and finally get 271 Twitter conversation threads (relabeled as "non-rumor" and "rumor",

[1]https://twitter.com/

[2]https://www.dropbox.com/s/46r50ctrfa0ur1o/rumdect.zip

[3]Researchers interested in gaining access to Science dataset should contact [43] directly.

[4]https://competitions.codalab.org/competitions/19938

respectively). Because the RumourEval19 contains rich textual information rather than diffusion patterns, in the experiments part, we use RumourEval19 to test our model performance when including textual features.

Table 7.1: Statistics of the datasets.

Statistic	Science	Twitter15	Twitter16	RumourEval19
# source tweets	16,901	739	404	271
# users	3,603,265	306,402	168,659	4,774
# edges of global graph	3,586,360	336,486	182,493	4,503
# non-rumors	14,442	370	199	167
# rumors	2,459	369	205	104
Max. # retweets	46,895	2,990	999	155
Min. # retweets	5	97	100	3
Avg. # retweets	213	493	479	18
Avg. # time length	749 Hours	743 Hours	167 Hours	37 Hour

For each tweet, we construct the diffusion graph and the global graph from the propagation threads (see Section 3.1 for recalling the formal definitions). In Twitter15/16, no user information is provided due to constraints set out in Twitter's terms of service. We crawl all the related user profiles via *Twitter API*[1] based on the provided user IDs. We follow the work of Liu et al. [100] and select 8 general user characteristics for experiments, which are summarized in Table 7.2. As for Science, all data, such as ids, were anonymized, so we directly use the user characteristics provided in this dataset, which are listed in Table 7.3. Here, the concrete definitions of each characteristic can be found in the supplementary of [43]. The RumourEval19 provided JSON files for each source tweet and its corresponding replies, and each file contains the complete information of the tweet and the users.

Table 7.2: Summary of user characteristics for Twiter15/16 and RumourEval19.

No.	Characteristic	Data Type
1	LENGTH OF USER NAME	Integer
2	USER COUNT CREATED TIME	Integer
3	LENGTH OF DESCRIPTION	Integer
4	FOLLOWERS COUNT	Integer
5	FRIENDS COUNT	Integer
6	STATUSES COUNT	Integer
7	IS VERIFIED	Binary
8	IS GEO ENABLED	Binary

7.1.2.3 Model-free evidence

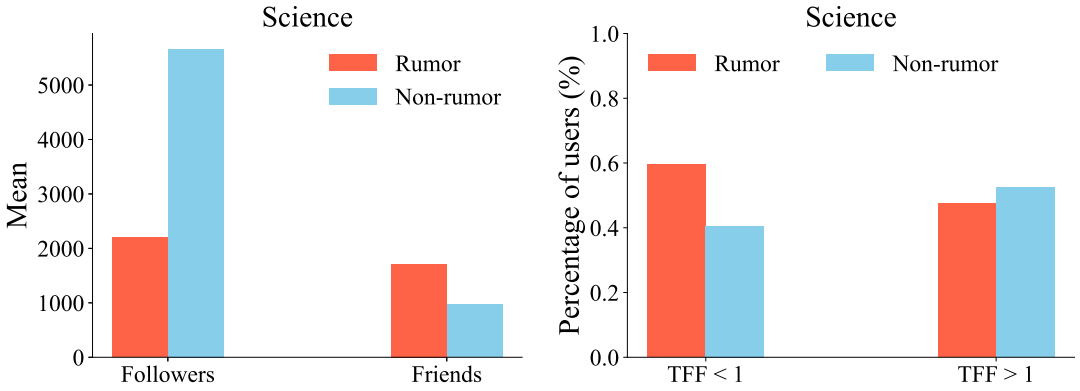
Following from our earlier **Hypothesis 1** that utilizing the information of all users who participated in a diffusion chain might improve rumor detection, we first check

[1]https://dev.twitter.com/rest/public

Table 7.3: Summary of user characteristics for Science.

No.	Characteristic	Data Type
1	USER COUNT CREATED TIME	Integer
2	FOLLOWERS COUNT	Integer
3	FRIENDS COUNT	Integer
4	ENGAGEMENT	Float
5	IS VERIFIED	Binary

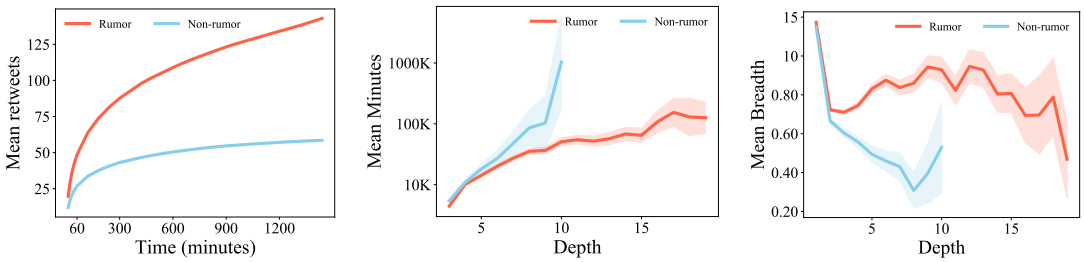
for any patterns or differences among involved participants in terms of their overall attributes across the rumors and non-rumors via analyzing the Science dataset.



(a) Follower/friends distribution **(b)** TFF distribution

Figure 7.1: Social influence/susceptibility analysis on Science.

- **Social influence/susceptibility: followers/friends.** Social influence may affect the speed and the depth of diffusion. On Twitter, a user's social influence can be measured by the size of their social circle in relation to two factors: the number of friends (i.e., users the specific user is following) and the number of followers (i.e., users following the specific user). We calculate the average followers/friends across all participants and find that this average in the rumors group is different from that in the non-rumors group (see Figure 7.1a). To compute the social influence of all participants, we define a new metric, *TFF* (the follower-friend-ratio [152]), which combines followers and friends: $TFF = \frac{\#\text{followers}}{\#\text{friends}}$. Users with $TFF < 1$ are less influential but with higher susceptibility, since they have fewer followers than friends, and extreme cases are fake users. In contrast, users with $TFF > 1$ are more influential, e.g., celebrity accounts. Figure 7.1b shows that a higher percentage of users with $TFF < 1$ are involved in rumors, while more influential users participate in non-rumors.
- **Structural and temporal impact of diffusion.** Many prior studies have demonstrated that falsehood diffuses significantly farther, faster, deeper, and



(a) Propagation speed **(b)** Depth-time **(c)** Depth-breadth

Figure 7.2: Structural and temporal analysis on Science.

more broadly than the truth in all categories of information [43]. This being the case, we expect to see that the spreading pattern, in terms of network structures and the temporal sequence of retweeting, should vary based on rumor type. To this goal, we analyze the propagation speed, depth-time distribution and depth-breadth distribution, respectively, which are shown in Figure. 7.2. In the diffusion graph, the depth of a node is the number of hops from the node to the source node, and the breadth at a specific depth of a graph is the total number of nodes at this depth level. From Figure. 7.2a, we can find that at the same time-scale, rumors can infect more users than non-rumors, which demonstrates that rumors spreading faster than non-rumors. In Figure. 7.2b, we measured the average time (in minutes) rumor or non-rumor tweets took to reach different depths, where we observe that (1) rumors can reach deeper users than non-rumors, and (2) rumors require less time to reach the same depth with non-rumors. In Figure. 7.2c, we plot the average breadth at every depth for rumors and non-rumors, which indicates that rumors spread broader than non-rumors, especially at a deeper level. These observations substantiate our belief that incorporating such structural and temporal information into the model may improve rumor detection performance.

7.1.3 Methodology

In this section, we first present a preliminary overview of rumor detection in our context and describe the overall framework of the proposed PLRD method, followed by details of each component in the model. As discussed above, two challenges need to be addressed when designing an effective rumor detection system: (1) how to incorporate fine-grained all-participant information in one model – not simply aggregating or concatenating – to capture rumor spreading patterns; and (2) how to effectively learn latent representations of users' propagation activities in a diffusion chain to capture fine-grained user representations. To answer these two questions, we first formally define our problem and describe its context.

Table 7.4: Main notations used throughout this chapter.

Symbol	Description
$\mathcal{G}, N$	Global graph, and the number of nodes of global graph.
G_i, G_i^T	Diffusion graph and inverse diffusion graph of m_i.
$\mathbf{E}^{\mathcal{G}}, \mathbf{e}_*^{\mathcal{G}}$	The user social homophily embedding matrix and social homophily vector for user u_*.
$\mathbf{T}_i$	Time-embedding of m_i.
$\mathbf{H}_i^{inf}$	The representations of the user influence.
$\mathbf{H}_i^{sus}$	The representations of the user susceptibility.
$\mathbf{H}_i^{temp}$	The representations of the user temporal information.
$\bar{\mathbf{U}}_i$	The representation after feature-level attention.
$\mathbf{U}_i^z$	The representation after VAE.
$\mathbf{U}_i^F$	The representation after concatenate operation.
$\mathbf{R}_i$	the final representation of tweet m_i.
$\hat{\mathbf{Y}}/\hat{\mathbf{y}}_*$	The predicted label.
$\mathbf{Y}/\mathbf{y}_*$	The ground truth.

7.1.3.1 Preliminaries and Problem Statement

In this section, we give the necessary background and formally define the rumor detection problem. We list the main mathematical notations used throughout the paper in Table 7.4.

In this study, we formalize our rumor detection problem as a supervised binary-class classification task. Suppose the input of the task is from a rumor detection dataset (e.g., Twitter) consisting of a set of posts (e.g., tweets) denoted as $M = \{m_i, i \in [1, |M|]\}$. Each m_i corresponds to its own diffusion process and all participating users, so that m_i can be represented by $\{C_i, \mathbf{U_i}\}$, where C_i and $\mathbf{U_i}$ are the cascade graph and the user characteristics matrix (see Definition 1 and Definition 5), respectively. The cascade graph C_i can be further broken down into a diffusion graph G_i and diffusion path $\mathcal{P}_i$ (see Definition 2 and Definition 3). In addition, we construct a global graph $\mathcal{G}$ based on all tweets M, to represent the social homophily among all users. See their formal definitions below.

Definition 13 ***Global Graph*** $\mathcal{G}$**.** *The global graph $\mathcal{G} = \{U, E\}$ is a collection of nodes and edges, which is constructed based on all posts in the dataset. $U = \bigcup_{i=1}^{|M|} U_i$ is a user set contains all users in dataset, and E is the edge set. An edge between u_i and u_j refers to these two users share the same tweet (or discuss the same topic).*

Definition 14 ***Rumor Detection.*** *Given a tweet $m_i = \{C_i(t_o), \boldsymbol{U}_i(t_o)\}$ within an observation window t_o (in our work, t_o is the total number of retweets), and the global graph $\mathcal{G}$, the goal of rumor detection is to learn a function $f(\hat{y}_i|m_i, \mathcal{G})$ to classify the source tweet m_i into one of the rumor categories, where the predicted result $\hat{y}_i$ represents either non-rumor or rumor.*

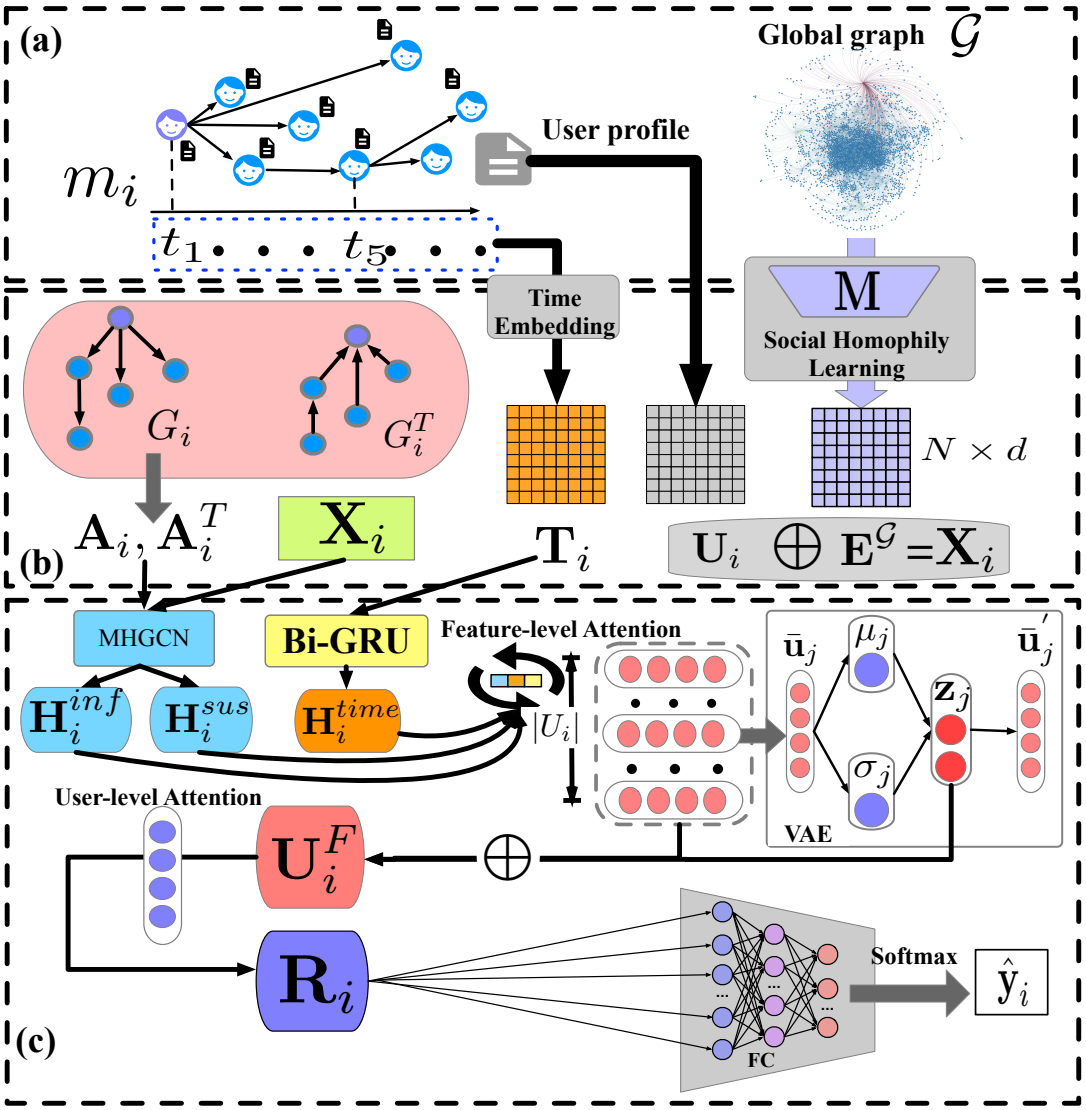


Figure 7.3: Overview of **PLRD**: (a) input of **PLRD**; (b) preprocessing layer; (c) fine-grained user representation learning and rumor detection layer.

7.1.3.2 Overall framework of PLRD

In this section, we describe our proposed PLRD rumor detection system. It consists of the following components (see Figure 7.3): (a) inputs, including (1) the propagation threads of tweet m_i, and (2) a global graph constructed on all tweets propagation threads; (b) the preprocessing layer, which consists of (1) constructing a diffusion graph and inverse diffusion graph based on propagation threads, (2) constructing a user characteristic matrix based on user profiles, (3) pre-training retweet time-stamps via positional encoding, and (4) pre-training global graph via randomized truncated singular value decomposition-based sparse matrix factorization; and (c) the fine-grained user representation learning layer and the rumor detection layer, in which we learn user influence and susceptibility via a multi-hop graph convolution layer, model user temporal feature via bi-directional GRU, then aggregate and enhance the learned multi-scale user representations through a feature-level attention

layer and a variational autoencoder. Finally, after a user-level attention layer, we feed the unique rumor representation into a rumor classifier. Specifically, we use several fully connected feedforward layers and a softmax output layer to generate a rumor prediction. Below, we explain each of the above components in detail.

7.1.3.3 Social homophily learning from global graph

As mentioned in section 7.1.3.1, we construct the global graph based on all the retweet threads in the dataset. Our goal is to capture social homophily for all users. The social homophily among users specifies that users with similar interests are more likely to closely connected [153]. We assume that the users who discuss the same post in social communities share homophilous relationships in our work. Then, we try to model social homophily in an unsupervised manner, which encourages users with shared social neighborhoods to have similar latent representation.

More specifically, we cast the problem of learning social homophily as the task of graph embedding. The global graph $\mathcal{G}$ always contains tens of thousands of nodes, and to model such a large graph effectively is a tough challenge in the field of graph representation learning [154, 155]. Inspired by the success of sparse matrix factorization (SMF) in large-sized graph representation learning, in our work, we use a randomized tSVD-based SMF to learn social homophily from the global graph. Here, tSVD is truncated singular value decomposition, which can prevent the problem of infeasible computation of factorization for a large-sized matrix [156, 157]. Specifically, given global graph $\mathcal{G}$, we can obtain the adjacency matrix $\mathbf{A}^{\mathcal{G}} \in \mathbb{R}^{N\times N}$ and diagonal degree matrix $\mathbf{D}^{\mathcal{G}} \in \mathbb{R}^{N\times N}$, N is the number of nodes in global graph. Each entry $\mathbf{A}^{\mathcal{G}}_{i,j}$ of $\mathbf{A}^{\mathcal{G}}$ equals to 1 when u_j and u_j share the same post or $i = j$, otherwise $\mathbf{A}^{\mathcal{G}}_{i,j} = 0$. And $\mathbf{D}^{\mathcal{G}}_{i,i} = \sum_j^N \mathbf{A}^{\mathcal{G}}_{i,j}$. To learn the embedding of $\mathcal{G}$ via randomized tSVD-based SMF, we first define a proximity matrix $\mathbf{M}^{\mathcal{G}}$ as:

$$\mathbf{M}^{\mathcal{G}}_{i,j} = \begin{cases} \ln p^{\mathcal{G}}_{i,j} - \ln(\lambda \mathcal{N}^{\mathcal{G}}_{E,j}), & (u_i, u_j) \in E \\ 0, & \text{otherwise} \end{cases} \tag{7.1}$$

where $p^{\mathcal{G}}_{i,j} = \mathbf{A}^{\mathcal{G}}_{i,j}/\mathbf{D}^{\mathcal{G}}_{i,i}$ indicates the weight of (u_i, u_j) in E. $\mathcal{N}^{\mathcal{G}}_{E,j}$ are the negative samples connected with user u_j, which can be defined as $\mathcal{N}^{\mathcal{G}}_{E,j} \propto (\sum_{i:(i,j)\in E} p_{i,j})^{3/4}$ ($y \propto x$ means that y and x are in a directly proportional relation) [158]. The goal of the global graph embedding is transformed to factorize the matrix $\mathbf{M}^{\mathcal{G}}$. Specifically, the step for the approximate matrix factorization of $\mathbf{M}$[1] are as follows: (1) we first look for a matrix $\mathbf{Q} \in \mathbb{R}^{N\times d}$ with d orthonormal columns that let $\mathbf{M} \approx \mathbf{Q}\mathbf{Q}^T\mathbf{M}$; (2) suppose we found such matrix $\mathbf{Q}$, we define $\hat{\mathbf{M}} = \mathbf{Q}^T\mathbf{M} \in \mathbb{R}^{d\times N}$, which is quite smaller compare with the original matrix $\mathbf{M}$. Then we have $\hat{\mathbf{M}} = \mathbf{S}\mathbf{\Sigma}\mathbf{V}^{\mathbf{T}}$, where $\mathbf{\Sigma}$ is the diagonal matrix with top-d singular values, and $\mathbf{S}, \mathbf{V} \in \mathbb{R}^{N\times d}$ are orthonormal matrices with d selected singular values; (3) finally, the factorization of $\mathbf{M}$ is approximate to $\mathbf{M} \approx \mathbf{Q}\mathbf{Q}^T\mathbf{M} = (\mathbf{Q}\mathbf{S})\mathbf{\Sigma}\mathbf{V}^{\mathbf{T}}$ and the calculation of the output

[1] for brevity, we ignore the superscript $\mathcal{G}$.

embeddings for the global graph is $\mathbf{E}^{\mathcal{G}} = \mathbf{Q}\mathbf{S}\mathbf{\Sigma}^{\mathbf{1/2}}$ and $\mathbf{E}^{\mathcal{G}} = \{\mathbf{e}_j^{\mathcal{G}} | j \in [1, N]\} \in \mathbb{R}^{N \times d}$. That is, for each user u_j in the global graph, was allocated a relative latent embedding, i.e., $\mathbf{e}_j^{\mathcal{G}}$, and users with similar preferences and behavior (i.e., interested in the same posts) will have similar embeddings.

