Exploring Trends and Patterns of Popularity Stage Evolution in Social Media

Qingchao Kong, Wenji Mao, Guandan Chen, and Daniel Zeng, *Fellow, IEEE*

***Abstract*—The popularity of online contents in social media frequently experiences ebb and flow, and thus its evolution often involves different stages, such as burst and valley. Exploring the patterns of popularity evolution, especially how burst forms and decays, and even further, predicting the trends of popularity evolution is both an important research topic and beneficial to support decision making for many applications, such as emergency management, business intelligence, and public security. Previous work on popularity prediction has focused on predicting the popularity volume of online contents, and at most, popularity burst and ignored the exploration of popularity evolution and the prediction of its stages. To fill this gap, in this paper, we propose our method for the popularity stage prediction problem both at the microscopic level and macroscopic level. At the microscopic level, we first extract multiple dynamic factors and infer future evolution stage by considering the contributions of different dynamic factors. At the macroscopic level, we extract the overall evolution patterns of popularity stages and adopt a pattern matching-based method to predict future popularity stages. We evaluate the proposed approach using tweets in SinaWeibo, the most popular Twitter-like social media platform in China. The experimental results show the effectiveness of our proposed approach in predicting popularity evolution stages.**

***Index Terms*—Online contents, popularity evolution, popularity stage prediction (PSP), social media analytics.**

I. Introduction

In RECENT years, social media sites have become important platforms for Web users to share information and express opinions. Social media platforms provide users various ways to interact with each other and their interactive behavior often makes some online contents become more *popular* than others. The popularity of online contents reflects how much user attention certain media contents can draw, which frequently experiences ebb and flow. Thus, the popularity of online contents often dynamically evolves over time and involves several different stages (such as burst and valley) and evolution patterns [1], [2].

Exploring the patterns of popularity evolution, especially how burst forms and decays, and even further, predicting the trends of popularity evolution is both an important research topic and beneficial to decision making for many applications, such as emergency management, business intelligence, and public security. For example, marketing campaigns [3], [4] in social networks can reach a much larger number of potential customers at a relatively low cost. For example, if the popularity stage of the promoted product is predicted to be in stage "valley" or "fall" soon, the campaign organizers can be signaled to take necessary actions to avoid the drop of user attention, depending on how serious the situation is. In emergency management situations, if the popularity stage of a social unrest is predicted to be in stage "rise" or "burst" soon, the government can initiate emergency planning in response to the specific predicted stage. To summarize, similar situations in social marketing, emergency management, and many other applications require deeper exploration of popularity evolution and prediction of its stages in support of better decision making.

Modeling and predicting the popularity of online contents is one of the main research issues in Web data analytics as well as many Web-based research areas [5]. Most of the previous research on popularity modeling and prediction have focused on predicting the total volume or level of popularity [6]–[10] and ignored the exploration of popularity evolution process and its trends, with the exception of studying a prominent stage, burst. Burst is usually referred to a certain feature rising sharply in frequency [11], [12], indicating sudden changes in social, economic, public, and/or personal status. Predicting the outbreak of a burst [13] and burst time [14] can facilitate many applications, such as monitoring of social network behavior [15] and public reactions [11]. However, burst is not generally the only stage in the popularity evolution process of online contents. Other distinct stages, such as valley, rise, and fall, are indispensable for exploring the trends and patterns of popularity evolution.

In this paper, we aim at modeling the dynamic evolution and focus on predicting multiple popularity stages of online contents in social media. Specifically, to capture the dynamic changes of the states in popularity evolution, we focus on the modeling of dynamic aspects of evolution process. To

Manuscript received April 1, 2018; accepted June 28, 2018. This work was supported in part by the Ministry of Science and Technology of China under Grant 2016QY02D0305, in part by the National Natural Science Foundation of China under Grant 71702181 and Grant 71621002, and in part by the Key Program of the Chinese Academy of Sciences under Grant ZDRW-XH-2017-3. This paper was recommended by Associate Editor E. Chen. *(Corresponding author: Wenji Mao.)*

Q. Kong and G. Chen are with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: qingchao.kong@ia.ac.cn; chenguandan2014@ia.ac.cn).

W. Mao and D. Zeng are with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: wenji.mao@ia.ac.cn; dajun.zeng@ia.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2018.2855806

predict popularity stages, including *burst*, *valley*, *rise*, and *fall*, we develop our approach both at the microscopic level and macroscopic level. At the microscopic level, as different dynamic factors have different contributions to the prediction problem, we propose an algorithm to combine the contributions of different dynamic factors. At the macroscopic level, as patterns capture the overall dynamic variations of popularity evolution, we make use of these patterns for popularity stage prediction (PSP).

This paper has made several contributions. We consider the evolution of popularity of online contents and first address the PSP problem. We consider the dynamic aspects of popularity evolution and propose our approach to the problem by identifying various dynamic factors and incorporating their contributions into the proposed algorithm. We also extract popularity stage patterns and employ them to further improve the overall performance of PSP.

II. Related Work

Popularity of online contents, such as tweets, threads, videos, images, is usually defined using some "numeric" measurements. For example, popularity of tweets is defined by the number of retweets or comments. The specific definition of popularity of online contents depends on the application context. In this paper, we define the popularity of a SinaWeibo tweet as the number of retweets, as the retweeting mechanism is the main driving force of information diffusion [16] in social networks.

The vast majority of previous work on popularity have focused on the prediction of popularity volume of online contents [5]. One of the early work on popularity prediction is the S-H model proposed by Szabo and Huberman [6]. The S-H model assumes that the (log-transformed) popularity at the early stage is linearly correlated with popularity at the future stage. Another early work is done by Borghol *et al.* [8]. They conduct a comprehensive analysis of the "content-agnostic factors" about the popularity of Youtube videos, and have similar findings: the most significant content-agnostic factors are the total number of previous views and the video age. However, these work only focus on the numerical factors of popularity at the early and future time, and ignore other important factors that influence popularity prediction.

