Neural Computing and Applications (2022) 34:13219–13235
https://doi.org/10.1007/s00521-022-07174-9

S.I.: DEEP LEARNING FOR TIME SERIES DATA

FEBDNN: fusion embedding-based deep neural network for user retweeting behavior prediction on social networks

Lidong Wang[1] · Yin Zhang[2] · Jie Yuan[3] · Keyong Hu[1] · Shihua Cao[1]

Received: 23 September 2021 / Accepted: 4 March 2022 / Published online: 6 April 2022

Abstract
Due to the fast growing amount of user generated content (UGC) on social networks, the prediction of retweeting behavior is attracting significant attention in recent years. However, the existing studies tend to ignore the influence of implicit social influence and group retweeting factor factors. Also, it is still challenging to consider all related factors into a unified framework. To solve the above disadvantages, we propose a novel deep neural network fusion embedding-based deep neural network (FEBDNN) through the perspective of user embedding and tweets embedding for the author and the user's historical tweets. Firstly, we propose dual auto-encoder (DAE) network for user embedding by integrating user's basic features, explicit and implicit social influence and group retweeting factor. Then, we utilize the attention-based F_BLSTM_CNN(A_F_BLSTM_CNN) model for historical tweets' representative embedding based on the combination of convolutional neural network (CNN) and bidirectional long short-term memory (BLSTM). Finally, we concatenate these embedding features into a vector and design a hidden layer and a fully connected softmax layer to predict the retweeting label. The experimental results demonstrate that the FEBDNN model compares favorably performance against the state-of-the-art methods.

Keywords Retweeting prediction · Deep neural network · Convolutional neural network · Dual auto-encoder · Social network

1 Introduction

With the fast growing of active users in recent years, social medias (e.g., Sina Weibo, Twitter, Instagram) have gradually become significant platforms for information collection and share their own opinions. The users on social networks create millions of tweets on various topics every day. Take Sina Weibo, for example, up to September 2020, there are totally 376 million active users for each month and 165 million active users for each day. Other social platforms also have a similar amount of users. Such a huge number of users make these platforms to be a popular way to get breaking news and entertainment.

The public opinion dissemination on social networks has the characteristics of content diversity, enormous people interactivity and fast retweeting speed, which will cause a great influence on public opinion analysis and public sentiment analysis [1]. Some rumors will start if some users produce some false or fake network public opinions [2–4]. Thus, it is very important to capture the development and the direction of rumor dissemination as soon as possible. Retweeting is the most straight and crucial way to spread these false information. Predicting the retweeting behavior in time is of crucial importance for the monitoring and guiding of the network public opinion [5]. Besides, retweeting prediction is also an important factor for the research of user recommendation [6], tweet recommendation [7] and hot-spot topic tendency monitoring. Therefore, it is urgent to propose an accurate retweeting prediction framework to achieve the function of public opinion early warning and even disaster prediction.

✉ Lidong Wang
wld@huqc.edu.cn

[1] Qianjiang College, Hangzhou Normal University, Hangzhou 310018, Zhejiang, China

[2] College of Science and Technology, Zhejiang University, Hangzhou 310012, Zhejiang, China

[3] Jiangsu Electric Power Information Technology Co. Ltd., Nanjing 210009, Jiangsu, China

The current research on the prediction of retweeting behavior mainly focuses on two perspectives, feature extraction and model construction. For example, Wang et al. [8] proposed a unified factorization model called Bayesian Poisson factorization (BPF++) to combine two factors, which were social influence and tweet similarity. Khan et al. [9] proposed two prediction models based on RNN and CNN to combine numeric and text features on tweets. Ameur et al. [10] proposed a contextual recursive auto-encoders embedding method for the content of comments and posts to predict user behavior (like and comment). Zhang et al. [11] incorporated structural, textual and temporal information into hierarchical Dirichlet process (HDP) model to conduct prediction task. In recent studies, the analysis and the mining of the influencing factors is still the main method for our task. Most of these methods focus on constructing a prediction model by using surface information, such as user attribute information, user's historical post content and user's following relationships. Actually, in addition to these surface features, the social influence and the group retweeting prior often have a great impact on the prediction accuracy, since that a user may be influenced by his/her group or his followees' retweeting behavior, even if he/she is not interested in the retweeted topics. The existing retweeting behavior prediction methods usually have the following disadvantages:

1. Some methods only use user attributes, post content and other surface features to predict, but ignore the implicit social influence between two users.
2. There exists group-aware prior on user retweeting behaviors. One user may retweet a post if most of the users in his/her community retweet this post, although the post is not in accord with the user's interests. However, most methods neglect the group-aware retweeting factor, which make the result cannot be apparently improved.
3. Many existing research improve the performance from the perspective of deep learning model, but it is still challenging to consider all these related factors into a unified framework.

To solve the above disadvantages, we propose a novel deep neural network called fusion deep neural network (FEBDNN) to incorporate several kinds of important factors into this unified framework with their embedding space. Specifically, our manuscript has the following contributions:

(1) We propose a unified deep neural network FEBDNN from the perspective of embedding learning that jointly combines the user embedding features and the tweets embedding features to improve the performance of retweeting prediction task.

(2) We propose dual auto-encoder (DAE) model to get a joint representation for user surface attributes, temporal information, explicit and implicit social influence. The DAE model is designed as an extension of traditional auto-encoder network. We also integrate the group retweeting factor into the joint embedding by leveraging the first-order and second-order neighborhood features.

(3) We build an attention-based model F_BLSTM_CNN(A_F_BLSTM_CNN) to get the embedding vector for the author and the user's historical tweets. The A_F_BLSTM_CNN model is designed as an attention-based deep neural network with the combination of convolutional neural network (CNN) and bidirectional long short-term memory (BLSTM) to capture the fine-grained local semantic features and to model complex semantics of word use and polysemy.

(4) We collect a new corpus Twitter dataset by ourselves for validation and conduct extensively experiments on two real-world datasets (Twitter and Sina Weibo). The experimental results demonstrate that the FEBDNN model compares favorably performance against the state-of-the-art methods.

To the best of our knowledge, our framework is the first study to incorporate all important factors into a unified model, including user surface features, temporal information, explicit and implicit social influence, group-aware retweeting factor and user/author's content information. Although the combination of CNN and BLSTM has been proposed in emotional analysis [12], we design them as a sequential structure in A_F_BLSTM_CNN model and add an extra attention mechanism to generate attention probabilities for different tweets in user/author's history. Besides, the FEBDNN framework also includes a novel joint embedding module DAE, which is an extension of traditional auto-encoder network with two inputs, two outputs and a common hidden layer. This structure is the first attempt to effectively incorporate group retweeting prior into user embedding. Thus, the technical steps in our method are different with the existing methods.

2 Related works

User's retweeting behavior is one of the important ways to spread information in social networks. The retweeting-related research mostly includes the prediction of retweeters, retweeting counts as well as tweets' spreading path. In this paper, we try to analyze and predict one user's retweeting behavior and deal with the problem of whether the user will retweet the query tweet or not, which is usually determined by many different factors. Aiming at predicting the user retweeting behavior on social networks, many researchers have deeply analyzed the user behavior through different methods. Current research mainly focuses on two aspects, feature extraction and prediction model construction.

In terms of prediction model construction, Jiang et al. [13] proposed a probabilistic matrix factorization method to combine obvious retweet data, social influence and tweet content to improve the prediction performance. But this method does not analyze the implicit social influence factors. Zhang et al. [14] analyzed the factors that affected the user's retweeting behavior and adopted factor graph model to learn the correlation between these factors. Petrovic et al. [15] used manual experiments to prove that the retweeting behavior of a single user can indeed be predicted by a supervised binary classification method, and proposed a passive–aggressive algorithm for retweeting prediction on all users. Tang et al. [16] extended the logistic regression algorithm by analyzing and characterizing the similarity between Sina Weibo users, and proposed a novel logical regression model for individual retweeting behavior (IRBLRUS) based on user similarity. Liang et al. [17] showed that the retweeting prediction problem was a one-class setting problem. By analyzing the basic factors affecting microblog retweeting, authors employed one-class collaborative filtering to measure the user's personal preferences and social influence. Liu et al. [18] adopted RBF (radical basis function) to model users retweeting behavior and then proposed a novel neural network model Cloud-RBFNN for fully expressing the fuzziness and randomness of user retweeting behavior. Wang et al. [19] presented a probabilistic model which incorporates multiple trust relationships between users into a traditional Bayesian Poisson factorization (BPF) model to predict retweeting behavior. Kushwaha et al. [20] presented a deep neural network framework based on LSTM model to classify tweets that may be retweeted with a high possibility. Dai et al. [21] considered both the user's own factors and external factors and proposed improved SVM model to predict the retweeting behavior of hot topics. Inspired by the image restoration technology, Xiao et al. [22] proposed a diffusion2pixel algorithm to transform the user relationship network of topic diffusion into image pixel matrix.

In terms of feature extraction, some context features are combined to perform our task. Boyd et al. [23] analyzed the user's historical retweet records and obtained multiple factors affecting user's retweeting behavior. Spiro et al. [24] statistically analyzed the retweeting time and proposed the influence of the time on the user's retweeting behavior. Zhang et al. [25] provided a definition of structural influence and pairwise influence to describe the local influence from active neighbors and proposed that if the user's neighbors are more active, the user is more likely to retweet the Weibo. This shows that the local structure and the retweeting behavior of neighbors have a certain influence on the predicted users. Zhang et al. [26] studied the influence of different structures composed of active neighbors and divided different structures into three categories according to the influence extent, which improved the F1 value of retweeting prediction task. Liu et al. [27] analyzed the historical retweeting records of Weibo users and proposed the definition of "user's activity" and "invisible Weibo" based on dynamic time window. This method fully considered the dynamic and regularity of retweeting behavior. Shi et al. [28] proposed a framework that related five major components involved in a social communication: (1) the information source, (2) the stimuli, (3) the information receiver, (4) the relationship between the source and the receiver and (5) the contextual factor, and an analysis on panel dataset indicates that all these components had significant impacts on individual retweeting decision. Rivadeneira et al. [29] presented a novel evidential reasoning (ER) prediction model called MAKER-RIMER to analyze the impact of different features. Jia et al. [30] extracted 19 features to predict by analyzing the relationship between high-retweeted microblog and low-retweeted microblog, the relationship between high-retweeted users and low-retweeted users and the relationship between high-retweeting users and low-retweeting users. Ma et al. [31] perceived the hop topic discussed by user's followees and analyzed user interests by using user's historical posts to perform prediction task. Yin et al. [32] learned the latent features and interactions of tweets, social relationships and the posting time through a deep learning model. Firdaus et al. [33] considered the emotion, topic preference and personality of a user to represent user's online behavior. Wang et al. [34] proposed a CH-Transformer to learn feature vector from numerical features and textual Features. Khan et al. [9] analyzed the impact of tweet text and user features on the information spreading on COVID-19.