7.1.3.4 Users' influence and susceptibility learning

The role of each participant in the diffusion process of m_i has two types, i.e., sender and receiver [159]. In previous diffusion research [160, 161], the role of the sender reflects a user's influence, that is, the ability to spread information to other users. On the opposite, the receiver role reflects the susceptibility of a user, i.e., the ability of a user to be infected by possible senders. In our work, we learn the influence and susceptibility for each user from the diffusion graph G_i of m_i. However, in the original diffusion graph G_i, the information passed from a sender to a receiver, so that modeling G_i can acquire influence for each user, is not efficient for susceptibility learning. To overcome this problem, we introduce an inverse diffusion graph G_i^T, which changes the direction of information propagation, i.e., from receiver to sender.

Inspired by the recent success of deep learning technologies in graph representation learning, such as graph convolutional network (GCN) [41, 61], and graph attention network (GAT) [109], in order to model the higher-order relationships among participants, we propose a multi-hop graph convolution layer (MHGCN) to extract user influence and susceptibility from G_i and G_i^T, respectively. The convolution kernel of MHGCN is defined as:

$$\mathbf{H} = g_\theta * \mathbf{X} = \sigma(\mathop{\Big\|}_{k \in \mathcal{O}} (\hat{\mathbf{A}}^{(k)} \mathbf{X} \mathbf{W}^{(k)})) \tag{7.2}$$

where $\mathop{\Big\|}_{k \in \mathcal{O}}$ represents the order-level concatenate, and σ is a non-linear activation function such as ReLU. $\hat{\mathbf{A}}^{(k)}$ denotes the normalized adjacency matrix $\hat{\mathbf{A}} \in \mathbb{R}^{|U| \times |U|}$ multiplied by itself k times, $|U|$ is the number of nodes in graph, and $\mathcal{O}$ is a set of integer adjacency powers from 0 to K, K is the max-order. The calculation of normalized adjacency matrix is denoted as $\hat{\mathbf{A}} = \bar{\mathbf{D}}^{-1}\bar{\mathbf{A}}$, where $\bar{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, and $\mathbf{I}$ is diagonal identity matrix. $\mathbf{X} \in \mathbb{R}^{N \times d_X}$ is the input graph signal, d_X is the dimension number. $\mathbf{W}^{(k)} \in \mathbb{R}^{d_X \times F}$ is the weight matrix for different order. Given the diffusion graph G_i and its inverse diffusion graph G_i^T, we have:

$$\begin{aligned} \mathbf{X}_i &= \text{Concat}(\mathbf{E}_i, \mathbf{U}_i), \\ \mathbf{H}_i^{inf} &= \sigma(\mathop{\Big\|}_{k \in \mathcal{O}} (\hat{\mathbf{A}}_i^{(k)} \mathbf{X}_i \mathbf{W}_1^{(k)})), \\ \mathbf{H}_i^{sus} &= \sigma(\mathop{\Big\|}_{k \in \mathcal{O}} ((\hat{\mathbf{A}}_i^{T})^{(k)} \mathbf{X}_i \mathbf{W}_2^{(k)})). \end{aligned} \tag{7.3}$$

$\mathbf{E}_i = \{\mathbf{e}_j^{\mathcal{G}} | u_j \in U_i\}$, where $\mathbf{e}_j^{\mathcal{G}}$ is looked up through global graph embedding matrix $\mathbf{E}^{\mathcal{G}}$ by given user id, and $\mathbf{U}_i$ is the user characteristics matrix. $\mathbf{H}_i^{inf}, \mathbf{H}_i^{sus} \in \mathbb{R}^{|U_i| \times F}$

are user influence and user susceptibility, respectively.

7.1.3.5 Users' temporal learning

We learn temporal information for users from the diffusion path $\mathcal{P}_i$. Specifically, by modeling the timestamp information we can extract the dynamic and temporal information for each participant user, which has been demonstrated to help rumor detection [162].

Assume that, the time window is $[t_1, t_o]$, we first split the time window into l disjoint time intervals, and compute the corresponding time interval for each retweet user u_j as $\text{pos} = \left\lfloor \frac{t_j - t_1}{t_o/l} \right\rfloor$, where t_0 is the timestamp for the source post user. Then, we use positional encoding introduced in the Transformer [131] to allocate initial embedding for each time interval.

$$\begin{aligned} TP(\text{pos})_{2d} &= \sin \frac{\text{pos}}{10000^{2d/d_{time}}}, \\ TP(\text{pos})_{2d+1} &= \cos \frac{\text{pos}}{10000^{2d/d_{time}}}. \end{aligned} \tag{7.4}$$

where $\text{pos} \in [0, l)$ denotes the time interval each user fall into, d refers to dimension, and d_{time} is the total dimensions of the time interval embedding. So that, for a given tweet m_i, we construct an initial time-embedding matrix denoted as $\mathbf{T}_i \in \mathbb{R}^{|U_i| \times d_{time}}$.

After that, we feed $\mathbf{T}_i$ into a Bi-directional GRU (Bi-GRU) [108] to learn the temporal information $\mathbf{H}_i^{time}$ for users.

$$\mathbf{H}_i^{time} = \text{Bi-GRU}(\mathbf{T}_i), \mathbf{H}_i^{time} \in \mathbb{R}^{|U_i| \times F} \tag{7.5}$$

7.1.3.6 Feature-level aggregation attention

After obtaining the multi-scale latent representation of users, i.e., $\mathbf{H}_i^{inf}$, $\mathbf{H}_i^{sus}$ and $\mathbf{H}_i^{time}$, we propose an attention-based method to capture the different importance among three types of representations. Let $\hat{\mathbf{u}}_j = \left[\mathbf{h}_j^{inf}, \mathbf{h}_j^{sus}, \mathbf{h}_j^{time}\right] \in \mathbb{R}^{3 \times F}$ denote the learned feature set for user u_j. The attention $\mathbf{a}_j$ for $\hat{\mathbf{u}}_j$ is calculated as:

$$\begin{aligned} \hat{\mathbf{u}}_j^{'} &= \tanh\left(\hat{\mathbf{u}}_j \cdot \mathbf{w}_j\right), \\ \mathbf{a}_j &= \text{softmax}\left(\hat{\mathbf{u}}_j^{'} \cdot \mathbf{w}_j^{'}\right), \end{aligned} \tag{7.6}$$

where $\mathbf{w}_j \in \mathbb{R}^{F \times F}$ and $\mathbf{w}_j^{'} \in \mathbb{R}^{F \times 1}$ are weight matrices, $\mathbf{a}_j \in \mathbb{R}^{3 \times 1}$ is the learned attention. Then the aggregated feature vector for user u_j is $\bar{\mathbf{u}}_j = \hat{\mathbf{u}}_j \cdot \mathbf{a}_j$, where $\bar{\mathbf{u}}_j \in \mathbb{R}^F$. Finally, we get the fused user feature vector matrix as $\bar{\mathbf{U}}_i = \{\bar{\mathbf{u}}_j | j \in [1, |U_i|]\}$.

7.1.3.7 VAE-based uncertainty learning

In most of the existing works, the learned $\bar{\mathbf{U}}_i$ can be directly fed into a classification layer to predict the label of m_i. In our work, motivated by the ability of variational autoencoders (VAE) [163] in coping with randomness and uncertainty, we employ VAE to capture the uncertainty in the learned user features. Let $f_{\text{Dec}}(\cdot)$, $f_{\text{Dec}}(\cdot)$ and $f_{\text{NN}}(\cdot)$ denote the encoder, decoder and neural network, respectively. Then the VAE-based uncertainty learning layer can be simply formalized as:

$$\begin{aligned} &\mathbf{z}_j = f_{\text{ENC}}(\bar{\mathbf{u}}_j), \bar{\mathbf{u}}_j^{'} = f_{\text{Dec}}(\mathbf{z}_j), j = 1, 2, \cdots, |U_i|. \\ &\mu_j = f_{\text{NN}}(\bar{\mathbf{u}}_j), \log \sigma_j^2 = f_{\text{NN}}(\bar{\mathbf{u}}_j), \mathbf{z}_j \sim \mathcal{N}(\mu_j, \sigma_j^2) \end{aligned} \tag{7.7}$$

where $\bar{\mathbf{u}}_j^{'}$ represents the reconstructed input features. $\mathbf{z}_j \in \mathbb{R}^{d_z}$ is the latent vector. Specifically, VAE gets μ and $\log \sigma^2$ from the encoder (we omit the subscript j for simplicity), and then samples latent representation $\mathbf{z}$ from Gaussian distribution via reparameterization trick, where $\mathbf{z} = \mu + \sigma\epsilon$ and $\epsilon \sim \mathcal{N}(0, 1)$. Then the decoder takes the latent representation $\mathbf{z}$ as input, and try to reconstruct the original input feature. In general, the marginal log-likelihood of $\bar{\mathbf{u}}$ – $\log p_\theta(\bar{\mathbf{u}}) = \log \int_{\mathbf{z}} p_\theta(\bar{\mathbf{u}}|\mathbf{z}) p(\mathbf{z}) d_{\mathbf{z}}$, which is intractable to compute effectively. Instead, we adopt variational inference by defining a simple parametric distribution over the latent variables $q_\phi(\mathbf{z}|\bar{\mathbf{u}})$ (a.k.a. f_{Enc} parameterized by ϕ), and maximizing the evidence lower bound (ELBO) on the marginal log-likelihood of each observation:

$$\begin{aligned} \log p_\theta(\bar{\mathbf{u}}) &= \log \int_{\mathbf{z}} p_\theta(\bar{\mathbf{u}}|\mathbf{z}) p(\mathbf{z}) d_{\mathbf{z}} \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\bar{\mathbf{u}})} \log[\frac{p_\theta(\bar{\mathbf{u}}, \mathbf{z})}{q_\phi(\mathbf{z}|\bar{\mathbf{u}})}] + \mathbb{KL}[q_\phi(\mathbf{z}|\bar{\mathbf{u}})||p_\theta(\mathbf{z}|\bar{\mathbf{u}})] \\ &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\bar{\mathbf{u}})}[\log p_\theta(\bar{\mathbf{u}}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\bar{\mathbf{u}})] \triangleq \text{ELBO}(\bar{\mathbf{u}}) \end{aligned} \tag{7.8}$$

To optimize the ELBO, we use a parametric inference network and reparameterization of $q_\phi(\mathbf{z}|\bar{\mathbf{u}})$ to alternatively maximize the following reformulation:

$$\begin{aligned} \text{ELBO}(\bar{\mathbf{u}}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\bar{\mathbf{u}})}[\log p_\theta(\bar{\mathbf{u}}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\bar{\mathbf{u}})] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\bar{\mathbf{u}})}[\log p_\theta(\mathbf{z}) + \log p_\theta(\bar{\mathbf{u}}|\mathbf{z}) - \log q_\phi(\mathbf{z}|\bar{\mathbf{u}})] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\bar{\mathbf{u}})} p_\theta(\bar{\mathbf{u}}|\mathbf{z}) - \mathbb{KL}[q_\phi(\mathbf{z}|\bar{\mathbf{u}})||p_\theta(\mathbf{z})] \end{aligned} \tag{7.9}$$

where $p_\theta(\bar{\mathbf{u}}|\mathbf{z})$ denotes the decoder and the first term of Equation (7.9) is the reconstruction loss, which is used to measure the likelihood value of the reconstructed features. The second term is the Kullback-Leibler (KL) divergence between the variational distribution $q_\phi(\mathbf{z}|\bar{\mathbf{u}})$ and the prior $p_\theta(\mathbf{z})$ (which is always ≥ 0). Therefore, the objective of maximizing ELBO of $\log p_\theta(\bar{\mathbf{u}})$ turns to minimize the Kullback-Leibler (KL) divergence. Through this VAE-based uncertainty learning layer, the learned latent representation for all users form a user latent representation matrix $\mathbf{U}_i^z = \{\mathbf{u}_j^z | j \in \mathbb{R}^{|U_i| \times d_z}\}$.

7.1.3.8 User-level aggregation attention

We concatenate $\bar{\mathbf{U}}_i$ and $\mathbf{U}_i^z$ at user-level to form a new user representation matrix $\mathbf{U}_i^F \in \mathbb{R}^{|U_i| \times (F+d_z)}$. Then we try to merge the user-level information to form an unique representation $\mathbf{R}_i$ for tweet m_i through attention sum-pooling operation:

$$\begin{aligned} \hat{a}_j &= \frac{\exp(\langle \mathbf{w}, \tanh(\mathbf{W}_a \mathbf{u}_j^F + \mathbf{b}_a) \rangle)}{\sum_{*:=1}^{|U_i|} \exp(\langle \mathbf{w}, \tanh(\mathbf{W}_a \mathbf{u}_*^F + \mathbf{b}_a) \rangle)}, \\ \mathbf{R}_i &= \sum_{j=1}^{N} \hat{a}_j \mathbf{u}_j^F \end{aligned} \tag{7.10}$$

where $\mathbf{W}_a \in \mathbb{R}^{(F+d_z) \times d}$, $\mathbf{b}_a \in \mathbb{R}^d$ and $\mathbf{w} \in \mathbb{R}^d$.

7.1.3.9 Rumor detection

Our ultimate goal is to predict the rumor label $\hat{y}_i$ of tweet m_i. We calculate this through feeding $\mathbf{R}_i$ into several fully connected layers and a softmax output layer, which is denoted as:

$$\hat{\mathbf{y}}_i = \text{softmax}\left(\text{FC}\left(\mathbf{R}_i\right)\right) \tag{7.11}$$

where $\hat{\mathbf{y}}_i$ is a vector of predicted probabilities of all rumor categories for the tweet m_i.

In the implementation, we train PLRD to estimate all the model parameters by minimizing the *cross-entropy* of the predictions $\hat{\mathbf{Y}}$ and the ground truth labels $\mathbf{Y}$. The prediction loss is:

$$\mathcal{L}_{pre} = -\frac{1}{|B|} \sum_{i=1}^{|B|} \sum_{c=0}^{1} y_{i,c} \log \hat{y}_{i,c} \tag{7.12}$$

where $|B|$ is the batch size, $y_{i,c}$ and $\hat{y}_{i,c}$ are the ground truth and predicted results for the i-th sample. That is, if the sample belongs to c-th class, $\hat{y}_{i,c}$ is 1; otherwise it is 0.

The total loss of PLRD should take the ELBO into consideration, that is:

$$\mathcal{L} = \mathcal{L}_{pre} - \frac{1}{|B|} \sum_{i=1}^{|B|} \text{ELBO}(\bar{\mathbf{U}}_i) \tag{7.13}$$

During training, the well-known stochastic gradient descent is applied to update parameters. Specifically, we use the adaptive learning rate optimization algorithm Adam [139] for model training. All hyper-parameters are tuned using the standard grid search for optimal results. The next section provides the details of the computational experiments.

7.1.3.10 Computational complexity analysis

In this section, we give a brief analysis of the computational complexity of PLRD. (1) The complexity for social homophily learning from global graph: as analyzed in [155], the overall complexity of this layer is $\mathcal{O}(d^2|U| + |E|)$, where d, $|U|$ and $|E|$ are the dimensions of user social homophily, number of nodes and edges in global graph, respectively. (2) The complexity for users' influence and susceptibility learning: we use a multi-hop graph convolutional layer to learn the users' influence and susceptibility (cf. Eq 7.2). Recall that, the dimensions of the input and the output are d_X and F, respectively, the max-order is K, and the normalized adjacency matrix $\hat{\mathbf{A}}$ is a sparse matrix with m nonzero elements. Therefore, for a single MHGCN layer, the computational complexity is $\mathcal{O}(F \times K \times m \times d_X)$. (3) The other parts of the PLRD are implemented by GRUs and MLPs. The time and space complexity are related to the input dimensions of latent variables. Since the users' social homophily are computed in preprocessing phase, the computational complexity of whole PLRD is therefore $\mathcal{O}(F \times K \times m \times d_X)$.

7.1.4 Evaluation

In this section we evaluate our proposed PLRD framework and demonstrate its practical utility through quantitative experiments.

7.1.4.1 Evaluation metrics and baselines

In our work, we use Accuracy (ACC), Precision (Pre), Recall (Rec), and F1 as the evaluation protocols to measure the models' performance. In particular, ACC measures the proportion of correctly classified tweets, while F1 is the harmonic mean of the precision and recall.

We compare our method with a battery of baselines, they are:

- **DTC** [23]: A decision tree-based classification model that combines manually engineered characteristics of tweets to compute the information credibility.
- **SVM-TS** [85]: A linear SVM-based time series model, which can capture the variation of a broad spectrum of social context information over time by converting the continuous-time stream into fixed time intervals.
- **GRU** [33]: An RNN-based model has been employed to learn the sequential cascading effect of tweets with high-level feature representations extracted from relevant posts over time.
- **TD-RvNN** [34]: A tree-structured RNN model for rumor detection, which embeds hidden indicative signals in the tree-structures and explores the importance of comments for rumor detection.

- **PPC_RNN+CNN** [100]: An early-stage rumor detection model through classifying news propagation paths with RNN and CNN, which learns the rumor representations through the characteristics of users and source tweets (for brevity, the model name is abbreviated to PPC).
- **dEFEND** [87]: A co-attention-based fake news detection model that exploits both news contents and user comments for fake news detection.
- **Bi-GCN** [16]: A GCN-based model exploiting the bi-directional propagation structures and comments for rumor detection.
- **GCAN** [102]: A co-attention network that detects true and false rumors based on the content of the source tweet and its propagation-based users. We also provide a variant of GCAN, denoted as GCAN-Text, which removes the source tweet features in the original inputs.