Recent work on popularity modeling methods can be divided into three categories, namely the discriminative methods, generative methods, and deep learning-based methods. Discriminative methods [6], [9], [10], [17]–[22] employ various features and train supervised learning classifiers to predict the popularity volume or level. The success of this type of methods largely depends on the factors (features) which may influence the popularity. To predict the popularity level (low or high) of micro-reviews in Foursquare, Vasconcelos *et al.* [21] consider three different types of factors that influence the popularity of micro-reviews, namely user factor, venue factor, and content factor, and extract as many as 125 features. To predict the popularity of photograph cascades in Facebook, Cheng *et al.* [18] extracts static features such as content features, original poster/resharer features, structural features of the cascades and temporal features. In general, discriminative methods are relatively flexible to incorporate various factors and effective in practice, but most of the them do not consider the dynamic evolution of popularity [23].

Generative methods usually build models that describe the generative process of user behavior [24]–[26] or the popularity time series directly [27]–[30]. For example, Zaman *et al.* [24] adopted a Bayesian approach to model the user retweet behavior, while Wang *et al.* [25] and Zhou *et al.* [26] employed a game-theoretic view to model the diffusion of user behaviors in social networks, such as adopting products or spreading rumors. Instead of modeling user behaviors, some researchers attempt to model the popularity time series directly. Zadeh and Sharda [27] modeled the popularity dynamics of the brand posts in Twitter using the Hawkes process. Their work mainly considers two factors: 1) the publishing time of a tweet and 2) the number of followers of each retweeter. Although generative methods can describe the generative process and usually have better interpretability, they often need strong assumptions and have relatively high computational complexity.

With the recent success of deep neural networks in recognition and prediction tasks, researchers propose to adopt deep learning-based methods to model the cascade process of online contents. Representative work includes DeepCas [31] and DeepHawkes [32] which achieve the state of the art popularity prediction performance. DeepCas first takes a random walk over the cascade graph extracted from user following network and then uses GRU with attention mechanism to predict popularity. DeepHawkes, in contrast, adopts GRU to encode the cascade paths, and then predict popularity using weighted summing of the cascade path features. Although deep learning-based methods can achieve good performance, they usually need much more training data, and the resulting model is weak in interpretability. Moreover, both the generative methods and deep learning-based methods lack a more detailed description of the popularity stage evolutions and fine-grained model for them.

Popularity evolution often involves different stages, and the *burst* stage, for example, is a prominent phenomenon in its evolution process [1]. Actually, the "burst of activity" phenomenon is often seen in e-mail streams and news articles [11], social network events [15], user interactions [33], and hashtags in Twitter [13], [34]. Traditional burst related research usually focuses on burst detection [11], [12] rather than burst prediction. Kleinberg [11] adopted a state-based model and regards the popularity evolution as transitions between different states, which have different emission rates of events. Wang *et al.* [14] recently proposed a feature-based model to predict burst time of cascades. To deal with the fact that the length of post lifecycles can be very different, Wang *et al.* [14], [35] segmented the whole lifecycle into fixed number of time windows and make predictions based on these time windows. However, the above work only considers popularity burst and ignores other important popularity evolution stages, such as valley, rise, and fall.

Another line of related work explores the patterns of social media evolution [1], [2], [36], [37]. Crane and Sornette [1] found that many popularity dynamics follow a power-law distribution, and categorize them into four classes based on the type of disturbance (endogenous/exogenous) and the ability of individuals to influence others to action (critical/subcritical). Yang and Leskovec [2] proposed the K-SC clustering algorithm and find six representative and distinctive patterns of temporal variations of Twitter hashtags. As the patterns of social media evolution imply how popularity might evolve, they can provide important relevant information for evolution stage prediction. Wu *et al.* [37] further proposed to model the popularity dynamics of online videos through explainable factors.

As we have discussed above, none of the existing work on popularity modeling and prediction aims at modeling and predicting the evolution stages of popularity, which is an important research topic and has many applications in different domains. In this paper, we focus on predicting different stages of popularity evolution. We employ various factors to model the dynamic aspects of the popularity evolution process, and propose the factor prior weighting (FPW) algorithm which considers the contributions of each dynamic factor to the PSP problem. We also extract the patterns of popularity evolution to facilitate the prediction of popularity stages. We finally conduct an experimental study for evaluation, and provide a case study to show the significance and implications of our approach.

III. Problem Formulation

Previous work, such as [11] and [14], have studied the burst phenomenon in many different domains. However, we argue that burst alone cannot fully describe the popularity evolution and there are other interesting evolution stages. In contrast to the burst stage, the valley stage is equally important. Besides these two stages, other two stages between burst and valley, that is, the rise and fall stages reflect the trends and tendency of popularity evolution. To provide a complete picture of how the popularity of online contents evolves over time, we use these four stages to fully describe the evolution of the popularity time series, namely burst, valley, rise, and fall.

The lifecycle of an online content refers to the time duration which begins when the online content gets published and ends when the online content receives no more comments, retweets and etc., depending on the specific measurement of popularity (see Fig. 1 for a graphic illustration). Similar to [14], we segment the whole lifecycle of each online content into K time windows which are of equal length in time. The difference between our approach and [14] is that, we do not assume there is *only* one burst stage in the popularity time series. Note that the lengths of time windows vary across different popularity time series, as their lifecycles can be very different. After the segmentation, we label each time window with one of the four stages.

To label the above popularity stages, we apply a simple yet empirically effective method to label each segment with one of the above four stages. For the kth time window, denote the number of newly published retweets at different time units (referred to a short time period) as $s_{k1}, s_{k2}, \ldots, s_{kn}$, where n is the number of time units in each time window. The kth time window is labeled as burst if

$$\frac{\max\{s_{ki} | i = 1, \ldots, n\}}{\frac{1}{nK} \sum_{k=1}^{K} \sum_{i=1}^{n} s_{ki}} > \tau_B. \tag{1}$$

Similarly, it is labeled as valley if

$$\frac{\min\{s_{ki} | i = 1, \ldots, n\}}{\frac{1}{nK} \sum_{k=1}^{K} \sum_{i=1}^{n} s_{ki}} < \tau_V \tag{2}$$

where τ_B and τ_V are thresholds. For each one of rest time windows, we fit a simple linear regression model using the least squares method. For the kth time window, the data points for model fitting are: $\{(i,\ s_{ki}), 1 \le i \le n\}$. So the slope of this regression model, denoted as w_k, can be calculated as

$$w_k = \frac{\sum_{i=1}^{n} s_{ki}(x_i - \bar{x})}{\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2} \tag{3}$$

where $x_i = i$ and $\bar{x}$ is the average of $x_i (1 \le i \le n)$, and w_k can be further simplified as follows:

$$w_k = \frac{6 \sum_{i=1}^{n} s_{ki}(2i - n - 1)}{n\left(n^2 - 1\right)}. \tag{4}$$

Then we label the kth time window as rise if $w_k > 0$, and label it as fall if $w_k < 0$. If $w_k = 0$, we simply label it using the same popularity stage as the previous time window.

