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

Chen, X.

Part II

Rumor Detection

Chapter 6

Modeling Hierarchical Diffusion for Rumor Detection

6.1 Chapter Overview

Conventional methods for rumor detection broadly fall into two groups: (1) *hand-crafted feature-based approaches* — mostly identifying and incorporating complicated hand-crafted features for rumor detection, including lexical features [23, 29], syntactic features [23, 79], visual features [24, 77], user profile related features [26, 83], and social relationship features [137, 138]. Their performance highly depends on the effectiveness of extracted features, which require extensive domain knowledge. (2) *credibility propagation-based approaches* [24, 31, 32], which aims to find the truth against conflicting information. These approaches usually leverage the inter-entity relations but heavily rely on the constructed credibility network for high rumor identification accuracy. Recent studies inspired by the successes of deep learning methods in many fields have developed various neural network-based models to learn several feature representations for rumor detection in an end-to-end way [16, 33, 34, 102]. Although these methods have shown performance improvements over the previous methods, they still face several critical limitations. First, most of the existing methods still require a large volume of textual data or a rich collection of users' comments as input [16, 34, 102]. In addition, previous works focused on either microscopic diffusion patterns that emphasize users' personal retweeting behavior or macroscopic diffusion structures depicting the full rumor in-network diffusion paths [16, 100].

To overcome the limitations mentioned above, in this chapter, we propose MMRD (**M**acroscopic and **M**icroscopic-aware **R**umor **D**etection), a novel deep learning-based framework for rumor detection. MMRD models the rumor diffusion from both macroscopic and microscopic perspectives through newly designed encoding components MacroE and MicroE and enhancing the diffusion representations through the cross-learning mechanism. We design a fusion gate to selectively aggregate learned

macroscopic and microscopic knowledge and introduce the attention mechanism to merge row-level information to form a unique rumor representation. The rumor prediction is generated based on the learned rumor representation. Moreover, a knowledge distillation technique is applied to further improve the model's detection performance. Our main contributions are summarized as follows:

- Firstly, we propose a new model to learn the representation of rumor through modeling the macroscopic and microscopic diffusion. The model is flexible and can be easily integrated into any existing approaches.
- Secondly, we design two encoding components for macroscopic and microscopic diffusion modeling, respectively, as well as the mechanism to control the information aggregation.
- Thirdly, MMRD employs a powerful technique–knowledge distillation to transfer knowledge from a teacher model to a student model, which further improves the model performance since the student capture more knowledge than the teacher.
- Finally, we conduct extensive evaluations on two benchmark datasets. The experimental results demonstrate that our model significantly outperforms existing baseline methods on rumor detection.

This chapter is based on the following publication[44]:

- **Chen, X.**, Zhou, F., Zhang, F., Bonsangue, M.: Modeling microscopic and macroscopic information diffusion for rumor detection. International Journal of Intelligent Systems36(2021) 5449–5471

6.2 Problem statement

We first borrow the definitions of macro-level and micro-level diffusion prediction from the field of information cascades modeling [71] to define macroscopic diffusion and microscopic rumor diffusion, and then give the formalized definition of rumor detection, which are formally defined as follows.

In information cascades modeling, the macro-level diffusion prediction aims at predicting the eventual size of a given cascade. Similarly, the macroscopic diffusion in our work refers to the evolution of the network scale, representing both the change of edges and nodes.

Definition 10 ***Macroscopic diffusion.*** *We denote the macroscopic diffusion as a diffusion graph $G = \{U, E\}$ (see Definition 2), where U is the user set comprising N users, and $E = \{(u_i, u_j)|u_i, u_j \in U\}$ represents a set of edges connecting pairs of users when u_j retweets u_i.*

Similar to micro-level diffusion prediction that aims to predict the next infected user, we define the user infected process as microscopic diffusion, i.e., who will engage in the information spreading and when this event (retweeting) occurs.

Definition 11 ***Microscopic diffusion.*** *We represent the microscopic diffusion as a user-time series* $\mathcal{P} = \{(u_1, t_1), \cdots, (u_j, t_j), \cdots, (u_N, t_N)\}$*, where* (u_1, t_1) *denotes user* u_1 *created source tweet at time* t_1*, and the rest of* (u_j, t_j) *tuples denote user* u_j *retweet the source tweet at time* t_j*. Here, all users are in chronological order according to their timestamps.* $\mathcal{P}$ *is also known as diffusion path in Definition 3*

Recall the definition of observed diffusion graph $G(t_o)$ and diffusion path $\mathcal{P}(t_o)$ in Definition 4, where t_o is the observation window, we now give a formal definition of rumor detection in this work:

Definition 12 ***Rumor detection.*** *Given a tweet* $m = \{G_m(t_o), \mathcal{P}_m(t_o)\}$ *within an observation window* t_o*, the goal of rumor detection is to learn a classification function* $f(m)$ *to classify* m *as a rumor or non-rumor.*

Others definitions, such as user vector, used in next sections, are formally given in Definition 5.

Table 6.1: Main notations used throughout this chapter.

Symbol	Description
$\mathbf{MacroE}(\cdot)$	Macroscopic diffusion encoding component.
$\mathbf{MicroE}(\cdot)$	Microscopic diffusion encoding component.
$\mathbf{H}_{Macro}$	The macroscopic diffusion representation learned by MacroE.
F^{Macro}	Dimension of macroscopic diffusion representation.
$\mathbf{H}_{Micro}$	The microscopic diffusion representation learned by MicroE.
F^{Macro}	Dimension of microscopic diffusion representation.
$\mathbf{H}_{Fuse}$	Importance-aware diffusion representation.
$\mathbf{H}_{Rumor}$	Rumor representation.
$\mathcal{K}$	A set of order powers.
K	Max order (i.e., max-value of $\mathcal{K}$).
$\hat{\mathbf{y}}, \mathbf{y}$	Prediction and the ground truth.

6.3 MMRD: Modeling Microscopic and Macroscopic Information Diffusion for Rumor Detection

The overall framework of the proposed model MMRD is shown in Figure 6.1. In particular, it consists of the following main components: (a) the input layer, which takes the observed diffusion graph $G(t_o)$ and diffusion path $\mathcal{P}(t_o)$ as inputs; (b) the normal training process, so-called the training process of a teacher model, the teacher model consists of (1) the macroscopic and microscopic diffusion encoding layer, including MacroE, MicroE, and cross-learning mechanism, (2) the fusion gate,