7.1.4.2 Experimental setup

We implement DTC with Weka[1], SVM-based models with scikit-learn[2], and other neural network-based models with Tensorflow[3]. All baselines follow the parameter settings in the original papers.

For PLRD, the learning rate is initialized at 0.001 and gradually decreases as the training proceeds. We set the embedding size d for the social homophily to 40. As for time-embedding, we set the total number of time intervals to be 100 and each interval represents 10 minutes. Retweets with a latency of more than 1,000 minutes would fall into the last time interval. The size of time-embedding is d_{time} set to 50. The hidden size F of user influence, susceptibility, and temporal information are set to 64; the hidden size d_z of the VAE-based uncertainty learning layer is set to 32. The batch size is set to 32 for Twitter15/16, 128 for Science and 16 for RumourEval19, and the training process is iterated upon for 500 epochs but would be stopped earlier if the validation loss does not decrease after 10 epochs. And we randomly choose 70% data for training and the rest of 10% and 20% for validation and testing. The experiments are conducted on a machine with an Intel i7-6700 3.40GHZ CPU and a single NVIDIA GeForce GTX 2080Ti. The time cost for model training is less than 10 minutes on all datasets used in this work.

7.1.4.3 Study on Twitter15/16

Table 7.5 and 7.6 summarizes the overall performance on Twitter15 and Twitter16 datasets. The last two rows show the performance of the complete version of our model PLRD and the improvement percentage compare with the second-best method, which basically yields much better performance than the other baseline

[1]https://www.cs.waikato.ac.nz/ml/weka/
[2]https://scikit-learn.org/
[3]https://www.tensorflow.org/

Table 7.5: Overall performance comparison of rumor detection on Twitter15. The best method is shown in **bold**, and the second best one is underlined. The number of retweets is 40.

	Twitter15			
Method	Acc	Pre	Rec	F1
DTC	0.495	0.494	0.481	0.495
SVM-TS	0.519	0.519	0.518	0.519
GRU	0.580	0.544	0.545	0.544
TD-RvNN	0.678	0.671	0.674	0.672
PPC	0.691	0.674	0.686	0.679
dEFEND	0.738	0.658	0.661	0.654
Bi-GCN	0.748	0.731	0.759	0.745
GCAN	0.875	0.825	0.829	0.825
GCAN-Text	0.683	0.705	0.652	0.678
PLRD	**0.934**	**0.928**	**0.929**	**0.927**
Improvement	**8.98%**	**12.5%**	**12.1%**	**12.4%**

Table 7.6: Overall performance comparison of rumor detection on Twitter16. The best method is shown in **bold**, and the second best one is underlined. The number of retweets is 40.

	Twitter16			
Method	Acc	Pre	Rec	F1
DTC	0.561	0.575	0.537	0.562
SVM-TS	0.693	0.692	0.691	0.692
GRU	0.554	0.514	0.516	0.515
TD-RvNN	0.661	0.632	0.641	0.636
PPC	0.655	0.632	0.651	0.641
dEFEND	0.702	0.637	0.638	0.631
Bi-GCN	0.711	0.709	0.710	0.716
GCAN	0.823	0.803	0.841	0.822
GCAN-Text	0.664	0.716	0.579	0.648
PLRD	**0.875**	**0.876**	**0.874**	**0.855**
Improvement	**6.32%**	**9.09%**	**3.92%**	**4.01%**

methods across all metrics. Note that we conduct a McNemar's test [144] between our PLRD and the best baseline based on the prediction results on the testing set. The p-values are $p < 0.001$ on both Twitter15 and Twitter16. Therefore, we can conclude that the performance between PLRD and GCAN exists with statistical significance.

We make the following additional observations. *O1:* The feature-based approaches – DTC and SVM-TS use hand-crafted features based on the overall statistics of tweets, which perform poorly. These two methods not sufficient to capture the generalizable features associated with tweets and the diffusion process. Notably, SVM-TS achieves

comparatively better performance than DTC because it utilizes an extensive set of features and focuses on retweets' temporal features. *O2:* Deep learning-based models achieve better performance than feature-based methods. GRU is the first deep-learning-based method for rumor detection, which performs worst among deep-learning-based baselines because it only relies on the temporal-linguistics features of the post sequence but ignores other useful information such as diffusion structures and user profiles. Both TD-RvNN and PPC outperform GRU, which indicates the effectiveness of modeling the propagation structure and temporal information in rumor detection. Moreover, PPC proves user profile features as important as text features for rumor detection. dEFEND utilize a co-attention mechanism to learn the correlation between news contents and user comments, which performs better than TD-RvNN and PPC but worse than Bi-GCN and GCAN. Bi-GCN and GCAN claim that they can learn structure information from graphs, and their performance indeed exceeds other baseline methods. However, Bi-GCN constructs the structural tree based on the replies, which can not reflect the full process of rumor diffusion. As for GCAN, it captures the similarity between users rather than propagation structural features. According to the results, GCAN performs much better than Bi-GCAN, because it takes both text information and user profiles into consideration. By comparing GCAN with its variants GCAN-Text, we can find that after removing text information, the performance of GCAN remarkably decrees, demonstrating that GCAN is not efficient to capture user-related features. *O3:* PLRD consistently outperforms all baselines on both Twitter15 and Twitter16. Compare to the best baseline method Bi-GCN, PLRD learns rumor representation from a participant-level without any text information, demonstrate the primary motivations of this work – i.e., users are the main contributor to the rumor propagation.

7.1.4.4 Study on Science

Table 7.7: Overall performance comparison of rumor detection on Science. The best method is shown in **bold**. The number of retweets is 40.

	Science			
Method	Acc	Pre	Rec	F1
PPC	0.655	0.649	0.568	0.606
GCAN-Text	0.671	0.646	0.622	0.634
PLRD	**0.768**	**0.727**	**0.814**	**0.768**

In this section, we conduct an experiment on Science dataset. Specifically, from the original Science dataset, we first filter out (1) the tweets with less than 10 retweets and more than 100; and (2) the tweets' with a diffusion period exceed 24 hours. After that, we have $3,493$ tweets in total, and the processed dataset is still highly imbalanced, e.g., only 610 tweets were labeled as "non-rumor", while the majority (i.e., $2,883$) were classified as "rumor". We randomly select $1,000$ items from the rumor set to make sure the number of tweets labeled as non-rumors is 50% of the

rumors, and finally, we get a new experiment dataset with 1,000 rumors and 610 non-rumors. Table 7.3 summarizes the performance comparison between PLRD and two state-of-the-art propagation-based baselines, i.e., PPC and GCAN-Text[1]. We can observe that our PLRD outperforms both PPC and GCAN-Text on all metrics, which indicates that our model is more effective and stable in extracting diffusion patterns of users even without any textual information.

7.1.4.5 Study on RumourEval19

Table 7.8: Overall performance comparison of rumor detection on RumourEval19. The best method is shown in **bold**. The result of the "Base" model has referenced the best method from the paper [150]. "w/o Text" means without textual features. The number of comments is 100.

	RumourEval19			
Method	Acc	Pre	Rec	Macro-F1
Base	–	0.596	0.603	0.577
TD-RvNN	0.667	0.641	0.673	0.615
Bi-GCN	0.734	0.733	0.735	0.661
PLRD	**0.813**	**0.826**	**0.885**	**0.788**
PLRD w/o Text	0.750	0.806	0.842	0.692

Since the RumorEval19 dataset has rich textual features, in this section we conduct an experiment on RumourEval19 to test the PLRD performance when including textual features. Specifically, we first use a pre-trained model – BERTweet [164] – to generate the tweet embedding for each source tweet and its corresponding comments, and then concatenate the source tweet embedding and comments embedding to form a textual embedding matrix $\mathbf{C} \in \mathbb{R}^{|U_i| \times d_{text}}$ for each tweet. Finally, we combine $\mathbf{C}$ with $\mathbf{X}$ to form the input of PLRD. For a comparison, we choose TD-RvNN and Bi-GCN as baselines, and also provide the result of the best method in [150] denotes as "Base". From Table 7.8, we can find that under the situation of unbalanced label distribution, our PLRD can still achieve competitive performance compared with other baselines on rumor detection. After deleting the textual features, the performance of PLRD slightly drops, which demonstrate that textual features are powerful and can help improve the model performance.

7.1.4.6 Ablation study

In this section, we conduct an ablation study on Twitter15 and Twitter16 to explore the effect of each component in PLRD. Towards that, we derive the following variants of PLRD:

[1]The Science dataset only provides anonymous user profile characteristics and propagation threads. No textual features are available. So we compare with PPC and GCAN-Text.

- **w/o user profiles (UP)**: In "**w/o UP**", we do not consider the user profile characteristics, which means we do not use $\mathbf{U}_i$ and only keep $\mathbf{E}_i$ as input features.
- **w/o social homophily (SH)**: In "**w/o SH**", we ignore the social homophily of users, which means we do not use $\mathbf{E}_i$ and only keep $\mathbf{U}_i$ as input features.
- **w/o user influence (UI)**: In "**w/o UI**", we do not capture user influence, which means ignore $\mathbf{H}_i^{inf}$.
- **w/o user susceptibility (US)**: In "**w/o US**", we do not capture user susceptibility, which means ignore $\mathbf{H}_i^{sus}$..
- **w/o user temporal (UT)**: In "**w/o UT**", we do not take the user temporal information into consideration, which means ignore $\mathbf{H}_i^{time}$.
- **w/o graph information (GI)**: In "**w/o GI**", we do not utilize any information from the global graph and the diffusion graph, which means we ignore users' social homophily, influence, susceptibility and only keep the users' temporal learning component. The input of this part is the concatenation of user features and temporal information.
- **w/o feature uncertainty (FU)**: In "**w/o FU**", we remove the VAE-based uncertainty learning layer and use $\bar{\mathbf{U}}_i$ directly.
- **w/o feature-level attention (FA)**: In "**w/o FA**", we remove the feature-level aggregation attention in PLRD and concatenate the different user features directly, i.e., $\bar{\mathbf{U}}_i = \text{concat}(\mathbf{H}_i^{inf}, \mathbf{H}_i^{sus}, \mathbf{H}_i^{time}) \in \mathbb{R}^{|U_i| \times 3F}$.
- **w/o user-level attention (UA)**: In "**w/o UA**", we do not allocate different importance for each user and directly use a sum-pooling to form the rumor representation.

Table 7.9: performance comparison between PLRD and its variants.

	Twitter15				Twitter16			
Method	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
w/o UP	0.853	0.842	0.838	0.828	0.838	0.858	0.847	0.840
w/o SH	0.906	0.894	0.910	0.896	0.802	0.821	0.786	0.794
w/o UI	0.868	0.863	0.871	0.859	0.800	0.859	0.871	0.867
w/o US	0.841	0.877	0.857	0.864	0.781	0.794	0.775	0.782
w/o UT	0.873	0.914	0.935	0.920	0.795	0.792	0.802	0.788
w/o GI	0.806	0.755	0.811	0.782	0.758	0.716	0.732	0.724
w/o FU	0.913	0.911	0.914	0.911	0.854	0.847	0.859	0.848
w/o FA	0.811	0.842	0.843	0.848	0.790	0.831	0.837	0.816
w/o UA	0.896	0.906	0.877	0.886	0.854	0.838	0.831	0.829
PLRD	**0.934**	**0.928**	**0.929**	**0.927**	**0.875**	**0.876**	**0.874**	**0.855**

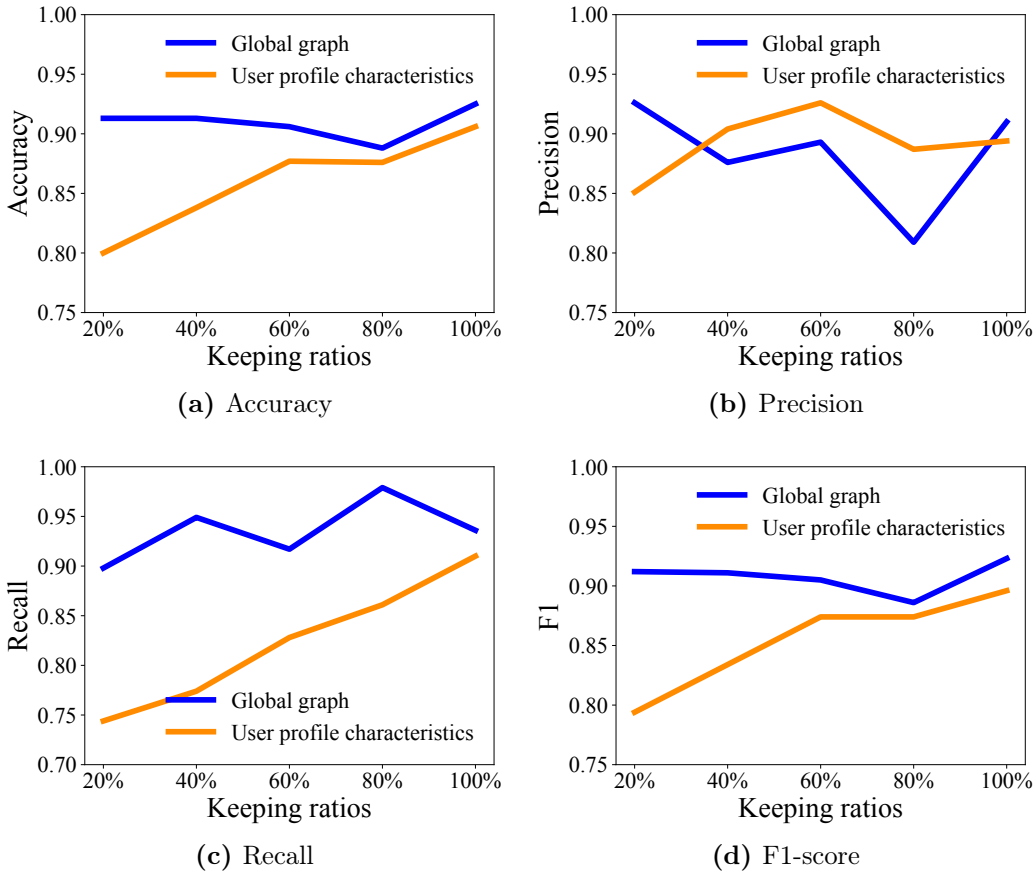
The results, shown in Table 7.9, indicate that the original PLRD outperforms these variants in terms of all metrics. From Table 7.9, we can observe that: (1) Both user profile features (w/o UP) and social homophily ('w/o SH) are reliable inputs of our model that because user profiles can be used to identify an individual, and social homophily can reflect the user preference. (2) User influence (w/o UI), susceptibility (w/o US), and temporal features (w/o UT) are indispensable for rumor detection. (3) The result of "w/o GI" performs worst among all variants, demonstrating that graph data (global graph and diffusion graph) provide considerable meaningful features, and are thus indispensable in rumor detection. (4) The fact that "w/o FU" provides lower performance compare with PLRD, reflects the benefit of modeling the feature uncertainty. (5) The two attention-based aggregation layers, i.e., feature-level aggregation attention (w/o FA) and user-level aggregation attention (w/o UA), play crucial roles in detecting rumors. Especially the feature-level aggregation attention (w/o FA), after removing it, the performance remarkably decreases, suggesting that distinguishing the importance of different scale of user features can improve detection performance. Similarly, "w/o UA" demonstrates that different users play different roles in rumor spreading.

7.1.4.7 Privacy-preserving study

In this section, we conduct experiments on Twitter15 dataset to test our PLRD's performance on privacy-preserving scenarios which can be summarized as: (1) randomly removing different proportions of edges in the global graph $\mathcal{G}$; (2) randomly masking different proportions of user characteristics $\mathbf{u}$, and (3) randomly removing different proportions of edges in the diffusion graph G. Note that, in scenario (1) and (2) we only keep related features as inputs, i.e., in scenario (1), we only use the user homophily $\mathbf{E}$ generated from the global graph as input, and in scenario (2) we only keep user characteristic $\mathbf{U}$ as input. Both scenarios are tested on 40 retweets.

Figure 7.4 plots the performance of PLRD in scenarios (1) and (2), which shows that even though we only keep 20% of edges in the global graph, the PLRD still achieves 90% accuracy. However, when masking 80% of user characteristics, the performance of PLDR drops significantly.

Figure 7.5 shows the performance of PLRD with the different numbers of retweets in scenario (3). We find that when we only observe few retweets, e.g., 10 and 40, the performance of the model decreases as the removal of edges in the diffusion graph. In contrast, when the number of observation retweets is sufficient enough such as 100, dropping a few edges would help improve the model performance. The reasons behind is that: (1) with the increase of observed retweets, there is a great possibility to introduce noise into the graph, which can be eliminated with random removal of graph edges; and (2) randomly drop a few of graph edges is widely used as a data augmentation method in graph representation learning field [165, 166], which can



(a) Accuracy **(b)** Precision

(c) Recall **(d)** F1-score

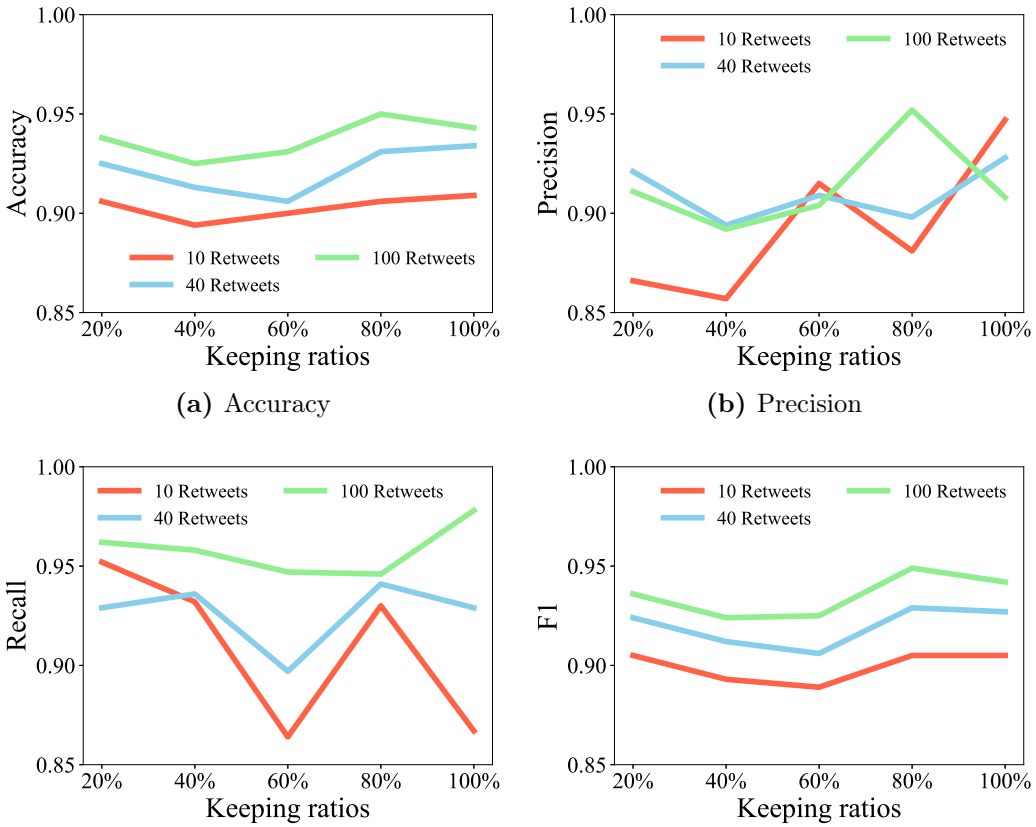
Figure 7.4: Evaluations on randomly removing different proportions of edges in the global graph and random masking different proportions of user characteristics.

improve model generalization and overcome the overfitting and over-smooth issues of graph neural networks.)