Given N time windows of an online content after its publishing time, the PSP problem is to predict the stage of the $(N + d)$th time window of this online content. We call the first N time windows *input intervals* and the following d time windows *response intervals*. Fig. 1 illustrates the above problem formulation.

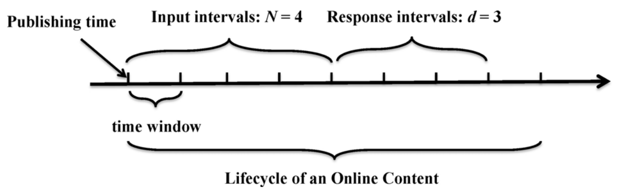

Fig. 1. Illustration of the problem formulation.

IV. Popularity Stage Prediction

We propose our approach to PSP in this section. Specifically, the PSP problem is treated as a multiclass classification task, i.e., the prediction result will be one of the four popularity stages, namely burst, valley, rise, and fall. For this classification task, we first consider the dynamic factors that influence how the popularity evolves and then propose the FPW algorithm to calculate how each dynamic factor [i.e., factor priors (FPs)] affects the PSP results. This information measures the importance of different dynamic factors and is used as weights of factors in the stage prediction algorithm. To capture the overall patterns of popularity stage evolution and their influence on the prediction problem, we apply the K-means clustering algorithm to extract popularity stage patterns and propose the PSP algorithm by combining the prediction results of the FPs and the clustering results.

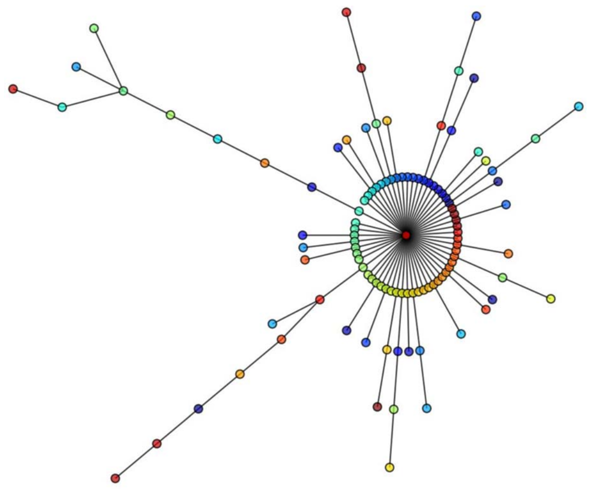

Fig. 2. Example of a retweet tree.

A. Factor Prior Weighting Algorithm

1) Dynamic Factors: We define "dynamic factors" as factors that continually change when more and more users retweet the original tweet. Based on this definition, dynamic factors characterize the dynamic evolution of retweeting cascades of tweets, as well as its popularity stage evolution. During the popularity evolution, retweets and users who issue the retweeting behavior are two key ingredients that drive the cascading process, so we extract dynamic factors from these two aspects.

The extracted dynamic factors capture the numeric and structural information of retweets and users, including number of retweets and users, as well as the retweet tree, user reply network, and social network. Here, we show how to calculate these dynamic factors.

1) *Number of Retweets and Users:* Note that in each time interval, only the number of newly added retweets and participating users are recorded. In addition, the resulting time series are transformed as in [38].
2) *Retweet Tree Structure:* Specifically, the retweet tree is constructed as follows: a) the original tweet is the Root node; b) if tweet A directly retweets the original tweet, add a new node A and a new link from A to the Root node; and c) if tweet A retweets other tweet B which retweets the original tweet, add a new node A and a new link from A to B. See Fig. 2 for an example of a retweet tree. We extract structural properties of the retweet tree as dynamic factors, which include max depth and average path length. Larger depth of a retweet tree means more heated discussions and can boost the popularity of the target tweet. The average path of length is also called Wiener index [18], and used to describe the balanceness of a retweet tree.
3) *User Reply Network Structure:* Different from the retweet tree, nodes in user reply network are users who issue the retweets. Specifically, the reply network is constructed as follows: a) the author of the original tweet is the first node in reply network, denoted as Root; b) if user A retweets the original tweet, add a new node A and a new link from A to Root; and c) if user A retweets a tweet published by user B, add a new node A and a new link from A to B. Dynamic factors of a reply network include link density and mean degree. Larger link density and mean degree means higher interacting frequency between users, and thus they are more likely to boost popularity burst.
4) *Social Network Structure:* When constructing a social network, we add a link from A to B if user A follows user B. Intuitively, when a user has a large friend network, i.e., has many followers or followees, her tweets will have a better chance to be retweeted. For the social network structure, we calculate the average number of followers and followees of users in each time window as the dynamic factors.

For each type of dynamic factor, we record their values in each time window, thus forming a time series.

2) Single Dynamic Factor-Based Prediction: Each dynamic factor corresponds to a time series, denoted as $v_1, v_2, \ldots, v_t$, where t is the number of time windows and v_i is the value of the dynamic factor at time i. Since there are multiple dynamic factors, so each tweet corresponds to multiple time series. For a single dynamic factor, features can be extracted from the corresponding time series, such as first-order derivatives and second-order derivatives. Given the above extracted features, we can build classification models, such as k-NN and SVM, to predict future popularity stage, since we have known the PSP problem can be transformed into a multiclass classification task.

3) Factor Prior Weighting Algorithm: The main idea behind the FPW algorithm (Algorithm 1) is that: different dynamic factors capture different aspects of the popularity evolution, and thus having different effects on the PSP task. In the FPW algorithm, the effects of each dynamic factor on the classification task is represented as a vector FP: $p_1, p_2, \ldots, p_R$, where R is the number of dynamic factors and p_r measures how dynamic factor r affects the popularity prediction performance, which is then used as a weight to represent the importance of dynamic factor r.

