Neurocomputing 333 (2019) 221–230

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

NPP: A neural popularity prediction model for social media content

Guandan Chen[a,b], Qingchao Kong[a,b,*], Nan Xu[a,b], Wenji Mao[a,b]

[a] State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[b] University of Chinese Academy of Sciences, Beijing 101408, China

A R T I C L E I N F O

Article history:
Received 6 March 2018
Revised 13 October 2018
Accepted 16 December 2018
Available online 29 December 2018

Communicated by Dr. Min Xu

Keywords:
Social media
Popularity prediction
Deep neural network
Attention mechanism

A B S T R A C T

Online interactive behaviors between Web users often make some social media contents go viral. The popularity of social media contents can help us understand public interest and attention behind user interactions, thus popularity prediction of online contents has become a key task in social media analytics and can facilitate many applications in different domains. However, it is a difficult task for two main reasons. Firstly, popularity can be affected by many factors such as user, text content and time. Secondly, social media data is often noisy, which may degrade the performance of the prediction model. To overcome these difficulties, in this paper, we design a deep learning based popularity prediction model, which extracts and fuses the rich information of text content, user and time series in a data-driven fashion. To deal with the noise in social media data, we incorporate attention mechanism to focus on more informative parts and suppress noisy ones. Experiments on real world datasets demonstrate the effectiveness of our proposed model.

1. Introduction

The continuous development of social media contents and the interactive behaviors between people often make some user generated contents spread quickly through the Internet. The popularity of social media contents can help us understand public interest and attention behind user interactions, which has profound influences on social, economic and governmental activities. Thus, modeling and predicting the popularity of online contents has become an important research topic in social media analytics and is beneficial to many applications in public management, business and security-related domains. For example, it can support emergency management by knowing the impact of natural disasters, terrorisms and crimes [1]. In business domain, it can help analyze the trends as well as the concerns of people, and provide valuable information for systemic risk modeling [2].

However, popularity prediction is a nontrivial task. Firstly, the amounts of users and user generated contents in social media are huge and they are multi-dimensional. For example, there are hundreds of millions of monthly active users on the Twitter platform, and the relationships between users are very complicated. Secondly, social media data are often noisy. For instance, there are many informal expressions, word morphs in social media. Thirdly, popularity may be affected by many factors, such as text content, user and time, which are intertwined with each other during the cascade process.

Early studies on popularity prediction mainly adopt feature-based methods [3–7]. These methods extract a large number of features related to user attributes, user network, text content and time series, and then train machine learning models to predict popularity. They can get relatively good performance when features are effective. However, as the performances of these methods rely heavily on feature engineering, which is time consuming, labor intensive, and requires much expert knowledge. In the case that the selected features are not appropriate for the task, the model can be brittle. While early studies depend on simple regression models, some recent studies focus on modeling the time series of popularity using stochastic processes (e.g. Hawkes processes [8]) [8–13]. However, they solely use time series information for the prediction task, and ignore other valuable information for popularity prediction. Moreover, these methods rely on strong hypotheses about the formation of popularity.

Recently, inspired by the outstanding performances of deep learning models in many fields, several deep learning based popularity prediction methods are proposed [14–16]. In these methods, DeepCas [15] and DeepHawkes [16] only take user network information into consideration, and ignore interactions between different kinds of information (i.e. text, user and time series), while the method in [14] only uses deep neural network to extract features from videos. All these related methods either require feature engi-

* Corresponding author at: State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China.

E-mail addresses: chenguandan2014@ia.ac.cn (G. Chen), qingchao.kong@ia.ac.cn (Q. Kong), xunan2015@ia.ac.cn (N. Xu), wenji.mao@ia.ac.cn (W. Mao).

https://doi.org/10.1016/j.neucom.2018.12.039
0925-2312/© 2019 Elsevier B.V. All rights reserved.

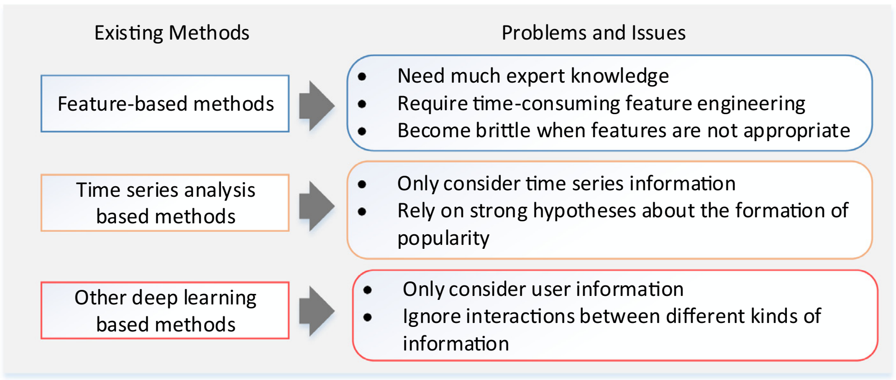
Existing Methods
Problems and Issues
Feature-based methods
Need much expert knowledge
Require time-consuming feature engineering
Become brittle when features are not appropriate
Time series analysis based methods
Only consider time series information
Rely on strong hypotheses about the formation of popularity
Other deep learning based methods
Only consider user information
Ignore interactions between different kinds of information

Fig. 1. Existing problems of popularity prediction.

neering, or only focus on part of the useful information for popularity prediction. In summary, Fig. 1 shows existing problems of popularity prediction models in the related work. To address the aforementioned challenges and overcome the weaknesses of existing models, we propose a deep neural network model to predict the popularity of social media content. It jointly models users participating in the online activities, user generated text contents as well as time series at an early stage. Our model takes multiple strategies to fuse different kinds of information by considering the interplay between them. Firstly, as users have different writing styles and word usage tendency, and users in the same community tend to have similar interests, we learn word and user embedding jointly. In addition, contents of different topics may have different popularity evolution patterns and different influential users. Thus, we adopt attention mechanism and a fusing layer to capture the interplay between different kinds of information.

The contributions of this paper are as follows. Firstly, we propose a neural model for popularity prediction, include time embedding and joint embedding of users and words. To focus on the informative parts, we also employ attention mechanism. Secondly, we propose three encoders by considering text content, user and popularity time series information for social media content. Thirdly, we use large social media datasets to show the effectiveness of our neural popularity prediction (NPP) model.

The rest of the paper is organized as follows. Section 2 reviews the related research on popularity prediction. Section 3 presents the details of our proposed method. In Section 4, we report our experimental study and discuss the results. Finally in Section 5, we conclude the paper and raise some future work.