7.1.4.8 Early detection

Another critical goal of rumor detection is to detect rumors as early as possible that is essential to stop their spread in a timely fashion. Next we investigate the performance of models on identifying rumors at an early stage. Here, we consider the early 50 retweets.

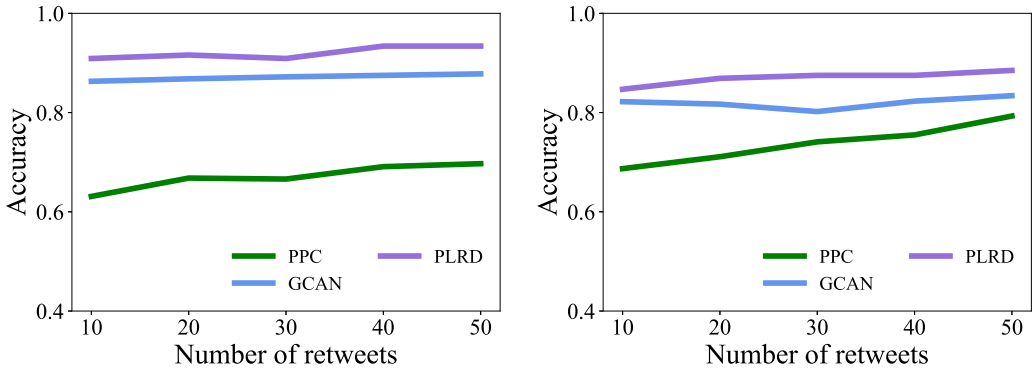
Figure 7.6 shows the performance comparison on early-stage detection between our PLRD and the selected baselines. Note that we omit the feature-based approaches (i.e., DTC and SVM-TS) and GRU since they did not show comparable performance, especially on early rumor detection. Moreover, we also ignore TvRvNN, dEFEND, and Bi-GCN, because these methods are built on the replies that may not exist in the early-stage. We observe that PLRD performs better than PPC and GCAN, especially when there are only a few observations. PLRD needs a short time to identify the misinformation because PLRD learns the rumor representation from a



(a) Accuracy **(b)** Precision

(c) Recall **(d)** F1-score

Figure 7.5: Evaluations on randomly removal of diffusion links based on Twitter15.

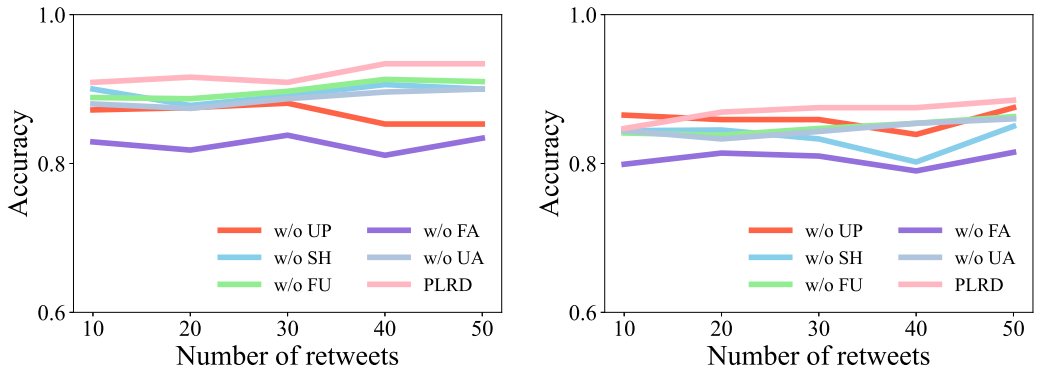


(a) Early 50 retweets (Twitter15). **(b)** Early 50 retweets (Twitter16).

Figure 7.6: Evaluations on early rumor detection.

participant-level and fuses users' multi-scale knowledge, such as user influence, user susceptibility, user temporal information, etc.

We also investigate the time-varying performance between PLRD and its variants. Specifically, we choose "w/o UP", "w/o SH", "w/o FU", "w/o FA" and "'w/o UA' as the comparison methods. The results show in Figure 7.7. We find that the performance



(a) Early 50 retweets (Twitter15). **(b)** Early 50 retweets (Twitter16).

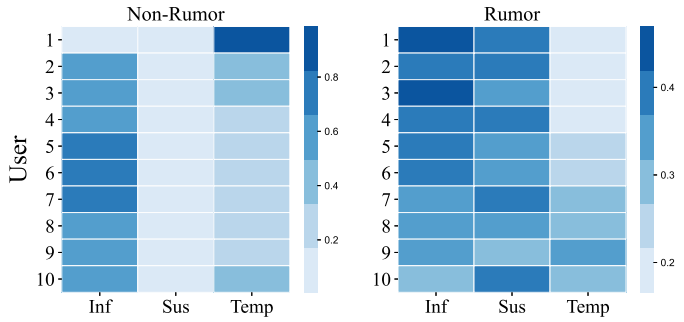
Figure 7.7: Evaluations on early rumor detection among variants of PLRD.

of PLRD surpasses all variants, and with the number of retweets increased, the accuracy of all methods grow to saturation.

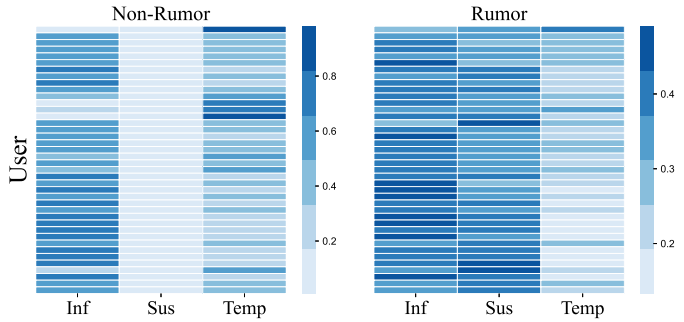
7.1.4.9 Interpretability analysis

The above ablation studies have shown the superiority of each component in PLRD. In this section, we provide more in-depth insights by visualizing features.

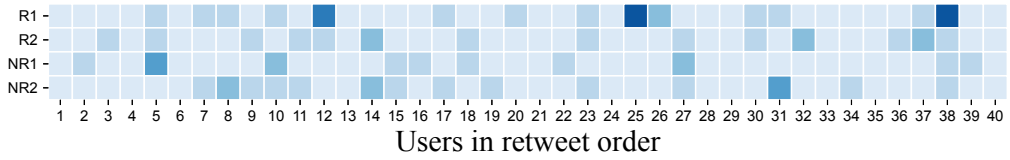
Figure 7.8 plots the importance of the feature-level aggregation layer and user-level aggregation layer. As for the feature-level aggregation attention, we randomly selected two different types of tweets in Twitter15 and plotted the importance of the different features. Figure 7.8a and Figure 7.8b show the results of previous 10 and 40 retweet users, respectively. Overall, we find that (1) the three types of features for each user have different importance; (2) attention distribution varies between rumor and non-rumor. Specifically, as for rumor tweets, participants try to affect others, while themselves are easier to expose in the misleading tweets. In contrast, in non-rumor tweets, participants are more influential compare with the susceptibility. Moreover, temporal information plays a crucial role in detecting both rumor and non-rumor. In Figure 7.8c, we investigate the role of the retweet users at the very beginning of the diffusion. As shown, the later users are more critical in rumor spreading, which confirms the hypothesis that rumors can spread deeper than non-rumors [43]. Moreover, to have an intuitive explanation regarding the superiority of each component in PLRD, we plot the learned latent representations (i.e.,$\bar{\mathbf{U}}$, $\mathbf{U}^z$, $\mathbf{U}^F$ and $\mathbf{R}$) using t-SNE [125]. Each point in the plot represents a tweet in the test set (tweets with similar latent vectors are closer in the plot), and different colors refer to different labels, i.e., green represents non-rumor, orange represents rumor. From Figure 7.9a, we see clear clustering phenomena by $\bar{\mathbf{U}}$. These latent vectors can already be used to predict directly. In contrast, Figure 7.9b "smoothes" this clustering effect by modeling the feature uncertainty, which should help explore more



(a) Feature-level aggregation attention (10 retweet user).



(b) Feature-level aggregation attention (40 retweet user).



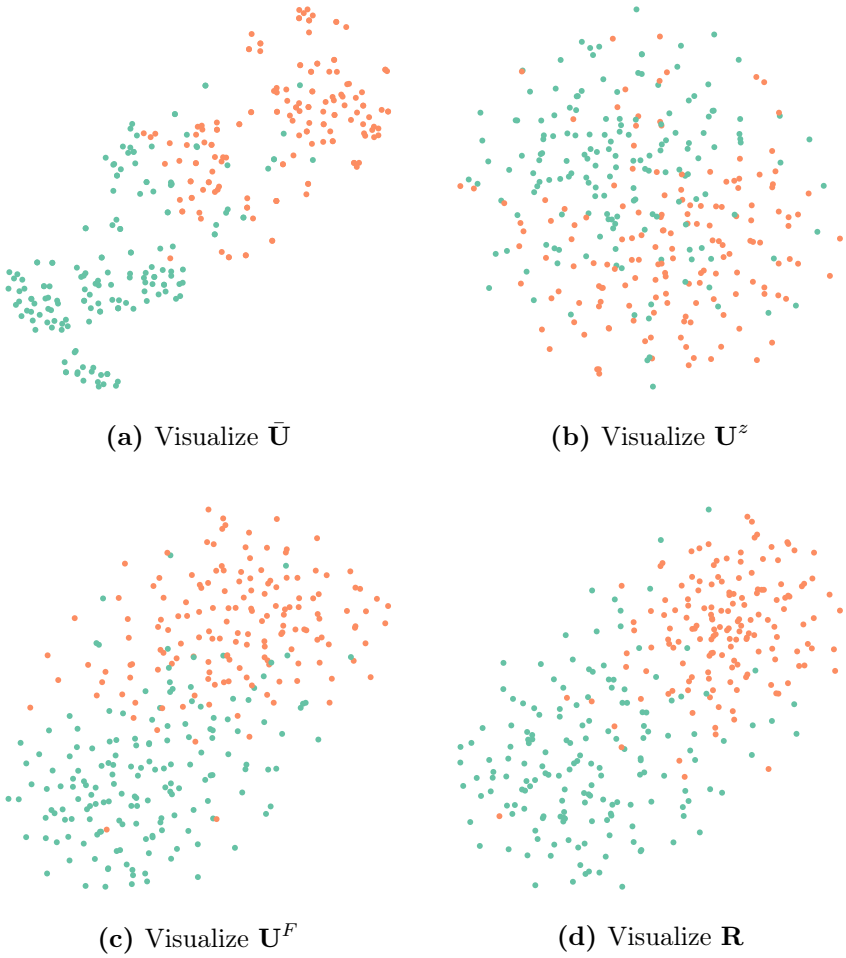
(c) User-level aggregation attention.

Figure 7.8: Attention visualization of PLRD. "Inf": influence; "Sus": susceptibility; "Temp": temporal.

possibilities. From Figure 7.9c – Figure 7.9d, we find that the model learned more suitable latent representations for prediction after a user-level attention layer.

7.1.5 Summary

In this first part of the chapter, we first provided empirical evidence that all participants in the diffusion chains of rumors exhibit different patterns than participants in the diffusion chains of non-rumors. Based on these findings, we proposed a novel fine-grained all- participant level rumor detection model, named PLRD (Participant-Level Rumor Detection). Specifically, PLRD learns fine-grained user representations, i.e., user influence, user susceptibility, and user temporal information from the propagation threads of a given post, and merges the learned features to form a unique rumor representation through a feature-level attention layer and a user-level attention layer. Moreover, a variational autoencoder used to capture uncertainty



(a) Visualize $\bar{\mathbf{U}}$ **(b)** Visualize $\mathbf{U}^z$

(c) Visualize $\mathbf{U}^F$ **(d)** Visualize $\mathbf{R}$

Figure 7.9: Visualization of the learned latent representation on Twitter15 using t-SNE. Each point is a sample from the test set. The color green represents non-rumor, and the orange one represents rumor.

from features further improves the learned rumor representation. Compared with existing rumor detection methods, PLRD makes predictions only based on user-level features learned from the diffusion process of posts, which overcomes the problem of overemphasizing the text features. We conducted experiments on four real-world datasets, Twitter15, Twitter16, Science and RumourEval19. The experiment results not only demonstrate that our model significantly outperforms the baselines regarding effective early detection, but also supports the hypothesis that the combination of various user information at a participant level in a diffusion chain will improve the performance of rumor discovery. Besides, our ablation study further demonstrates that each part in our model is indispensable for rumor detection.

7.2 UMLARD: Multi-view Learning with Distinguishable Feature Fusion for Rumor Detection

7.2.1 Section Overview

In section 7.1, we proposed a novel participant-level rumor detection model – PLRD, which can learn fine-grained all-participant patterns throughout the whole diffusion process, including social patterns (user social homophily), diffusion patterns (user influence and susceptibility), temporal patterns (how fast the news is propagated in the social network), all-participant profile patterns. Based on the experimental results compared with existing state-of-the-art deep learning-based rumor detection methods, we have seen that PLRD achieves significant improvements. In this section, we propose a new model UMLARD – User-aspect Multi-view Learning with Attention for Rumor Detection model, which can make even better predictions than PLRD and other recent works. UMLARD inherits the advantages of PLRD, i.e., solving the same limitations that are also considered by PLRD:

(L1) *Lack of systematic user-aspect rumor modeling*: Recent research [43] reveals that humans are the principal "culprits" in spreading false news. Existing studies either directly aggregate users' profile information as model inputs [85, 86, 102], which only pay attention to the local structure correlations among propagated users and the sequential propagation patterns [100] (i.e., user temporal features), or focus on learning the global structure of rumor diffusion [16] (i.e., user structural features). Essentially, these works learn the rumor representation from an event-level, and still lacks a unified framework that can learn rumor diffusion while extracting meaningful features from user-aspect.

(L2) *Indistinguishable importance of both features and users*: Different features play different roles in rumor detection at different phases of propagation. As information spreads, for example, the effect of structural information and temporal information on discriminating rumors becomes different [84, 85]. Also, users may either unconsciously forward some unproven news, or deliberately propagate the fake news in the information spread [26]. Understanding the efficacy of features and individual users at the same time would help detect rumors, which, however, has not been well investigated in existing studies.

(L3) *Limited interpretability*: Most existing studies focus on explaining the news content, e.g., discover the important sentence in the articles or emotional words in comments [87], to interpret the detection results. However, these works cannot explain critical features beyond text and determine user's roles in rumor propagation.

Besides all those points above, UMLARD also improves on PLRD by overcoming the following limitation is not solved by PLRD:

(L4) *Entangled high-level feature learning*: Existing works learn high-level representations (e.g., structural or temporal) for rumor detection by exploiting user profiles or pretrained textual features as model inputs. While improving detection performance, it is difficult to demonstrate the effectiveness and high-level representations of the model, because (1) the learned representations are entangled with the original input [102], and (2) the models use the same input [44] to learn different high-level features.

To develop UMLARD and address the limitations L1–L4 above, inspired by recent progress in multi-view learning [167, 168, 169], we initiate the attempts to capture the principal characteristics of users and rumors by learning multiple distinct features. Multi-view learning is a promising learning paradigm that jointly models different views of the same input data for improving learning performance [170]. For example, a web page can be described in forms of text, video, and image [171] simultaneously. By exploring the complementarity and consistency of different views, it can further improve the model performance [172]. Specifically, we exploit different views to represent an instance for comprehensively describing the information of the instance. We first abstract the user-aspect features of the users engaged in the diffusion process as user profile-view, user structural-view, and user temporal-view, and then incorporate different views to predict the credibility of the given information. Specifically, UMLARD exploits different embedding methods to learn the view-specific high-level representations of a given post from the hierarchical diffusion process and user profiles. To understand the importance of each view and the role of the user, UMLARD employs a view-wise attention network and a capsule attention network to incorporate both view-level and user-level features. It allows us to better discriminate feature influence and the effect of user behaviors in spreading rumors.

Our main contributions towards rumor detection problem provide:

- **User-aspect feature extraction (L1)**: We conceptualize user-aspect features as different views, including profile-view, structural-view, and temporal-view, and present a novel model to learn different views for each user who engaged in the information diffusion.
- **View-specific embedding methods (L2)**: UMLARD utilizes different embedding methods to learn view-specific high-level representations based on different inputs: (1) an attention-based layer aims to learn user profile-view by assigning different importance to features in user profiles; (2) an improved GCN-based network to learn structural-view from the diffusion network while considering the direction of information dissemination, taking the adjacency matrices of diffusion networks as input; and (3) a time-decay LSTM considers the influence of users and is used for temporal-view learning based on the diffusion path taking two types of embeddings as inputs, i.e., static-embedding and dynamic-embedding.

- **Distinguishable hierarchical feature fusion (L3)**: We design a hierarchical feature fusion mechanism to unify the knowledge from different perspectives, which consists of two components: (1) a view-wise attention layer to capture the features from different views; and (2) a capsule attention layer to differentiate the most related users.
- **Explainable prediction results (L4)**: UMLARD explains the significance of features according to the learned attention values. Specifically: (1) the dimensional-wise attention network shows the importance of different characteristics in the user profiles; (2) the view-wise attention results tell how the users play different roles in different phases of rumor propagation; and (3) from the capsule attention results, one can easily understand which users play critical roles in detecting the rumors.

This section is based on the following publication [46]:

- **Chen, X.**, Zhou, F., Trajcevski, G., Bonsangue, M.: Multi-view Learning with Distinguishable Feature Fusion for Rumor Detection. Knowledge-Based Systems 240 (2022) 108085

7.2.2 Problem Statement

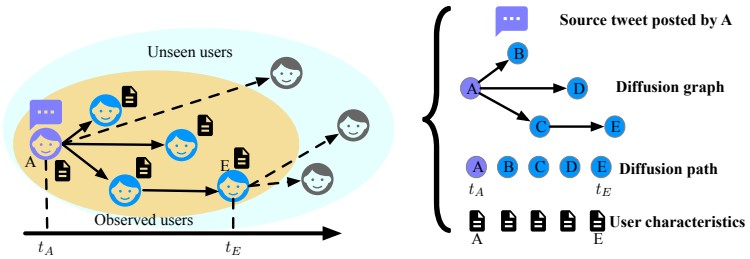


Figure 7.10: An example of the extracted information from a tweet diffusion.