Next we describe the details of the FPW algorithm. Given a training dataset T and the set of dynamic factors, the FPW algorithm calculates the FPs. For each dynamic factor r, we first construct a dataset consisting of the time series of all the training samples in T with respect to dynamic factor r, denoted as T_r. The labels of the samples in T_r are the same as those of the samples in T. Using the time series in T_r as features, we build a multiclass classifier, such as SVM, denoted as clf_r. Next we use clf_r to classify the samples in T_r and store the prediction results in $pred_r$. The FP p_r is calculated as the $F1$-score using $pred_r$ and the true labels of the training samples in T_r.

B. Popularity Stage Clustering

Previous studies on temporal patterns of memes [2], [36] and user interactions [33] have shown that, temporal popularity variations follow a limited number of patterns. Based on this finding, we attempt to extract the stage evolution patterns and predict future popularity stage by measuring the

Algorithm 1 FPW Algorithm

```
Input:
    $T$ : Training dataset
    $D$ : Set of dynamic factors
Output:
    $FP : p_1, p_2, \ldots, p_R$
Algorithm:
1. FOR each dynamic factor $r \in D$
        1.1 Construct a dataset with respect to dynamic factor $r$ of all the training samples in $T$, denoted as $T_r$
        1.2 Build a multiclass classifier using $T_r$, such as SVM, denoted as $clf_r$
        1.3 FOR each training sample $i \in T_r$
            1.3.1 Use $clf_r$ to classify sample $i$ and store the prediction result in $pred_r$
            END-FOR
        1.4 Calculate the $F1$-score using $pred_r$ and the true labels of the training samples in $T_r$, denoted as $p_r$
    END-FOR
2. Return $FP$
```

TABLE I
POPULARITY STAGE CLUSTERING RESULTS

Index	Popularity Stage Evolutions										Cluster Size
1	F	F	R	F	F	F	R	F	F	F	1639
2	B	B	B	B	B	B	B	B	B	B	1312
3	F	V	V	V	R	F	V	V	V	V	1067
4	B	B	B	F	F	F	F	F	R	R	982

similarity between the target popularity evolution and the extracted patterns.

We apply the K-means clustering algorithm to cluster stage sequences of tweets. Note that the distance between two stage sequences is defined as the number of different stages at corresponding positions of stage sequences. For illustration purpose, we randomly select 5000 tweets and apply the clustering algorithm to find cluster centroids, where K is set to be 10 and the number of clusters, denoted as M, is 4. Table I shows the four cluster centroids ("*B*" stands for burst, "*V*" for valley, "*R*" for rise, and "*F*" for fall).

As shown in Table I, the first cluster centroid corresponds to the type of tweets, which may receive some retweets at the beginning and sometime during its lifecycle, but never has the opportunity to reach the burst threshold. The second cluster centroid corresponds to the type of tweets which have a large amount of retweets during all its lifecycle. The popularity evolution of tweets in the third cluster quickly drops to valley stage and almost keeps in this stage till the end. This corresponds to the type of tweets which has very limited number of retweets after its publishing time. For the fourth cluster centroid, the burst stage does not last very long and then begins to fall.

C. Popularity Stage Prediction Algorithm

In this section, we propose the PSP algorithm (see Algorithm 2) by combining the priors from the microscopic level (i.e., FP obtained from the FPW Algorithm) and the macroscopic level (i.e., clustering results from K-means clustering algorithm). The algorithm employs the FP-based stage prediction method as well as the clustering-based stage prediction method, where the FP-based and clustering-based methods refer to the methods which only use the FPs and clustering results, respectively. Their prediction results are then combined using a balancing factor α.

Next we describe the details of the PSP algorithm. Given a training dataset T, a target test data o, FPs (i.e., FP), the M clustering centroids produced by K-means clustering algorithm, and the balancing parameter α, the stage prediction algorithm gives the stage prediction result for target sample o. The algorithm first initializes the prediction results of the FP-based and clustering-based prediction methods (denoted as score_{FP} and $\text{score}_{\text{cluster}}$, respectively) (step 1) with zero vectors of length 4, the element of which represents the estimated confidence of each popularity stage, i.e., burst, valley, rise, and fall.

In the algorithm, step 2 describes the FP-based stage prediction method. It calculates each element of score_{FP} for each dynamic factor. For dynamic factor r, first denote the time series for r of target sample o as o_r (step 2.1), and then construct a dataset T_r using all the training samples for factor r (step 2.2, similar to step 1.1 in the FPW algorithm). A multiclass classifier is built using T_r, such as SVM, denoted as clf_r, which is then used to predict the class label (i.e., popularity stage) of o_r (step 2.3). The predicted probabilities of each popularity stage are stored in q_r, which is also a vector of length 4 (step 2.4). To combine the prediction results of different classifiers which are built using different dynamic factors, step 2.5 adopts the weighted soft voting framework [39]. Specifically, weighted soft voting framework is defined as

$$H^j(o) = \sum_{i=1}^{T} q_i h_i^j(o) \tag{5}$$

where o is the target test data, T is the number of classifiers (i.e., number of dynamic factors in our case), $h_i^j(o)$ is the predicted probability for class j by classifier i, q_i is the weight of classifier i, and $H^j(o)$ is the combined prediction result for target test data o for class j. In our problem setting, we simply accumulate the prediction results using score_{FP} for each classifier built using different dynamic factors

$$\text{score}_{\text{FP}} = \text{score}_{\text{FP}} + h_r \times p_r \tag{6}$$

where h_r is the predicted probability by dynamic factor r, and p_r is the FP which is regarded as weight for dynamic factor r.