3 Proposed method

Given a social network $G = (U, E), U = \{u_i\}(i = 1, \ldots, n)$ represents a set of users, $E = \{u_i, u_j\}(i, j = 1, \ldots, n)$ represents the social relationship between u_i and u_j. For a query tweet t_q, we use $y_i \in \{0, 1\}$ to represent whether the user u_i retweets the twee t_q or not. In other words, the task of our paper is to predict y_i as a positive output or a negative output. In this way, the problem of user identification problem can be converted to the following problem: Given a query tweet t_q and a user u_i, how to predict the corresponding retweeting label $y_i, y_i \in Y$?

We design a deep neural network FEBDNN to combine user embedding (including surface attribute features, temporal information, group-aware retweeting factor) and tweets embedding (including the user's historical tweets and the author's historical tweets). It has the following steps:

(1) For user embedding, we first extract the user basic features, temporal information and social influence between the user and the author, and denote them as $\mathbf{v}_u$. Next, we extract these features from the user's first-order and second-order neighborhood N,$u_k \in N, k \neq i$, and average these features as $\overline{\mathbf{v}}_u$. Then, to incorporate the group retweeting factor into the user embedding, we propose a DAE (dual auto-encoder) network to obtain the joint representation for $\mathbf{v}_u$ and $\overline{\mathbf{v}}_u$.

(2) For tweeting embedding, we propose A_F_BLSTM_CNN to embed the content of the author's historical tweets and the user's historical tweets. We utilize BLSTM before CNN model to exploit previous and future context with respect to current position for sequence learning in both the forward and backward direction in two layers, and utilize an attention mechanism to generate attention probabilities for different tweets.

(3) After extracting the user embedding features and the tweets embedding features, we employ a concatenation layer to combine the information from the following vectors:

$$\mathbf{V} = c[\mathbf{v}_{e\_u}, \tilde{\mathbf{v}}_h^a, \tilde{\mathbf{v}}_h^u], \tag{1}$$

where $\mathbf{v}_{e\_u}$ is the user embedding vector, $\tilde{\mathbf{v}}_h^u$ is the embedding of user's historical tweets, and $\tilde{\mathbf{v}}_h^a$ is the embedding of author's historical tweets. We design a hidden layer and a fully connected softmax layer to predict the retweeting label. In this way, the problem of retweeting prediction problem is converted to a binary classification problem.

Figure 1 shows the framework of our method. In the following, we introduce how to generate the features $\mathbf{v}_{e\_u}$,$\tilde{\mathbf{v}}_h^u$ and $\tilde{\mathbf{v}}_h^a$.

3.1 User embedding

To effectively get a high-quality representation of the users, we need to analyze the user features that may affect the user's retweeting behavior from different perspectives. Most of recent studies only extract user's basic surface features, such as the number of followees, the number of posts and temporal information, but ignore implicit social influence features and group retweeting factor. In the following, we will introduce how to get an embedding space for these features.

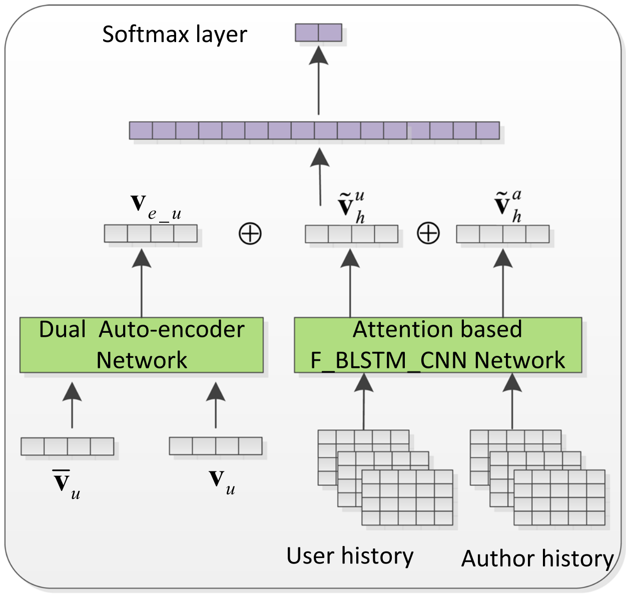


Fig. 1 The architecture of the FEBDNN model

3.1.1 User attributes and temporal information

Most social networks allow one user to follow the other users. When user u_i follows u_j, u_i can retweet a tweet that is published (or retweeted) by u_j. u_i can be considered as the follower of u_j, and u_j can be considered as the followee of u_i. User attributes mainly include the number of followees, the number of followers, the total number of tweet, the number of retweeted tweets, the authentication label and the degree of retweeting activity, which can be obtained directly from the user data. These features are denoted as $n_1, n_2, n_3, n_4, n_5, n_6$, respectively. The degree of retweeting activity n_6 is defined as the ratio of the number of retweetings within a certain period to the total number of tweets. It can measure the user's tendency to retweet. We concatenate these feature after standardization by the following equation:

$$f' = \frac{f - f_{\min}}{f_{\max} - f_{\min}}, \tag{2}$$

where f represents the original features, $f_{\min}$ and $f_{\max}$ represent the min value and the max value of one kind of features. It has to be noted that we do not standardize n_5 and n_6 since they are already in the range of [0,1]. The steps can be shown in Fig. 2.

Besides the above features, the temporal information is also an important factor for our task. It has been reported that one new tweet is usually retweeted in 24 h or smaller

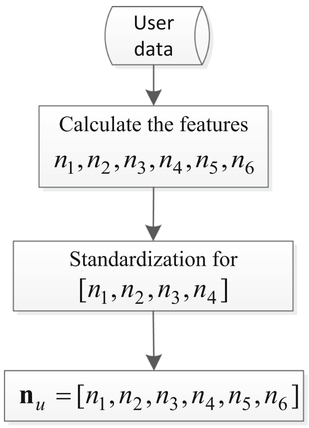

User data
Calculate the features $n_1, n_2, n_3, n_4, n_5, n_6$
Standardization for $[n_1, n_2, n_3, n_4]$
$\mathbf{n}_u = [n_1, n_2, n_3, n_4, n_5, n_6]$

Fig. 2 The steps of user attribute extraction

time period after its first publication, and after that the retweeting number decreases apparently [13]. Concerning the ideas of "earliest influence," "recent influence" and "average influence" in sociology, we extract four temporal information. The first one is the time span between the time when the tweet is published and the time when the user wants to predict, we denote it as s_1; The second one is the time span between the time when the tweet t is published and the first time when the user posts a tweet after t's publishing, we denote it as s_2; The third one is the time span between the time when the tweet is published and the most recent time when the user posts a tweet, we denote it as s_3; The last one is the time span between the time when the tweet's published and the time when the user's active neighbors retweet the tweet, we denote it as $\mathbf{s}_4$. We set $\mathbf{s}_4$ as a 20-dimensional vector. Specifically, we randomly select 20 active neighbors to record the time span. If the user does not have 20 neighbors or the neighbor does not retweet the tweet, we fill in the vector with 0. We concatenate these features after standardization by Eq. (2).

3.1.2 Social influence

Social influence means the dependency of the retweeting behavior between two users, which is also an important factor for the retweeting behavior. As shown in Fig. 3, u_1 follows the users u_6, u_7, u_8, u_9. It is obvious that the behavior of u_1 will be influenced by the behavior of u_6, u_7, u_8, u_9 if they have similar topic preference. This factor has been considered as explicit influence in many recent studies [35, 36]. Actually, there exist some implicit social relations between these users that play an important role in influencing one users' behaviors. For example, if u_1, u_2, u_3, u_4, u_5 have one or more common followees, they may tend to have similar retweeting behavior. If two users follow more common followees, more similar retweeting behaviors can be observed. We name this "co-follow" relation as ***common following degree***. Similarly, if two users have retweeted the same tweets before, they tend to retweet the same tweet in the future. We name this "co-retweet" relation as ***common retweeting degree***. The user group $\{u_1, u_2, u_3, u_4, u_5\}$ having "co-follow" relationship and the user group $\{u_1, u_{10}, u_{11}, u_{12}, u_{13}\}$ having "co-retweet" relationship tend to retweet the same tweets. Besides, if two users follow each other, they are more likely to retweet the same tweet. We name this relation as ***mutual following degree***. If two users have retweeted each other's tweet more frequently, they are more likely to have similar topic preference and therefore are more likely to retweet the same tweet in the future. We name this relation as ***mutual retweeting degree***. To effectively obtain the user embedding, we should consider both explicit and implicit social influence. In the following, we will introduce how to calculate these factors.