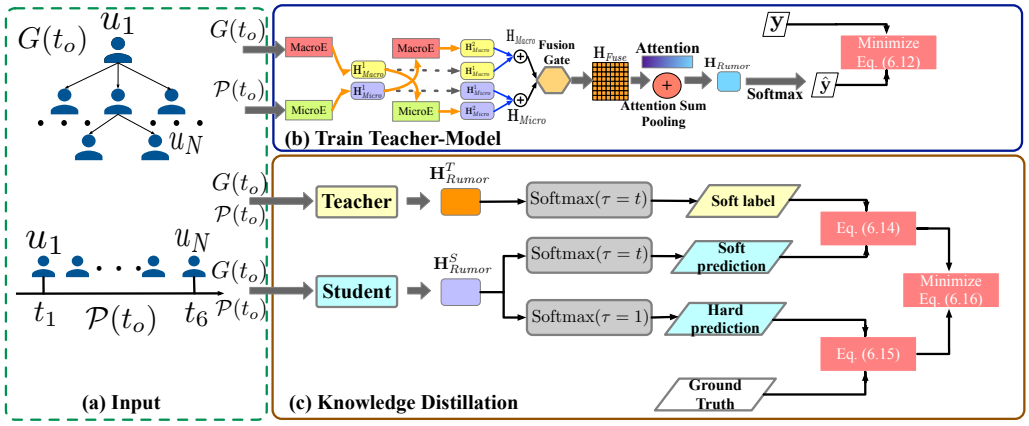
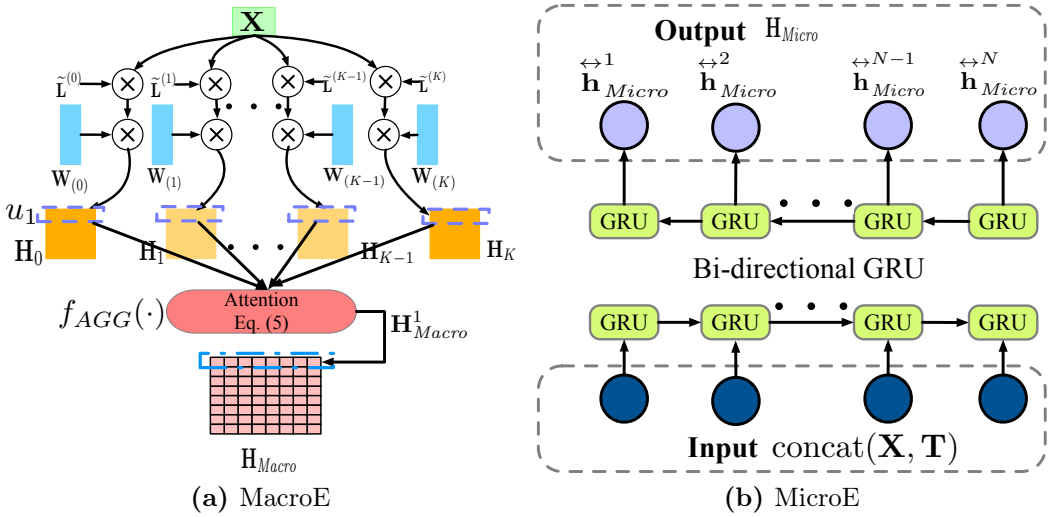


Figure 6.1: Overview of **MMRD**: (a) inputs of MMRD; (b) normal training process of MMRD; (c) the process of train MMRD with knowledge distillation.

and (3) the rumor detection layer; (c) knowledge distillation phase, which is used to further improve the model performance. With this model in mind, we first introduce two essential structural encoding components – **MacroE** and **MicroE**, and then discuss how to generate the unique rumor representation based on the two modules that will preserve both macroscopic and microscopic diffusion properties. Finally, we introduce how to use the knowledge distillation technique to develop a powerful student model for rumor detection.



(a) MacroE **(b)** MicroE

Figure 6.2: Illustration of diffusion encoding components.

6.3.1 Macroscopic Diffusion Encoding Component

The macroscopic diffusion of a tweet m reflects its diffusion scale. In our work, we cast the macroscopic diffusion modeling as learning the latent structural patterns from its diffusion graph $G_m(t_o)$. Inspired by the recent success of graph neural networks in processing the graph structural data, e.g., graph convolutional network (GCN) [41, 61] and graph attention network (GAT) [109], we implement the

macroscopic diffusion encoding component (**MacroE**) based on vanilla GCN [61]. The vanilla GCN is a multi-layer structure that contains several convolution layers, which is defined as:

$$\begin{aligned} \mathbf{H}^{(j+1)} &= \sigma\left(\widetilde{\mathbf{A}}\mathbf{H}^{(j)}\mathbf{W}^{(j)}\right) \\ \widetilde{\mathbf{A}} &= \mathbf{D}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I}_N)\mathbf{D}^{-\frac{1}{2}} \end{aligned} \tag{6.1}$$

where $\mathbf{H}^{(j)} \in \mathbb{R}^{N\times d_j}$ and $\mathbf{H}^{(j+1)} \in \mathbb{R}^{N\times d_{j+1}}$ are the input and output for layer j, $\mathbf{W}^{(j)} \in \mathbb{R}^{d_j\times d_{j+1}}$ is a trainable weight matrix and $\sigma(\cdot)$ is an activation function (e.g., Relu). $\widetilde{\mathbf{A}}$ is a symmetrically normalized adjacency matrix with self-connections, and $\mathbf{D}$ is a diagonal degree matrix. The adjacency matrix $\mathbf{A}$ and degree matrix $\mathbf{D}$ are expressed as the following:

$$\begin{aligned} \mathbf{A}_{ij} &= \begin{cases} 1 & \text{if } (u_i,u_j)\in E \text{ and } i\neq j, \\ 0 & \text{otherwise.} \end{cases} \\ \mathbf{D}_{ii} &= \sum_j \mathbf{A}_{ij} \end{aligned} \tag{6.2}$$

The initial input of the first GCN layer $\mathbf{H}^{(0)} = \mathbf{X}$, which is formed by user vectors, i.e., $\mathbf{X} = \{\mathbf{u}_1, \cdots, \mathbf{u}_N\} \in \mathbb{R}^{N\times F^{user}}$. Even the vanilla GCN shows powerful ability in graph embedding, it still faces some limitations: (1) it focuses on undirected graphs rather than the directed graph [10]; and (2) the nodes receive latent representations only from their immediate neighbors, cannot be summarized as higher-order adjacency information [111, 130].

To overcome the aforementioned limitations of GCN in modeling the directed graph and learning higher-order interactions, in this work, we reference the work of CasCN [10] (ref. Chapter 4) and MixHop [126], and extend the vanilla GCN. Finally, we propose a directed multi-hop graph convolutional network with attention aggregation as the MacroE (Figure 6.2a). The convolutional kernel of MacroE is defined as:

$$\begin{aligned} \mathbf{H}_{Macro} &= f_{AGG}\left[\sigma(\widetilde{\mathbf{L}}^{(k)}\mathbf{X}\mathbf{W}_{(k)})_{k\in\mathcal{K}}\right] \\ &= \sigma(f_{AGG}\left[\widetilde{\mathbf{L}}^{(0)}\mathbf{X}\mathbf{W}_{(0)} \vdots \widetilde{\mathbf{L}}^{(1)}\mathbf{X}\mathbf{W}_{(1)} \vdots \cdots \vdots \widetilde{\mathbf{L}}^{(K)}\mathbf{X}\mathbf{W}_{(K)}\right]) \\ &= \sigma(f_{AGG}[\mathbf{H}_{(0)} \vdots \mathbf{H}_{(1)} \vdots \cdots \vdots \mathbf{H}_{(K)}]) \end{aligned} \tag{6.3}$$

where $\mathbf{X} \in \mathbb{R}^{N\times F}$ is the input feature matrix and $\mathbf{H}_{Macro} = \{\mathbf{h}^1_{Macro}, \cdots, \mathbf{h}^N_{Macro}\} \in \mathbb{R}^{N\times F^{Macro}}$ is the output of MacroE. In order to capture the directional information from the diffusion graph, we replace the $\widetilde{\mathbf{A}}$ with $\widetilde{\mathbf{L}}$ – normalized Laplacian for directed

graph. The calculation of $\widetilde{\mathbf{L}}$ is defined as:

$$\begin{aligned}\mathbf{P} &= (1-\alpha)\frac{\mathbf{E}}{N} + \alpha\left(\mathbf{D}^{-1}\mathbf{A}\right),\\ \mathbf{L} &= \Phi^{\frac{1}{2}}\left(\mathbf{I}-\mathbf{P}\right)\Phi^{-\frac{1}{2}},\\ \widetilde{\mathbf{L}} &= \frac{2}{\lambda_{max}}\mathbf{L}-\mathbf{I}\end{aligned} \tag{6.4}$$

where $\mathbf{P}$ is a transition probability matrix, $\mathbf{E}\in\mathbb{R}^{N\times N}$ is an all–one matrix. $\alpha\in(0,1)$ is an initial probability used to restrict the state transition matrix $\mathbf{D}^{-1}\mathbf{A}$ a strongly connected matrix [10]. Φ is a diagonal matrix with entries $\Phi(v,v)=\phi(v)-\phi(v)$ is the column vector of $\mathbf{P}$ [117], and λ_{max} denotes the largest eigenvalue of $\mathbf{L}$.