Step 4 of the PSP algorithm describes the clustering-based method, which predicts the popularity stage based on the similarity of the popularity stage sequence of target sample o and the clustering centroids. Denote seq as the popularity sequence of o and the length of seq as l. For each clustering centroid c_m, the algorithm first calculates the shortest distance $d_{\min}$ between c_m and seq by letting it slide along c_m, and $j_{\min}$ is denoted as the offset corresponding to $d_{\min}$ (step 4.1). The algorithm then accumulates the confidence score using $d_{\min}$

$$\text{score}_{\text{cluster}}[g] = \text{score}_{\text{cluster}}[g] + \exp(-d_{\min}) \tag{7}$$

where g is the popularity stage of $c_m[j_{\min} + l + d]$ and d is the length of the response interval (step 4.2). In step 5, score_{FP}

Algorithm 2 PSP Algorithm

```
Input:
$o$ : Target test data
$T$ : Training dataset
FP : $p_1, p_2, \dots, p_R$
$c_1, c_2, \dots, c_M$: the $M$ clustering centroids
$\alpha$ : Balancing factor between the FP-based and clustering-based method
Output:
Stage prediction result for test data $o$
Algorithm:
1. Vector $score_{FP}$ and $score_{cluster}$ are confidence scores for the FP-based and the clustering-based prediction method, respectively, and both of them are initialized to be [0, 0, 0, 0]
2. FOR each dynamic factor $r \in D$
      2.1 Denote the time series for factor $r$ of the target sample $o$ as $o_r$
      2.2 Construct a dataset using all training samples for factor $r$, denoted as $T_r$
      2.3 Build a multiclass classifier using $T_r$, such as SVM, denoted as $clf_r$
      2.4 Use $clf_r$ to classify $o_r$ and get the probability estimates for each class, stored in $h_r$
      2.5 Add factor prior $p_r$: $score_{FP} = score_{FP} + h_r \times p_r$
   END-FOR
3. Denote $seq$ as the popularity stage sequence of target sample $o$ and the length of $seq$ as $l$
4. FOR each clustering centroid $c_m$
      4.1 Calculate the shortest distance $d_{min}$ between $seq$ and
          $c_m : j_{\min} = \arg\min_{j=1...K-l} dis(seq, c_m[j : j+l])$ and
          $d_{\min} = dis(seq, c_m[j_{min} : j_{min} + l])$
      4.2 $score_{cluster}[g] = score_{cluster}[g] + exp(-d_{min})$, where $g$ is the stage of $c_m[j_{min} + l + d]$
   END-FOR
5. $score = \alpha \times score_{FP} + (1\text{-}\alpha) \times score_{cluster}$
6. Return $\arg\max_{1 \le i \le 4} score[i]$
```

and $\text{score}_{\text{cluster}}$ are normalized to 1 over four popularity stages before they are combined using a balancing parameter α

$$\text{score} = \alpha \times \text{score}_{\text{FP}} + (1-\alpha) \times \text{score}_{\text{cluster}}. \tag{8}$$

The final prediction result is given by $\arg\max_{1 \le i \le 4} \text{score}[i]$ (step 6).

The FP-based method shares some similarities with the IPW algorithm which is proposed in our previous work [19]. Both algorithms are classification algorithms, and can be regarded as one type of ensemble methods based on dynamic factors which combine different classifiers to improve the prediction results. However, they are different from each other in many aspects. The FP-based method is a multiclass classification method while IPW is a two-class classification method. The IPW algorithm is based on *kNN* while the FP-based method is based on multiclassification methods, such as SVM, logistic regression, and *kNN*. The most important difference is that IP matrix in the IPW algorithm stores how each dynamic factor affect the prediction result of each training sample, while FP stores how each dynamic factor affect the prediction result of each class. This difference allows the FP-based method to save much computation time for training and storage.

V. Experiment

A. SinaWeibo Dataset

SinaWeibo (weibo.com) is one of the largest social networking sites in China, with over 500 million registered users by the end of 2012 and about 50 million daily active users. Similar to Twitter, SinaWeibo users can publish original tweets, retweet others' tweets, or mention other users by using "@." We use a publicly available SinaWeibo dataset [40] with about 1.8 million users with 300 million following relationships, 232 000 SinaWeibo tweets and 30 million retweets. In our experiment, we filter out tweets with less than 50 retweets, resulting in about 86 000 original tweets with 28 million retweets.

For this classification task, we construct the training and test dataset as follows: each training data corresponds to a sliding time window with N successive time windows, denoted as $[t, t+N]$, where t is the index of time windows, and time interval $[t,\ t+N]$ is called input interval (see Fig. 1). For each test data, we segment the time interval starting from publishing time to prediction time (i.e., the time to make predictions) into N time windows, so the input interval of a test data is $[0, N]$.

For evaluation, we adopt the fivefold cross evaluation method and report the averaged $F1$-score. For the test dataset, we begin to predict when the tweet has more than 30 retweets. We also adopt the under-sampling technique [41] to keep the datasets balanced between the four classes, i.e., four evolution stages.

In this paper, the thresholds to label popularity stages are set by the experts in security informatics [42]. Specifically, τ_B and τ_V are chosen to be 3 and 0.1, respectively. By default, the lifecycle of the tweet is segmented into 10 time windows with 5 time units in each time window, i.e., $K = 10$ and $n = 5$.

B. Baseline Methods

We compare our methods with three baseline methods, namely *voting method* [39], *static factor-based method*, *dynamic factor-based method*, and three deep learning-based methods: 1) long short term memory (LSTM) [43]; 2) DeepCas [31]; and 3) DeepHawkes [32]. We use SVM as the base classifier for the first three baseline methods as well as for our own methods.

1) Voting Method: The FPW algorithm proposed in this paper can be regarded as a method to combine different classifiers to gain better performances. As a basic ensemble method, voting method is chosen as a baseline. Specifically, voting method proceeds as follows: first, for each dynamic factor, we use the value of the dynamic factor at different time windows as features; then we build a classifier using these features to predict future popularity stage; and we take a vote to make the final prediction among the prediction results by different dynamic factors.

2) Static Factor-Based Method: The static factor-based method only considers the values of each dynamic factor as features at the last time window of the input interval. Compared to the dynamic factor-based method below, it ignores the dynamic time series information of all factors.

TABLE II
PREDICTION RESULTS FOR EACH STAGE

Method	B	V	R	F	Avg.
Voting	0.52	0.41	0.24	–	0.29
Static factor	0.83	0.55	–	0.16	0.41
Dynamic factor	0.83	0.53	–	–	0.35
LSTM	0.37	0.37	-	0.14	0.22
DeepCas	0.45	0.41	0.34	0.11	0.33
DeepHawkes	0.43	0.54	0.12	0.18	0.32
Clustering	0.75	–	0.24	0.46	0.37
PSP-clustering	0.83	**0.69**	**0.43**	0.24	0.55
PSP	**0.86**	0.67	0.31	**0.47**	**0.58**

3) Dynamic Factor-Based Method: The dynamic factor-based method uses the time series of all dynamic factors to form a large time series as features. The difference between the dynamic factor-based method and our proposed methods is that it did not consider FP information and the combination of different factors.