2. Related work

Popularity reflects the level of public attention toward a Web content (e.g. a hashtag, an event, a video, etc.), which is quantitatively represented by a measure, either a numerical value (e.g. the number of related tweets, the number of views) or a range of values. Popularity prediction aims to predict this measure of public attention using the information at an early stage. Existing approaches to popularity prediction fall into four categories: feature engineering, time series analysis, cascade process analysis and deep learning based methods.

Feature engineering based methods mainly focus on feature design. The features used in the related work are mainly extracted from user attributes, user network, text content and time series at an early stage. Tsur and Rappoport [3] extract a lot of lexicon features as well as some topic, user and time series features, and then adopt different machine learning models to predict topic popularity in future days. Aiello et al. [4] predict the popularity of hashtags on the next day, using features such as language model divergence and topic distribution etc. Weng et al. [5] propose some community and user network features. To predict the popularity of news articles, Tsagkias et al. [6] propose many features, include surface features (e.g. publication time, length of the content and number of hyperlinks), accumulation features (e.g. number of articles at the same time), texture features (e.g. term frequency of keywords), semantic features (e.g. number of location entities, person entities and organization entities), and physics features (e.g. temperature). Bandari et al. [7] take the number of shares as the popularity of news articles. They adopt features such as the average popularity of the corresponding category and news source. They also consider the text subjectivity as well as named entities in the text. A major drawback of these methods is that they usually require domain knowledge and hand-crafted efforts so as to design useful features.

Time series analysis based methods mainly take advantage of the evolution information of popularity at an early stage. Early studies use a simple linear function to predict popularity [17]. Some other works use multivariate linear regression [18], ARMA model [19], or multiplicative seasonal Holt–Winters model [20]. These methods assume that future popularity has a linear or nonlinear relationship with early popularity. Some works assume that the popularity evolution processes can be described by several temporal patterns [18,21], and thus they use most similar popularity evolution processes to make predictions. Recently, many works model the spread of contents as stochastic processes [8–13]. Hawkes [8] is the state of the art method among them. It first employs Hawkes processes to model the spreading processes of contents, then uses both learned stochastic processes parameters and some other features to train a random forest model. Although these stochastic models elegantly explain the formation process of popularity, most of them only rely on time series information and ignore other important information for the prediction, with the exception of [8] using some other features.

Popularity prediction is similar to cascade size prediction, which aims to predict the cascade size in the future using the cascade state at an early stage. Cascade process analysis based methods predict popularity by modeling the cascade process in user network. Zhang et al. [22] simulate the spreading process of an event, using similarity between user history publications and the event content to compute user interest toward the event. DeepCas [15] constructs a cascade network by taking the users involved in the cascade as nodes and the following relationships as edges. It takes random walks on the cascade network. Then, it uses Recurrent Neural Network (RNN) with attention mechanism to extract features from the walk sequences for popularity prediction. These methods mainly consider information of user network, and do not make fully use of other information from text content and time series.

There are a few popularity prediction methods based on the deep neural network. Trzcinski et al. [14] adopt LRCN [23] to extract features from video content, and the extracted features are then used to predict the number of views at a future time. However, their method is limited to video popularity prediction. DeepCas [15] is also a deep neural network based method, but it only uses user network information. DeepHawkes [16] is an extension of Hawkes method. The original Hawkes method only uses the number of followers as user feature, while DeepHawkes uses user embedding vector and adopts RNN to encode cascade paths. However, DeepHawkes only consider user information and cascade path information. If we take every post or retweet as an element in a sequence, the propagation process can be viewed as a sequence. Among neural network models, RNN has gradually become the basis of many models dealing with sequence data, e.g. next basket prediction [24], machine translation [25] and dialogue system [26]. RNN has a lot of variations including LSTM [27] and GRU [28]. However, RNN assumes dependency changes monotonously along the whole sequence, while the time interval between actions may be varied in social media. Moreover, social media platform has time-varying activity level (depending on that it is working hours or rest time etc.), which is hard to be captured by ordinary RNN.

To address the above issues, in this paper, we propose a neural popularity prediction model for social media content, which adopts time embedding enhanced RNN and fuses text content, user as well as time series information. To focus on the informative parts of information and reduce noise, we incorporate attention mechanism in our model. We take Twitter as the representative social media platform, and focus on the prediction of the range of popularity, because for many applications, it is more practical than predicting the exact number. We also conduct experiment on real world datasets and experimental results show the effectiveness of our proposed model.

3. The proposed NPP model

In this section, we first formulate the popularity prediction problem. As our model uses vector representations of time, users and words, we then present our time embedding enhanced RNN, as well as our joint embedding model of users and words. We finally describe the details of our NPP model, which encodes information of social media content and predicts popularity.

3.1. Problem formulation

We define popularity as a numerical value related to a social media content (e.g. the number of retweets for a single tweet, or the number of tweets discussing an event). Here, a social media content refers to a tweet, a hashtag or an event. We transform the popularity prediction task to predicting whether the future popularity will exceed a given threshold. Specifically, given a social media content, we observe the discussion of the social media content during time period $[T_s, T_s + t_o]$, including related tweets, authors of the related tweets as well as the publication timestamps of the related tweets, where T_s is the publication time of the first tweet related to the social media content, and t_o measures how long we observe. Denoting the number of tweets about the social media content from T_s to $T_s + t_r$ ($t_r > t_o$) as V, where t_r is the lifecycle of the social media content, our goal is to predict whether V will exceed the threshold δ.

3.2. Time embedding enhanced RNN

Recently, RNN has been successfully employed to model sequence dependency in many tasks. However, ordinary RNN assumes monotonously temporal dependency when modeling the sequence data. In our popularity prediction task, adjacent tweet pairs may have different time intervals. In general, shorter time interval indicates more heated discussion about a social media content. However, ordinary RNN is hard to capture this difference. Moreover, users in social media may have different activity levels at different times, e.g. users may be more active at the rest time and not so active during working hours. Considering the above differences, we propose a time embedding enhanced RNN.

The input of an RNN is a sequence $[x_1, x_2, \ldots, x_n]$, where x_i is a vector that summarizes information at current step. At each step, an RNN takes the last state h_{i-1} and current input x_i to update the current state h_i, i.e.

$$h_i = f(x_i, h_{i-1})$$

where f is the update function for RNN. In our model, the input x_i is the vector representation of the i-th related tweet or its author and h_i is a vector that summarizes information at the i-th step as well as its context.