$\mathcal{K}$ is a set of integer order powers – the value of $\mathcal{K}$ is from 0 to K, and $\mathbf{W}_k\in\mathbb{R}^{F\times F^{Macro}}$ is the weight matrix for k-hops. $\widetilde{\mathbf{L}}^{(k)}$ denotes the normalized Laplacian matrix $\widetilde{\mathbf{L}}$ multiplied by itself k times, and its value represents the probability connecting path from vertex u_i to vertex u_j in k-hops. Specifically, $\widetilde{\mathbf{L}}^{(0)}=\mathbf{I}$ is an identity matrix. Through the Laplacian matrix's multiple powers, MacroE mixes the feature representation of higher-order neighbors in one graph convolutional layer.

$f_{AGG}(\cdot)$ is an aggregation function, which is used to fuse the latent representation from different orders. In most of the existing works [112, 126], $f_{AGG}(\cdot)$ is similar to the pooling methods in CNNs, which can be a mean-pooling function, max-pooling function, or sum-pooling function. However, the distance of the message passing for each node is different, i.e., different nodes have different max-orders. In this work, we implement the aggregation function via the order attention mechanism at the node-level. As for each node u_j in the diffusion graph $G_m(t_o)$, it has a set of latent representations $\mathbf{h}^{u_j}_{(0)},\cdots,\mathbf{h}^{u_j}_{(K)}$ from K-orders. Then the order attention of u_j is calculated as:

$$\begin{aligned}a^{u_j}_{(k)} &= \frac{\exp(\langle\mathbf{w}_{u_j},\tanh(\mathbf{W}_{u_j}\mathbf{h}^{u_j}_{(k)}+\mathbf{b}_{u_j}\rangle))}{\sum_{*=1}^{K}\exp(\langle\mathbf{w}_{u_j},\tanh(\mathbf{W}_{u_j}\mathbf{h}^{u_j}_{(*)}+\mathbf{b}_{u_j}\rangle))},\\ \mathbf{h}^{u_j}_{Macro} &= \sum_{k=1}^{K}a^{u_j}_{(k)}\mathbf{h}^{u_j}_{(k)}\end{aligned} \tag{6.5}$$

where $\mathbf{W}_{u_j}\in\mathbb{R}^{F^{Macro}\times d}$, $\mathbf{b}_{u_j}\in\mathbb{R}^d$ and $\mathbf{w}_{u_j}\in\mathbb{R}^d$. So that, the aggregation function f_{AGG} is formulated as $f_{AGG}=\{\mathbf{h}^{u_j}_{Macro}=\text{Attention}(\mathbf{h}^{u_j}_{(0)},\cdots,\mathbf{h}^{u_j}_{(K)}),|j\in\{1,\cdots,N\}\}$. The calculation process of MacroE is outlined in Algorithm 7.

MacroE vs. GCN: As depicted above, the convolutional kernel in MacroE for one single order is similar to a single layer of GCN, i.e., $\widetilde{\mathbf{L}}^{(k)}\mathbf{X}\mathbf{W}_{(k)}$ and $\widetilde{\mathbf{A}}\mathbf{H}^{(j)}\mathbf{W}^{(j)}$, respectively. The main differences between our MacroE and GCN are: (1) we use normalized directed Laplacian $\widetilde{\mathbf{L}}$ to replace the symmetrically normalized adjacency matrix $\widetilde{\mathbf{A}}$ in GCN, which introduces the directional information of edges into the

Algorithm 7: Calculation of MacroE (Eq. (6.5)).

Input: Feature matrix $\mathbf{X}$, normalized laplacian matrix $\widetilde{\mathbf{L}}$, a set of order powers $\mathcal{K}$ and its max-value $K = \max(\mathcal{K})$.
Parameters: $\{\mathbf{W}_{(k)}\}_{k\in\mathcal{K}}$.
1: $\mathbf{B} := \mathbf{X}$
2: **for** $k = 0$ to K **do**
3: **if** $k = 0$ **then**
4: $\mathbf{B} := \mathbf{IB}$
5: **else**
6: $\mathbf{B} := \widetilde{\mathbf{L}}\mathbf{B}$
7: **end if**
8: $\mathbf{H}_{(k)} := \mathbf{B}\mathbf{W}_{(k)}$
9: **end for**
/*From step 1 to 9, complete the calculation of $(\widetilde{\mathbf{L}}^{(k)}\mathbf{X}\mathbf{W}_{(k)})_{k\in\mathcal{K}}$ in Eq. (6.3)*/
10: $\mathbf{H}_{Macro} := f_{AGG}([\mathbf{H}_{(0)}, \cdots, \mathbf{H}_{(k)}, \cdots, \mathbf{H}_{(K)}])$ via Eq. (6.5).
11: **return** $\sigma(\mathbf{H})$

convolution rather than only considering the link information between nodes; and (2) our MacroE can learn high-order information for each node by using one single layer; however, GCN relies on multi-layers and may introduce the over-smoothing issue in learning node feature representations [112].

6.3.2 Microscopic Diffusion Encoding Component

The microscopic diffusion encoding component (**MicroE**, shown in Figure 6.2b) aims to capture temporal patterns from the user engagement time series $\mathcal{P}_m(t_o)$. Inspired by the success of RNNs in sequential modeling, we employ a Bi-directional GRU (Bi-GRU) [108] as the encoding component, where the hidden states are used to memorize the diffusion history. At each step t_j, Bi-GRU takes the feature vector and previous hidden state as inputs and computes the updated hidden state as:

$$\overset{\leftrightarrow}{\mathbf{h}}^{j} = \text{Bi-GRU}(\mathbf{x}^j, \mathbf{h}^{j-1}), \overset{\leftrightarrow}{\mathbf{h}}^{j} \in \mathbb{R}^{F^{Micro}} \tag{6.6}$$

Then, the output of MicroE module is a sequence of hidden states $\mathbf{H}_{Micro} = \{\overset{\leftrightarrow}{\mathbf{h}}^{j}_{Micro} | j \in \{1, \cdots, N\}\} \in \mathbb{R}^{N\times F^{Micro}}$