4) Deep Learning-Based Methods: Popularity prediction can be viewed as a time series prediction problem, so we use LSTM to model the evolutions of popularity using the extracted dynamic factors at each time window. We also compare the prediction performance with two other deep learning-based methods from related work: DeepCas [31] and DeepHawkes [32]. Since PSP is a classification problem in this paper, we change the output of the above two methods to predict discrete popularity stages instead of exact popularity values and adopt cross-entropy as loss function, while keeping the design of the neural network architecture as it is.

In addition, to test the performance of our clustering-based stage prediction method, we also compare the FP-based prediction method without clustering (denoted as PSP-clustering), the clustering-based method (denoted as Clustering), and the FP-based prediction method with clustering (denoted as PSP).

C. Experimental Results and Analysis

1) Popularity Stage Prediction Results: We first compare the prediction results of each stage using our proposed methods and the baseline methods. In the following experiments, parameter settings are as follows: $N = 4$, $d = 1$, $K = 10$, $M = 4$, and $\alpha = 0.5$. We shall change each one of these parameters and see how they affect the prediction performance in Section V-D. Note that the $F1$-score for multiclass classification is calculated by averaging the $F1$-scores of each class and "–" is used to denote a $F1$-score which is lower than 0.05.

As we can see from Table II, all methods, except the voting method, produce relatively good results for the burst stage. We can also see that, all deep learning-based methods perform much worse than the proposed methods, which is also true in other parameter settings in the following sections. Possible reasons for this are: 1) the LSTM model incorporates too much history information which introduces much noise for prediction and 2) DeepCas and DeepHawkes both need retweet graphs of users, however, the input interval may be too short to generate any informative retweet structures; also, they are only suitable for predicting exact popularity values and not for PSP.

TABLE III
STAGE/NONSTAGE PREDICTION RESULTS

Method	B/Non-B	V/Non-V	R/Non-R	F/Non-F
CPB	0.80	-	-	-
PSP-clustering	**0.91**	0.82	0.69	0.51

Our proposed methods in this paper, i.e., PSP-clustering and PSP method, also have higher $F1$-score for the valley stage, and higher average $F1$-score than all other methods because they keep a reasonably good performance balance between all four stages. All the methods have relatively low $F1$-score on stage R and F, because according to our popularity time series segmentation method, R-stage and F-stage prediction involves not only predicting the popularity volume but also the trends.

The prediction results also show that, by combining the clustering-based method, the performance of the PSP-clustering method gets improved. One observation of the prediction results is that this method and the clustering-based method usually make different types of prediction mistakes and thus they can compensate each other in many situations. For R-stage prediction, our approach makes the most mistakes by predicting as V or F stages. One possible reason is: there are more V and F stages in clustering centroids (see Table I) and our algorithm tends to make V-stage or F-stage predictions when incorporating clustering. For F-stage prediction, our algorithm exhibits similar behaviors, and makes most mistakes by predicting it as V-stage.

2) Burst Prediction Results: Apart from PSP, our proposed methods can also predict popularity burst, i.e., whether there will be burst in future time windows, by regarding all other stages (valley, rise, and fall) as nonburst stage. We compare our methods with the recently proposed CPB method [14], [35]. Besides, we also report prediction results of other stage/nonstage settings in Table III, such as valley and nonvalley. In this setting, we have set N to be 8 and d to be 1.

Experimental results in Table III show that, our proposed method can effectively predict burst stage with higher $F1$-score (and also with higher AUC and accuracy which we have omitted) over the CPB method.

D. Effects of Parameters on Prediction Results

Now we examine the effects of parameters on prediction results, including the length of input interval N, the length of response interval d, the number of time windows K, and the balancing parameter α. In the following experiments, we set one of them with different values while keeping other ones fixed.

1) Effects of the Length of Input Interval (N): For our proposed method, N controls how much input to use to train the base classifiers, while in the stage prediction algorithm, N controls the length of the time window of test data: larger N means smaller time window length. Other parameter settings,

TABLE IV
PSP RESULTS WITH DIFFERENT *N*

Method	*N*=1	*N*=2	*N*=3	*N*=4	*N*=5	*N*=6
Voting	0.21	0.27	0.29	0.29	0.27	0.27
Static factor	0.34	0.46	0.36	0.41	0.35	0.36
Dynamic factor	0.34	0.52	0.39	0.35	0.34	0.36
LSTM	0.13	0.20	0.22	0.22	0.30	0.27
DeepCas	0.23	0.26	0.32	0.33	0.29	0.37
DeepHawkes	0.25	0.35	0.35	0.32	0.34	0.31
Clustering	0.53	0.53	0.48	0.37	0.19	**0.52**
PSP-clustering	0.47	**0.6**	0.57	0.55	0.52	0.49
PSP	**0.53**	**0.6**	**0.58**	**0.58**	**0.55**	0.51

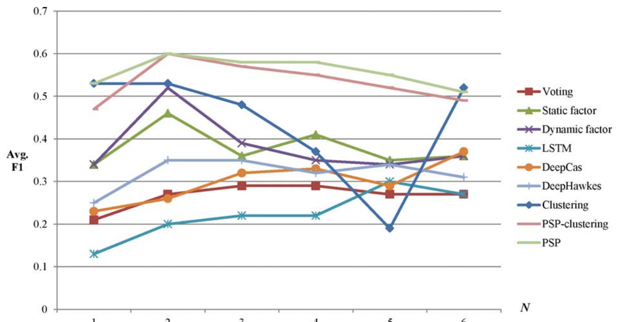

Fig. 3. PSP results with different *N*.

TABLE V
PSP RESULTS WITH DIFFERENT *d*

Method	*d* = 1	*d* = 2	*d* = 3	*d* = 4	*d* = 5
Voting	0.29	0.28	0.33	0.34	0.24
Static factor	0.41	0.50	0.44	0.5	0.49
Dynamic factor	0.35	0.38	0.43	0.51	0.44
LSTM	0.22	0.21	0.39	0.41	0.47
DeepCas	0.33	0.28	0.36	0.33	0.37
DeepHawkes	0.32	0.37	0.33	0.36	0.37
Clustering	0.37	0.21	0.57	0.52	0.47
PSP-clustering	0.55	0.49	0.58	0.58	**0.53**
PSP	**0.58**	**0.52**	**0.59**	**0.59**	**0.53**

including *d*, *K*, and α, are the same as in Section V-C, which is also true for the following experiments. Table IV and Fig. 3 show the PSP results with different *N*.