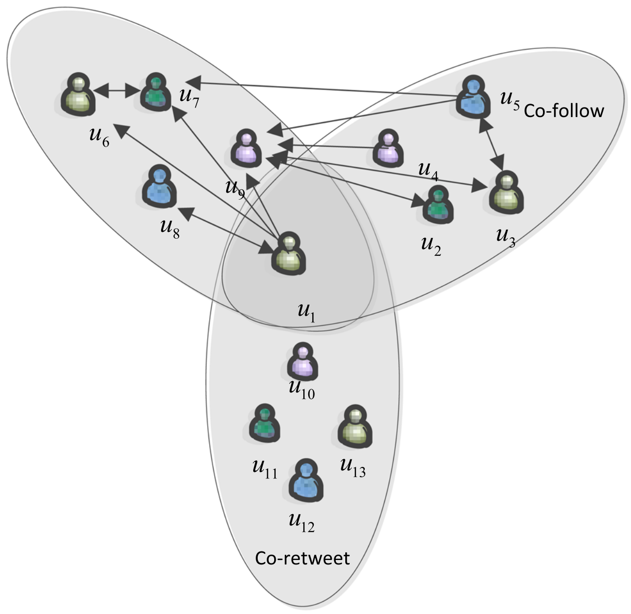


Fig. 3 Social influence between two users

(1) Topic preference similarity

If two users u_i and u_j have similar topic preference, they tend to have a similar retweeting behavior for one tweet t. Due to the sparsity of short tweets, we perform topic analysis on aggregated user history [37]. We aggregate the historical tweets of user u_i as the document d_i and aggregate the historical tweets of user u_j as the document d_j, and then we utilize standard LDA (Latent Dirichlet Allocation) model to calculate the probability distribution of top 30 topics between d_i and d_j. Next, we use cosine similarity to calculate the topic similarity for the top 30 topics. After that, we can get a 30-dimensional topic preference feature vector.

(2) Mutual following degree and common following degree

We set the mutual following degree to 1 if two users u_i and u_j follow each other, and set the mutual following degree to 0 if they do not follow each other.

When two users follow a large number of identical users, they are more likely to retweet the same tweet simultaneously. Thus, the common following degree between two users is defined as the ratio of the number of their common followees to the total number of their followees, which is defined as follows:

$$C_{ij} = \frac{U_i \cap U_j}{U_i \cup U_j}, \tag{3}$$

where U_i denotes the number of u_i's followees.

(3) Mutual retweeting degree and common retweeting degree

If two users u_i and u_j have retweeted each other's tweet more frequently, they are more likely to retweet the same tweet in the future. The mutual retweeting degree between two users can be defined as:

$$R_{ij} = \max\left(\frac{T_{ij}}{T_i}, \frac{T_{ji}}{T_j}\right), \tag{4}$$

where T_{ij} indicates the number of u_j's tweets retweeted by u_i, T_i indicates the total number of tweets retweeted by u_i. Equation (4) shows that if R_{ij} is higher, u_i and u_j are more likely to retweet each other's tweets.

The number of tweets retweeted by two users measures their common interest and their retweeting tendency. The common retweeting degree can be calculated as the ratio between the common retweeting number and the total number of their retweeting number, which is defined as:

$$M_{ij} = \frac{T_i \cap T_j}{T_i \cup T_j}. \tag{5}$$

Except for the above features, we also extract some content information in tweets, such as the number of times that the query tweet is retweeted, whether the query tweet's content contains URL, pictures, videos or @, hot topic label, etc. These specific information cannot be extracted directly in tweet content embedding space, but can affect the user's tweeting intention. After extracting the above features, we concatenate these features as $\mathbf{v}_u$ after standardization. Table 1 lists the details of three kinds of basic features.

3.1.3 Dual auto-encoder (DAE) network

After extracting the above features, we obtain a k-dimensional basic attribute space $\mathbf{v}_u$. However, besides these attributes, group retweeting behavior also has an important impact on the prediction of one user's retweeting behavior. As shown in Fig. 3, the retweeting behavior of u_1 will not only be affected by the connected users, but can also be affected by some indirectly connected users, such as the users (u_2 and u_3) from the second-order neighborhood. In other words, the user retweeting behavior on social networks has global influence between two users. If many user's friends retweet a tweet, he will probably retweet it too, although he is not interested in it. We name this kind of potential influence as ***group retweeting factor***. However, the existing prediction models often ignore this factor, which makes it difficult to obtain a high accuracy. To incorporate the group retweeting factor into our model, we first average the basic attributes of the users from u_i's first-order and second-order neighborhood, and denote it as $\overline{\mathbf{v}}_u$. Then, we design a dual auto-encoder network DAE network to extract the fusion embedding for $\overline{\mathbf{v}}_u$ and $\mathbf{v}_u$. Based on the above analysis, we can see that the user embedding features in our model contain implicit local and global structural information, which comes from the social influence between two users and the average feature vector calculated from the user's first-order and the second-order neighborhood.

The DAE network, which extends from a traditional deep auto-encoder model, is designed as a two-input and two-output network with a common fully connected hidden layer. The architecture of the DAE network is shown in Fig. 4. It uses two separate deep encoders and two separate decoders. In this way, the attribute information $\overline{\mathbf{v}}_u$ and $\mathbf{v}_u$ are tightly inter-connected, which ensures that the low-dimensional embedding feature space can preserve the attributes from both the user and his neighbors. In this way, some features that reflect the group retweeting prior can also be preserved and incorporated into the output of DAE network. As shown in Fig. 4, the encoder part contains the layers $\{\mathbf{h}_l^1, \mathbf{h}_r^1, \mathbf{h}^2\}$, the decoder part contains the layers $\{\hat{\mathbf{h}}_l^1, \hat{\mathbf{h}}_r^1, \hat{\mathbf{h}}^2\}$.

The right part of DAE network is the encoding for the user attribute vector $\mathbf{v}_u$, and the left part is for the user's neighborhood attribute vector $\overline{\mathbf{v}}_u$. Take $\overline{\mathbf{v}}_u$ for example, the latent representation vectors for $\overline{\mathbf{v}}_u$ are represented as $\mathbf{h}_l^1$ and $\mathbf{h}^2$ in the combination module. The differences between our model and traditional auto-encoder are the combination part and the dispatch part. The combination part combines the node attribute vectors from the user and his neighborhood, which can preserve both of their attribute proximity. The dispatch part dispatches the

Table 1 The details of basic features

Number	Category	Information
1	User attributes	The number of followees
2		The number of followers
3		The authentication label
4		The number of tweets
5		The number of retweeted tweets
6		The degree of retweeting activity
7	Temporal information	The time span between the time when the tweet is published and the time when the user wants to predict
8		The time span between the time when the tweet *t* is published and the first time when the user posts a tweet after *t*'s publishing
9		The time span between the time when the tweet is published and the most recent time when the user posts a tweet
10		The time span between the time when the tweet's published and the time when user's active neighbors retweet the tweet
11	Social influence	Topic preference similarity between the user and the author
12		Mutual following degree between the user/the user's active neighbors and the author
13		Common following degree between the user/the user's active neighbors and the author
14		Mutual retweeting degree between the user/the user's active neighbors and the author
15		Common retweeting degree between the user/the user's active neighbors and the author

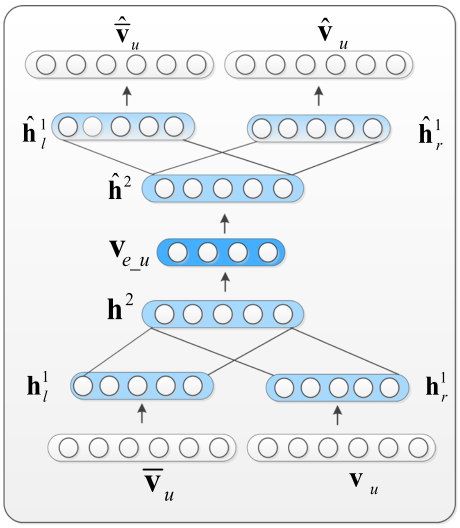


Fig. 4 The architecture of DAE model

embedding vector $\mathbf{v}_{e\_u}$ back to the hidden vectors, which are represented as $\hat{\mathbf{h}}^2$ and $\hat{\mathbf{h}}_l^1$. Formally, the relationship between these layers in the encoder module can be represented as follows:

$$\begin{cases} \mathbf{h}_l^1 = \sigma(\mathbf{W}_l^1\mathbf{v}_a + \mathbf{b}_l^1) \\ \mathbf{h}_r^1 = \sigma(\mathbf{W}_r^1\mathbf{v}_u + \mathbf{b}_r^1) \\ \mathbf{h}^2 = \sigma(\mathbf{W}_l^2\mathbf{h}_l^1 + \mathbf{b}_l^2 + \mathbf{W}_r^2\mathbf{h}_r^1 + \mathbf{b}_r^2) \\ \mathbf{v}_{e\_u} = \sigma(\mathbf{W}^3\mathbf{h}^2 + \mathbf{b}^3) \end{cases} \tag{6}$$

where $\sigma(\cdot)$ denotes a nonlinear activation function, and we utilize tanh function in our experiments. $\mathbf{W}^m$ and $\mathbf{b}^m$ denote the weight matrix and bias vector in the m - th layer. For example,$\mathbf{W}_l^1$ represents the weight matrix in the hidden layer 1 in the left part of DAE network.