6.3.3 Macroscopic and Microscopic Cross-learning

After introducing the necessary encoding components, we go to describe how to apply them to learn the latent representations from macroscopic and microscopic diffusion, summarized into two steps. In the first step, we train MacroE and MicroE separately. MacroE takes the feature matrix $\mathbf{X}$, the normalized Laplacian matrix $\widetilde{\mathbf{L}}$ and max-order number K as inputs. As for MicroE, we first represent the infected

timestamp of each user into one-hot vector $\mathbf{t}_j \in \mathbb{R}^{d_{time}}$, and then concatenate the timestamp vector matrix $\mathbf{T} = \{\mathbf{t}_1, \cdots, \mathbf{t}_N\}$ with $\mathbf{X}$ to form the input $\hat{\mathbf{X}}$ for MicroE. Specifically, assume that, the time window is $[t_1, t_o]$, and we first split the time window into l disjoint time intervals, and then compute the corresponding time interval for each retweet user u_j as $t^j_{int} = \left\lfloor \frac{t_j - t_1}{t_o/l} \right\rfloor$, where t_1 is the timestamp for the source post user, and t_j is timestamp for user u_j. Finally, each user's timestamp is falling into corresponding time intervals and each interval is related to a one-hot embedding, thus, for u_j its timestamp embedding equals to the related time-interval embedding. Note that, in our work, the initial feature matrix $\mathbf{X}$ is extracted from users' profiles. Figure 6.3 shows a toy example of the model inputs. The outputs of first step are $\mathbf{H}^1_{Macro}$ and $\mathbf{H}^1_{Micro}$, respectively:

$$\begin{aligned} \mathbf{H}^1_{Macro} &= \text{MacroE}(\mathbf{X}, \widetilde{\mathbf{L}}, K) \\ \mathbf{H}^1_{Micro} &= \text{MicroE}(\text{concat}(\mathbf{X}, \mathbf{T})) \end{aligned} \tag{6.7}$$

In the second step, we train MacroE and MicroE in a cross-learning manner. Specifically, we use $\mathbf{H}^1_{Macro}$ to train a new MicroE, and vice versa. The outputs of second step are $\mathbf{H}^2_{Macro}$ and $\mathbf{H}^2_{Micro}$:

$$\begin{aligned} \mathbf{H}^2_{Macro} &= \text{MacroE}(\mathbf{H}^1_{Micro}, \widetilde{\mathbf{L}}, K) \\ \mathbf{H}^2_{Micro} &= \text{MicroE}(\mathbf{H}^1_{Macro}) \end{aligned} \tag{6.8}$$

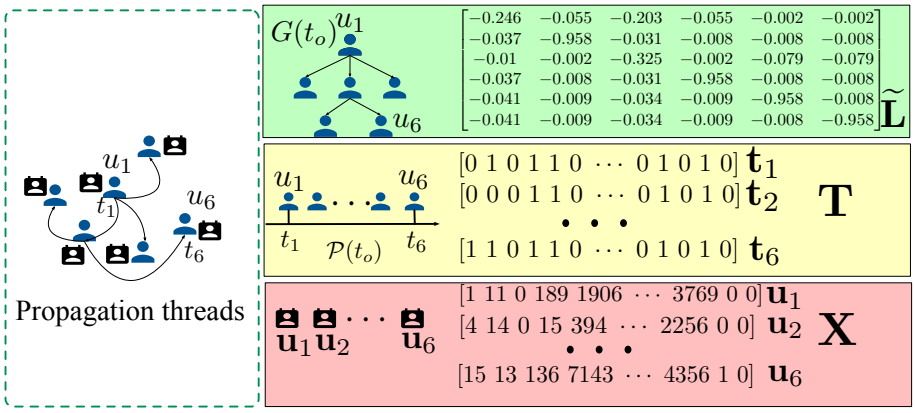


Figure 6.3: A toy example of the model inputs.

6.3.4 Feature fusion via hybrid aggregation layer

We concatenate $\mathbf{H}^1_{Macro}$ with $\mathbf{H}^2_{Macro}$ to form $\mathbf{H}_{Macro} \in \mathbb{R}^{N \times 2F^{Macro}}$, and concatenate $\mathbf{H}^1_{Micro}$ with $\mathbf{H}^2_{Micro}$ to form $\mathbf{H}_{Micro} \in \mathbb{R}^{N \times 2F^{Micro}}$. Thus, for each tweet m_i, we have a macroscopic representation $\mathbf{H}_{Macro}$ and a microscopic representation $\mathbf{H}_{Micro}$. In most of the existing works, after getting $\mathbf{H}_{Macro}$ and $\mathbf{H}_{Micro}$, they will concatenate them directly, however, this operation ignores the different dependence on the two different representations. In our work, in order to effectively aggregate the learned representations, inspired by the gate mechanism [108] and attention mechanism [131], we

design (1) a fusion gate to fuse $\mathbf{H}_{Macro}$ and $\mathbf{H}_{Micro}$ to form $\mathbf{H}_{Fuse}$, and (2) row-level attention to aggregate features to merge a unique representation $\mathbf{H}_{Rumor}$.

To selectively integrate the important information of two representations, we employ a concise and effective fusion gating mechanism that produces an importance-aware diffusion representation $\mathbf{H}_{\text{Fuse}}$ as follows:

$$\begin{aligned} \mathbf{G} &= \text{sigmoid}(\mathbf{W}^1_{gate}\mathbf{H}_{Macro} + \mathbf{W}^2_{gate}\mathbf{H}_{Micro} + \mathbf{b}_{gate}) \\ \mathbf{H}_{Fuse} &= \mathbf{G} \odot \mathbf{H}_{Macro} + (1 - \mathbf{G}) \odot \mathbf{H}_{Micro} \end{aligned} \tag{6.9}$$

where $\mathbf{W}^1_{gate}, \mathbf{W}^2_{gate} \in \mathbb{R}^{2F' \times 2F'}$, and $\mathbf{b}_{gate} \in \mathbb{R}^{2F'}$. Note that, here $F' = F^{Macro} = F^{Micro}$. $\mathbf{G}$ is used to drop trivial parts of macroscopic representation and add important information from microscopic representation. The rationale behind this design is that the representation fusion $\mathbf{H}_{Fuse} = \{\mathbf{h}^1_{\text{Fuse}}, \cdots, \mathbf{h}^N_{\text{Fuse}}\} \in \mathbb{R}^{N \times 2F'}$ would be aware of the different importance of macroscopic and microscopic diffusion.

Then, we merge the row-level information of $\mathbf{H}_{Fuse}$ to form an unique representation $\mathbf{H}_{Rumor}$ for tweet m through attention sum-pooling operation:

$$\begin{aligned} a_j &= \frac{\exp(\langle \mathbf{w}, \tanh(\mathbf{W}_a \mathbf{h}^j_{\text{Fuse}} + \mathbf{b}_a \rangle))}{\sum_{*=1}^{N} \exp(\langle \mathbf{w}, \tanh(\mathbf{W}_a \mathbf{h}^*_{\text{Fuse}} + \mathbf{b}_a \rangle))}, \\ \mathbf{H}_{Rumor} &= \sum_{j=1}^{N} a_j \mathbf{h}^j_{Fuse} \end{aligned} \tag{6.10}$$

where $\mathbf{W}_a \in \mathbb{R}^{2F' \times d}$, $\mathbf{b}_a \in \mathbb{R}^d$ and $\mathbf{w} \in \mathbb{R}^d$.