The experimental results show that, the voting method always performs the worst. The $F1$-score of the static factor-based and dynamic factor-based methods keep around 0.4 as *N* gets larger. Our proposed PSP-clustering and PSP methods achieve better prediction results with different *N* settings. There is one interesting observation: as *N* gets larger, the $F1$-score first increases and then decreases, except for DeepCas. One possible reason for this is that, larger *N* implies more input and the prediction results get better; but too much historic information tends to mislead the algorithm.

2) Effects of the Length of Response Interval (D): Larger *d* means predicting more distant future ahead. Table V and Fig. 4 show the PSP results with different *d*.

As we can see from Table V and Fig. 4, similar to previous parameter settings, our proposed methods perform the best

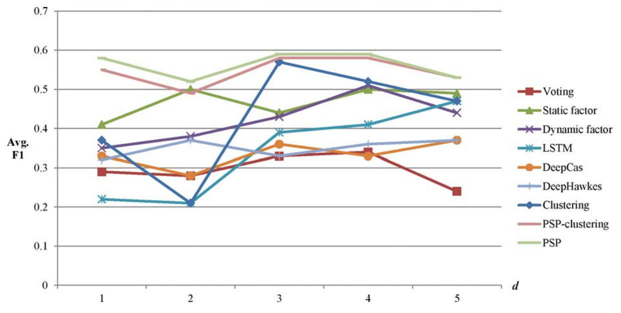

Fig. 4. PSP results with different *d*.

TABLE VI
PSP RESULTS WITH DIFFERENT *K*

Method	*K*=8	*K*=10	*K*=15	*K*=20	*K*=25	*K*=30
Voting	0.25	0.29	0.16	0.24	0.23	0.24
Static factor	0.46	0.41	0.35	0.34	0.34	0.36
Dynamic factor	0.38	0.35	0.34	0.34	0.34	0.35
LSTM	0.31	0.22	0.30	0.27	0.28	0.25
DeepCas	0.36	0.33	0.28	0.27	0.24	0.25
DeepHawkes	0.36	0.32	0.27	0.28	0.32	0.34
Clustering	0.20	0.37	0.53	0.47	0.31	0.23
PSP-clustering	0.53	0.55	0.40	0.47	0.41	0.44
PSP	**0.58**	**0.58**	**0.56**	**0.55**	**0.58**	**0.58**

over other baseline methods. The prediction results of nearly all the methods are not very stable when *d* is changing except for the LSTM method, which reflects the dynamic characteristics of the popularity evolution of tweets. As *d* get larger, the prediction performance of the LSTM method steadily increases, which indicates that LSTM is better at modeling long term relations than other baseline methods.

3) Effects of the Number of Time Windows (K): With fixed *N*, *d*, *M*, and α, larger *K* means smaller length of time windows. With smaller length of time windows, we are actually predicting nearer future (as *d* is fixed), but with less dynamic evolution information as input at the same time (as *N* is fixed). For stage evolution clustering, both of small and large number of evolution stages in the lifecycle of tweets may fail to capture the overall patterns and trends. In summary, the effect of parameter *K* on prediction performance is rather complicated. See Table VI and Fig. 5 for the experimental results.

As can be seen, as *K* increases, nearly the performances of all the methods (including PSP-clustering) drop in general, which means that the effect of less dynamic evolution input is more evident. However, the PSP method is an exception which shows that the prediction results of PSP-clustering method and clustering-based method can be a good compensate for each other.

4) Effects of the Number of Clusters (M): We also experiment with multiple values of *M* to see the effects of *M* on the clustering-based prediction results. From Table VII and Fig. 6, we can see that the performance of the clustering-based approach is becoming much worse when *M* is larger than 4. Through observing the clustering results, we find that larger *M* results in more patterns of popularity stage evolutions,

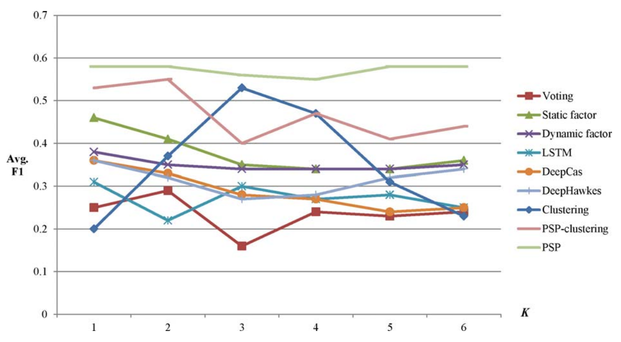

Fig. 5. PSP results with different K.

TABLE VII
PSP RESULTS WITH DIFFERENT M

Method	$M = 2$	$M = 4$	$M = 6$	$M = 8$	$M = 10$
Clustering	0.33	0.37	0.18	0.15	0.14
PSP-clustering	**0.55**	0.55	**0.55**	0.55	**0.55**
PSP	**0.55**	**0.58**	0.53	**0.56**	**0.55**

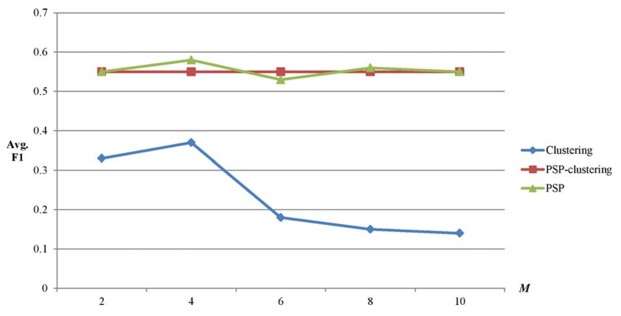

Fig. 6. PSP results with different M.

TABLE VIII
PSP RESULTS WITH DIFFERENT α

α	0	0.2	0.4	0.5	0.6	0.8	1
PSP	0.37	0.41	0.52	**0.58**	0.57	0.56	0.55

but usually many of them are not significantly different from each other.