During the decoder module, we input the embedding vector $\mathbf{v}_{e\_u}$ and reconstruct $\mathbf{v}_u$ to $\hat{\mathbf{v}}_u$,$\bar{\mathbf{v}}_u$ to $\hat{\bar{\mathbf{v}}}_u$. The relationship between the layers in the decoder module can be represented as follows:

$$\begin{cases} \hat{\mathbf{h}}^2 = \sigma(\hat{\mathbf{W}}^3\mathbf{v}_{e\_u} + \hat{\mathbf{b}}^3) \\ \hat{\mathbf{h}}_l^1 = \sigma(\hat{\mathbf{W}}_l^2\hat{\mathbf{h}}^2 + \hat{\mathbf{b}}_l^2) \\ \hat{\mathbf{h}}_r^1 = \sigma(\hat{\mathbf{W}}_r^2\hat{\mathbf{h}}^2 + \hat{\mathbf{b}}_r^2) \\ \hat{\bar{\mathbf{v}}}_u = \sigma(\hat{\mathbf{W}}_l^1\hat{\mathbf{h}}_l^1 + \hat{\mathbf{b}}_l^1) \\ \hat{\mathbf{v}}_u = \sigma(\hat{\mathbf{W}}_r^1\hat{\mathbf{h}}_r^1 + \hat{\mathbf{b}}_r^1) \end{cases}. \tag{7}$$

The meaning of the variables in the above equations is similar to the variables in Eq. (6). Each auto-encoder should make sure that the input feature vector has to be as similar to the reconstruction space as possible. To train the DAE network, we need to minimize the distance L between the input vectors and its reconstructions for all the instances in the network.

$$L = \sum_i \left(\left\| \bar{\mathbf{v}}_u^i - \hat{\bar{\mathbf{v}}}_u^i \right\|^2 + \left\| \mathbf{v}_u^i - \hat{\mathbf{v}}_u^i \right\|^2 \right). \tag{8}$$

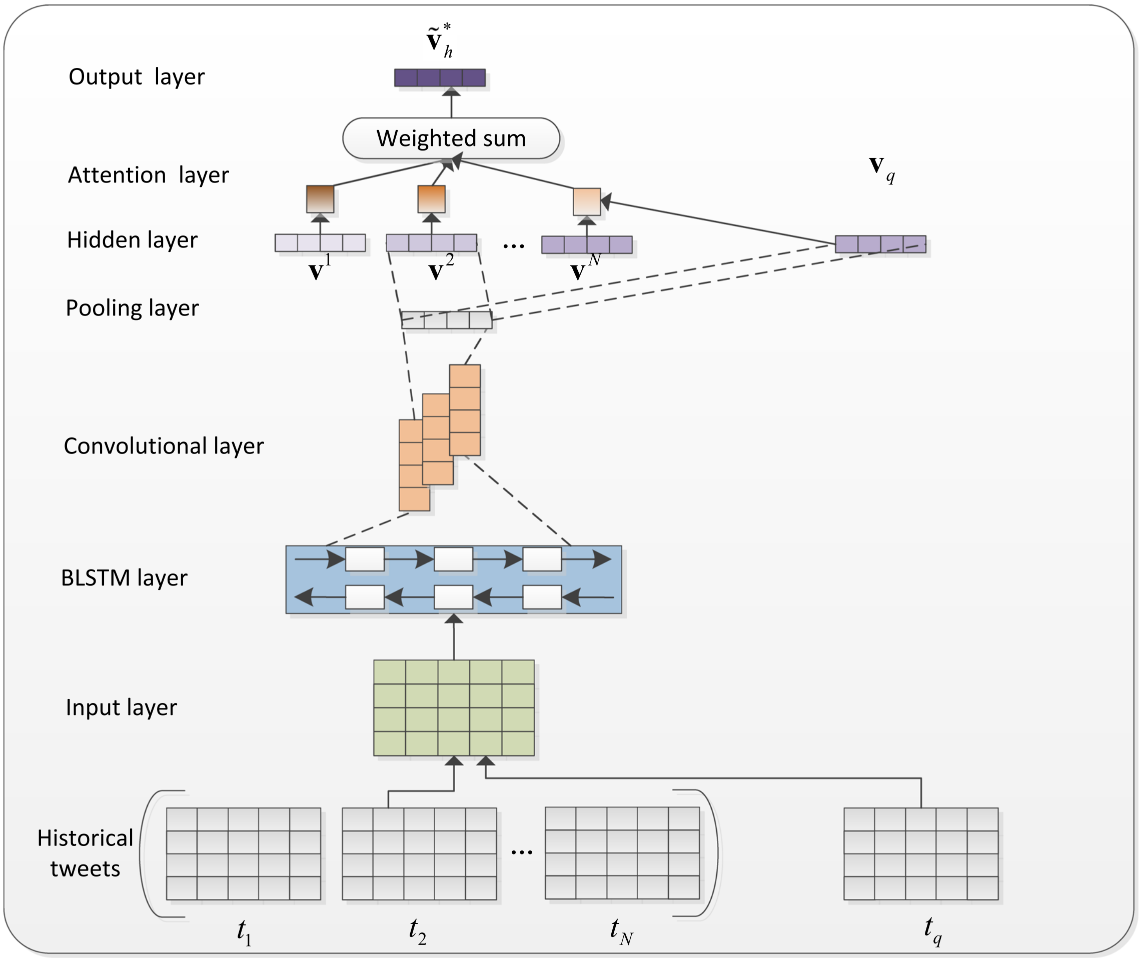


Fig. 5 The architecture of the A_F_BLSTM_CNN model

3.2 Tweets embedding

3.2.1 The overview of A_ F_BLSTM_CNN

It has been proved that CNN can obtain good performance for embedding local semantic information for short texts. However, a short tweet tends to contain context semantic information and apparent sequential features, which cannot be captured only by CNN model. Therefore, we propose an attention-based F_BLSTM_CNN(A_F_BLSTM_CNN) network combining BLSTM [38] and CNN to encode the content of the author's historical tweets and the content of user's historical tweets. Besides, not all of history tweets contribute equally to the embedding of user interests. Thus, to model the representation of user's historical tweet, we introduce an attention mechanism to achieve the different weights for different tweets in history memory. Figure 5 shows the architecture of the A_F_BLSTM_CNN model. This model can take the historical tweets from the author/user and the query tweet as input. In Fig. 5, $t_1, t_2, \ldots, t_N$ represents user/author history,$\mathbf{v}q$ is the embedding of the query tweet, $\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^N$ are the embedding vectors of $t_1, t_2, \ldots, t_N$, $\tilde{\mathbf{v}}_h^*$ is the final representation for user/author's historical tweets.

3.2.2 Model construction

There are several steps to encode tweets in the user/author's historical tweets. A tweet can be represented as a sequence of words $t = \{w_1, w_2, \ldots, w_n\}$, where n is the number of words. We summarize the steps of A_F_BLSTM_CNN construction in Algorithm 1.

```
Algorithm1 The construction of A_F_BLSTM_CNN model
Input: User/author history $\{t_1, t_2, ..., t_N\}$, the query tweet $t_q$, filter size $l$, the size of word embedding $k$, number of filters $m$
Output: a new representation of user/author's historical tweets $\tilde{\mathbf{v}}_h^*$.
1  for j = 1: $N$ do
2      Get a word vector $\mathbf{e}_i$ through word2vec model for each word in the tweet $t_j$;
3      Input word vectors to BLSTM to get an embedding space $\mathbf{t}'$ for tweet $t_j$ by Eq.(10)-(13);
4      Design a convolutional layer and calculate local feature vector $\mathbf{a}$ by Eq.(14)-(15);
5      Generate several filter matrices $\mathbf{m} \in \mathbf{R}^{l\times k}$ for each different $l$ and calculate the vector $\mathbf{z}$ for $\mathbf{t}'$ through a pooling layer by Eq.(16);
6      Input $\mathbf{z}$ to a non-linear full connection hidden layer to get a new embedding space $\mathbf{v}^j$;
7  end for
8  Use the query tweet's embedding vector $\mathbf{v}_q$ to generate attention probability over $\mathbf{v}^j$ by Eq.(17)-(18);
9  Calculate the new representation of user/author's historical tweets $\tilde{\mathbf{v}}_h^*$ by Eq.(19).
```

In the following, we describe each step in detail.

Firstly, each word w_i can be mapped into a word vector $\mathbf{e}_i$ through word2vec model and keep them static. In each tweet matrix, each column is a feature vector and represents a word. The feature matrix of tweet t can be denoted as:

$$\mathbf{t} = \mathbf{e}_1 \oplus \mathbf{e}_2 \oplus \cdots \oplus \mathbf{e}_n. \tag{9}$$

In this way, we obtain a word-level representation for the tweet t.

Secondly, we employ BLSTM for further embedding for the tweet t to get a high-quality representation $\mathbf{t}'$, which can ideally model complex semantics of word use and polysemy. LSTM (long short-term memory) is a specific RNN model, which uses three multiplicative gates to solve the gradient vanishing and gradient exploration problem in RNN. It has been reported that BLSTM can better understand both the future and the past context information, and it is suitable to be used on short tweets [39]. BLSTM is composed of two LSTM models, one of which processes the context features forward, while the other processes the context features backward. The architecture of the further embedding learning for tweet content is shown in Fig. 6. We take the word $\mathbf{e}_t$ as input and outputs hidden states $\mathbf{h}_t$ and $\mathbf{h}'_t$, which can be represented as follows:

$$\mathbf{h}_t = \mathrm{LSTM}_f(\mathbf{e}_t, \mathbf{h}_{t-1}), \tag{10}$$

$$\mathbf{h}'_t = \mathrm{LSTM}_b(\mathbf{e}_t, \mathbf{h}'_{t+1}). \tag{11}$$

Finally, we concatenate the forward hidden states and the backward hidden states after BLSTM modeling:

$$\mathbf{w}_t = [\mathbf{h}_t, \mathbf{h}'_t], \tag{12}$$

where $\mathbf{h}_t$ is the forward output of BLSTM,$\mathbf{h}'_t$ is the backward output of BLSTM. In this way, each word can obtain different embeddings in different context sentences and can disambiguate the meaning of words using their context. Thus, the embedding for tweet can be represented as:

$$\mathbf{t}' = \mathbf{w}_1 \oplus \mathbf{w}_2 \oplus \cdots \oplus \mathbf{w}_n. \tag{13}$$

Thirdly, we design a convolutional layer to extract local features for the input representation matrix. We generate a filter matrix $\mathbf{m} \in \mathbf{R}^{l\times k}$, where l means the window size and k is the dimension of word vector $\mathbf{w}_i$. For example, we can use the following equation to generate a new feature for $\mathbf{w}_{j:j+l-1}$ with window size l:

$$a_j = \sigma(\mathbf{m} \cdot \mathbf{w}_{j:j+l-1} + b), \tag{14}$$

where b is a bias term,σ is a nonlinear function. We use ReLu function in our experiments. We employ $\mathbf{m}$ to different continuous parts of the feature matrix $\{\mathbf{w}_{1:l}, \mathbf{w}_{2:l+1}, \cdots, \mathbf{w}_{n-l+1:n}\}$ to generate a feature vector $\mathbf{a}$:

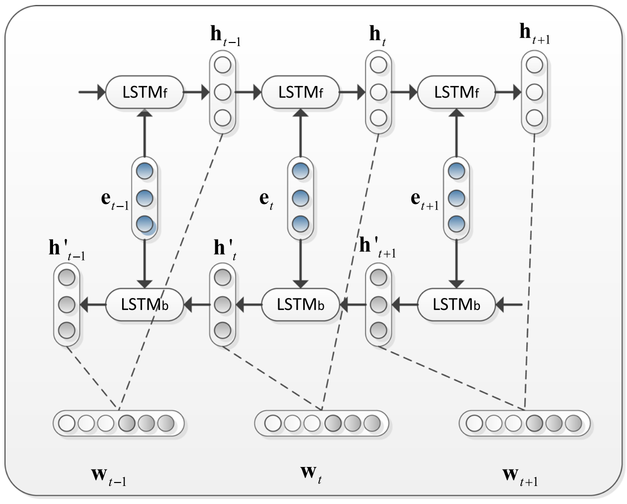


Fig. 6 BLSTM for tweets embedding

$$\mathbf{a} = [a_1, a_2, ..., a_{n-l+1}]. \tag{15}$$

The length of the output of pooling layer depends on the number of words in the tweet t. We need to combine the tweet embedding feature with other features to generate a global feature vector. Thus, we need a fixed length for the tweet encoded features. To solve this problem, we use max pooling operation to extract the max value for each $\mathbf{a}$. In this way, the most important feature can be extracted by keeping the highest value.

$$\hat{a} = \max(a_1, a_2, ..., a_{n-l+1}). \tag{16}$$

We vary the window size and obtain several filters for each size to obtain multiple features. These features are concatenated into a fixed length feature vector $\mathbf{z} = [\hat{a}_1, \hat{a}_2, ..., \hat{a}_m]$(note that here we have m filters). Thus, the output of the pooling layer is a feature vector with a fixed length.

Finally, to make full use of rich features obtained from the pooling layer, we use a nonlinear full connection hidden layer and set tanh as the activation function to make the output embedding space and the user embedding space in the same range. The output embedding vector is denoted as $\mathbf{v}^i$,$\mathbf{v}^i \in R^d$.

Following the above steps, we get an embedding vector for each tweet in user/author's historical tweets. However, not all tweets in the user/author's history contribute equally to the modeling of history embedding. Thus, we utilize an attention mechanism to get a new representation of user/author's historical tweets $\tilde{\mathbf{v}}_h^*$ based on the tweet's attention probability distributions. We use the query tweet's embedding vector $\mathbf{v}_q$ to query the representations of each history tweet and generate attention probabilities over the author's tweet histories and the user's tweet histories:

$$\mathbf{s}_h^N = (\mathbf{W}_h^q \mathbf{v}_q)^T \mathbf{W}_h^N \mathbf{v}^N, \tag{17}$$

$$\boldsymbol{\alpha}_N^h = \text{softmax}\left(\mathbf{s}_N^h / \sqrt{d_k}\right), \tag{18}$$

$$\tilde{\mathbf{v}}_h^* = \sum_{i=1}^{N} \alpha_i^h \mathbf{v}^i, \tag{19}$$

where N is the number of historical tweets of the user/author, $\mathbf{v}^N \in R^{d \times N}$ is the embedding matrix for all tweets, $*$ denotes the user or the author's historical tweets and $\sqrt{d_k}$ denotes the scaling factor,$\mathbf{W}_q^h \in R^{d_k \times d}$,$d_k < d$.

3.3 Training

After the above features are extracted, we use a concatenation layer to combine different features. Then, we design a nonlinear hidden layer to make full use of the features obtained from the concatenation layer. This layer can be represented as follows:

$$\mathbf{h} = \sigma(\mathbf{w}_c \cdot c[\mathbf{v}_{e\_u}, \tilde{\mathbf{v}}_h^a, \mathbf{v}_h^u] + \mathbf{b}), \tag{20}$$

where $\mathbf{w}_c$ is the parameter matrix,$\mathbf{b}$ is the bias term, σ is a nonlinear function, which is designed as tanh function. We also utilize dropout for the regularization by randomly setting the elements in feature vector to zero with probability p.

We define all parameters in our model as $\boldsymbol{\theta}$. It has to be noted that the user embedding is designed as an extension of Auto-encoder, which can be trained independently under an unsupervised way. Thus, the parameters in DAE network are not included in $\boldsymbol{\theta}$. For the training feature set $\mathbf{x}_i \in X$, and its corresponding tweeting label set $Y = \{y_1, y_2, ...y_m\}$, the network needs to calculate a value $s(y_i)$ for each $\mathbf{x}_i$, then we use a softmax function operation to convert this value to a probability distribution:

$$p(y_i = j|\mathbf{x}_i) = \frac{\exp(s(y_i))}{\sum_k \exp(s(y_k))} \tag{21}$$
$$s(y_i) = \mathbf{W}_j \cdot \mathbf{h}_i + b_j,$$

where $y_i \in \{0, 1\}$.$\mathbf{W}_j$ denotes the weight vector for i-th class. The objective of our training process is to minimize the following log-likelihood function:

$$\boldsymbol{\theta}* = \arg\min \sum_{\mathbf{x}_i \in X} -\log(p(y_i|\mathbf{x}_i)). \tag{22}$$

We utilize mini-batch SGD [40] to optimize the training process. The details of some hyper-parameter are discussed in Sect. 4.3.4.

4 Experiments

4.1 Datasets

We conducted several experiments on the following two datasets:

1. Sina Weibo dataset [13]

This dataset contains 1,787,443 core users and their social relationships. Each user has 1000 most recent microblogs for each user. Each microblog contains id, original microblog id, user id, author id, content, time and so on. The original microblog id indicates whether microblog is retweeted or not. The user attributes include name, gender, verification status, #followers, #followees, creating time and so on.

2. Twitter

We collected a new corpus by ourselves for validation from Twitter API.[1] We randomly select several users from

[1] https://developer.twitter.com.

Twitter and iteratively collect their followers and followees. We get 400 historical tweets posted by each user. Then we remove some non-English tweets, the tweets less than five words and the tweets contain some unexplainable characters. We finally collect 5213 users for training and testing. Each user contains 300 historical tweets. Each tweet contains id, original tweet id, author id, user id, content, time and so on. The user attributes include screen name, account name, location, #followers, #followees, individual resume, creating time and so on.

Suppose that u_i is u_k's follower, if u_i retweets the microblog in the dataset, we consider it as a positive instance; if u_i is not observed to retweet the microblog posted by u_k, we consider it as a negative sample.

4.2 Experiment settings

FEBDNN model contains several hyper-parameters. We set them empirically in our experiment. Table 2 shows the details of some crucial hyper-parameters. The details of some crucial parameters analysis is provided in Sect. 4.3.4. The experiments were implemented by using Python, Keras library and Tensorflow.

The complexity of A_F_BLSTM_CNN contains three parts, which are CNN part, BLSTM part and Attention part. The complexity of CNN part is $O[M^2 \cdot (l \cdot k) \cdot c_{\text{in}} \cdot c_{\text{out}}]$, where M is the size of the output feature map,$l \cdot k$ is the size of filter, c_{in} is the number of units in input layer, c_{out} is the number of units in output layer. The complexity of BLSTM part is determined by the number of cell blocks, the size of cell blocks, the number of hidden units, etc. Since we directly utilize Keras library to implement BLSTM part, we use W to denote the total number of weights optimized in the network. Thus, the complexity of BLSTM can be denoted as $O(W)$. The complexity of Attention part is $O(N \cdot d^3)$, where N is the number of historical tweets of the user/author,d is the embedding dimension.

The DAE network is trained independently. 5 hidden layers are set in the DAE network, including two hidden layers in encoder module, two hidden layers in decoder module and one fusion hidden layer. The embedding size of $\mathbf{v}_{e\_u}$ is 64. The batch size in DAE network is set to 50 for both two datasets. We use word2vec model to train the word embedding for each tweet. The metric for our evaluation contains precision, recall and F1_score.

To validate the effectiveness of our algorithms, we conducted several experiments:

(1) To evaluate the performance influenced by several factors, we conducted several experiments with a single factor, such as the user embedding, the user embedding without group retweeting factor, tweets embedding, tweets embedding without BLSTM module;
(2) Performance comparison on retweeting prediction between FEBDNN and other state_of_the_art algorithms;
(3) Performance comparison on retweeting path prediction between FEBDNN and other baseline algorithms.

To support the above experiments, we compared our framework with other traditional and state-of-the-art methods:

(1) SVM: We extract the user basic features, temporal information and social influence between the user and the author, and denote them as $\mathbf{v}_u$. Next, we extract the average features from the user's first-order and second-order neighborhood N,$u_k \in N, k \neq i$, and denote them as $\overline{\mathbf{v}}_u$. Then, we average the embedding vectors of all the words as the feature vector of the tweet and also average all the tweets' embedding vectors for the user/author's historical tweets. Finally, we use these features to train a SVM model to make a prediction.
(2) LR: We implement the algorithm proposed in [41], which uses Logistic Regression algorithm to classify each tweet as positive or negative. The tweets are encoded by TF-IDF, and all the tweets' embedding vectors are averaged as the user/author's historical tweets. Similar to the SVM, we consider the same information to train a LR model to make a prediction.