6.3.5 Rumor detection and optimization

Subsequently, $\mathbf{H}_{Rumor}$ is used to generate the corresponding binary prediction vector $\hat{\mathbf{y}} = [\hat{y}_0, \hat{y}_1]$, where $\hat{y}_0, \hat{y}_1$ indicate that the prediction probabilities of the label being 0 and 1, respectively, via a fully connected layer and the Softmax function:

$$\hat{\mathbf{y}} = \text{Softmax}\left(\text{FC}\left(\mathbf{H}_{Rumor}\right)\right). \tag{6.11}$$

In our implementation, we train all the model parameters by minimizing the *cross-entropy* between $\hat{\mathbf{y}}$ and $\mathbf{y}$:

$$\mathcal{L} = -\frac{1}{|B|} \sum_{i=1}^{|B|} \sum_{c=0}^{1} y_{i,c} \log \hat{y}_{i,c} + \lambda \left\| \Theta \right\|_2^2, \tag{6.12}$$

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. $\|\Theta\|_2^2$ is the L_2 regularizer over all the model parameters Θ, and λ is the trade-off coefficient. The optimization can be solved by stochastic gradient

descent–based optimization approaches, such as Adam [139] and RAdam [140]. The above computation process of our MMRD model is outlined in Algorithm 8.

Algorithm 8: Training process of MMRD.

Input: A set of tweets $M = \{m_i\}_{i=1}^{|M|}$, each tweet $m_i = \{G_i(t_o), \mathcal{P}_i(t_o)\}$, the max-order number K.
Output: MMRD-optimized parameters Θ.

initialize Θ
while Θ has not converged **do**
  for each tweets batch B **do**
    for each tweet m_i in tweets batch B **do**
      /* (1st) Train MacroE and MicroE separately. */
      1st macroscopic diffusion encoding: $\mathbf{H}^1_{Macro} \leftarrow \mathbf{X}, \widetilde{\mathbf{L}}, K$ via Eq.(6.3);
      1st microscopic diffusion encoding: $\mathbf{H}^1_{Micro} \leftarrow \hat{\mathbf{X}}$ via Eq.(6.6);
      /* (2nd) Cross-learning MacroE and MicroE */
      2nd macroscopic diffusion encoding: $\mathbf{H}^2_{Macro} \leftarrow \mathbf{H}^1_{Micro}, \widetilde{\mathbf{L}}, K$ via Eq.(6.3);
      2nd microscopic diffusion encoding: $\mathbf{H}^2_{Micro} \leftarrow \mathbf{H}^1_{Macro}$ via Eq.(6.6);
      /* Concatenate operation */
      $\mathbf{H}_{Macro} = \text{concat}(\mathbf{H}^1_{Macro}, \mathbf{H}^2_{Macro})$
      $\mathbf{H}_{Micro} = \text{concat}(\mathbf{H}^1_{Micro}, \mathbf{H}^2_{Micro})$
      macroscopic and microscopic representation fusion:
      $\mathbf{H}_{Fuse} \leftarrow \mathbf{H}_{Macro}, \mathbf{H}_{Micro}$ via Eq.(6.9);
      Attention sum-polling: $\mathbf{H}_{Rumor} \leftarrow \mathbf{H}_{Fuse}$ via Eq.(6.10);
      Estimate the probability $\hat{\mathbf{y}}$ via Eq.(6.11);
    end for
    $\mathcal{L} \leftarrow$ Eq.(6.12);
    $\Theta \leftarrow RAdam(\mathcal{L})$
  end for
end while

6.3.6 Rumor detection with knowledge distilling

In order to further improve the model performance on rumor detection task, we inspired by knowledge distillation technique [141]. The knowledge distillation technique, which involves capturing the "dark knowledge" from a teacher model to guide the learning of a student network, has emerged as an essential technique for model improvement. We first train a teacher model via Algorithm 8, and then transfer the knowledge from the teacher model to a student model. Here in our work, the student model has the same model architecture as the teacher model (self-distillation [142, 143]). Before introducing the concrete training procedure of MMRD with knowledge distillation, we first give the definition of the softmax with temperature:

$$q_i = \text{softmax}(\mathbf{H}, \tau) = \frac{exp(\mathbf{H}/\tau)}{\sum_j exp(\mathbf{H}/\tau)} \tag{6.13}$$

where τ is a temperature that is normally set to 1. We use a higher value for temperature τ to produce a softer probability distribution over the class, which brings the advantage that the information carried by the negative label will be relatively amplified, and the model training will pay more attention to the negative label.

The concrete training procedure of the knowledge distillation is listed in Algorithm 9, and Figure 6.1–(c) gives a visualization of Algorithm 9. The objective function of the knowledge distillation is a weighted average of two different objective functions. The first loss function is the cross-entropy with the soft targets and it is computed using the same high temperature $\tau = t$ in the softmax of the student model as was used for generating the soft targets from the teacher model.

$$\mathcal{L}_{soft} = -\sum_{i=1}^{|B|} \bar{\mathbf{y}}_i^T \log \bar{\mathbf{y}}_i^S \tag{6.14}$$

where $\bar{\mathbf{y}}_i^T = \text{softmax}(\text{FC}(\mathbf{H}_{Rumor}^T), \tau = t)$ is soft output from teacher model, and $\bar{\mathbf{y}}_i^S = \text{softmax}(\text{FC}(\mathbf{H}_{Rumor}^S), \tau = t)$ is soft output from student model.

The second loss function is the cross-entropy with the ground truth. This is computed using exactly the same logits in softmax of the student model but at a temperature of 1.

$$\mathcal{L}_{hard} = -\sum_{i=1}^{|B|} \mathbf{y}_i \log \hat{y}_i^S \tag{6.15}$$

where $\mathbf{y}_i$ is the ground truth and $\hat{y}_i^S = \text{softmax}(\text{FC}(\mathbf{H}_{Rumor}^S), \tau = 1)$ is the hard output of student model. Finally, the objective function of knowledge distillation is:

$$\mathcal{L}_{KD} = (1 - \beta)\mathcal{L}_{soft} + \beta\mathcal{L}_{hard} \tag{6.16}$$

where β is the balance weight, which always been a considerably lower value since the amplitude of the gradients produced by the scale of the soft output as $1/\tau^2$. This ensures that the relative contributions of the hard and soft targets remain roughly unchanged [141].

6.4 Evaluating MMRD

In this section present the findings from our experimental evaluations. We compare the performance of our MMRD with the state-of-art baselines on rumor detection, and we also investigate the effects of different components by comparing several variants of MMRD. Specifically, we aim at providing empirical evaluations to answer the following research questions:

- **Q1** How does MMRD perform compared with the state-of-the-art baselines on rumor detection?

Algorithm 9: Training procedure of MMRD with knowledge distillation.

Input: A set of tweets $M = \{m_i\}_{i=1}^{|M|}$, each tweet $m_i = \{G_i(t_o), \mathcal{P}_i(t_o)\}$, the max-order number K, temperature τ.

Output: **Student**-optimized parameters Θ.

Pre-train a **Teacher** model via Alogrithm 8.
initialize Θ in **Student** model.
while Θ has not converged **do**
  for each tweets batch B **do**
    for each tweet m_i in tweets batch B **do**
      $\mathbf{H}_{Rumor}^{T} \leftarrow$**Teacher**
      /* Train **Student** model via step 1 to step 9 in Alogrithm 8.*/
      $\mathbf{H}_{Rumor}^{S} \leftarrow$ **Student**
      /* Soft outputs from **Teacher***/
      $\bar{\mathbf{y}}_i^T = \text{softmax}(\text{FC}(\mathbf{H}_{Rumor}^{T}), \tau = t)$
      /* Soft outputs from **Student***/
      $\bar{\mathbf{y}}_i^S = \text{softmax}(\text{FC}(\mathbf{H}_{Rumor}^{S}), \tau = t)$
      /* Hard outputs from **Student** */
      $\hat{y}_i^S = \text{softmax}(\text{FC}(\mathbf{H}_{Rumor}^{S}), \tau = 1)$
      calculate loss $\mathcal{L}_{KD}$ via Eq. 6.16
    end for
    $\Theta \leftarrow \text{RAdam}(\mathcal{L}_{KD})$
  end for
end while

- **Q2** How does each component of MMRD contribute to the performance?
- **Q3** Can MMRD detect rumor at an early stage?