To capture the time information in modeling the sequence, we add a time embedding vector to every input of RNN. A time embedding vector consists of three parts:

(1) The embedding vector of the hour.
(2) The embedding vector of the weekday.
(3) The embedding of the distance to the time when we make prediction, denoted as $s(t_o - t_i)$, where s is the embedding function, t_o is the observation time, and t_i is the publication time of the i-th tweet (or retweet).

The embedding vectors of the hour and the weekday are optimized during the training process. There are several options for the embedding function $s(t_o - t_i)$, and we choose the cosine function for its effective representation ability. The value at the j-th position in the embedding vector would be

$$s_j(t) = \cos(\alpha j t),\ j \in [1, K]$$

The parameter α is optimized during training process, and K is an hyper-parameter, representing for the dimension of the embedding vector.

By considering time embedding, we change the original RNN update method into

$$h_i = f([x_i, te(t_i)], h_{i-1})$$

where $te(t_i)$ is the time embedding of input at the i-th step. Thus, at each step, the update of state will consider both the input and time information.

3.3. Joint embedding of users and words

An embedding model aims to learn a continuous vector representation for each word or user. There are many widely used word embedding models, e.g. Word2vec [29], Glove [30]. These word embedding models are learned from unlabeled text corpus by predicting co-occurrence relationships in a window, and can capture the semantics of words. There are also many network embedding models [31,32] that learn user embedding by predicting edges in user network or co-occurrence relationships in random walk sequences. These network embedding models can capture community or structure information of a user in the user network. However, it is hard to capture the relations between users and words by learning word embedding and user embedding in separate models. For instance, users with different interests may have different word usage tendency. Thus, we learn the user and word embedding in a joint model.

For our joint embedding model, there are three parts in the training objective, namely word co-occurrence objective L_w, user co-occurrence objective L_u, and word usage tendency objective L_{uw}.

For word embedding, we decompose each word into two vector representations. We call the two representations as "input" representation and "output" representation following Word2vec [29]. In word co-occurrence objective, we want to distinguish the co-occurrence word pairs from randomly sampled word pairs using these vector representations. Here, a co-occurrence word pair consists of two words that show in the same fix-length window in a sentence. Specifically, this objective tries to minimize

$$L_w = \log\sigma\left({v'_{w_O}}^\top v_{w_I}\right) + \sum_{i=1}^{k} E_{w_i\sim p_{noise}(w)} \log\sigma\left(-{v'_{w_i}}^\top v_{w_I}\right)$$

where w_O and w_I belong to a co-occurrence word pair, and w_i and w_I belong to a randomly sampled word pair, σ is sigmoid function, v'_{w_O}, v'_{w_i} are the "output" representations for word w_O and w_i respectively, while v_{w_I} is the "input" representation of the word w_I, k is the number of randomly sampled word pairs corresponding to each co-occurrence word pair, E represents for expectation, and w_i is a word randomly sampled from the noise distribution $p_{noise}(w)$. We use the same choice as in [29] for $p_{noise}(w)$, i.e. the unigram distribution raised to the 3/4rd power.

To capture user network information, we first take random walks on the user network to produce several user sequences, and learn user representation by predicting user co-occurrence in these sequences. To this end, the user co-occurrence objective tries to distinguish user co-occurrence pairs from randomly sampled user pairs using user representations, which is similar to the word co-occurrence objective:

$$L_u = \log\sigma\left({v'_{u_O}}^\top v_{u_I}\right) + \sum_{i=1}^{k} E_{u_i\sim p_{noise}(u)} \log\sigma\left(-{v'_{u_i}}^\top v_{u_I}\right)$$

where u_O and u_I belong to a co-occurrence user pair, and u_i and u_I belong to a randomly sampled user pair, v'_{u_O} and v'_{u_i} are "output" representations for user u_O and u_i, while v_{u_I} is the "input" representation for user u_I.

To capture word usage tendency of users, we jointly learn user and word embedding by predicting the author of a tweet. Specifically, we map each tweet and author into vector representations, then our embedding model tries to discriminate between the true author of the tweet and randomly sampled users. Let $[w_1, w_2, \dots, w_n]$ be the word sequence of the tweet, u_I be the true author of the tweet, and u_i be a randomly sampled user. The word usage tendency objective L_{uw} is computed as:

$$v_t = Att_{self}([v_{w_1}, v_{w_2}, \dots, v_{wn}])$$

$$L_{uw} = \log\sigma\left({v'_{u_I}}^\top v_t\right) + \sum_{i=1}^{k} E_{u_i\sim p_{noise}(u)} \log\sigma\left(-{v'_{u_i}}^\top v_t\right)$$

where Att_{self} represents for self-attention mechanism. It maps a sequence of vectors into one vector by weighted summing and we will introduce the detail of self-attention mechanism in the following section. $v'_{u_I}$and v'_{u_i} are "output" representations of user u_I and u_i, respectively, while v_{w_i} is the "input" representation of the i-th word in the word sequence. v_t is the vector representation of the tweet.

3.4. The structure of NPP model

The popularity of a social media content is affected by many factors, such as influential users, text content and time. Existing popularity prediction methods either require time-consuming feature engineering, or only focus on one part of information for popularity prediction (e.g. time series or user network). To overcome these drawbacks, we propose our neural popularity prediction model NPP.

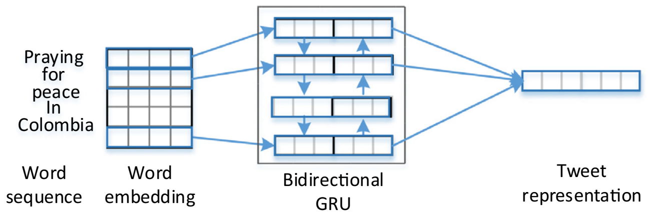

Fig. 2. Structure of tweet encoder.

The NPP model consists of four parts, namely, text content encoder, user encoder, time series encoder and fusing layer. The first three encoders are to learn text content, user and time series representations from data respectively. The fusing layer then combines features produced by these three encoders and outputs the prediction result.

3.4.1. Text content encoder

Each social media content consists of a sequence of related tweets or retweets, and each tweet/retweet is a sequence of words. Thus, we use a hierarchical neural network. At the first level, it uses a tweet encoder to encode each tweet into a vector representation, and at the second level, it obtains the text content representation from the sequence of tweet representations.