Suppose we have a set of tweets $M = \{m_i, i \in [1, |M|]\}$, where each tweet m_i is a quadruplet representing the corresponding diffusion process and the users enrolled: $m_i = \{G_i, \mathcal{P}_i, \mathbf{U}_i, \mathbf{C}_i\}$, where $G_i, \mathcal{P}_i, \mathbf{U}_i, \mathbf{C}_i$ are diffusion graph, diffusion path, user characteristic matrix and the content vector of source tweet, respectively. The concepts of diffusion graph, diffusion path and user characteristic matrix are formally defined in Definition. 2, 3 and 5. And the definition of tweet content is shown as follow:

Definition 15 ***Tweet Content.*** *For a tweet m_i, the text content $\boldsymbol{C}_i$ is considered to be a sequence of words – i.e., $\boldsymbol{C}_i = [\boldsymbol{w}_{i1}, \boldsymbol{w}_{i2}, \ldots, \boldsymbol{w}_{iL}] \in \mathbb{R}^{L \times d_{word}}$, where L is the number of words in source tweet.*

We note that each word is represented by a d_{word}-dimension vector using a particular word embedding technique, e.g., word2vec.

We summarize the (definitions of the) symbols used in this section in Table 7.10. We note that, in the sequel, whenever there is no ambiguity, we may omit the double-subscript from the notation (i.e., whenever we are unambiguously working with one specific tweet m_i, we may drop i from the sequences denoting users, time-stamps, etc.).

Table 7.10: Main notations used throughout this chapter.

Symbol	Description
$\mathbf{p}_*$	the user profile vector of each user.
$\mathbf{g}_*$	The pre-trained node embedding of each user.
$\mathbf{e}_*^s$	The static embedding of each user.
$\mathbf{e}_*^d$	The dynamic embedding of each user.
d_{user}	The hidden size of the profile-view.
d_{stru}	The hidden size of the structural-view.
d_{temp}	The hidden size of the temporal-view.
d_{word}	The hidden size of the word embedding.
d_{view}	The hidden size of the multi-view layer.
$\mathbf{H}_i^{User}$	The representations of the profile-view.
$\mathbf{H}_i^{Stru}$	The representations of the structural-view.
$\mathbf{H}_i^{Temp}$	The representations of the temporal-view.
$\mathbf{H}_i^{Text}$	The representations of the content feature.
$\mathbf{V}_i^{'}, \mathbf{s}_{in}$	the representation after view-wise attention and capsule attention for tweet m_i.
$\mathbf{H}_i^{Rumor}$	the final representation of tweet m_i.
$\hat{\mathbf{Y}}/\hat{\mathbf{y}}_*$	The predicted label.
$\mathbf{Y}/\mathbf{y}_*$	The ground truth.

We now formally define the rumor detection problem that we study as follows:

Definition 16 ***Rumor Detection.*** *Given a tweet $m_i = \{G_i, \mathcal{P}_i, \boldsymbol{U}_i, \boldsymbol{C}_i\}$ within an observation window t_o, our rumor detection goal is to learn a function f from labeled claims, i.e., $f(\hat{\boldsymbol{y}}_i | G_i, \mathcal{P}_i, \boldsymbol{U}_i, \boldsymbol{C}_i; t_o)$, where the predicted result $\hat{\boldsymbol{y}}_i$ takes one of the four finer-grained classes: non-rumor, false rumor, true rumor, and unverified rumor (as introduced in [84]).*

7.2.3 Methodology

In this section, we first introduce the preliminaries and basic notations, and then formalize the problem studied in this paper. Subsequently, we present the details of the proposed UMLARD framework.

As illustrated in Figure 7.11, UMLARD consists of three main components: (1) *Representation learning layer* that simultaneously extracts user-aspect features from the profile-view, structural-view, and temporal-view, while embedding the source tweet content into low-dimensional space; (2) *Hierarchical fusion layer* that fuses the learned representation at both view-level and user-level; and (3) *Rumor detection*

layer that makes use of a fully connected layer to predict the labels of tweets, based on the learned user-aspect knowledge and tweet content.

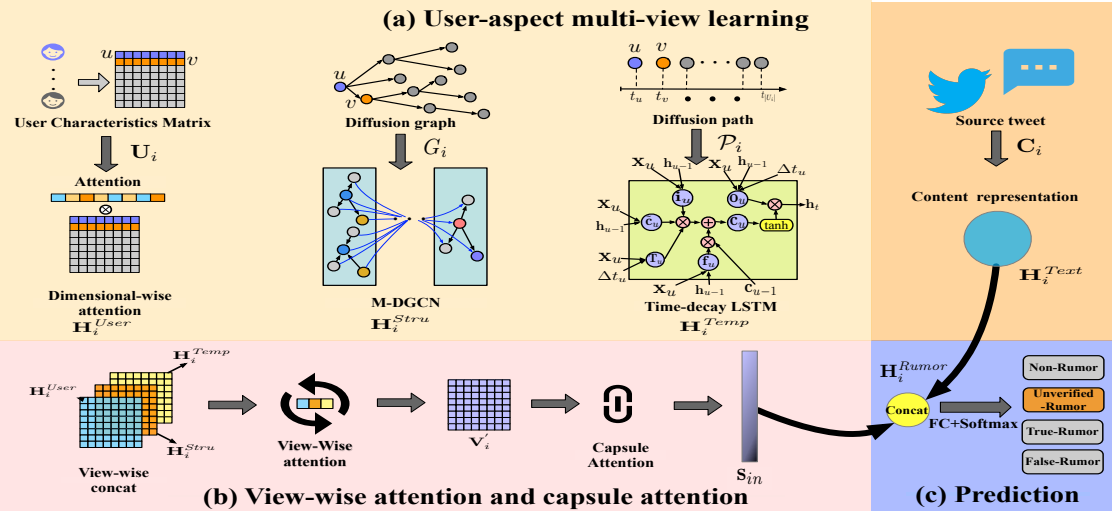


Figure 7.11: An overview of UMLARD. (a) The inputs of UMLARD are the observed diffusion network, the diffusion path, the user characteristic matrix, and the content of the source tweet. It uses a dimensional-wise attention layer, a multi-layer diffusion graph convolutional network (M-DGCN), and a time-decay LSTM to learn the latent representations from the three kinds of inputs, respectively. (b) It learns to discriminate the role of three-views and the importance of users in identifying misinformation. (c) Finally, we concatenate the learned features with text content to perform classification.

7.2.3.1 Learning the User Profile-View

User profiles have been demonstrated to be strong indicators when detecting rumors [26, 83]. The user profile characteristics are either explicit (e.g., username and geolocations) or implicit (e.g., gender and age). However, accessing the implicit features may not always be feasible due to the privacy concerns of many OSNs. Therefore, we consider the following eight explicit features, grouped in two major categories, which can be typically accessed in most OSNs:

- **Profile-Related features** include five basic user description fields: the screen name that the user identify herself; the user's self description; the attribute indicating whether the account has been verified by the platform; the geographical location of the user; and the UTC time that the user account was created on the social platform.
- **Influence-Related features** include three attributes describing user activities and social relations: the number of posts issued by the user, the number of followers, and the mutual follower-ship.

For each user u_j in a tweet m_i, we concatenate the profile characteristics into one feature vector, and then form the user characteristic matrix $\mathbf{U}_i \in \mathbb{R}^{|U_i| \times d_{user}}$ by

concatenating all user vectors for the users involved in spreading the tweet.

To provide explanations on which characteristics are useful for rumor detection, we design a dimensional-wise attention layer to assign weights to each dimension of user profiles. Its aim is to learn how to discriminate the importance of different characteristics. First, we expand $\mathbf{U}_i$ as a sequence of 1-dimensional "channels" for the features, i.e., $\mathbf{U}_i \in \mathbb{R}^{|U_i| \times 1 \times d_{user}}$, where $|U_i|$, 1 and d_{user} can be regarded as the height, width and channel of an image (similarly to the channels for each of the primitive colors – red, green and blue – in image processing). Then, we use a global average pooling (GAP) to aggregate the global information into a dimensionality-wise descriptor $\mathbf{z} \in \mathbb{R}^{d_{user}}$, where $\mathbf{z} = \frac{1}{|U_i| \times 1} \sum_{h=1,w=1}^{|U_i|,1} \mathbf{U}_i(h, w)$. To capture the dimensional-wise dependencies, we employ two fully connected layers with non-linearity – i.e., dimensionality-reduction layer and dimensionality-increasing layer:

$$\begin{aligned} \mathbf{f}_{red} &= \tanh(\mathbf{W}_1 \mathbf{z} + \mathbf{b}_1), \\ \mathbf{f}_{inc} &= \text{softmax}(\mathbf{W}_2 \mathbf{f}_{red} + \mathbf{b}_2), \end{aligned} \tag{7.14}$$

where $\mathbf{W}_1 \in \mathbb{R}^{\frac{d_{user}}{r} \times d_{user}}$ and $\mathbf{W}_2 \in \mathbb{R}^{d_{user} \times \frac{d_{user}}{r}}$ are parameter matrices, $\mathbf{b}_1 \in \mathbb{R}^{\frac{d_{user}}{r}}$ and $\mathbf{b}_2 \in \mathbb{R}^{d_{user}}$ are biases, and r is the reduction ratio. Thus, the final output of the user profile-view becomes:

$$\mathbf{H}_i^{User} = \mathbf{U}_i \mathbf{f}_{inc} + \mathbf{U}_i, \tag{7.15}$$

where $\mathbf{H}_i^{User} \in \mathbb{R}^{|U_i| \times d_{user}}$, $\mathbf{f}_{inc}$ denotes the attention score allocating different importance to each dimension of $\mathbf{U}_i$ through the multiplication operation, i.e., $\mathbf{U}_i \mathbf{f}_{inc}$. The operation of plus $\mathbf{U}_i$ is borrowed from the idea of skip connections [66].

The objective of dimensional-wise attention layer is to obtain a new user characteristic matrix through correlation training between the user profile's different characteristics by assigning different dimensions of the matrix with the different weights during training the model. In general, the contributing characteristics would be strengthened. Since the trivial characteristics should be weakened, we can also reduce the noise brought by non-critical characteristics, thereby improving the accuracy of the detection task. This effect is especially valuable for early-stage rumor detection. For example, when the number of participating users and the corresponding profiles are limited, it is particularly important to encourage the fundamental characteristics to explain rumor identification decisions. We will provide visual explanations in Sec. 7.2.4.

7.2.3.2 Learning the User Structural-View

The structural information of users who participate in spreading a tweet is extracted from the diffusion graph, which aims to capture the degree of connection, similarity, distance, and even community, etc., between users [99]. Inspired by the recent successes of network representation learning methods in processing graph-structured

data [41, 61, 115, 173], we define a multi-layer diffusion graph convolutional network (M-DGCN) as user structural-view encoder, in which the propagation rule of diffusion convolutional network is defined as:

$$\mathbf{H}^{(l+1)} = \sigma((\theta_O(\mathbf{D}_O^{-1}\mathbf{A}) + \theta_I(\mathbf{D}_I^{-1}\mathbf{A}^T))\mathbf{H}^{(l)}), \tag{7.16}$$

where θ_O and θ_I are filter parameters; $\mathbf{D}_O^{-1}\mathbf{A}$ and $\mathbf{D}_I^{-1}\mathbf{A}^T$ are transition matrices of the forward diffusion process and the reverse one, respectively – $\mathbf{D}_O$ and $\mathbf{D}_I$ represent out-degree diagonal matrix and in-degree diagonal matrix, respectively; $\sigma(\cdot)$ denotes activation function, i.e., ReLU$(\cdot)$ here; $\mathbf{H}^{(l)} \in \mathbb{R}^{|U|\times F}$ is the matrix of activation in the l-th layer – $|U|$ is the number of users in the diffusion graph and F is the dimension of the output. The difference between our M-DGCN and previous graph convolutional network [41, 61] is that the Chebyshev kernel in M-DGCN is equal to 1, whereas we stack a couple of such layers to aggregate the information from the distant nodes rather than the K-localized convolutions. In this layer, the initial input $\mathbf{H}^{(0)}$ is obtained from a pre-trained network embedding layer which maps a user u_j to it's D-dimensional representation $\mathbf{g}_j \in \mathbb{R}^D$, which allows the varying-size diffusion networks learning.

In order to reduce over-fitting for diffusion convolutional network, we employed a recently developed technique *DropEdge* (cf. [165]) for robust structural-view learning. That is, we randomly drop edges from the input diffusion graphs to generate different copies with a certain ratio in each training epoch. More specifically, suppose the total number of edges in the diffusion graph is $|E|$ and the dropping rate is r_{drop}. The adjacency matrix after dropout is computed as $\hat{\mathbf{A}} = \mathbf{A} - \mathbf{A}_{drop}$, where $\mathbf{A}_{drop}$ is the matrix constructed using $|E| \times r_{drop}$ edges randomly sampled from the original edge set E. After the diffusion convolutional layer, the diffusion graph G_i is represented as a vector matrix $\mathbf{H}_i^{Stru} \in \mathbb{R}^{|U_i|\times d_{stru}}$.

The structural-view $\mathbf{H}_i^{Stru}$ learned through M-DGCN represents the role of a node (i.e., a user) in the information spreading. M-DGCN not only models the propagation direction of information between spreaders but also aggregates high-order structural details, including the cascade virality, spreading patterns, etc., which may facilitate the rumor identification. We note that in [84] it has been demonstrated that the rumors have similar propagation patterns.

7.2.3.3 Learning the User Temporal-View

Users' engagement time and the sequential patterns of retweets also play an essential role in detecting rumors [33, 100]. We capture this view of users based on the diffusion path. Each user in the diffusion path would be assigned two types of embeddings: a static-embedding and a dynamic-embedding.

- **Static-embedding** refers to the relative position j $(1 \leq j \leq |U_i|)$ for each user u_j in the sequence. We encode this information based on the chronological

order of retweet times, and the users with the same retweet time will have the same position embedding. Inspired by the self-attention [131], we obtain the static-embedding $\mathbf{e}_j^s$ using a positional-encoding technique based on sine and cosine functions of frequencies:

$$\begin{aligned} \mathbf{PE}(j)_{2d} &= \sin(j/10000^{2d/d_e}), \\ \mathbf{PE}(j)_{2d+1} &= \cos(j/10000^{2d/d_e}) \end{aligned} \tag{7.17}$$

where d_e is an adjustable dimension and $1 \leq d \leq d_e/2$ denotes the dimension index in $\mathbf{e}_j^s$. The basic idea of this choice is to allow the model attending the relative position of the users. For details of this formula, refer to [131].

- **Dynamic-embedding** initializes user representations as one-hot vector $\mathbf{q} \in \mathbb{R}^N$, where N denotes the total number of users in the dataset. All users are associated with a specific embedding matrix $\mathbf{E} \in \mathbb{R}^{N \times d_e}$, where d_e is an adjustable dimension. Matrix $\mathbf{E}$ converts each user u_j into a unique representation vector as $\mathbf{e}_j^d = \mathbf{qE}, \mathbf{e}_j^d \in \mathbb{R}^{d_e}$. In this way, the user embedding matrix $\mathbf{E}$ can be learned during training, supervised by the downstream task, i.e., rumor detection in this work.

Subsequently, we use an RNN model (e.g., LSTM [40]) to learn the temporal dependence of the diffusion. However, the influence of retweet users will diminish over time, and the "vanilla LSTM" is not capable of capturing this time-decay effect of information diffusion. To address this issue, we introduce a time-gate inspired by [174] into the LSTM.

The time-gate not only controls the influence of $\mathbf{x}_j$ – the combination of static and dynamic embeddings – on the current step, but also caches the time interval between consecutive retweets to model the time-decay effect. Specifically, a time-decay LSTM unit takes: $\mathbf{x}_j$, previous hidden state $\mathbf{h}_{j-1}$, and time interval Δ_{t_j} as inputs – and outputs the current hidden state $\mathbf{h}_j$ using:

$$\begin{aligned} \mathbf{x}_j &= \mathbf{e}_j^s + \mathbf{e}_j^d, \\ \mathbf{i}_j &= \sigma\left(\mathbf{W}_{xi}\mathbf{x}_j + \mathbf{U}_{hi}\mathbf{h}_{j-1} + \mathbf{b}_i\right), \\ \mathbf{f}_j &= \sigma\left(\mathbf{W}_{xf}\mathbf{x}_j + \mathbf{U}_{hf}\mathbf{h}_{j-1} + \mathbf{b}_f\right), \\ \mathbf{T}_j &= \sigma\left(\mathbf{W}_{xT}\mathbf{x}_j + \tanh\left(\mathbf{W}_{tt}\Delta t_j\right) + \mathbf{b}_T\right), \\ \mathbf{o}_j &= \sigma\left(\mathbf{W}_{xo}\mathbf{x}_j + \mathbf{U}_{ho}\mathbf{h}_{j-1} + \mathbf{W}_{to}\Delta t_j + \mathbf{b}_o\right), \\ \widetilde{\mathbf{c}}_j &= \tanh\left(\mathbf{W}_{xz}\mathbf{x}_j + \mathbf{U}_{hz}\mathbf{h}_{j-1} + \mathbf{b}_z\right), \end{aligned} \tag{7.18}$$

where $\sigma(\cdot)$ is the sigmoid function; $\mathbf{i}_j, \mathbf{f}_j, \mathbf{T}_j, \mathbf{o}_j, \widetilde{\mathbf{c}}_j, \mathbf{b}_*$ are the input gate, forget gate, time gate, output gate, new candidate vector and bias vector, respectively. The matrices $\mathbf{W}_{x*} \in \mathbb{R}^{d_e \times d_{temp}}$, $\mathbf{W}_{t*} \in \mathbb{R}^{1 \times d_{temp}}$ and $\mathbf{U}_{h*} \in \mathbb{R}^{d_h \times d_{temp}}$ represent the different gate parameters. In particular, the memory cell $\mathbf{c}_j$ is updated by replacing

the existing memory unit with a new cell $\mathbf{c}_j$ as:

$$\mathbf{c}_j = \mathbf{f}_j \odot \mathbf{c}_{j-1} + \mathbf{i}_j \odot \mathbf{T}_j \odot \widetilde{\mathbf{c}}_j, \tag{7.19}$$

where $\odot$ denotes the element-wise multiplication. The hidden state is then updated by:

$$\mathbf{h}_j = \mathbf{o}_j \odot \tanh\left(\mathbf{c}_j\right), \tag{7.20}$$

Finally, the representation vector for the temporal-view is $\mathbf{H}_i^{Temp} = \left\{\mathbf{h}_j^{Temp} | j \in [1, |U_i|]\right\}$, where $\mathbf{H}_i^{Temp} \in \mathbb{R}^{|U_i| \times d_{temp}}$. Note that the temporal-view of the user obtained by the time-decay LSTM reflects each user's influence on the subsequent participators in the message diffusion.