Table 2 Hyper-parameters

Parameter	Sina dataset	Twitter dataset
Filter sizes	[1,2]	[1,2,3]
Feature maps	100 for each filter size	100 for each filter size
Dropout rate	0.5	0.5
Learning rate	0.01	0.01
The dimension of word embedding	300	300
The dimension of tweet embedding	100	100
The batch size	60	50

(3) BERT: The BERT pre-trained language model[2] is learned based on large-scale corpora and can calculate context representation for each word and enhance the representation ability for each sentence. In our task, we need to use BERT to get the embedding space for the user history and the author history. Firstly, we input each tweet into BERT to get the embedding space by using the CLS-token output. We use 12-layer BERT-base model in our experiment. In this model, the train-batch-size is set to 16, the learning rate is set to 0.0001, and the drop out rate is set to 0.5. Then, we use the attention mechanism to obtain the embedding of user/author history $\tilde{\mathbf{v}}_h^*$. The following steps are the same as in our model.

(4) ASC-HDP [11]: In this method, we combine user attributes, author attributes and content information to generate this topical model. We also average the embedding vectors of all the words in a tweet as the content information. We use the source code released in original paper.

(5) C_RBF [18]: In this model, we design a RBF neural network with cloud model to predict user's retweeting behavior. The cloud model is combined to optimize the activation function in the hidden layers of RBF model. The input of the model is set to the features used in our model. We obtain the source code released in original paper.

(6) AUT-MSAM [31]: This model utilizes a novel masked self-attentive model and a hierarchical attention mechanism to jointly perceive hot topics and user interests. We calculate the history interests similarity between author and user in our experiment, and randomly sample 30 tweets from users' history, and set the parameters as the default values in the original paper.

(7) DAE_UE: DAE_UE is designed as a variant of FEBDNN to test the performance of user embedding features. In this method, we remove the tweet encoding and the group retweeting factor, so that the DAE module is converted to a traditional autoencoder network to embed the user feature $\mathbf{v}_u$, not including $\overline{\mathbf{v}}_u$. Then, the output embedding vector is directly set as the input of a MLP.

(8) DAE: Based on DAE_UE, this method adds the factor of group tweeting prior in the user embedding process. The architecture of DAE can be seen in Fig. 4.

(9) A_F_BLSTM_CNN: This is the attention-based F_BLSTM_CNN in our paper. The architecture of A_F_BLSTM_CNN can be seen in Fig. 5. In this method, we only concatenate user history embedding feature and author history embedding feature in Eq. (20).

(10) F_CNN: F_CNN is designed as a variant of A_F_BLSTM_CNN to test the performance of history tweet encoding only by CNN model. We remove the BLSTM part in A_F_BLSTM_CNN.

4.3 Experimental results

4.3.1 Performance on different factors

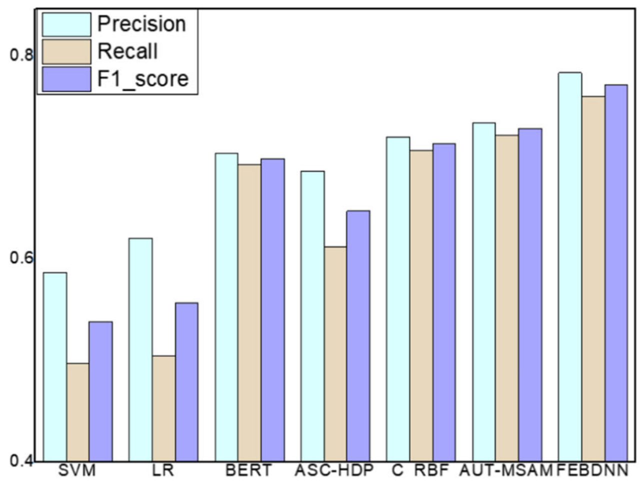
To evaluate the performance influenced by a single factor, we conducted several experiments by DAE_UE, DAE, F_CNN, A_F_BLSTM_CNN and FEBDNN, respectively. For two datasets, since the ratio between the positive samples and the negative samples is unbalanced, we sampled a balanced dataset with a ratio of 1:1. Specifically, we randomly sampled a negative sample for each positive sample. We evaluated our performance in terms of precision, recall and F1_score.

Table 3 shows the performance on the variants of FEBDNN. These experimental results show that A_F_BLSTM_CNN obtains better performance than DAE. It means that the tweets embedding module for the query tweet and user/author's historical tweets plays a more important role for our prediction task. DAE obtains a little improvement over DAE_UE (+ 3.20% in terms of F1 on Sina, + 5% in terms of F1 on Twitter), which demonstrates that incorporating group tweeting prior factor into DAE model can improve the prediction performance. A_F_BLSTM_CNN performs better than F_CNN on both two datasets. This is due to that A_F_BLSTM_CNN implements BLSTM for further embedding on the microblogs/tweets, which can model the complex semantic of the words and obtain a better representative embedding vector. Compared with the methods based on a single factor, we can see a clear improvement on FEBDNN when we combine the user embedding feature and the tweets embedding feature together. This indicates that all of these factors can

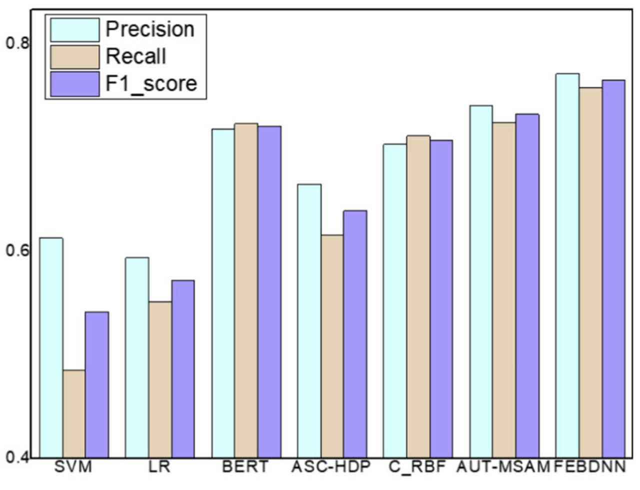
Table 3 Prediction performance of the variants of FEBDNN

Methods	Sina			Twitter		
	Prec	Rec	*F*1	Prec	Rec	*F*1
DAE	0.722	0.654	0.686	0.723	0.623	0.669
DAE_UE	0.691	0.622	0.654	0.644	0.596	0.619
F_CNN	0.681	0.624	0.651	0.628	0.587	0.607
A_F_BLSTM_CNN	0.745	0.695	0.719	0.753	0.621	0.681
FEBDNN	**0.784**	**0.761**	**0.772**	**0.772**	**0.759**	**0.765**

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



(a) Sina



(b) Twitter

Fig. 7 The experimental results on different methods

contribute a lot on predicting retweet behavior, and FEBDNN is indeed an effective way to incorporate these factors into a unified framework. Also, FEBDNN improves a lot on recall, this demonstrates that the extracted embedding features can detect more potential positive instances.

4.3.2 Performance on different methods

We compared FEBDNN with other traditional and state_of_the_art methods, such as SVM, LR, BERT, ASC-HDP, C_RBF and AUT-MSAM. The results are shown in Fig. 7.

As shown in Fig. 7, it is clear that FEBDNN performs best compared with other methods. The performance of SVM and LR model is unsatisfactory. They only use the original feature vector as the input without embedding analysis, which cannot catch the good representation of features. The results of BERT-only model can achieve better results than SVM and LR model. This is because the BERT model can achieve high-quality embedding space, which can represent word syntax, word semantics and place ambiguous words into distinct embedding space. Compared with BERT, FEBDNN produces about 5.8% improvement on F1 on two datasets. This demonstrates that A_F_BLSTM_CNN can obtain better representative embedding space than BERT. This may due to that the embeddings from the pre-trained Bert cannot suit very well for our task, and the tweeting embedding based on CLS-token output cannot capture complex semantic information. Compared with ASC-HDP, FEBDNN produces about 15% improvement on F1_score. ASC-HDP only uses the user basic features and content information to generate a topical model and neglects the group retweeting factor and the social influence. Actually, the social relationships between users will cause mutual influence on their retweeting behaviors and will even change the user's own interests and cause the local consistency of the users' retweeting behavior. Thus, it is important to incorporate the social influence and the group retweeting factor into user embedding analysis. Compared with AUT-MSAM, FEBDNN produces about 4.3% improvement on F1_score. This shows that considering users' interest similarity cannot obtain better performance than the perspective of user embedding and effective tweet encoding. Also, AUT-MSAM cannot reflect the factor of group retweeting factor and social influence between the user and the author, which may lead to an unsatisfactory result.

Compared with C_RBF, FEBDNN produces about 5.8% improvement on F1_score. Although C_RBF is designed as a deep neural network model, and it incorporates cloud model to model the ambiguity and randomness for the relationships between the features and user behaviors, the features are extracted without embedding analysis, which would result in a decrease in precision and recall. The results also show that the perspective of the embedding analysis for user features and tweet features can apparently benefit our performance.

4.3.3 Prediction of retweeting path

In order to verify the performance of our algorithm in social network message diffusion, we also conducted some experiments on the tweet retweeting path prediction. When a tweet is retweeted by several cascade users until no user retweets the tweet, we define these cascade users as a cascade set $c_i = \{u_1, u_2, ..., u_{|c_i|}\}$.$|c_i|$ is the length of the retweeting path. For this retweeting path, the prediction is successful when the retweeting behaviors of all users are all accurately predicted.

We compared our method with BERT model and information diffusion model FOREST [42] and SNIDSA [43]. If a tweet is retweeted by several cascade users, it can be considered as a kind of information diffusion. FOREST combines GRU model and structural context information, which comes from user attributes and structural information extracted by neighborhood sampling [44]. We implement RNN-based microscopic diffusion prediction objective for our task, which predicts the next infected user who may retweet the tweet given last m infected users who have retweeted the tweet. The window size m is set to $|c_i|$-1 in our experiment. SNIDSA employs RNN to model the sequential information and incorporates the attention mechanism into RNN to capture the diffusion context. The experimental results are shown in Table 4. It has to be noted that if we want to predict the retweeting path with $|c_i|= 3$, it has to be conducted based on the retweeting path with $|c_i|= 2$.