Tweet encoder. Fig. 2 shows the structure of the tweet encoder. We embed each word w^i_j in a tweet into a low dimension vector x^i_j using our joint embedding model, where w^i_j is the j-th word of the i-th tweet (or retweet). Then we use bidirectional Gated Recurrent Units (BiGRU) to encode the sequence. BiGRU maps a sequence to another sequence by considering context information of each word, i.e.

$$[h^i_1, h^i_2, \dots, h^i_n] = BiGRU([x^i_1, x^i_2, \dots, x^i_n])$$

where each h^i_j summarizes the context information of the word w^i_j. To get the representation of a tweet, we introduce attention mechanism to extract and aggregate the most important states for popularity prediction. This operation produces one single vector by weighted summing the sequence of vectors, denoted as

$$s'_i = Att_{self}([\mathrm{h}^i_1, h^i_2, \dots, h^i_n])$$

where $s_i{}'$ is a vector representation for the i-th tweet and Att_{self} represents for self-attention mechanism. The detail of self-attention mechanism is as follows.

Self-attention mechanism. Attention mechanism learns to focus on more informative parts and suppress noise, and maps a sequence of states into a vector representation by weighted summing the states in the sequence. The weight can be interpreted as attention weight on the corresponding state in a neural metaphor. Specifically, let $[h_1, h_2, \dots, h_n]$ be a sequence with length n. The self-attention mechanism outputs a vector v using

$$u_i = \tanh(Wh_i + b)$$

$$\alpha_i = \frac{\exp(u_s^\top u_i)}{\sum_t \exp(u_s^\top u_i)}$$

$$v = \sum_i \alpha_i h_i$$

where each state h_i is mapped into a hidden space to get u_i. The attention weight α_i is the normalized value of the dot product between u_i and u_s. Here, u_s is a parameter optimized during training.

Text content encoder. After applying tweet encoder to each tweet, we can get the tweet vector representations, denoted as $[s'_1, s'_2, \dots, s'_n]$. However, this representation does not contain time information. To capture time information of tweets, we adopt the time embedding enhanced RNN. Its input at the i-th step is the concatenation of the time embedding and the tweet representation s_i', i.e.

$$s_i = \left[s_i', te(t_i)\right]$$

where t_i represents for the publish time of the i-th tweet, and $te(t_i)$ is the time embedding of t_i.

Then we use a bidirectional GRU and attention mechanism to get the text content representation as follows:

$$\left[h_1^{tc}, h_2^{tc}, \ldots, h_n^{tc}\right] = BiGRU([s_1, s_2, \ldots, s_n])$$

$$v_{tc} = Att_{self}\left(\left[h_1^{tc}, h_2^{tc}, \ldots, h_n^{tc}\right]\right)$$

where h_i^{tc} is the i-th state produced by the GRU in the text content encoder, and v_{tc} is a vector representation for text content. Att_{self} is self-attention mechanism.

3.4.2. User encoder

In the user encoder, we use time-embedding enhanced RNN and attention mechanism to model the user sequence. Each user is represented as a low dimension vector U_i' using our joint embedding model, which encodes the structure, community and interest information of the user in the network. Let U_i be the concatenation of U_i' and the time embedding, then we use bidirectional GRU and an attention layer to get a higher level representation of the user sequence. Specifically, the representation of the user sequence is computed from

$$[h_1^u, h_2^u, \ldots, h_n^u] = BiGRU([U_1, U_2, \ldots, U_n])$$

$$v_u = Att_{self}([h_1^u, h_2^u, \ldots, h_n^u])$$

where h_i^u is the i-th state produced by the bidirectional GRU in the user encoder, and v_u is a vector representation for users.

3.4.3. Time series encoder

We divide the early observation time $[T_s, T_s + t_o]$ into m time windows with fixed window width Δt. Each time widow corresponds to one feature vector, denoted as f_i. The feature vector f_i includes the number of tweets in this time window, average and maximum follower number of authors of these tweets, and time embedding. Thus the whole time series is represented as $[f_1, f_2, \ldots, f_m]$. We again use the bidirectional GRU and attention mechanism to map this sequence of feature vectors to a vector representation, i.e.

$$\left[h_1^{ts}, h_2^{ts}, \ldots, h_m^{ts}\right] = BiGRU([f_1, f_2, \ldots, f_m])$$

$$v_{ts} = Att_{self}\left(\left[h_1^{ts}, h_2^{ts}, \ldots, h_m^{ts}\right]\right)$$

where h_j^{ts} is the j-th state produced by the bidirectional GRU in the time series encoder, and v_{ts} represents for a vector representation for time series information.

3.4.4. Fusing layer

The above encoded text content, user and time series representations are concatenated into one feature vector after batch normalization [33] in the fusing layer. We choose batch normalization here because features from different kinds of information may be in different distributions. Let $\tilde{v}_{tc}, \tilde{v}_u, \tilde{v}_{ts}$ be the normalized vector representations of text content, user sequence and time series respectively. The fusing vector representation is computed as follows:

$$v_f = [\tilde{v}_{tc}, \tilde{v}_u, \tilde{v}_{ts}]$$

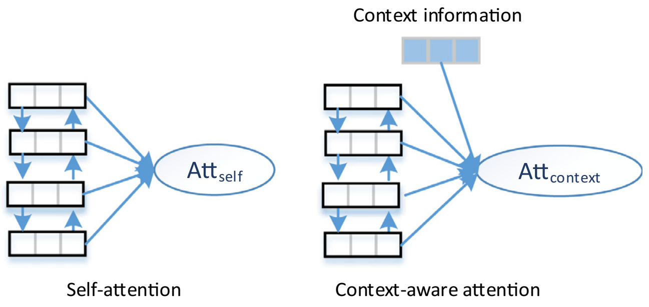

Fig. 3. Difference between self-attention and context-aware attention.

Considering the interplays between different kinds of information as described in the earlier sections, we further adopt context-aware attention mechanism to refine feature representations. Specifically, the refined text content representation v_{tc}', user representation v_u' and time series representation v_{ts}' are computed respectively as follows:

$$v_{tc}' = Att_{context}\left(\left[h_1^{tc}, h_2^{tc}, \ldots, h_n^{tc}\right], v_f\right)$$

$$v_u' = Att_{context}\left([h_1^u, h_2^u, \ldots, h_n^u], v_f\right)$$

$$v_{ts}' = Att_{context}\left(\left[h_1^{ts}, h_2^{ts}, \ldots, h_m^{ts}\right], v_f\right)$$

where h_i^c, h_i^u are the i-th state produced by bidirectional GRUs in the text content encoder and the user encoder respectively. h_j^t is the j-th state produced by the bidirectional GRU in the time series encoder. And $Att_{context}$ is context-aware attention mechanism. The detail of context-aware attention mechanism is as follows.