7.2.3.4 View-Wise Attention for View-Level Feature Fusion

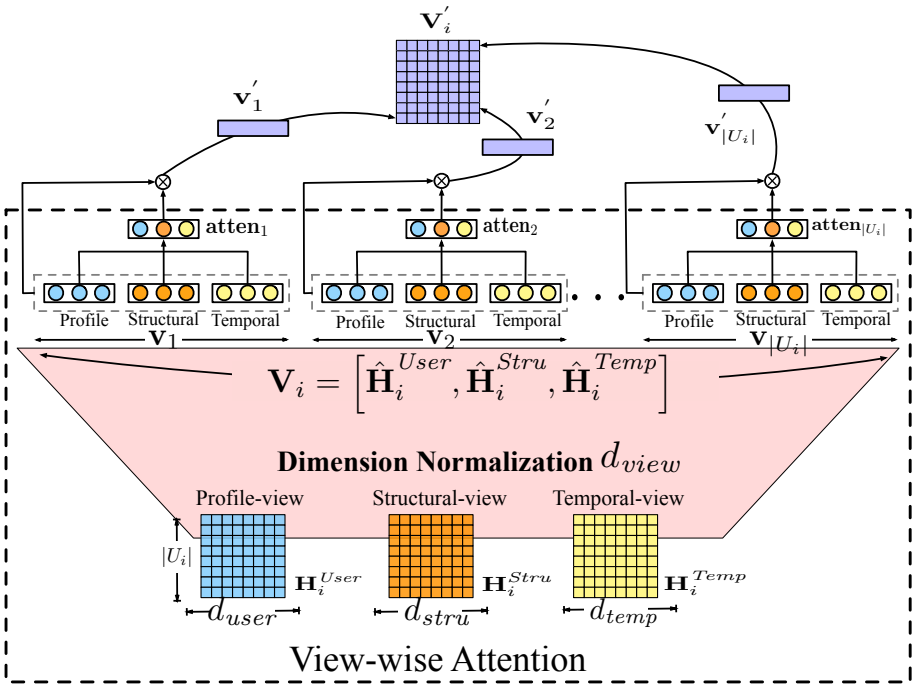


Figure 7.12: Illustration of view-wise attention.

After obtaining the latent representation for each view, we need to fuse the multi-view features. Rather than directly concatenating different aspects, as often done in the existing solutions [91, 94, 175], we present a method to capture the differences between different views. The primary motivation stems from the observation that various views are not equally relevant in the task of rumor identification. Towards that, we propose a view-wise attention layer to prioritize the fundamental views for each user.

As depicted in Figure 7.12, the view-wise attention layer takes profile-view, structural-view, and temporal-view as input and generates the attention score for each view at

the user-level. Specifically, it first normalizes the dimensions of the three views' vectors as d_{view} via a fully connected layer. Let $\mathbf{V}_i = \left[\hat{\mathbf{H}}_i^{User}, \hat{\mathbf{H}}_i^{Stru}, \hat{\mathbf{H}}_i^{Temp}\right]$ denote the feature set after dimension normalization. Each vector $\mathbf{v}_j = \left[\hat{\mathbf{h}}_j^{User}, \hat{\mathbf{h}}_j^{Stru}, \hat{\mathbf{h}}_j^{Temp}\right] \in \mathbf{V}_i$ represents a view feature set for a specific user u_j engaged in spreading tweet m_i. Then, the view-wise attention layer calculates the attention score $\mathbf{atten}_j \in \mathbb{R}^{1\times 3}$ for each view of the user-level feature set $\mathbf{v}_j \in \mathbb{R}^{d_{view}\times 3}$ as:

$$\bar{\mathbf{v}}_j = \tanh\left(\mathbf{W}_v \cdot \mathbf{v}_j\right), \tag{7.21}$$

$$\mathbf{atten}_j = \mathrm{softmax}\left(\mathbf{w}_v^T \cdot \bar{\mathbf{v}}_j\right), \tag{7.22}$$

where $\mathbf{W}_v \in \mathbb{R}^{d_{view}\times d_{view}}$, $\mathbf{w}_v \in \mathbb{R}^{d_{view}}$ are learnable projection parameters during training, $\bar{\mathbf{v}}_j = \left[\bar{\mathbf{h}}_j^{User}, \bar{\mathbf{h}}_j^{Stru}, \bar{\mathbf{h}}_j^{Temp}\right]$. Here, the view-wise attention layer first computes the hidden representation of $\mathbf{v}_j$ through multiplying it with $\mathbf{W}_v$ to get $\bar{\mathbf{v}}_j$, which is implemented with a fully connected layer without bias. It measures the weight of a view as the similarity of $\bar{\mathbf{h}}_j^*$ ($* \in \{User, Stru, Temp\}$) with a view-level context vector $\mathbf{w}_v$ and finally obtains a normalized weight through a softmax function. Each entry of $\mathbf{atten}_j$ represents an importance score for a specific view of user j.

Finally, the fused multi-view feature vector $\mathbf{v}_j^{'}$ for user u_j can be calculated as:

$$\mathbf{v}_j^{'} = \mathbf{atten}_j \cdot \mathbf{v}_j, \tag{7.23}$$

where $\mathbf{v}_j^{'} \in \mathbb{R}^{d_{view}}$. The fused multi-view feature vector for each user forms the multi-view matrix, denoted as $\mathbf{V}_i^{'} = \left\{\mathbf{v}_1^{'}, \mathbf{v}_2^{'}, \cdots, \mathbf{v}_{|U_i|}^{'}\right\}$, where $\mathbf{V}_i^{'} \in \mathbb{R}^{|U_i|\times d_{view}}$.

7.2.3.5 Capsule Attention for User-level Feature Fusion

Most of existing works [16, 100, 102] would directly use $\mathbf{V}_i^{'}$ for rumor detection. However, that does not properly discriminate different users, contrary to the fact that different users in a tweet propagation network may contribute differently to classifying the tweet. In our UMLARD, we introduce a capsule attention layer inspired by the recent success of capsule networks [133, 176, 177]. The Capsule network was first proposed in [133] and the main idea is to replace the scalar-output feature detectors in traditional neural networks with vector-output capsules, and train the model by the dynamic routing algorithm. It can be regarded as a parallel attention mechanism that allows each underlying capsule to attend to higher capsules at different importance.

In UMLARD, the capsule attention chooses the most related underlying vectors dynamically to form the only upper capsule via an unsupervised routing-by-agreement mechanism, which also avoids the intensive computation raised by a huge amount of parameters used in multi-layer attention. More precisely, in the n-th iteration,

the upper capsule $\mathbf{s}_{in}$ is calculated by:

$$\mathbf{s}_{in} = \sum_{j}^{|U_i|} \mathbf{a}_j \hat{\mathbf{v}}_j, \hat{\mathbf{v}}_j = \mathbf{W}\mathbf{v}_j^{'}, \tag{7.24}$$

where the coupling coefficient $\mathbf{a}_j$ indicates the contributions of a user capsule to the upper capsule – namely, the attention score of each user. $\mathbf{W} \in \mathbb{R}^{d_{view} \times d_{caps}}$ is the transform matrix that guarantees the feature representation ability of the center vector after clustering, and identifies the order of input features. Note that before the last iteration we add a normalization $\widetilde{\mathbf{s}}_{in} = \mathbf{s}_{in} / \|\mathbf{s}_{in}\|$ in $\mathbf{s}_{in}$ to overcome the information loss caused by the original CapsAtt [176].

The coupling coefficient $\mathbf{a}_j \in \mathbb{R}^{|U_i| \times 1}$ is determined by a "routing softmax" whose initial logit is denoted as $\mathbf{b}_j$, where $\mathbf{b}_j$ is the log prior probability that the j-th user capsule should be coupled to the upper capsule $\mathbf{s}_{in}$. The coefficient is calculated by:

$$\mathbf{a}_j = \frac{\exp(\mathbf{b}_j)}{\sum_k^{|U_i|} \exp(\mathbf{b}_k)}, \tag{7.25}$$

The log prior is initialized with zero and then updated by adding agreements between the user capsule and the upper capsule:

$$\mathbf{b}_j = \mathbf{b}_j + \hat{\mathbf{v}}_j \cdot \widetilde{\mathbf{s}}_{in}, \tag{7.26}$$

These agreements are added to log priors after each routing, i.e., the output capsule $\mathbf{s}_{in}$ represents the feature matrix after correlation learning, which can be easily coupled into the model for downstream tasks, in our case the rumor detection.

7.2.3.6 Tweet Content Representation

Tweet content is one of the most important features in rumor detection [23, 29, 78], and has been extensively studied in the literature [16, 33, 80, 102, 178], where various natural language processing (NLP) techniques have been exploited for learning informative signals from the textual content. Though content learning is beside the scope of this thesis, we describe a simple CNN layer for text representation learning from the input of word embedding matrix for completeness. A single CNN layer is denoted as:

$$\mathbf{h}_m = \sigma(\mathbf{W} * \mathbf{w}_{m:m+d-1}), \tag{7.27}$$

where $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{L-d+1}\}$ is the extracted feature map, and $\mathbf{W} \in \mathbb{R}^{d \times d_{word}}$ is the convolutional kernel with d as size of the receptive filed, and σ as non-linearity. Then max-pooling operation is used over the feature map to generate the output representation $\hat{\mathbf{H}}$. In our work, we use multiple CNN layers with different receptive fields to obtain multiple features, and then concatenate all outputs to form the tweet content representation $\mathbf{H}_i^{Text}$.

7.2.3.7 Training Objective

Finally, we concatenate content representation $\mathbf{H}_i^{Text}$ and capsule attention $\mathbf{s}_{in}$ to merge the information as:

$$\mathbf{H}_i^{Rumor} = \text{concat}(\mathbf{H}_i^{Text}, \mathbf{s}_{in}) \tag{7.28}$$

which is subsequently used for predicting the label $\hat{\mathbf{y}}_i$ of tweet m_i via a fully connected layer and the softmax function:

$$\hat{\mathbf{y}}_i = \text{softmax}\left(\text{FC}\left(\mathbf{H}_i^{Rumor}\right)\right). \tag{7.29}$$

We train all the parameters by minimizing the *cross-entropy* of the predictions $\hat{\mathbf{Y}}$ and the ground truth labels $\mathbf{Y}$ as:

$$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}) = -\sum_{i \in |M|} \mathbf{y}_i \log \hat{\mathbf{y}}_i + \lambda \left\|\Theta\right\|_2^2, \tag{7.30}$$

where $\|\Theta\|_2^2$ is the L_2 regularizer over all the model parameters Θ, and λ is the trade-off coefficient. In this work, we use *RAdam* [140] as optimizer. The whole training process of UMLARD is outlined in Algorithm 10.

Algorithm 10: Training of UMLARD.

Input: A set of tweets $M = \left\{m_i\right\}_{i=1}^{|M|}$, each tweet $m_i = \left\{G_i, \mathcal{P}_i, \mathbf{U}_i, \mathbf{C}_i\right\}$.
Output: Predicted labels $\hat{\mathbf{Y}}$ for all tweets.

repeat
  for m_i in a batch **do**
    Profile-view learning: $\mathbf{H}_i^{User} \leftarrow \mathbf{U}_i$ via Eq.(7.14) and Eq.(7.15);
    Structural-view learning $\mathbf{H}_i^{Stru} \leftarrow G_i$ via Eq.(7.16);
    Temporal-view Learning $\mathbf{H}_i^{Temp} \leftarrow \mathcal{P}_i$ via Eq.(7.18) - Eq.(7.20);
    Content representation: $\mathbf{H}_i^{Text} \leftarrow \mathbf{C}_i$ via Eq.(7.27);
    Nomalize dimensions:
    $\mathbf{V}_i = [\hat{\mathbf{H}}_i^{User}, \hat{\mathbf{H}}_i^{Stru}, \hat{\mathbf{H}}_i^{Temp}] \leftarrow [\mathbf{H}_i^{User}, \mathbf{H}_i^{Stru}, \mathbf{H}_i^{Temp}]$;
    View-wise attention learning: $\mathbf{V}_i^{'} \leftarrow \mathbf{V}_i$ via Eq.(7.21) to Eq.(7.23);
    Capsule attention learning: $\mathbf{s}_{in} \leftarrow \mathbf{V}_i^{'}$ via Eq.(7.24);
    Merge $\mathbf{H}_i^{Text}$ and $\mathbf{s}_{in}$ via Eq.(7.28);
    Estimate the probability $\hat{\mathbf{y}}_i$ via Eq.(7.29);
    Compute loss $\mathcal{L}(\mathbf{y}_i, \hat{\mathbf{y}}_i)$, via Eq.(7.30);
    Update parameters using RAdam.
  end for
until convergence;

7.2.3.8 Computational Complexity

We finalize this section with a discussion of the computational complexity of UMLARD, analyzed in two categories.

— *Complexity of multi-view representation learning* is influenced by four main components:

(1) As for **profile-view** that only uses dimensional-wise attention to allocate varying weights to each dimension, the computational complexity stems from the two fully connected layers, i.e., $\mathcal{O}(2d_{user}^2/r)$. Because the dimension of user characteristic d_{user} is very small, this computational cost is typically negligible.

(2) We use a multi-layer diffusion convolutional network for the **structural-view** learning (cf. Eq.(7.16)), which can be decomposed into two parts with the same time complexity, i.e., $\mathbf{D}_I^{-1}\mathbf{A}$ and $\mathbf{D}_O^{-1}\mathbf{A}^T$. Since the two matrices are very sparse, the time complexity is $\mathcal{O}(|E|)$, i.e., linear with the number of edges. Specifically, in a two-layer M-DGCN, the computational complexity is $\mathcal{O}(|E|DF_1F_2)$, where D, F_1 and F_2 are the input feature size, and the hidden size for the first and the last M-DGCN layer, respectively.

(3) The **temporal-view** is learned through a time-decay LSTM. The computational complexity of original LSTM per time step is $\mathcal{O}(1)$ due to LSTM is local in space and time [40]. Compared with LSTM, the only difference of our time-decay LSTM is an extra time-gate that controls the influential decreasing with time. This operator introduced extra parameters that requires $4(d_e d_{temp} + d_{temp}^2 + d_{temp}) + d_e d_{temp} + 3d_{temp}$ complexity. Besides, the dynamic embedding in UMLARD needs $N \times d_e$ parameters.

(4) For the **source tweet** representation learning, the CNN layers have the time complexity of $\mathcal{O}(\sum_{l=1}^{L}(M_l^2 K_l^2 C_{l-1} C_l))$, where L is the total number of CNN layers; K_l C_{l-1}, C_l are kernel size, input channel number and output channel number for l-th layer; output size is $M_l = (X_l - K_l)/Stride + 1$ and X_l is the input feature size. Overall, this component requires $\sum_{l=1}^{L}(K_l^2 C_{l-1} C_l)$ parameters.

— *Complexity of fusion layers.* In the hierarchical fusion layers, the time and space complexities of both view-wise attention and capsule attention are related to the input and output dimensions of the latent variables. In view-wise attention, it introduces $d_{view} \times d_{view} + |U_i| \times d_{view}$ parameters. As for the capsule attention layer, the parameter size is $d_{view} \times d_{caps}$, where d_{view} and d_{caps} represent view size and capsule size, respectively.

7.2.4 Evaluation

We now present the findings from our experimental evaluations. We compare the performance of our UMLARD with the state-of-art baselines on rumor detection, and we also investigate the effects of different components by comparing several variants of UMLARD.

Specifically, we would aim at providing quantitative characterization of the following research-related questions:

Table 7.11: Statistics of the datasets.

Statistic	Twitter15	Twitter16	Weibo
# source tweets	1,482	809	4,664
# users	477,009	286,657	2,746,818
# non-rumors	370	199	2,351
# false-rumors	369	205	2,313
# true-rumors	372	207	–
# unverified-rumors	371	198	–
Max. # retweets	2989	3058	59,318
Min. # retweets	55	73	10
Avg. # retweets	398	422	816
Avg. # time length	1,268 Hours	828 Hours	1,811 Hours

- **Q1**: How does UMLARD perform on rumor detection compare with the state-of-the-art baselines?
- **Q2**: What is the effect of each component of UMLARD?
- **Q3**: Can UMLARD detect rumors in early stages of their propagation?
- **Q4**: Can UMLARD explain the model behavior and the predicted results?

7.2.4.1 Experimental Settings

Following is the description of the main aspects of our experimental setup.

1) Datasets: We conduct our experiments on the three real-world datasets[1]: *Twitter15*, *Twitter16* [84] and *Weibo* [33]. In each dataset, a group of widespread source tweets along with their propagation threads with time stamps are provided. We construct propagation paths and diffusion networks from the propagation threads, which are also used for user temporal-aspect embedding and user structural-aspect embedding.

Different from the experiment settings in previous PLRD (Section 7.1), in this section, we consider the two Twitter datasets as multi-class datasets, i.e., each source tweet is annotated with one of the four class labels, i.e., *non-rumor*, *false-rumor*, *true-rumor*, and *unverified-rumor*, while the Weibo dataset contains binary labels: *false-rumor*, *non-rumor* – the labeling rules follow the method in [33]. The statistics of the three datasets are shown in Table 7.11. We extract the same user characteristics as PLRD for both Twitter and Weibo dataset, as shown in Table 7.2. Specifically, as for the Weibo dataset, we directly extract these eight characteristics from the JSON files in the original dataset. And the way to split the dataset into training, validation, and testing set follows the same setting in PLRD (see Section 7.1.4.2).

[1]https://www.dropbox.com/s/7ewzdrbelpmrnxu/rumdetect2017.zip?dl=0

2) Baselines: We compare UMLARD with following state-of-the-art rumor detection baseline models:

- **DTC** [23], **SVM-TS** [85], **GRU** [108], **TD-RvNN** [34], **PPC_RNN+CNN (PPC)**, **Bi-GCN** [16], **GCAN** [102], the description of these baseline methods can be found in Section 7.1.4.1. We also compared UMLARD with **PLRD** [45] (Section 7.1).
- **SVM-RBF** [86]: A support vector machine (SVM) based model that uses radius basis function (RBF) as the kernel and leverages the handcrafted features of posts for rumor detection.
- **PLAN** [179]: A hierarchical token- and post-level attention model for rumor detection, which models pairwise interactions between tweets via the self-attention mechanism.
- **Bi-GCN-U** [16]: A variant of Bi-GCN, which uses user profile characteristics to replace the comment features.
- **STS-NN** [180]: A rumor detection model based on spatial-temporal neural networks. It treats the spatial structure and temporal structures as a whole to learn a fine-grained rumor representation.
- **GCAN-G** [102]: A variant of GCAN, which uses the diffusion graph to replace the user similarity graph.
- **RDEA** [103]: A self-supervised rumor detection model. On the basis of Bi-GCN [16], RDEA improves the rumor representations and alleviates limited data issues through event augmentation and contrastive learning.