6.4.1 Datasets

We conducted our experiments on two real-world datasets: Twitter15, Twitter16. Both Twitter15 and Twitter16 datasets were collected by Ma et al. [33]. Each dataset contains a collection of source tweets with its corresponding propagation threads. The original datasets were constructed for multi-class classification, and we removed the tweets labeled as "unverified" or "true rumor" since they were beyond our research interest, and only keep "non-rumor" and "false-rumor" labels as ground truth in both datasets. We built the macroscopic diffusion graph and microscopic diffusion path for each source tweet from its propagation threads. The statistics of the datasets are presented in Table 6.2. The user profiles were crawled via Twitter API based on the provided user IDs, and for a fair comparison, we follow PPC_RNN+CNN [100] that extracts eight types of characteristics, including, (1) length of a user name; (2) created time of a user account; (3) length of description; (4) followers count; (5) friends count; (6) statuses count; (7) whether the user is verified; and (8) whether the geographical information is enabled.

Table 6.2: Statistics of the datasets.

Statistic	Twitter15	Twitter16
# source tweets	739	404
# non-rumor	370	199
# rumor	369	205
# users	306,402	168,659
Max. # retweets	2,990	999
Min. # retweets	97	100
Avg. # retweets	493	479
Avg. # time length	743 Hours	167 Hours

6.4.2 Baselines

We compare our model with a series of state-of-the-art baseline approaches for rumor detection:

- **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.

- **SVM-RBF** [86]: A SVM-based model that uses radius basis function (RBF) as the kernel, and leverages the handcrafted features of posts for rumor detection.
- **GRU** [33]: An RNN-based model, which learns temporal patterns and content features from user comments for rumor detection.
- **TD-RvNN** [34]: A top-down tree-structured RNN model that explores the importance of propagation structure for rumor detection.
- **PPC_RNN+CNN** [100]: A model combines RNN and CNN for early rumor detection, which learns the rumor representations through the characteristics of users.
- **Bi-GCN** [16]: A GCN-based model exploring rumor dissemination through bi-directional propagation structures and text contents for rumor detection.
- **GCAN** [102]: A state-of-the-art co-attention network for rumor detection, which learns the rumor representation based on the tweets content and the corresponding retweet users.

6.4.3 Parameter Settings and Evaluation Metrics

We implement DTC with Weka[1], SVM-TS and SVM-RBF with scikit–learn[2], and other deep learning-based baselines and our MMRD with Tensorflow[3]. The hyperparameters of baselines are the same as the settings described in the original papers.

Note that, in our work, MMRD only takes the user profiles and timestamps as inputs, and ignores the content features, such as source tweet text and comments, for a fair comparison, we implement some variants for the baselines by changing the initial inputs. Specifically, as for TD-RvNN and Bi-GCN, we use user profile features to replace the comment features, and the variants of these two baselines are denoted as TD-RvNN$_{(\text{User})}$ and Bi-GCN$_{(\text{User})}$, respectively. As for GCAN, we remove the source tweet features in the original inputs which termed as GCAN-Text.

The main hyperparameters in our MMRD are tuned as follows. The batch size is 32. The output dimension of MacroE $F^{Macro} = 64$, and the hidden sizes of both the forward GRU and backward GRU units are $F^{Micro} = 32$. The max-order number K is 3. The number of time intervals l is 100 and the embedding size for each timestamp vector d_{time} is 50. The learning rate for both the teacher training phase and knowledge distillation is 0.001 and the balance weight β in distillation is 0.3. The temperature τ in knowledge distillation is 2.5. The training process is iterated upon for 500 epochs but would be stopped earlier if the validation loss does not

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

decrease after 10 epochs. And we randomly choose 70% data for training and the rest of 10% and 20% for validation and testing. In this work, we measured the detection deadline by the number of retweets, i.e., the first k-th retweets. In the overall performance, the baselines and our MMRD consider the first 40-th retweets. We choose accuracy (ACC), precision (Pre), recall (Rec) and F-score (F1) as the evaluation protocols to measure the models' performance in this work.

Table 6.3: Overall performance comparison of rumor detection on Twitter15. The best method is shown in **bold**, and the second best one is underlined.

	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
SVM-RBF	0.535	0.552	0.521	0.536
GRU	0.580	0.544	0.545	0.544
TD-RvNN	0.628	0.594	0.616	0.604
PPC_RNN+CNN	0.691	0.674	0.686	0.679
Bi-GCN	0.748	0.731	0.759	0.745
GCAN	0.835	0.825	0.829	0.825
TD-RvNN$_{(User)}$	0.678	0.671	0.674	0.672
Bi-GCN$_{(User)}$	0.820	0.846	0.824	0.834
GCAN-Text	0.683	0.705	0.652	0.678
MMRD	**0.922**	**0.922**	**0.923**	**0.922**
Improvement	**10.41%**	**11.76%**	**11.34%**	**11.76%**

Table 6.4: Overall performance comparison of rumor detection on Twitter16. The best method is shown in **bold**, and the second best one is underlined.

	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
SVM-RBF	0.711	0.724	0.709	0.716
GRU	0.554	0.514	0.516	0.515
TD-RvNN	0.633	0.619	0.610	0.614
PPC_RNN+CNN	0.655	0.632	0.651	0.641
Bi-GCN	0.711	0.709	0.710	0.716
GCAN	0.823	0.803	0.841	0.822
TD-RvNN$_{(User)}$	0.661	0.632	0.641	0.636
Bi-GCN$_{(User)}$	0.814	0.815	0.816	0.816
GCAN-Text	0.664	0.716	0.579	0.648
MMRD	**0.876**	**0.877**	**0.874**	**0.875**
Improvement	**6.44%**	**9.22%**	**3.92%**	**6.45%**

6.4.4 Overall performance (Q1)

The overall performance is shown in Table 6.3 and 6.4, from which we can find that our MMRD model consistently outperforms all baselines on both Twitter15 and Twitter16 datasets. In addition to the overall superiority of our model, we have the following observations.

First, compared to the deep learning-based methods, feature-based methods such as DTC, SVM-TS, and SVM-RBF are not competitive because their performance heavily depends on the hand-crafted features. However, designing effective features is time-consuming and requires extensive field-specific knowledge. Furthermore, the performance gain of SVM-TS over DTC lies in its capability of considering time information. On the other hand, SVM-RBF performs slightly better than SVM-TS, suggesting that the kernel-based SVM is better than linear SVM but is still limited to the quality of hand-crafted features.

Second, among all the deep learning-based baselines, GRU, as the early deep learning-based work for rumor detection, performs the worst, primarily because it only relies on temporal-linguistics of the repost sequence but ignores other informative signals such as diffusion structures and user profiles. In addition, both TD-RvNN and Bi-GCN explore the dissemination of rumors on the basis of GRU and learn textual information from replies (i.e., the retweets with comments). However, their performance is not competitive when there are few comments or replies. Bi-GCN generally performs well than TD-RvNN, demonstrating that GCN is a powerful graph learning model compared with tree structure RNN. PPC_RNN+CNN performs relatively well than GRU and TD-RvNN, implying that user-profile information is more informative than text information in rumor detection, the reason is that compared with the replies, in reality, there exist more retweets without any comments, however, the user information of such retweets is acquirable. The same observations can find when compare Bi-GCN with its variant Bi-GCN$_{(\text{User})}$ and TD-RvNN with TD-RvNN$_{(\text{User})}$.