Context-aware attention mechanism. In the self-attention mechanism, we only use the information from the state itself, while context-aware attention mechanism takes other context information to compute attention weight, as Fig. 3 shows. This is important for popularity prediction for the following reasons:

(1) A specific topic may be more welcome in some communities, and thus be more popular.
(2) Users play different roles in the cascade process in different kinds of events, for example, a user may be more influential in the political area, but less influential in the economic area.
(3) Different topics and different communities may have different activity levels, resulting in different popularity evolving processes.

Therefore, we employ context-aware attention mechanism to capture the interplays between different kinds of information.

The output of context-aware attention mechanism is computed as

$$u_i = \tanh\left(Wh_i + W_c v_{context} + b\right)$$

$$\alpha_i = \frac{\exp(u_s^\mathsf{T} u_i)}{\sum_t \exp(u_s^\mathsf{T} u_i)}$$

$$v = \sum_i \alpha_i h_i$$

where $v_{context}$ is the context vector, which summarizes the external information. And the representation of the whole sequence is v, which is obtained by weighted summing the states.

After obtaining the refined feature representations v_{tc}', v_u' and v_{ts}', we adopt a dense layer to encode the social media content feature further, i.e.

$$v_m' = \sigma\left(W_{ds}\left[v_{tc}', v_u', v_{ts}'\right] + b_{ds}\right)$$

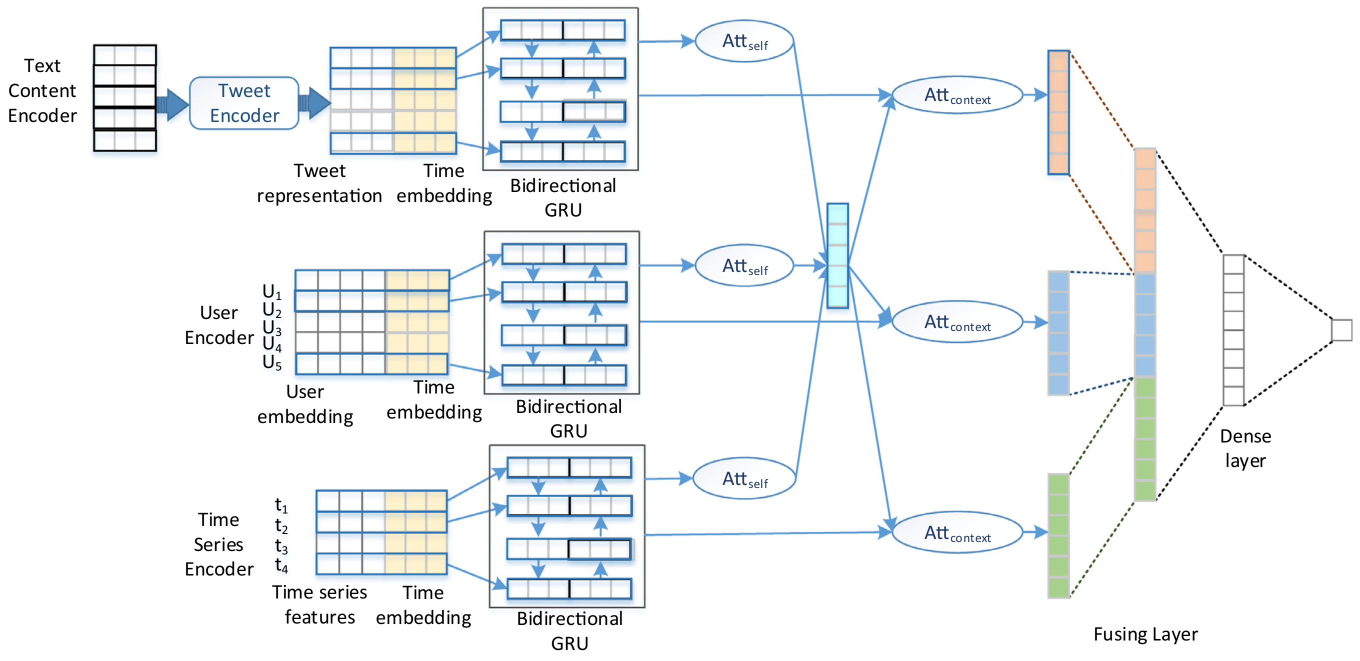

Fig. 4. Structure of the NPP model.

where σ represents for sigmoid function. W_{ds} and b_{ds} are trainable weights.

Then another dense layer is used to produce popularity prediction result, i.e.

$$\hat{y} = \sigma\left(W_p v_m' + b_p\right)$$

The larger $\hat{y}$ is, the higher probability we believe that the event will going to be popular. W_p and b_p are trainable weights.

3.5. Optimization

We use cross-entropy as our loss function, i.e.

$$L = -\sum_{i=1}^{M} \left(y_i log\ \hat{y}_i + (1 - y_i)\log\left(1 - \hat{y}_i\right)\right).$$

Here, y_i and $\hat{y}_i$ are the true label and prediction result of the i-th social media content respectively. M is the total number of social media contents. L is the loss we want to optimize. We adopt Adagrad [34] to optimize the parameter weights, which is a widely used optimization method. To avoid overfitting, we choose L1 regularization with 1e-8 for the dense layer. We also adopt early stopping strategy in our training process.

Fig. 4 summarizes the structure of our proposed model NPP. The NPP model uses three encoders to learn text content, user and time series representations from data respectively. The fusing layer then combines features produced by these three encoders and outputs the prediction result.

4. Experiment

4.1. Datasets and experimental setting

To compare our proposed model with methods from related work, we collected two datasets from Twitter, which is one of the largest social media platforms. Table 1 shows the summary statistics of our datasets.

Retweet dataset: This dataset was collected using the Twitter public API (https://developer.twitter.com) from Aug. 9, 2016 to Dec. 10, 2016. We filter out tweets with less than 5 retweets and sample 20,000 tweets as our dataset. Each sample corresponds a tweet, which contains the tweet content, retweets of this tweet during observation time, the authors of these retweets, and publication timestamps of these retweets. We measure the popularity of a tweet using the number of its retweets.

Event dataset: This dataset was also collected using Twitter public API from Aug. 9, 2016 to Dec. 10, 2016. Users usually use a hashtag to denote the event that they discuss, but sometimes one event may correspond to several hashtags, and some hashtags may not discuss real world events. To ensure the quality of the dataset, we manually remove non-event hashtags and merge hashtags that discuss the same event. In this dataset, each sample corresponds to an event, including tweets discussing the event during observation time, the authors of these tweets and publication timestamps of these tweets. We measure the popularity of an event using the number of its related tweets.