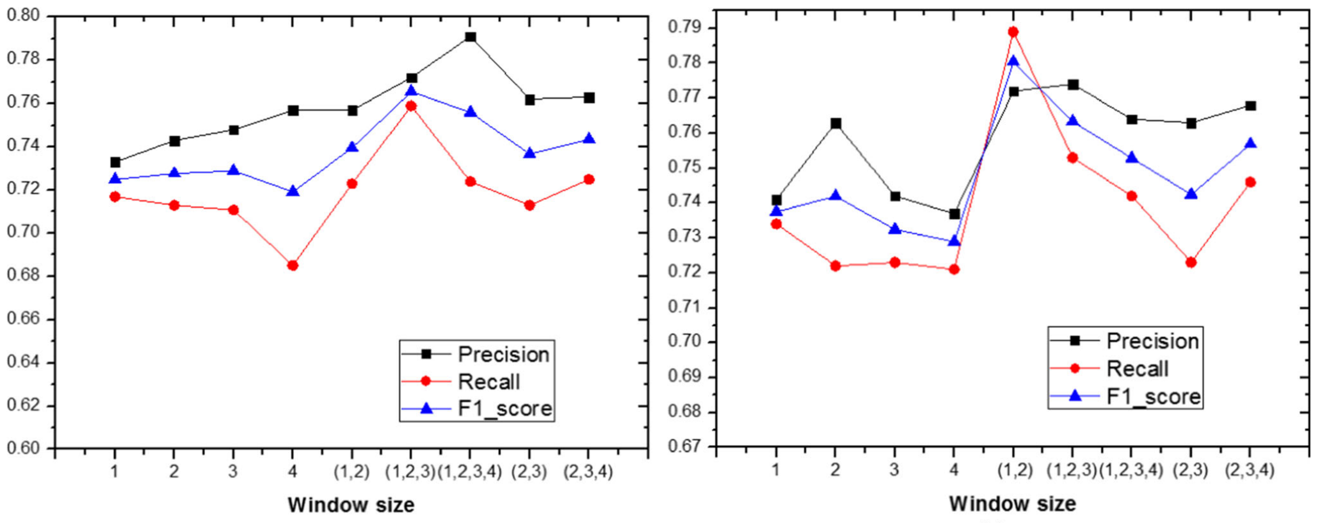
As shown in Table 4, FOREST outperforms SNIDSA and performs an average improvement of 7.1% on two datasets. This is due to the encoding of structural context in FOREST, which considers second-order neighborhoods while SNIDSA only considers the first-order neighborhoods. BERT and FEBDNN can achieve better performance than information diffusion models. FEBDNN performs best. It consistently outperforms other methods, except for the long prediction path($|c_i| = 6$). This result can demonstrate that the systemically combination of basic features, social influence and group-aware retweeting factor can achieve better prediction performance, while FOREST and SNIDSA only consider attribute and structural information. Furthermore, the user embedding features in our model contain implicit local and global structural information, which comes from the social influence between two users and the average feature vector calculated from the user's first-order and the second-order neighborhood.

4.3.4 Parameter analysis

The parameters in our model are mostly included in the DAE part and the A_F_BLSTM_CNN part. The DAE part

Table 4 The performance of the prediction of retweeting path

$\lvert c_i \rvert$	Sina				Twitter			
	BERT	FEBDNN	FOREST	SNIDSA	BERT	FEBDNN	FOREST	SNIDSA
2	0.705	**0.784**	0.683	0.619	0.719	**0.772**	0.706	0.628
3	0.591	**0.628**	0.532	0.502	0.547	**0.642**	0.523	0.498
4	0.497	**0.542**	0.458	0.386	0.428	**0.489**	0.462	0.422
5	0.402	**0.437**	0.351	0.317	0.392	**0.417**	0.387	0.310
6	0.307	**0.325**	0.213	0.192	0.201	0.302	**0.305**	0.215



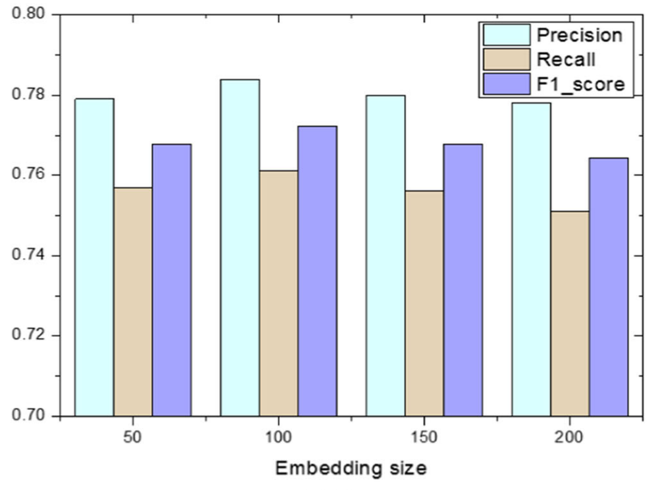
(a) Twitter **(b)** Sina

Fig. 8 Performance on different window sizes

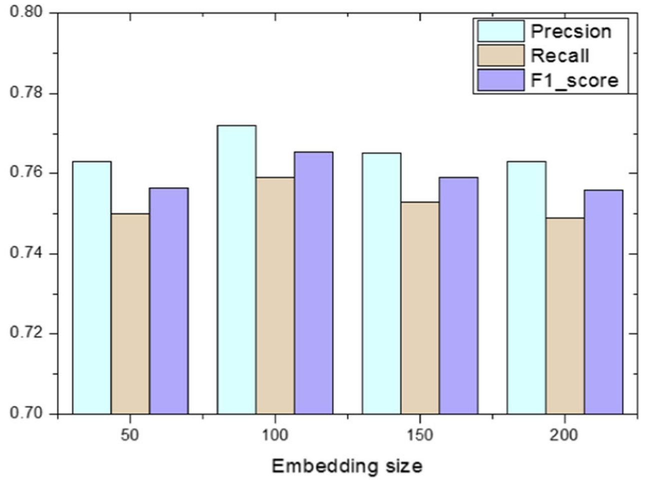
is trained independently, we do not discuss this part in this subsection. The A_F_BLSTM_CNN part has several important hyper-parameters. They are: (1) the window size of the filter mask; (2) the tweet embedding size; (3) the batch size. When we evaluate one parameter's performance, the other parameters are set to an optimal value. Based on the experimental results, we can observe that the proposed model could achieve stable performance, in the condition of various parameter settings.

Figure 8 lists the performance on different window sizes of filter masks in tweet embedding. Firstly, we set the window size to [1–4] to test the performance on different window sizes, respectively. Then we test the performance on a serial of the combinations (1,2), (1,2,3), (1,2,3,4), (2,3), (2,3,4). From the results of a single window size, we find that when the window size is set to 3 on Twitter dataset, it gets the best performance. This is because that the window size 3 can catch the information from trigrams, and the window size 1 and 2 mean that the encoding steps focus on unigrams and bigrams, respectively. Trigrams may contain more semantic and context information. However, when the window size is set to 2 on Sina dataset, it gets the best performance. This is because the semantic structure in Chinese words is different with the semantic structure in English words. From the results of the combinations, we find that the window size (1,2,3) gets the best performance on Twitter dataset, and the size (1,2) gets the best performance on Sina dataset.

Figure 9 shows the performance of different embedding sizes of the tweet. We vary the size from 50 to 200 for both

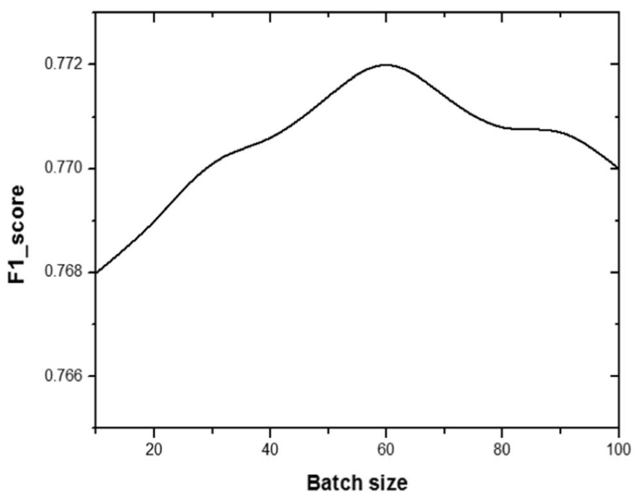


(a) Sina

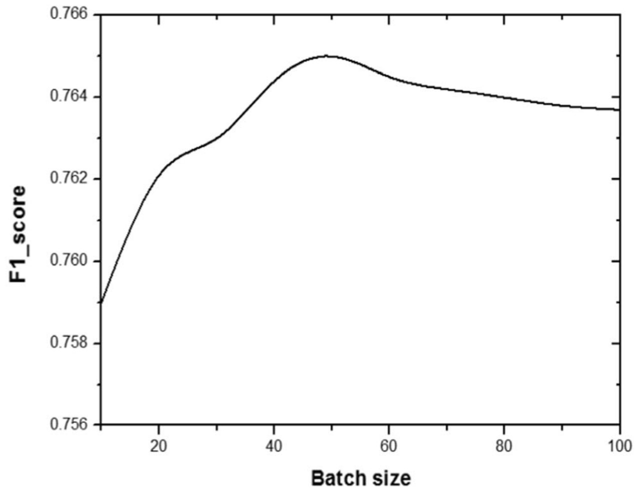


(b) Twitter

Fig. 9 Influence of the embedding size



(a) Sina



(b) Twitter

Fig. 10 Influence of the batch size

two datasets. The results in Fig. 9a show that the embedding size 100 gets the best performance on Sina dataset. We can get better results when we increase the size from 50 to 100, and the results of the size 200 are worse than the size 150. Thus, the embedding size 100 is good enough to represent the tweet in semantic space. We can observe the similar results on Twitter dataset.