On the other hand, GCAN takes both text information and user-profile information as input and indeed outperforms other baselines. By comparing GCAN with its variant GCAN-Text, we can find that the performance of GCAN still heavily relies on text information. This is because it models the structural information from the user similarity matrix rather than the retweet network, which may be insufficient in capturing user interactions, and due to the two datasets were existed for a long time, when we try to crawl the user profiles for all users in the datasets, we find that some user accounts do not exist anymore, and it causes difficulties in constructing user similarity graph. Besides that, compare GCAN-Text with Bi-GCN$_{(\text{User})}$, the results of Bi-GCN$_{(\text{User})}$ far exceed GCAN-Text, this observation illustrates the diffusion graph is more powerful than user similarity graph in detecting rumor when ignoring the textual features. To further illustrate that our MMRD indeed significantly outperforms the baselines, we conduct a McNemar's test [144] between our MMRD

and the best baseline (GCAN) based on the prediction results on the testing set, and the p-values are 0.001 and 0.013 on Twitter15 and Twitter16, respectively. As $p < 0.05$ on both Twitter15 and Twitter16, we can conclude that MMRD significantly outperforms GCAN.

Our MMRD, in contrast, learns rumor representation from macroscopic and microscopic diffusion without any textual information, suggesting the possibility of detecting rumors by completely exploiting rumors' diffusion patterns. However, the performance of MMRD can be further improved by taking into account other information such as textual information.

6.4.5 Ablation study (Q2)

In order to answer the RQ2, we conduct several ablation studies from the following perspectives: (1) we first propose five variants of MMRD and compare their performance on both Twitter15 and Twitter16; then, (2) we compare the performance of MMRD in without knowledge distillation and cross distillation settings; finally, (3) we pick up two special parameters to test the model's accuracy change brings by them when changing their value.

6.4.5.1 Variants comparison

We conducted an ablation study to explore each component's effect in MMRD by removing a particular component from the original MMRD. Towards that, we derive the following variants of MMRD:

- **-AGG_Atten**: In "-AGG_Atten", we use sum-pooling function to replace the attention aggregation function f_{AGG} in MacroE.
- **-Gate**: In "-Gate", we remove the fusion gate from the MMRD, i.e., concatenate $\mathbf{H}_{Macro}$ and $\mathbf{H}_{Micro}$ directly ($\mathbf{H}_{Fuse} = \text{concat}(\mathbf{H}_{Macro}, \mathbf{H}_{Micro})$).
- **-Atten**: In "-Atten", we replace the attention sum-pooling with normal sum-pooling, that is $\mathbf{H}_{Rumor} = \sum_{j=1}^{N} \mathbf{h}_{Fuse}^{j}$.
- **-GCN**: In "-GCN", we replace the convolution kernel in MacroE with a vanilla GCN layer.
- **-Time**: In "-Time", we ignore the timestamp information, i.e., the input feature of the first MicroE are user profile features.

The results of the ablation study are summarized in Table 6.5, where we can observe that:

(1) The accuracy of "-Atten" remarkably decreases compared with other variants, which indicates that user-level attention sum-pooling can learn the importance of each user in rumor diffusion since it allocates different significance to each row (that correlated to a specific user) of $\mathbf{H}_{Fuse}$. The visualization of the attention weights is

Table 6.5: Performance comparison between MMRD and its variants.

	Twitter15				Twitter16			
Method	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
-AGG_Atten	0.854	0.855	0.855	0.855	0.826	0.827	0.824	0.825
-Gate	0.875	0.875	0.874	0.875	0.845	0.845	0.844	0.844
-Atten	0.831	0.835	0.829	0.832	0.795	0.799	0.769	0.784
-GCN	0.851	0.851	0.851	0.851	0.807	0.807	0.806	0.807
-Time	0.878	0.878	0.879	0.878	0.845	0.863	0.839	0.851
MMRD	**0.922**	**0.922**	**0.923**	**0.922**	**0.876**	**0.877**	**0.874**	**0.875**

depicted in Figure 6.4, which further proves the effectiveness. From Figure 6.4, we also find that the later users are more critical in rumor spreading, which confirms the hypothesis that rumors can spread deeper than non-rumors [43]. (2) Using the fusion gate to control the dependency on macroscopic diffusion and microscopic diffusion will improve the model performance as achieved by the "-Gate". (3) The results of "-GCN" demonstrate that multi-hop and directional information are essential for macroscopic diffusion modeling, and the performance of "-AGG_Atten" worse than MMRD, which further demonstrates that as for each node, their order-dependency is different. (4) As for "-Time", it shows the importance of the timestamp information in capturing microscopic diffusion.

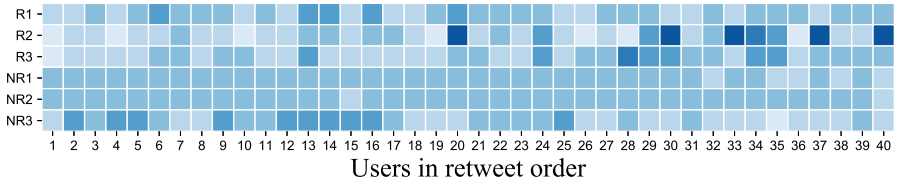
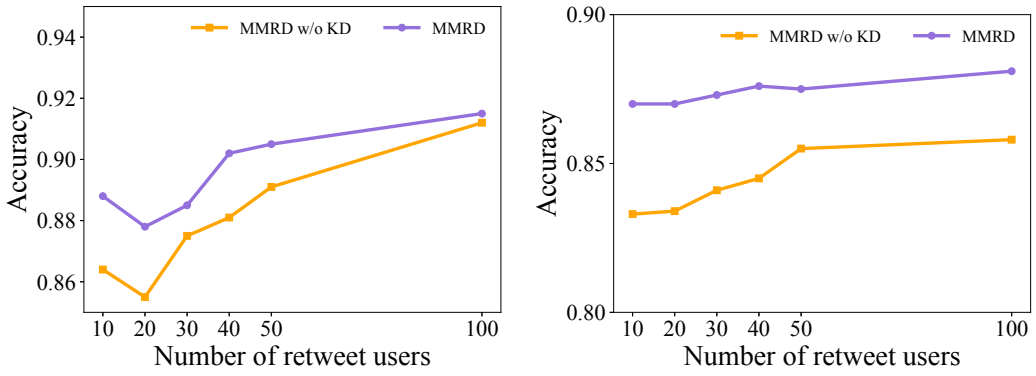


Figure 6.4: Visualization of attention weights in attention sum-pooling, which randomly choose 3 rumors and 3 non-rumors from Twitter15. Dark colors refer to a higher value.

6.4.5.2 Performance on knowledge distillation

In our work, one of the most important components is the use of knowledge distillation to enhance model performance. In order to test the performance of knowledge distillation (for briefly, simplify as KD), in this section, we conduct experiments on removing KD and using cross KD, respectively.