We set the threshold to regard a sample as popular or not according to the implication of Pareto Principle (or 80–20 rule) [35], which indicates that roughly 80% of people focus on around 20% of contents in social media. It has been used in many popularity prediction works [36,37]. Therefore, in our datasets, the top 20% most popular source tweets/events are considered as “popular”, and we randomly sample the same number of source tweets/events from the rest as “unpopular”. For both datasets, the observation time t_o is set to 1, 6, 12 or 24 h, respectively. The fixed window width Δt in the time series encoder is set to 10 min. In our experiment, the training set contains 70% of the whole datasets, the validation set contains 10%, and the test set contains 20%.

4.2. Baseline methods

We compare our method to the following baseline methods and use accuracy as the evaluation metric.

(1) *Tsur's* [3]: This method mainly uses lexicon features, associated with a few user, user interaction and time series features to predict popularity.

Table 1
Statistics of datasets.

Retweet dataset	#source tweets	#retweets	avg. #retweets per source tweet	max #retweets per source tweet
	20,000	915,139	46	1910
Event	#events	#tweets	avg. #tweets per event	max #tweets per event
dataset	41,035	4,561,375	111	234,944

Table 2
Comparison with baseline methods.

Method	Retweet dataset				Event dataset			
	Observation time (h)				Observation time (h)			
	1	6	12	24	1	6	12	24
Tsur's	0.757	0.794	0.810	0.843	0.612	0.653	0.682	0.722
Aiello's	0.804	0.851	0.876	0.901	0.633	0.701	0.728	0.781
Hawkes	0.733	0.732	0.740	0.745	0.590	0.669	0.709	0.749
DeepHawkes	0.782	0.854	0.870	0.903	0.588	0.691	0.725	0.791
DeepCas	0.794	0.853	0.875	0.893	0.580	0.658	0.707	0.780
NPP	**0.828**	**0.865**	**0.889**	**0.911**	**0.698**	**0.743**	**0.768**	**0.824**

(2) *Aiello's* [4]: This work proposes many novel features based on language model, network structure and time series to predict popularity.
(3) *Hawkes* [8]: This method models popularity dynamics as Hawkes processes, then combines Hawkes processes parameters, a few user and text content features to predict popularity.
(4) *DeepHawkes* [16]: This deep neural network based method is an extension of Hawkes method. The original Hawkes method [8] only uses the number of followers as user feature, while DeepHawkes uses user embedding and adopts GRU to encode cascade paths.
(5) *DeepCas* [15]: This method focuses on the cascade process of a social media content. It takes random walks on the cascade graph to produce a large number of sequences, and then uses GRU and attention mechanism to predict the popularity.

4.3. Results

4.3.1. Comparison with baseline methods

Table 2 shows the prediction accuracies of our proposed model NPP and other comparison methods. As shown in Table 2, NPP outperforms all the baseline methods: in the retweet dataset, NPP improves accuracies by around 1% compared to the best baseline method; in the event dataset, NPP improves accuracies by about 4% under all observation time settings.

Aiello's method and Tsur's method only use hand-crafted features, while our method learns the high-level feature representations in a data-driven manner, which shows the superiority of the deep learning based methods.

All models have lower accuracies when the observation time is short, because there is little information for popularity prediction, especially for Hawkes, DeepHawkes and DeepCas methods. However, as observation time gets longer, accuracies of these methods can be very close to Aiello's method. Different from all other methods, our proposed NPP model combines text content, user and time series information, which enables NPP to have a good performance in all observation time settings.

4.3.2. Comparison with NPP variations

Effectiveness of different encoders: To show the effectiveness of different encoders, we compare with the following variations of our model using the event dataset. We only use the event dataset for evaluation in the following sections, because we find the event dataset is more difficult according to Table 2.

Table 3
Comparison with NPP variations.

Method	Observation time (h)			
	1	6	12	24
Text	0.662	0.718	0.758	0.809
User	0.650	0.704	0.742	0.797
Time series	0.679	0.712	0.749	0.798
Text + user	0.660	0.731	0.768	0.822
Text + time series	0.695	0.740	0.758	0.805
User + time series	0.692	0.735	0.759	0.809
NPP	**0.698**	**0.743**	**0.768**	**0.824**

(1) *text*: Only uses features produced by the text content encoder to predict popularity.
(2) *user*: Only uses features produced by the user encoder to predict popularity.
(3) *time series*: Only uses features produced by the time series encoder to predict popularity.
(4) *text + user*: Combines the text content encoder and the user encoder.
(5) *text + time series*: Combines the text content encoder and the time series encoder.
(6) *user + time series*: Combines the user encoder and the time series encoder.

Table 3 shows the popularity prediction performances of different variations of our proposed model. Among the three encoders, time series encoder achieves the best performance when the observation time is 1 h, but text content encoder performs best when the observation time gets longer. This shows that time series information is important when the observation time is short, but text information is more powerful when observation time gets longer. By comparing text + user and user + time series, we can have similar conclusions. By combining all three encoders, we can improve accuracies by 2–3% compared to individual encoders.

Effectiveness of attention mechanism: We compare different attention mechanism strategies using the event dataset in Table 4. The comparison methods are as follows:

(1) *NPP (Without attention)*: Removes attention mechanism from NPP.
(2) *NPP (Self-attention)*: Only uses text content information, user information, and time series information to compute attention weights in three encoders without context-aware information.

Table 4
Comparisons of models with different attention mechanism settings.

Method	Observation time (h)			
	1	6	12	24
NPP (Without attention)	0.678	0.735	0.755	0.804
NPP (Self-attention)	0.696	0.733	0.762	0.814
NPP	**0.698**	**0.743**	**0.768**	**0.824**

Table 5
NPP with/without time embedding.

Method	Observation time (h)			
	1	6	12	24
NPP (without time embedding)	0.624	0.705	0.742	0.794
NPP (with time embedding)	**0.698**	**0.743**	**0.768**	**0.824**

Table 6
Comparison with different word and user embedding models.

Model	Observation time (h)			
	1	6	12	24
Random	0.684	0.708	0.755	0.800
Word2vec	0.686	0.731	0.760	0.804
Node2vec	0.692	0.723	0.762	0.816
Word2vec + Node2vec	0.693	0.735	0.763	0.817
NPP	**0.698**	**0.743**	**0.768**	**0.824**

The experimental results show that introducing attention mechanism can improve accuracies by around 2%. With context–aware attention mechanism, NPP can improve the accuracies by around 1% compared to NPP (Self-attention), which considers context information when computing attention weights.

Effectiveness of embedding model: We compare the NPP with time embedding and without time embedding in Table 5. The experimental results show that our time embedding model is effective by improving the accuracy by about 3%.