3) Implementation details: We implement DTC with Weka[1], SVM-based models with scikit-learn[2], and other neural network-based models with Tensorflow[3]. All baselines follow the parameter settings in the original papers. For UMLARD, the learning rate is initialized at 0.001 and gradually decreases as the training proceeds. We use word2vec to initialize the word embeddings with $d_{word} = 300$ dimensions, and the convolution kernel size is set to $[3, 4, 5]$, and per size with 100 kernels. The embedding size for structural view d_{stru} and temporal view d_{temp} of users are both set to 64; the view size d_{view} is also set to 64, as is the the capsule size; and the iteration number varies between 2 and 4. The batch size is 64; and the rate of dropout in the main neural networks is 0.5; the dropout rate in DropEdge is 0.2. The training process is iterated upon for 200 epochs, but would be stopped earlier if the validation loss does not decrease after 10 epochs.

4) Evaluation metrics: We use accuracy (ACC) and F-measure (F1) as the evaluation protocols to measure the models' performance. Specifically, ACC measures

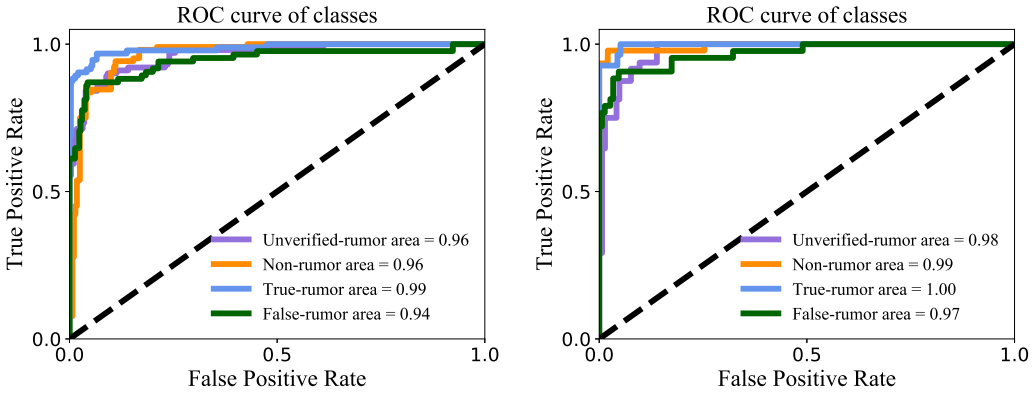
[1]https://www.cs.waikato.ac.nz/ml/weka/
[2]https://scikit-learn.org/
[3]https://www.tensorflow.org/

the proportion of correctly classified tweets, while F1 is the harmonic mean of the precision and recall values averaged across four classes. As for the Weibo dataset, we also report the precision and recall results.

Table 7.12: Overall performance comparison of rumor detection on Twitter15 (the observation window is set to the previous 40 retweets). "UR": unverified-rumor; "NR": non-rumor; "TR": true-rumor; "FR": false-rumor. The best method is shown in **bold**, and the second best is shown as underlined. A paired t-test is performed and $*$ indicates a statistical significance $p < 0.05$ compared to the best baseline method (RDEA).

Model	Twitter15				
	ACC.	F1			
		UR	NR	TR	FR
DTC	0.454	0.415	0.733	0.317	0.355
SVM-RBF	0.318	0.218	0.225	0.455	0.082
SVM-TS	0.544	0.483	0.796	0.404	0.472
GRU	0.646	0.608	0.592	0.792	0.574
TD-RvNN	0.723	0.654	0.682	0.821	0.758
PPC	0.697	0.689	0.760	0.696	0.645
PLAN	0.787	0.775	0.7754	0.768	0.807
Bi-GCN	0.829	0.752	0.772	0.885	0.847
Bi-GCN-U	0.778	0.764	0.741	0.853	0.752
GCAN	0.808	0.690	**0.930**	0.812	0.758
GCAN-G	0.750	0.731	0.754	0.823	0.678
STS-NN	0.808	0.779	0.786	0.860	0.808
RDEA	0.835	0.819	0.786	0.887	0.837
PLRD	0.622	0.519	0.832	0.438	0.596
UMLARD	**0.857***	**0.835***	0.840*	**0.906***	**0.848***

7.2.4.2 Overall Performance (Q1)



(a) Twitter15 **(b)** Twitter16

Figure 7.13: ROC curve comparison for each information type. Area under curve of ROC (AUC) is presented after the legend.

Table 7.12, Table 7.13, and Table 7.14 report the performance comparison among

Table 7.13: Overall performance comparison of rumor detection on Twitter16 (the observation window is set to the previous 40 retweets). "UR": unverified-rumor; "NR": non-rumor; "TR": true-rumor; "FR": false-rumor. The best method is shown in **bold**, and the second best is shown as underlined. A paired t-test is performed and $*$ indicates a statistical significance $p < 0.05$ compared to the best baseline method (RDEA).

Model	Twitter16				
	ACC.	F1			
		UR	NR	TR	FR
DTC	0.465	0.403	0.643	0.419	0.393
SVM-RBF	0.321	0.419	0.037	0.423	0.085
SVM-TS	0.574	0.526	0.755	0.571	0.420
GRU	0.633	0.686	0.593	0.772	0.489
TD-RvNN	0.737	0.708	0.662	0.835	0.743
PPC	0.702	0.608	0.711	0.816	0.664
PLAN	0.799	0.779	0.754	0.836	0.821
Bi-GCN	0.837	0.818	0.772	0.885	0.847
Bi-GCN-U	0.786	0.733	0.783	0.875	0.767
GCAN	0.765	0.784	0.848	0.678	0.754
GCAN-G	0.721	0.642	0.690	0.799	0.732
STS-NN	0.829	0.838	0.775	0.899	0.809
RDEA	0.848	0.868	0.729	0.922	0.823
PLRD	0.646	0.618	0.698	0.609	0.445
UMLARD	**0.901***	**0.822***	**0.965***	**0.960***	**0.855***

UMLARD and baselines on three datasets, from which we have the following observations:

O1: Feature-based approaches such as SVM-TS, SVM-RBF, and DTC perform poorly. These methods use hand-crafted features based on the overall statistics of tweets, but are not sufficient to capture the generalizable features associated with tweets and the process of information diffusion. Notably, SVM-RBF performs worse than the other two methods on two Twitter datasets. However, it achieves the best performance among the feature-based modes on Weibo dataset, because it selects the features based on Weibo that are hard to be generalized to other social platforms such as Twitter. SVM-TS achieves relatively better performance because it utilizes an extensive set of features and primarily focuses on retweets' temporal traits.

O2: Deep learning-based models perform significantly better than feature-based methods. As the first work exploiting RNN for efficient rumor detection, GRU only relies on temporal-linguistics of the repost sequence while ignoring other useful information such as diffusion structures and user profiles. TD-RvNN and PPC_RNN+CNN outperform GRU, which indicates the effectiveness of modeling the propagation structure and temporal information in rumor detection. The performance of PLAN slightly exceeds TD-RvNN and PPC_RNN+CNN, because it still mainly focuses on textual information and ignores structural features of rumor propagation.

Table 7.14: Overall performance comparison of rumor detection on Weibo (the observation window is set to the previous 40 retweets). "NR": non-rumor; "FR": false-rumor. The best method is shown in **bold**, and the second best is shown as underlined. A paired t-test is performed and ** indicates a statistical significance $p < 0.01$ compared to the best baseline method (RDEA).

Model	Weibo						
	ACC	NR			FR		
		Prec.	Rec.	F1	Prec.	Rec.	F1
DTC	0.731	0.715	0.747	0.730	0.747	0.715	0.731
SVM-RBF	0.741	0.738	0.747	0.742	0.745	0.735	0.740
SVM-TS	0.780	0.801	0.753	0.780	0.762	0.808	0.784
GRU	0.762	0.803	0.715	0.757	0.728	0.809	0.767
TD-RvNN	0.832	0.832	0.812	0.821	0.821	0.861	0.841
PPC	0.845	0.870	0.810	0.839	0.810	0.883	0.844
PLAN	0.857	0.829	0.904	0.857	0.893	0.805	0.835
Bi-GCN	0.891	0.892	0.892	0.890	0.891	0.891	0.890
Bi-GCN-U	0.864	0.896	0.818	0.860	0.830	0.910	0.868
GCAN	0.880	0.911	0.861	0.885	0.866	0.929	0.896
GCAN-G	0.831	0.815	0.824	0.819	0.847	0.815	0.831
STS-NN	0.875	0.881	0.866	0.865	0.851	0.872	0.852
RDEA	0.911	0.902	0.923	0.907	**0.913**	0.899	0.901
PLRD	0.899	0.936	0.863	0.900	0.862	**0.946**	0.904
UMLARD	**0.928****	**0.942****	**0.965****	**0.924****	**0.894****	0.944**	**0.928****

O3: Bi-GCN, GCAN, STN-SS, and RDEA have considered structural or temporal information, and thus outperform other baselines. In particular, Bi-GCN constructs the diffusion graph based on user replies, i.e., the retweets with comments, which may not reflect the whole structure of rumor dispersion. In contrast, GCAN models the structural information from the user similarity matrix rather than propagation network. Therefore, according to the results, Bi-GCN performs much better than GCAN, because it takes the comments information into consideration. Besides, the bi-directional GCN is more effective in learning propagation structures than vanilla GCN used in GCAN. Although STS-NN extracts both structural and temporal features for rumor detection, STS-NN still performs worse than Bi-GCN, because it fails to discriminate the spatial structures and the temporal patterns. RDEA improves the performance of Bi-GCAN via introducing contrastive learning and event augmentations, which alleviate the influence of limited data issue. However, this method still faces the same problem as Bi-GCN, i.e., reply network is not enough to represent the full information diffusion process. Through comparing UMLARD with Bi-GCN-U and GCAN-G, we find that the performance of Bi-GCN-U and GCAN-G drops significantly. This result indicates that these methods heavily depend on the input features and are ineffective in extracting diffusion patterns as our method. PLRD does not perform very well at Twitter15 and Twitter16 while showing competitive results on the Weibo dataset, since the label of Twitter15 and Twitter16 becomes more fine-grained, and PLRD is not sensitive to the un-verified and true rumors.

O4: UMLARD consistently outperforms all other baselines across all datasets. Compare to the best baseline RDEA, UMLARD models rumor diffusion from multi-view perspective that allows the model to discriminate the importance of features and users in spreading the tweets. These results also validate one of our primary motivations, i.e., various features play different roles in spreading the rumors, and users are the main contributor to the misinformation propagation.

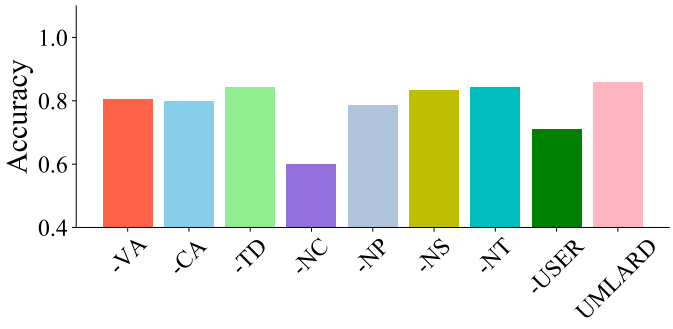
Finally, we scrutinize the performance of UMLARD on discriminating against the individual type of information on Twitter15 and Twitter16. Figure 7.13 plots the ROC curves of the model performance on four different kinds of tweets. We find that our model achieves the best identification results on true-rumors, which indicates that the characteristics of true-rumors are more distinctive from other types of messages. This result also implies that our model is more expressive on a binary classification task that only needs to classify tweets as rumors or truths (cf. the results on Weibo in Table 7.13). In practice, however, unverified-rumors and false-rumors are noisy signals that require careful treatment, which is a promising way of further improving the detection accuracy.

7.2.4.3 Ablation Experiments (Q2)

In this section, we conduct an ablation study to explore the effect of each component in UMLARD. Towards that, we derive the following variants of UMLARD:

- **-VA**: In -VA, ignores the different importance of different views, i.e., it removes the view-wise attention layer.
- **-CA**: In -CA, replaces the capsule attention layer with a fully connected layer.
- **-TD**: In -TD, neglects the time decay effect of retweet behaviors which is replaced by a vanilla LSTM [40] to learn sequential retweet behavior.
- **-NC**: In -NC, removes the content feature of the source tweet but keeps the temporal, profile, and structural features.
- **-NP**: In -NP, disregards the profile features of users but retains temporal, structural, and content features.
- **-NS**: In -NS, ignores the structural features of users but keeps temporal, profile, and content features.
- **-NT**: In -NT, ignores the temporal features of users but keep structural, profile, and content features.
- **-USER**: In -USER, ignores the user-aspect features (i.e., temporal, structural, and profile) that only retains the content feature.

Figure 7.14 illustrates the performance of the variants, where we can observe that:



(a) Twitter15

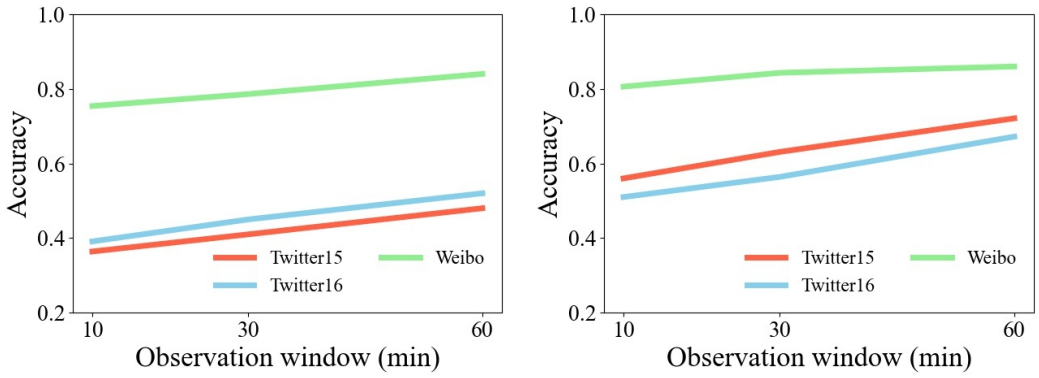
(b) Twitter16

(c) Weibo

Figure 7.14: Ablation study of UMLARD. Two attention mechanisms can significantly improve the detection performance by distinguishing the importance of features and users. Tweet content and profile information are two most informative features on rumor detection.

(1) The content of tweet (**-NC** and **-USER**) is still the most critical signal of discriminating rumors among various features. Without it, the model performance would significantly drop, as observed in many previous works [16, 102]. However, only based on the content feature is insufficient to develop an effective rumor detection model that can identify different types of rumors with high accuracy.

(2) Profile information (**-NP**) is another reliable indicator to detect the rumors because it is a straightforward but useful method to identify the users that spread the misinformation intentionally [26, 83].



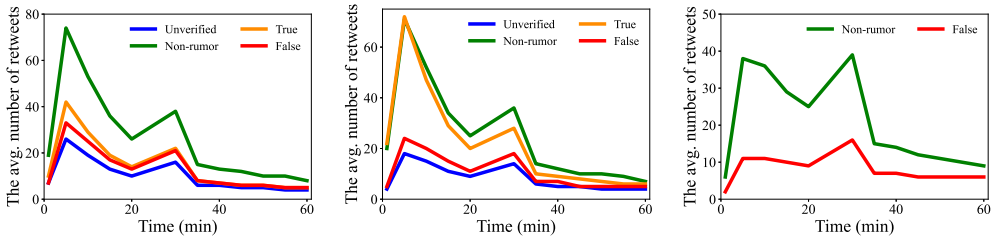
(a) Structural view **(b)** Temporal view

Figure 7.15: Study on structural and temporal view. We only use structural or temporal views to detect rumors in 10, 30, and 60 mins. The results demonstrate the importance of both views in rumor detection.

(3) Though both structural (**-NS**) and temporal information (**-NT**) are informative, they are not as important as contents of tweets and user profiles. This result also explains why the methods proposed in [100], and [16] do not show comparable performance as ours – the former mainly focuses on modeling the temporal information of retweets, whereas the latter one relies on graph neural networks to exploit the diffusion structures. We also conduct additional experiments to demonstrate the importance of structural and temporal features once the input contains enough information, especially in a binary classification task (e.g., Weibo). The results are shown in Figure 7.15. We find that as for Twitter datasets, the detection performance based on structural features grows slightly but is still not good enough as the type of rumors is fine-grained, making it challenging to learn discriminative structural features. As for the Weibo dataset, both structural and temporal features are helpful for rumor detection even in a short time.

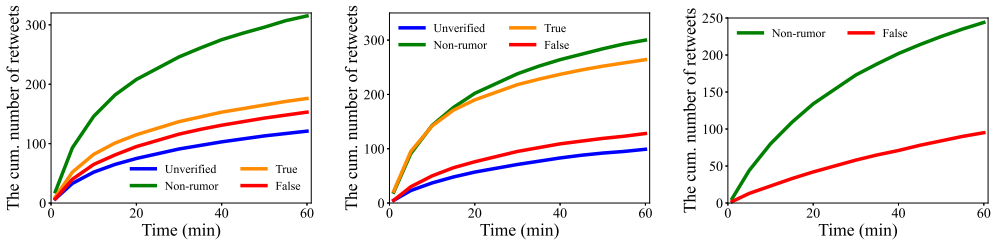
In order to demonstrate our findings in Figure 7.15, we conduct statistical analysis of the datasets and plot the temporal and structural propagation patterns in Figure 7.16–7.19. We find that the differences in temporal patterns are more obvious compared with the structural patterns. In addition, the differences between true and false rumors in Weibo are more significant than the discrepancy between the fine-grained types of rumors in Twitter datasets.

(4) The two attention mechanisms proposed in this work, i.e., view-wise attention (**-VA**) and capsule attention (**-CA**), play a crucial role on identifying the misinformation – the importance of which even exceed temporal features and diffusion patterns. This result also suggests that distinguishing the significance of different views of users can improve classification performance. Similarly, different users play different roles in spreading misinformation, e.g., users may intentionally mislead others or unknowingly retweet doubtful news. However, examining users' purposes is beyond the scope of this work and is left as our future work.



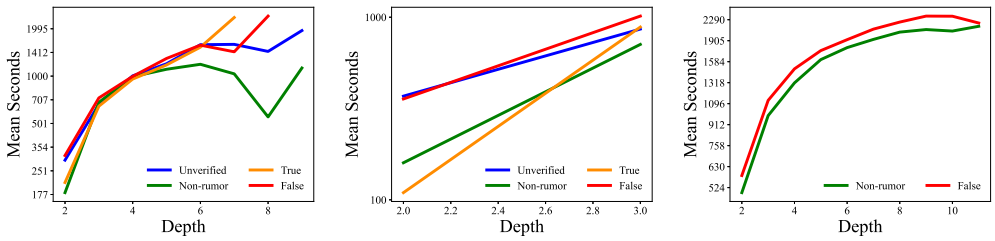
(a) Twitter15 **(b)** Twitter16 **(c)** Weibo

Figure 7.16: The average number of retweets for different types of rumors at different timestamps.