Figure 6.5 shows the results when removing the KD, we find that, after removing KD, although the model still can achieve better performance compared with the baselines in Table 6.3 and 6.4, it can be further improved by using KD to transfer knowledge from a teacher model to a student model. Besides that, the effect of KD is more remarkable on the Twitter16 dataset, it yields a large performance interval between "MMRD" and "MMRD w/o KD".



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

Figure 6.5: The effectiveness of knowledge distillation. The number of observed retweet users per source tweet varies from 10 to 100, and we plot the corresponding detection accuracy of MMRD with and without knowledge distillation. "MMRD w/o KD" denotes MMRD without knowledge distillation.

Figure 6.6 shows the comparison between different strategies of cross KD. Specifically, we train the teacher model and student model based on different datasets and then test the student's performance on both Twitter15 and Twitter16 datasets. For example, "T15/S16" means we first train a Teacher model "T15" on Twitter15 dataset and then distill the model on Twitter16 to get a student model "S16", and finally use the "S16" model to perform rumor detection on Twitter15 and Twitter16, respectively. From Figure 6.6, we observe that the performance of MMRD is much better when both the Teacher model and Student model train on the same dataset, this is because of some dataset-specific reasons, such as diffusion scale, the number of non-exist users, user-specific feature (e.g., create time), etc.

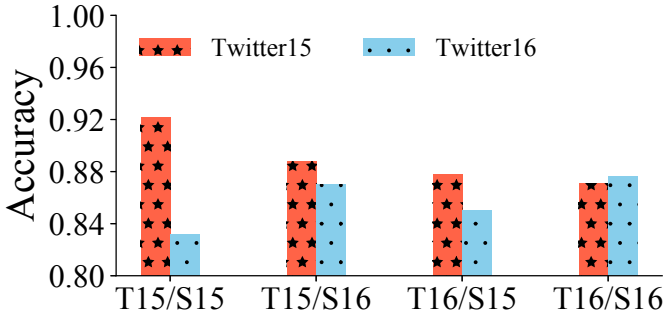
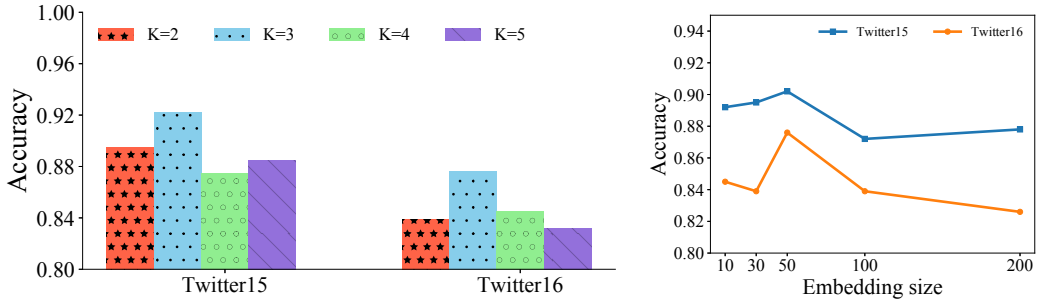


Figure 6.6: Cross knowledge distillation. The number of observed retweet users per source tweet sets to 40. Each bar represents the detection accuracy and the labels of the x-axis denote the datasets used when training the teacher model and student model. E.g., "T15" and "T16" denote that we train the teacher model on Twitter15, and Twitter16, respectively; "S15" and "S16" means that we learn the student model via distilling knowledge on Twitter15 and Twitter16, respectively.

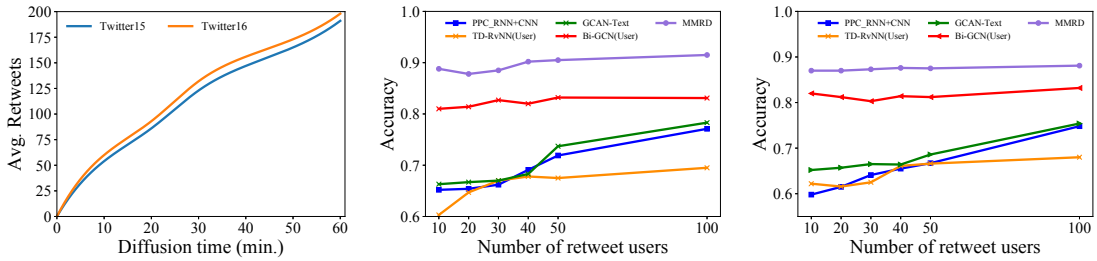
6.4.5.3 Parameter analysis

From all parameters in MMRD, we choose two special parameters to conduct parameter analysis experiments – the value of max-order K and the time embedding size d_{time}. The results shows in Figure 6.7. From both Figure 6.7a and Figure 6.7b, we find that by blindly increase the number of K and d_{time}, the model accuracy not improve, instead, decreased. And when set $K = 3$ and $d_{time} = 50$, the model achieves the best performance. Moreover, the embedding size d_{time} with small values achieve better performance than large values. And Figure 6.7a also demonstrates that take the higher-order of node interaction into consideration is useful when modeling macroscopic diffusion of tweets.



(a) Vary K **(b)** Vary embedding size

Figure 6.7: Results of parameter analysis on Twitter15 and Twitter16 when the number of observed retweet users per source tweet sets to 40. (a) Performance on different max-order value K, ranging from 2 to 5. (b) Performance on different embedding size d_{time} of timestamp vector $\mathbf{T}$.



(a) Average propagation speed **(b)** Twitter15 **(c)** Twitter16

Figure 6.8: Evaluations on early rumor detection. (a) The average propagation speed of tweets calculated based on Twitter15 and Twitter16 datasets. (b) and (c) plot the detection accuracy when the number of observed retweet users per source tweet are in the range of $[10, 20, 30, 40, 50, 100]$ on Twitter15 and Twitter16, respectively.

6.4.6 Early detection (Q3)

Detecting rumors as early as possible is crucial for public opinion control. Figure 6.8a shows the average propagation speed of messages on twitter calculated based on

Twitter15 and Twitter16. We find that within 60 minutes, both Twitter15 and Twitter16 have a diffusion speed near 190 retweets. And when the time is extremely small, i.e., within 30 minutes, the average retweets of both two datasets are close to 100. So, to investigate the performance of models on identifying rumors at an early stage, here, we consider the number of observed retweet users per source tweet from the list $[10, 20, 30, 40, 50, 100]$. Besides, for a fair comparision, we choose user profile-based mtheods, i.e., "TD-RvNN$_{(\text{User})}$", "PPC_RNN+CNN", "Bi-GCN$_{(\text{User})}$" and "GCAN-Text" as contrast methods. Figure 6.8b and Figure 6.8c show the performance comparison of early-stage detection between our MMRD and selected baselines. We can see that MMRD performs better, especially when there are only a few observations, i.e., MMRD achieves almost 89% and 87% accuracy on Twitter15 and Twitter16, respectively, even with only 10 retweet user observations.

6.5 Summary

This paper proposed a novel rumor detection model named MMRD, which can effectively and efficiently summarize a unique representation for each rumor propagation through capturing the dissemination patterns from both macroscopic and microscopic diffusion levels. Simultaneously, MMRD leverages the knowledge distillation technique to transfer knowledge from a pretraining teacher model to a student model which further improves the model detection performance. The experimental results based on two real-world Twitter data sets demonstrate that our method achieves state-of-the-art performance on rumor detection and also effective in detecting rumors at an early stage. Besides that, MMRD detects rumor via learning its spreading process, which can help us to develop rumor spreading models.