Figure 10 shows the performance of different batch sizes. It has been reported that the batch size for training the neural networks should not be too big or too small. If we set the batch size to the size of whole dataset, it may cause the local optimal problem. Small batch size may introduce gradient correction for the noise and more probably to find the optimal value. In our experiments, we vary the batch size from 10 to 100. As shown in Fig. 10, the optimal batch size is different for different datasets. For Sina dataset, the performance increases when we vary the batch size from 10 to 60. However, we get worse performance when the batch size is greater than 60. For Twitter dataset, we can obtain the optimal batch size 50.

5 Conclusion

In this paper, we propose a novel integrated neural network FEBDNN to predict the user retweeting behavior. Considering that most existing research focuses only on the surface features or network structures, our method incorporates lots of necessary and important factors into a unified framework from the perspective of user embedding and tweeting embedding. We propose a novel network DAE to conduct the user embedding through the combination of surface features, social influence between the user and the author and the group retweeting factor. The social influence shows the topic similarity and both the explicit and implicit structure influence between two users. Besides, our work is the first exploration to incorporate the group retweeting factor into our model. For tweet embedding, we propose A_F_BLSTM_CNN to utilize an attention mechanism based on the combination of CNN and BLSTM for deep representative embedding for history tweets' content, which can enhance the representative ability for the tweets, and also can represent the interests of the user and the author. Our method can not only catch the user surface feature and tweet content features, but also considers the structure information and group-aware retweeting prior, which are very important supplementary information for our task. Experimental results on two real social networks show that our proposed unified model can achieve better performance than state-of-the-art methods. Besides retweeting, our model can also be applied to other behaviors across social networks [45], such as liking, commenting or favoriting.

Acknowledgements This study is supported by Zhejiang Provincial Natural Science Foundation of China under Grant No. LY19F020022, China Knowledge Centre for Engineering Sciences and Technology (CKCEST), Joint Funds of the Zhejiang Provincial Natural Science Foundation of China under Grant No. LHY21E090004.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Narwal R, Aggarwal H (2022) Predicting online game-addicted behaviour with sentiment analysis using twitter data[M]. In: Machine learning, advances in computing, renewable energy and communication. Springer, Singapore, pp 505–517
2. Sang CY, Liao SG (2020) Modeling and simulation of information dissemination model considering user's awareness behavior in mobile social networks[J]. Physica A 537:122639
3. Kim J, Bae J, Hastak M (2018) Emergency information diffusion on online social media during storm Cindy in US[J]. Int J Inf Manage 40:153–165
4. Liu C, Zhou N, Zhan XX et al (2020) Markov-based solution for information diffusion on adaptive social networks[J]. Appl Math Comput 380:125286
5. Neubaum G, Krämer NC (2017) Monitoring the opinion of the crowd: Psychological mechanisms underlying public opinion perceptions on social media[J]. Media Psychol 20(3):502–531
6. Zhou C, Bai J, Song J, et al (2018) Atrank: an attention-based user behavior modeling framework for recommendation[C]. In: Proceedings of the AAAI Conference on Artificial Intelligence 32(1)
7. Karidi DP, Stavrakas Y, Vassiliou Y (2018) Tweet and followee personalized recommendations based on knowledge graphs[J]. J Ambient Intell Humaniz Comput 9(6):2035–2049
8. Wang S, Li C, Wang Z et al (2020) BPF++: a unified factorization model for predicting retweet behaviors[J]. Inf Sci 515:218–232
9. Khan P I, Razzak I, Dengel A, et al (2021) Understanding information spreading mechanisms during COVID-19 pandemic by analyzing the impact of tweet text and user features for retweet prediction[J]. arXiv preprint. arXiv:2106.07344
10. Ameur H, Jamoussi S, Hamadou AB (2019) A deep neural network model for predicting user behavior on facebook[C]. In: 2019 International Joint Conference on Neural Networks (IJCNN). IEEE, pp 1–8
11. Zhang Q, Gong Y, Guo Y, et al (2015) Retweet behavior prediction using hierarchical dirichlet process[C]. In: Proceedings of the AAAI Conference on Artificial Intelligence. 29(1)
12. Liu Z, Zhang D, Luo G et al (2020) A new method of emotional analysis based on CNN-BiLSTM hybrid neural network[J]. Clust Comput 23(4):2901–2913
13. Jiang B, Lu Z, Li N, et al (2018) Retweet prediction using social-aware probabilistic matrix factorization[C]. In: International conference on computational science. Springer, Cham, pp 316–327
14. Zhang J, Tang J, Li J et al (2015) Who influenced you? predicting retweet via social influence locality[J]. ACM Trans Knowl Discov Data (TKDD) 9(3):1–26
15. Petrovic S, Osborne M, Lavrenko V (2011) Rt to win! predicting message propagation in twitter[C]. In: Proceedings of the international AAAI conference on web and social media. 5(1)

16. Tang X, Miao Q, Quan Y et al (2015) Predicting individual retweet behavior by user similarity: a multi-task learning approach[J]. Knowl-Based Syst 89:681–688
17. Jiang B, Liang J, Sha Y, et al (2016) Retweeting behavior prediction based on one-class collaborative filtering in social networks[C]. In: Proceedings of the 39th international ACM SIGIR conference on research and development in information retrieval. pp 977–980
18. Liu Y, Zhao J, Xiao Y (2018) C-RBFNN: a user retweet behavior prediction method for hotspot topics based on improved RBF neural network[J]. Neurocomputing 275:733–746
19. Shaoqing W, Cuiping L, Zheng W et al (2019) Prediction of retweet behavior based on multiple trust relationships[J]. J Tsinghua Univ (Science and Technology) 59(4):270–275
20. Kushwaha AK, Kar AK, Ilavarasan PV(2021) Predicting retweet class using deep learning[J]. In: Trends in deep learning methodologies, pp 89–112
21 Dai T, Xiao Y, Liang X et al (2021) ICS-SVM: a user retweet prediction method for hot topics based on improved SVM[J]. Digit Commun Netw. https://doi.org/10.1016/j.dcan.2021.07.003
22 Xiao Y, Huang Z, Li Q et al (2022)Diffusion pixelation: a game diffusion model of rumor and anti-rumor inspired by image rest oration[J]. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2022.3144310
23. Boyd D, Golder S, Lotan G (2010) Tweet, tweet, retweet: conversational aspects of retweeting on twitter[C]. In: 2010 43rd Hawaii international conference on system sciences. IEEE, pp 1–10
24. Spiro E, Irvine C, DuBois C, et al (2012) Waiting for a retweet: modeling waiting times in information propagation[C]. In: 2012 NIPS workshop of social networks and social media conference. http://snap.stanford.edu/social2012/papers/spiro-dubois-butts.pdf. Accessed. 2012, 12
25. Zhang J, Liu B, Tang J, et al (2013) Social influence locality for modeling retweeting behaviors[C]. In: Twenty-third international joint conference on artificial intelligence
26. Zhang J, Tang J, Zhong Y, et al (2017) Structinf: mining structural influence from social streams[C]. In: Proceedings of the AAAI Conference on Artificial Intelligence, 31(1)
27. Liu W, He M, Wang LH et al (2016) Research on microblog retweeting prediction based on user behavior features[J]. Chin J Comput 39(10):1992–2006
28. Shi J, Lai KK, Hu P et al (2017) Understanding and predicting individual retweeting behavior: receiver perspectives[J]. Appl Soft Comput 60:844–857
29. Rivadeneira L, Yang JB, López-Ibáñez M (2021) Predicting tweet impact using a novel evidential reasoning prediction method[J]. Expert Syst Appl 169:114400
30. Jia K, Zhang X (2019) Micro-blog retweeting prediction based on combined-features and random forest[C]. In: CCF conference on computer supported cooperative work and social computing. Springer, Singapore, pp 429–440
31. Ma R, Hu X, Zhang Q, et al (2020) Hot topic-aware retweet prediction with masked self-attentive model[C]. In: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, pp 525–534
32. Yin H, Yang S, Song X, et al (2020) Deep fusion of multimodal features for social media retweet time prediction[J]. World Wide Web, pp 1–18
33. Firdaus SN, Ding C, Sadeghian A (2021) Retweet prediction based on topic, emotion and personality[J]. Online Soc Netw Media 25:100165
34 Wang J, Yang Y (2021) Tweet retweet prediction based on deep multitask learning[J]. Neural Process Lett. https://doi.org/10.1007/s11063-021-10642-3
35. Yuan N J, Zhong Y, Zhang F, et al (2016) Who will reply to/retweet this tweet? The dynamics of intimacy from online social interactions[C]. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, pp 3–12
36. Zhang Q, Gong Y, Wu J, et al (2016) Retweet prediction with attention-based deep neural network[C]. In: Proceedings of the 25th ACM international on conference on information and knowledge management, pp 75–84
37. Steinskog A, Therkelsen J, Gambäck B (2017) Twitter topic modeling by tweet aggregation[C]. In: Proceedings of the 21st Nordic conference on computational linguistics, pp 77–86
38. Ray A, Rajeswar S, Chaudhury S (2015) Text recognition using deep BLSTM networks[C]. In: 2015 eighth international conference on advances in pattern recognition (ICAPR). IEEE, 1–6
39. Peters M, Neumann M, Iyyer M, et al (2018) Deep contextualized word representations[C]. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)
40. Duchi JC, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res 12:2121–2159
41. Machuca CR, Gallardo C, Toasa RM (2021) Twitter sentiment analysis on coronavirus: machine learning approach[C]. J Phys 1828(1):012104
42. Yang C, Tang J, Sun M, et al (2019) Multi-scale information diffusion prediction with reinforced recurrent networks[C]. In: IJCAI, pp 4033–4039
43. Wang Z, Chen C, Li W (2018) A sequential neural information diffusion model with structure attention[C]. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp 1795–1798
44. Hamilton W L, Ying R, Leskovec J (2017) Inductive representation learning on large graphs[C]. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp 1025–1035
45. Wang L, Zhang Y, Hu K (2021) FEUI: Fusion Embedding for User Identification across social networks[J]. Appl Int. https://doi.org/10.1007/s10489-021-02716-5