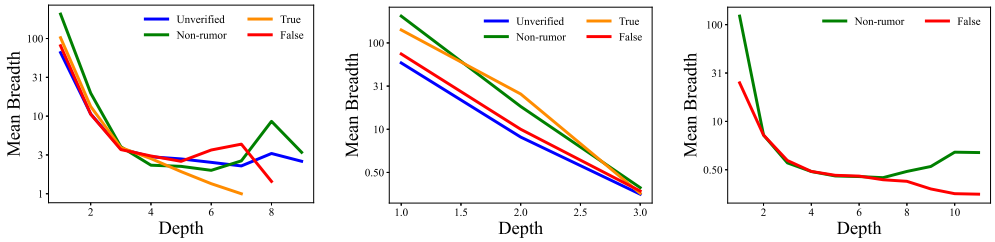
(a) Twitter15 **(b)** Twitter16 **(c)** Weibo

Figure 7.17: The cumulative number of retweets for different types of rumors at different timestamps.



(a) Twitter15 **(b)** Twitter16 **(c)** Weibo

Figure 7.18: The average time (in seconds) required to reach the same network depth. The observation window 60 minutes.

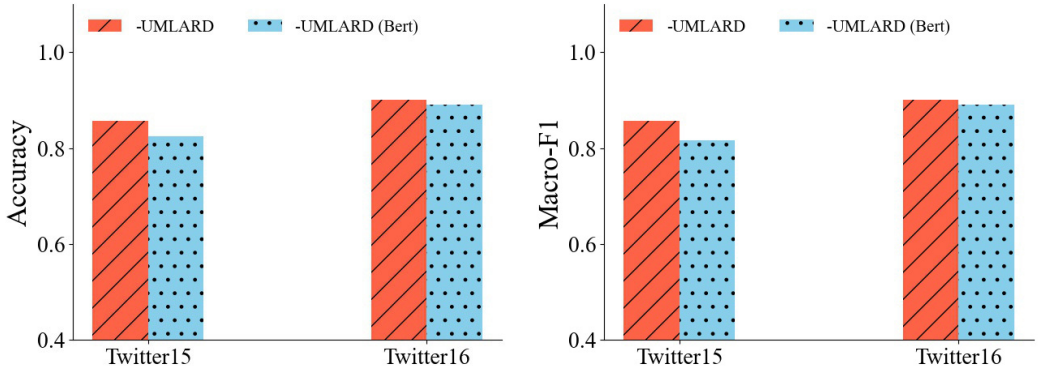


(a) Twitter15 **(b)** Twitter16 **(c)** Weibo

Figure 7.19: The average network breadth for different types of rumors. The observation window is 60 minutes.

(5) Finally, the discrepancy between UMLARD and **-TD** indicates the gain of modeling time decay in retweet cascades. In other words, both real information and false

information will significantly reduce their influence over time.



(a) Accuracy **(b)** Macro-F1

Figure 7.20: Content-aspect study of UMLARD.

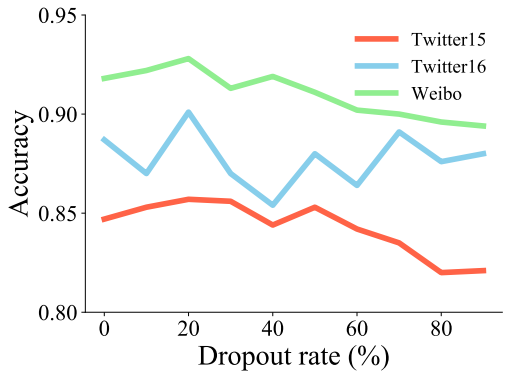


Figure 7.21: DropEdge Study of UMLARD.

To further investigate the content-aspect effect, we examine the influence of different word embedding methods. Specifically, we use the state-of-the-art Bert-based pretraining model [181] to replace the word2vec and then compare the performance on accuracy and macro-F1. In our work, we choose BERT-Base[1], which was trained on a large text corpus (e.g., Wikipedia). The results are shown in Figure 7.20. We can observe that the performance of Bert-based UMLARD is surprisingly lower. This happens due to the characteristics of tweet text, which are *short*, *sparse*, *sporadic* and written *casually*. Therefore, the Bert-based pretraining techniques that are usually trained on large-scale language corpus are difficult to directly used for short-text tasks such as Twitter content embedding. This conjecture is in accordance with some recent observations on [164].

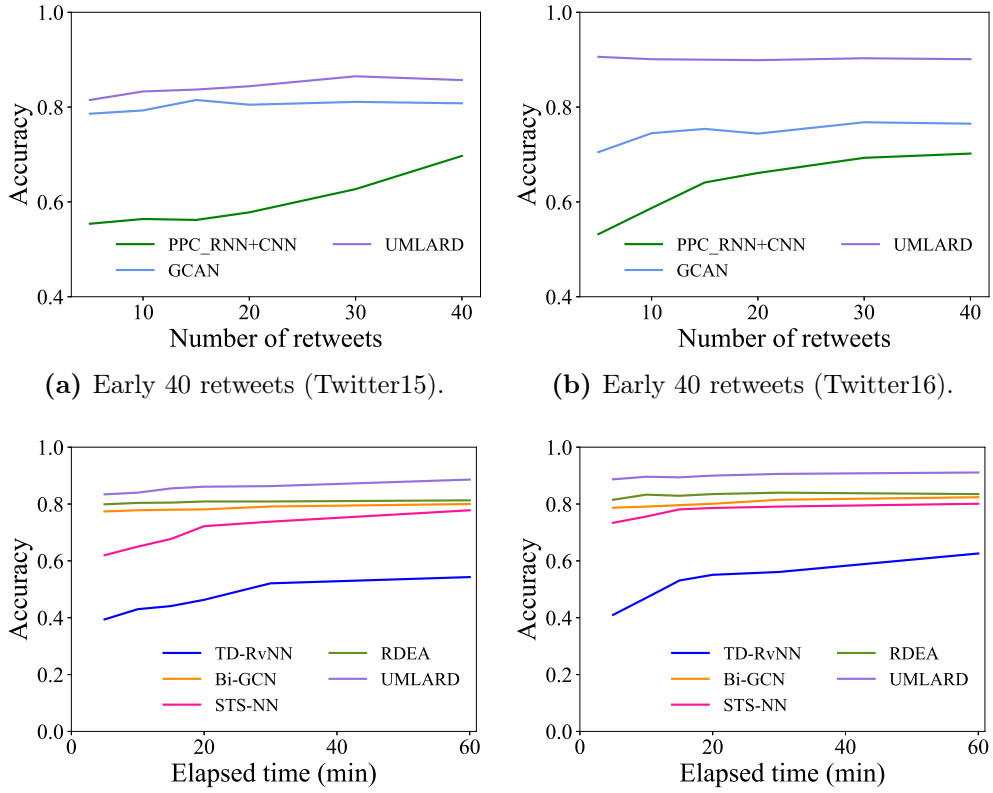
Furthermore, at the end of this section, we conduct an extra experiment to demonstrate the effectiveness of the "DropEdge" technical used in the data prepossessing. The dropout rate is set from 0 to 0.9, and the experimental results are shown in

[1]https://github.com/google-research/bert

Figure 7.21. We find that slightly dropping the edges in the diffusion graph would improve the model performance.

7.2.4.4 Performance on Early Detection (Q3)

Another important goal of rumor detection is to detect misinformation as early as possible and stop its spread in a timely fashion. Now we investigate the performance of models on identifying rumors at early-stage. Here, we consider two metrics for gauging the observation windows of information spread, i.e., the previous 40 retweets and the propagation in the first hour. In this section, the experiments of early detection are conducted on Twitter15 and Twitter16.



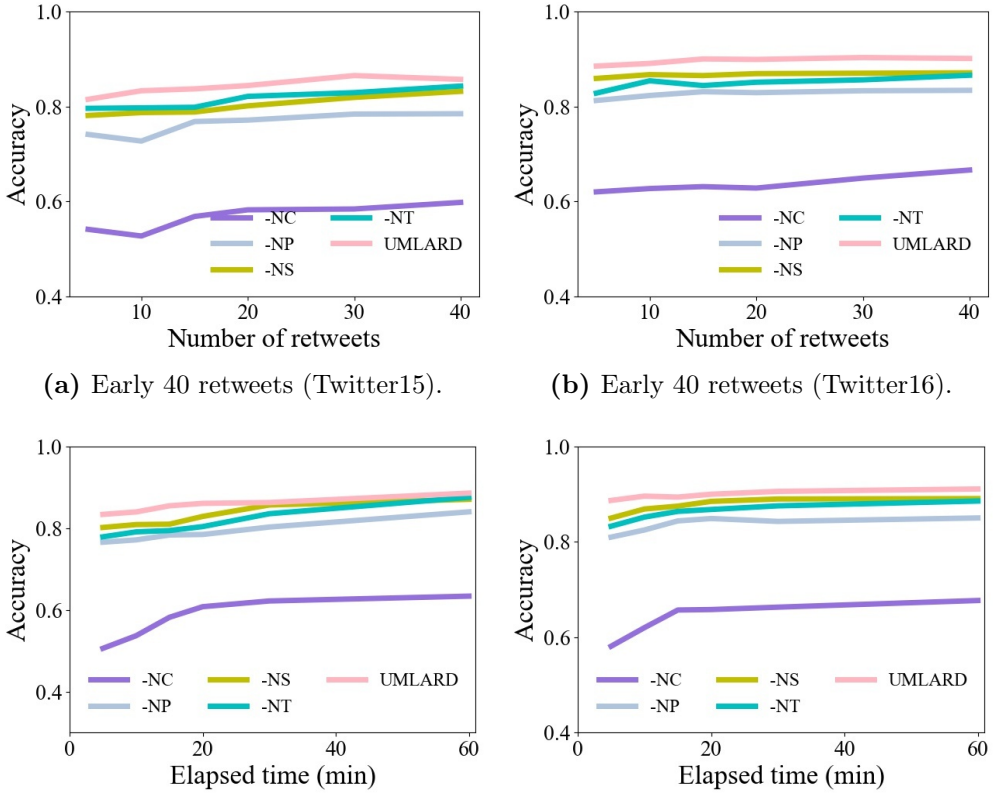
(a) Early 40 retweets (Twitter15). **(b)** Early 40 retweets (Twitter16).

(c) Early 1-hour retweets (Twitter15). **(d)** Early 1-hour retweets (Twitter16).

Figure 7.22: Evaluations on early rumor detection. (a) and (b): PPC_RNN+CNN and GCAN are cascade length-based methods. (c) and (d): Tv-RvNN, Bi-GCN, STS-NN and RDEA are built on the user comments that may not exist in early-stage retweets – hence, we observe their performance over time.

Figure 7.22 shows the performance comparison on early-stage detection between our UMLARD and the baselines. Note that we omit the feature-based methods and credibility-based approaches since they did not show comparable performance, especially on early rumor detection. We observe that UMLARD performs better, especially when there are only a few observations. UMLARD needs a short time to identify the misinformation because it fuses the multi-view knowledge of users.

For example, understanding the role of a user in spreading information is vital since tweets' size, spread speed and patterns are different. Moreover, UMLARD is capable of discriminating the importance of features even with few observations, which means the interference caused by the trivial or useless features would be dampened during training the model. In all cases, their early detection accuracy grows at the early stage of propagation. However, we find that the performance of our model demonstrates obvious advantage as time goes on.



(a) Early 40 retweets (Twitter15). **(b)** Early 40 retweets (Twitter16).

(c) Early 1-hour retweets (Twitter15). **(d)** Early 1-hour retweets (Twitter16).

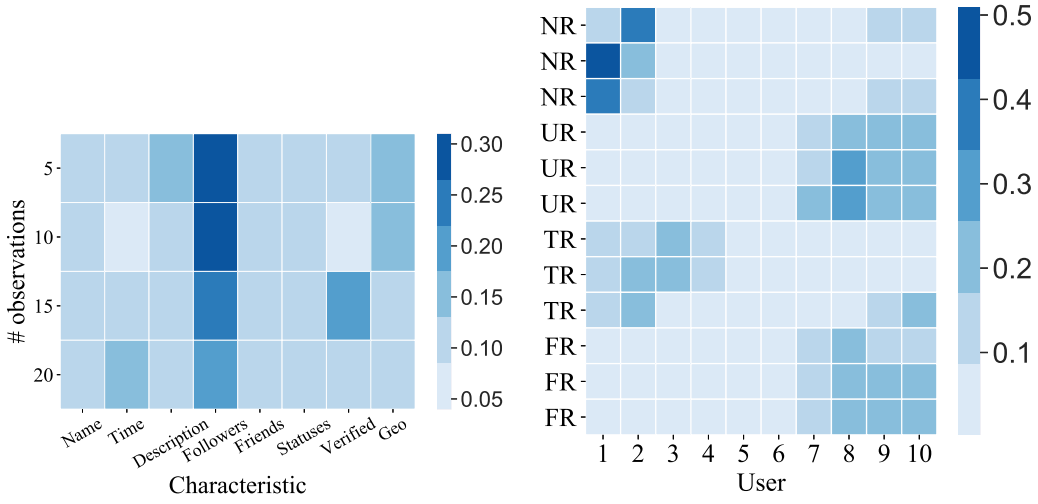
Figure 7.23: Evaluations on early rumor detection among variants of UMLARD. (a) and (b): The model performance using early 40 retweets. (c) and (d): The models are trained with early one hour observations.

We also investigate the time-varying performance between the variants and the full UMLARD. As shown in Figure 7.23, we find that the accuracy of all methods grows to saturation with increasing the number of retweets or time elapsed. Moreover, from Figure 7.23c and 7.23d, we can observe that the performance of -NP, -NT, and -NS is very close to the full UMLARD, because the models have acquired enough knowledge to detect rumors within a short observation time.

7.2.4.5 Interpretability Analysis (Q4)

The above experimental results have shown the superiority of the proposed hierarchical attentions. Namely, they can effectively discriminate the importance of

multi-views of users and the roles of users in spreading the (mis)information. Here, we provide more in-depth insights into the two components by visualizing the hierarchical attention layers in UMLARD.



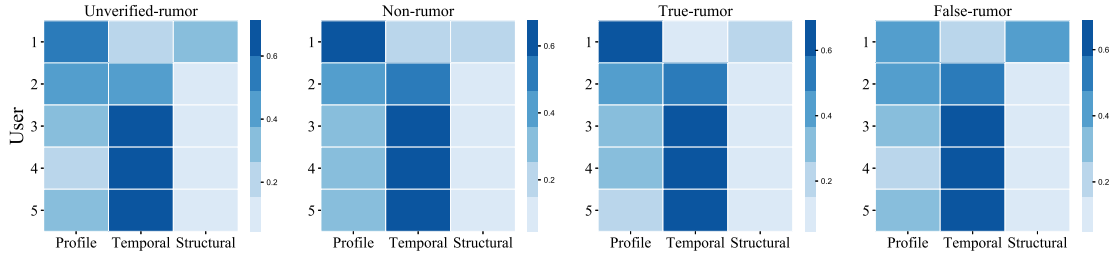
(a) User profile attention. **(b)** Capsule attention.

Figure 7.24: Visualization of user profiles importance and the role of earlier spreaders (Twitter15).

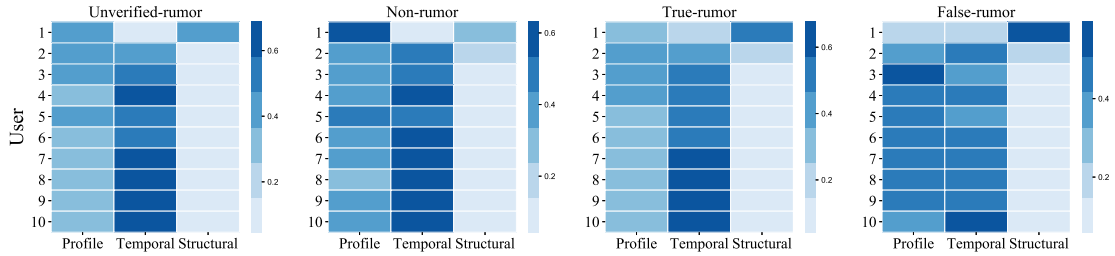
Figure 7.24 shows the importance of user-profiles and users themselves – the higher the value, the more important the feature or the user. Figure 7.24a plots the importance of eight user profile characteristics, where we vary the number of observed retweets between $\{5, 10, 15, 20\}$. We can observe that the follower counts is the most informative feature, followed by the register time, verified account, and geo-enabled features, consistent with the findings in [26, 83], i.e., the users enrolled in spreading of rumors have fewer followers.

In Figure 7.24b, we investigate the role of the retweet users at the very beginning of the cascade. As shown, the earlier users are more important for detecting *non-rumors* (NR) and *true-rumors* (TR). To the contrary, the later participators are important for detecting *unverified-rumors* (UR) and *false-rumors* (FR). This phenomenon shows that authoritative users usually spread TRs and NRs at the beginning of spreading information. URs and FRs, after the false information spread a while, will see an influx of massive malicious users, who would pretend these tweets as real information.

We now discuss the impact of the different views of users in rumor detection. We randomly selected four different types of tweets in Twitter15 and plots the importance of different views. Figure 7.25a and Figure 7.25b show the results of previous 5 and 10 retweet users, respectively. Overall, we can see that the three views of each user in these tweets have different importance. Specifically, when there are few observations (e.g., only 5 retweet users), the profile view and the temporal view of the



(a) View-wise Attention, observation length = 5.



(b) View-wise Attention, observation length = 10.

Figure 7.25: Visualization of the different user-aspect importance (Twitter15).

users dominate the rumor detection performance. As the number of retweet users increases, the structural information becomes more and more important. This result can be understood intuitively: In reality, at the very beginning the participants directly retweet the information from the source spreader, which leads to the similar propagation structures of information cascades. However, users are different from each other in profile and the time of retweeting, which are, consequently, the most important views for early-stage misinformation detection. Besides, by comparing different types of information, the non-rumor and the true-rumor have very similar weight distribution over different users' views, as observed in Figure 7.24b.

7.2.5 Summary

In this work, we presented UMLARD – a novel model for rumor detection which fuses multiple information contexts pertaining to users of social networks. Combining multiple views of users aspects and discriminating the importance of spreaders and user-aspect information, we successfully identified users' roles in different stages of rumor diffusion. UMLARD significantly outperforms previous methods in terms of misinformation classification and rapid rumor detection. Our approach is also notable in its strength of interpreting model behaviors and the predicted results. The experiments conducted on real Twitter datasets and Weibo dataset support the hypothesis that characteristics of user-profiles, aspects view of participants, as well as user's engagement time and tweets' diffusion patterns, can contribute to the misinformation prediction from the collective signals. Besides, our experimental results on early-detection discern several vital features of false information.