To show the effectiveness of joint embedding of words and users, the following embedding models are taken into consideration for comparison.

(1) *Random*: Randomly initializes embedding of users and words, and optimize the embedding vectors during training popularity prediction model.
(2) *Word2vec*: Uses Word2vec as word embedding model, while user embedding vectors are randomly initialized and optimized during training popularity prediction model.
(3) *Node2vec*: Uses Node2vec as user embedding model, while word embedding vectors are randomly initialized and optimized during training popularity prediction model.
(4) *Word2vec + Node2vec*: Uses Word2vec as word embedding model and Node2vec as user embedding model. Word2vec + Node2vec trains word embedding and user embedding separately, while out proposed model NPP trains them jointly.

Table 6 shows the comparison results of different embedding models. The Random initialized model performs worse than other comparison methods, which shows that appropriate user and word embedding are important for popularity prediction. Through jointly modeling users and words, NPP model further improves accuracies by more than 0.5% compared to Word2vec + Node2vec.

4.4. Illustrations of learned feature representations

As for text content encoder, we compare different attention weights for words and tweets. Specifically, we select several tweets with high or low attention weights, and highlight two to three words with high attention weights in the tweet (see Table 7).

We find that the words with large attention weights seem to be reasonable. Firstly, words that express emotions usually have higher attention weights, e.g. “please”, “lord”, “horrible”. It is interesting that these words may not be adjective, e.g. “could u imagine”. Secondly, some verbs (e.g. “lost”) and named entities (e.g. “Amnesty International”) also have higher attention weights. Some of these words may relate to hot topics recently, which inspires us to update the popularity prediction model regularly. As for tweets, it seems that tweets with high attention weights usually carry strong emotions or may cause strong emotion.

Next, we qualitatively study the connection between learned user representations and some well-known network properties. For each event, we construct a user local network from the users who participate in the discussion using their following relationships. Then we layout the local networks to a 2-D space by feeding

Table 7
Attention weights in text content encoder.

Topic 1: #PrayForLouisiana	
Tweet Attention weights	Word Attention Weights (We highlight the words with high attention weights)
high	RT this is awful **please** pray
	Even though we ’ve **lost** everything glad **to make** it out safe and help rescue people in the process
	Lord we **’ve** been so divided in our state and country Let us all come together
	More caskets coming up Could u **imagine** getting a call **that** you have to rebury a family member
low	And the flooding continues....
	we went from sweating in 100° weather to our streets being flooded
	United states is about to lose a state bc Louisiana about to float away
	My momma friend house This rain aint playing at all!!
Topic 2: #SiegeKillsYemenis	
Tweet Attention weights	Word Attention Weights (We highlight the words with high attention weights)
high	**Amnesty International accused** the US government of deadly hypocrisy
	21 Million **Yemenis** in **urgent** need of humanitarian aid
	SHOCKING US made bombs falling on and **killing** civilians
	Child starving dying of hunger **screaming** in pain appealing to conscience of the world
low	the sick ones first
	Yemeni governorates have seen strikes on agricultural & food production infrastructure
	“Yemen’s access food threatened, supply route targeted by airstrikes” #SiegeKillsYemenis
	“Is Yemen Too Much for the World to Take?”

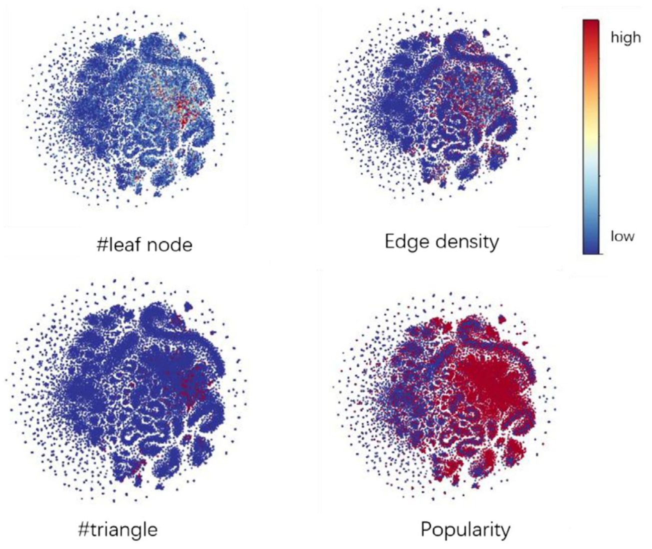

Fig. 5. Visualization of user feature.

learned representations from user encoder to t-SNE [38], which can model similar objects by nearby points. In Fig. 5, each point corresponds to the user local network of an event, and each subfigure is colored using a hand-crafted network feature. If we can observe some patterns of the distribution of colors in the figure, it suggests some connection between the learned representations and hand-crafted features.

As Fig. 5 shows, our learned representation could capture some hand-crafted network features. For example, the local networks with a large number of leaf nodes distribute on the center right part of the sub figure. By observing the distribution of events with high popularity, we also found that these hand-crafted features have some connection to the future popularity.

5. Conclusions

In this paper, we propose a neural popularity prediction model for social media contents. The proposed model consists of three encoders, which learn high-level representations of text content, user and time series in a data-driven approach. Attention mechanism is introduced to make the model focus more on informative parts and suppress noisy ones. We also propose time embedding enhanced RNN to capture different time interval of tweets, and time-varying activity level of social media platform. Experimental results show that the popularity prediction model can benefit from time embedding, as well as joint learning embedding of users and words. The illustrations further demonstrate that our model can assign reasonable attention weights on text content, and the learned user representations have some connection to handcrafted network features. In general, the empirical studies verify the effectiveness of our proposed model compared to the baseline methods.

For future work, we shall explore more information in social media for popularity prediction. As currently we mainly take the publications of tweets as a sequence, we shall consider the utilization of user interaction behaviors such as retweet and mention. We shall also consider the interactions between different social media contents and different platforms.

Acknowledgments

This work is supported in part by the National Key R&D Program of China under Grant #2016QY02D0305, National Natural Science Foundation of China under Grants #71702181, #71621002 and #11832001, and National Science and Technology Major Project under Grant #2018ZX10201001.

References

[1] D. Pohl, A. Bouchachia, H. Hellwagner, Online indexing and clustering of social media data for emergency management, Neurocomputing 172 (2016) 168–179.
[2] P. Cerchiello, P. Giudici, G. Nicola, Twitter data models for bank risk contagion, Neurocomputing 264 (2017) 50–56.
[3] O. Tsur, A. Rappoport, What's in a hashtag? Content based prediction of the spread of ideas in microblogging communities, in: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, ACM, 2012, pp. 643–652.
[4] L.M. Aiello, et al., Sensing trending topics in Twitter, IEEE Trans. Multimed. 15 (6) (2013) 1268–1282.
[5] L. Weng, F. Menczer, Y.-Y. Ahn, Predicting successful memes using network and community structure, in: Proceedings of International Conference on Web and Social Media, ICWSM, 2014.
[6] M. Tsagkias, W. Weerkamp, M. De Rijke, Predicting the volume of comments on online news stories, in: Proceedings of the 18th ACM Conference on Information and Knowledge Management, ACM, 2009, pp. 1765–1768.
[7] R. Bandari, S. Asur, B.A. Huberman, The pulse of news in social media: forecasting popularity, in: Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media, 2012.
[8] S. Mishra, M.-A. Rizoiu, L. Xie, Feature driven and point process approaches for popularity prediction, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, ACM, 2016, pp. 1069–1078.
[9] H. Shen, D. Wang, C. Song, Modeling and predicting popularity dynamics via reinforced Poisson processes, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014, pp. 291–297.
[10] P. Bao, H.W. Shen, X. Jin, X.Q. Cheng, Modeling and predicting popularity dynamics of microblogs using self-excited Hawkes processes, in: Proceedings of International Conference on World Wide Web, 2015, pp. 9–10.
[11] Q. Zhao, M.A. Erdogdu, H.Y. He, A. Rajaraman, J. Leskovec, SEISMIC: a self-exciting point process model for predicting tweet popularity, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 1513–1522.
[12] M.A. Rizoiu, L. Xie, S. Sanner, M. Cebrian, H. Yu, P.V. Hentenryck, Expecting to be HIP: Hawkes Intensity Processes for social media popularity, in: Proceedings of International Conference on World Wide Web, 2017, pp. 735–744.
[13] B. Samanta, A. De, A. Chakraborty, N. Ganguly, LMPP: a large margin point process combining reinforcement and competition for modeling hashtag popularity, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, 2017, pp. 2679–2685.
[14] T. Trzcinski, P. l. Andruszkiewicz, T. Bochenski, and P. law Rokita, "Recurrent Neural Networks for Online Video Popularity Prediction," arXiv:1707.06807, 2017.
[15] C. Li, J. Ma, X. Guo, Q. Mei, DeepCas: an end-to-end predictor of information cascades, in: Proceedings of the 26th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2017, pp. 577–586.
[16] Q. Cao, H. Shen, K. Cen, W. Ouyang, X. Cheng, DeepHawkes: bridging the gap between prediction and understanding of information cascades, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, ACM, 2017, pp. 1149–1158.
[17] G. Szabo, B.A. Huberman, Predicting the popularity of online content, Commun. ACM 53 (8) (2010) 80–88.
[18] H. Pinto, J.M. Almeida, M.A. Gonçalves, Using early view patterns to predict the popularity of youtube videos, in: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, ACM, 2013, pp. 365–374.
[19] G. Gürsun, M. Crovella, I. Matta, Describing and forecasting video access patterns, in: Proceedings of IEEE INFOCOM, 2011, IEEE, 2011, pp. 16–20.
[20] Y. Hu, C. Hu, S. Fu, P. Shi, B. Ning, Predicting the popularity of viral topics based on time series forecasting, Neurocomputing 210 (2016) 55–65.
[21] S. Kong, F. Ye, L. Feng, Predicting future retweet counts in a microblog, J. Comput. Inf. Syst. 10 (4) (2014) 1393–1404.
[22] X. Zhang, X. Chen, Y. Chen, S. Wang, Z. Li, J. Xia, Event detection and popularity prediction in microblogging, Neurocomputing 149 (2015) 1469–1480.
[23] J. Donahue, et al., Long-term recurrent convolutional networks for visual recognition and description, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2625–2634.
[24] F. Yu, Q. Liu, S. Wu, L. Wang, T. Tan, A dynamic recurrent model for next basket recommendation, in: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, ACM, 2016, pp. 729–732.
[25] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, in: Proceedings of Advances in Neural Information Processing Systems, 2014, pp. 3104–3112.
[26] I.V. Serban, A. Sordoni, Y. Bengio, A.C. Courville, J. Pineau, Building end-to-end dialogue systems using generative hierarchical neural network models, in: Proceedings of AAAI, 2016, pp. 3776–3784.
[27] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[28] J. Chung, C. Gulcehre, K.H. Cho, and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” arXiv:1412.3555, 2014.
[29] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of Advances in Neural Information Processing Systems, 2013, pp. 3111–3119.
[30] J. Pennington, R. Socher, C.D. Manning, Glove: global vectors for word representation, in: Proceedings of Empirical Methods in Natural Language Processing, EMNLP, 14, 2014, pp. 1532–1543.
[31] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 701–710.
[32] A. Grover, J. Leskovec, Node2vec: scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 855–864.
[33] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: Proceedings of the 32nd International Conference on Machine Learning, ICML-15, 2015, pp. 448–456.
[34] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (7) (2011) 257–269.
[35] M.E. Newman, Power laws, Pareto distributions and Zipf's law, Contemp. Phys. 46 (5) (2005) 323–351.
[36] P.J. McParlane, Y. Moshfeghi, J.M. Jose, Nobody comes here anymore, it's too crowded; predicting image popularity on flickr, in: Proceedings of International Conference on Multimedia Retrieval, ACM, 2014, p. 385.
[37] S. Cappallo, T. Mensink, C.G. Snoek, Latent factors of visual popularity prediction, in: Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, ACM, 2015, pp. 195–202.
[38] L.v.d. Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (2008) 2579–2605 no. Nov.

Chen Guandan received his B.S. degree in automation from Xiamen University. He is currently working towards his Ph.D. degree at the Institute of Automation, Chinese Academy of Sciences. His research interests include deep neural network, social media analytics and data mining.

Kong Qingchao received his Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences. He is currently assistant professor at the Institute of Automation, Chinese Academy of Sciences. His research interests include popularity modeling and prediction, user behavior modeling, social media analytics and data mining.

Xu Nan received his B.S. degree in automation from North China Electric Power University (Baoding). He is currently working towards his Ph.D. degree at the Institute of Automation, Chinese Academy of Sciences. His research interests include multimodal sentiment analysis and rumor detection.

Mao Wenji received her Ph.D. degree in computer science from the University of Southern California. She is currently professor at the Institute of Automation of the Chinese Academy of Sciences and member of the State Key Laboratory of Management and Control for Complex Systems. She is also professor at the University of Chinese Academy of Sciences. Her research interests include artificial intelligence and social computing.