5) Effects of the Balancing Parameter (α): The balancing parameter α combines the prediction results of the clustering-based method and PSP-clustering by assigning different weights for these two methods. By changing the value of α, we can observe the prediction performance of the PSP method with different weight assignments. Table VIII and Fig. 7 show the corresponding prediction results.

Obviously, setting α to 0 corresponds to only using the clustering-based method, while setting α to 1 corresponds to only using PSP-clustering. From Table VIII and Fig. 7, we can see that, as α increases, the prediction performance first increases rapidly and then decreases slowly. One possible reason is that the prediction performance of PSP-clustering method is better than that of the clustering-based method in this parameter setting (as well as most other parameter settings), and so assigning more weight to the PSP-clustering method can increase the overall performance. However, when α exceeds 0.5, the performance of the

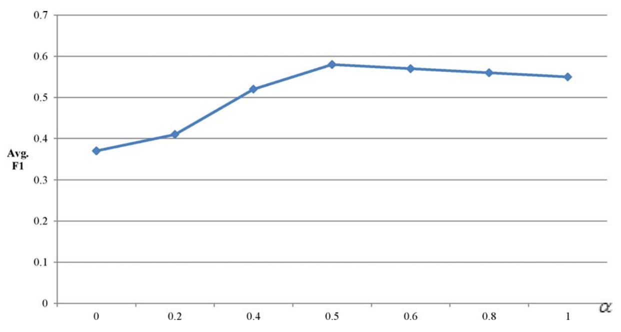

Fig. 7. PSP results with different α.

combined approach begins to drop slowly. This indicates that keeping a good balance is important to ensure a good overall performance of PSP.

E. Case Study

To provide a better understanding of our proposed approach and its implications, here we present a case study in the context of emergency management. Specifically, we select one original tweet (together with all its retweets)[1] published by Shi Wang, the chairman of Wanke (a top real estate company in China) and an opinion leader in Weibo. In this tweet, he stated that "It is reported that half of the top twenty polluted cities are in China" and "The air pollution has become a big threat for the mental and physical health of Chinese people," which attracts a large number people and over 26 000 retweets during its whole lifecycle.

To predict popularity stages of this tweet, we first extract the dynamic factors for each input time interval. The corresponding values for the dynamic factors are given in Table IX ($N = 4, d = 1, K = 10, M = 4$, and $\alpha = 0.5$). We then apply the PSP algorithm to calculate the confidence score using the extracted dynamic factors (i.e., score_{FP}), which is [0.95, 0, 0.02, 0.03] in this case. It means that, at the microscopic level, the algorithm estimates the confidence of each popularity stage B, V, R, and F as 0.95, 0, 0.02, and 0.03, respectively. Besides, the PSP algorithm also calculates the similarities of the stage sequences with clustering centroids at the macroscopic level. For this tweet, the input stage sequence (i.e., the first four time intervals) is labeled as [Burst, Burst, Burst, Burst]. According to the algorithm, we can get the similarities between stage sequences and clustering centroids (i.e., $\text{score}_{\text{cluster}}$), which is [0.48, 0, 0.52, 0] in this case. By combining score_{FP} and $\text{score}_{\text{cluster}}$, we can get the final confidence score, which is [0.715, 0, 0.27, 0.015], and thus the algorithm predicts that the popularity of this tweet will be in *burst* stage. According to the actual data, this is a correct prediction for the next time interval.

We further show the popularity stage evolution for the first three hours of this tweet in Fig. 8, where the length of a time interval is 15 min. The x-axis is the time interval index, and the y-axis is the number of retweets within each time unit. The popularity stage for each time interval is given in the figure. Besides the first series of burst stages in early time

[1]https://www.aminer.cn/influencelocality

TABLE IX
EXTRACTED DYNAMIC FACTORS

Dynamic Factors \ Time Interval Index		1	2	3	4
Number of retweets		65	65	58	49
Number of users		65	65	58	49
Retweet tree	Max depth	1	1	1	1
	Avg. path length	1.98	1.99	1.99	2.0
User reply network	Link density	0.009	0.005	0.003	0.002
	Mean degree	1.98	1.99	1.99	2.0
Social network	Avg. of followees	519	502	520	469
	Avg. of followers	7827	29339	4682	23537

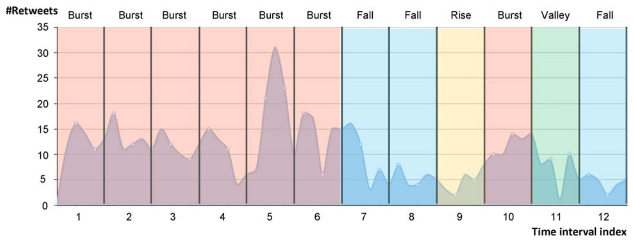

Fig. 8. Popularity stage evolution of an example tweet.

intervals, from the figure, we can also observe another *burst* stage in the tenth time interval. The burst stage occurs after two consecutive *fall* stages and one *rise* stage—a representative pattern as shown in [2], i.e., a large peak followed by a smaller peak. In general, PSP helps answer the questions such as the rise or fall trend of a tweet during each time interval, whether a tweet is going to burst, how long a burst stage will last, and when a tweet will receive few attention (i.e., valley).

In summary, different from previous work on popularity prediction, our proposed method can predict the trend of popularity evolution of user generated online contents. Moreover, by recursively applying the proposed method, it can provide continuous PSP results over time. Given recent popularity stages and the predicted popularity stage at a particular future time, decision makers can be more informed to quickly respond and take appropriate actions for emergency management, and to prevent hot discussions from becoming a public opinion crisis.

VI. CONCLUSION

In this paper, we focus on the popularity evolution of online contents and address the problem of PSP. We attempt to solve this problem by considering the dynamic aspects of popularity evolution at two levels. At the microscopic level, we identify various numeric and structural factors and propose the FPW algorithm to combine the prediction results produced by different dynamic factors. At the macroscopic level, we apply clustering algorithm to extract popularity stage patterns and adopt a pattern matching method to predict future popularity stages. The experimental results show that our proposed methods achieve better performance against the baseline methods and thus verify the effectiveness of our proposed approach to PSP.

REFERENCES

[1] R. Crane and D. Sornette, “Robust dynamic classes revealed by measuring the response function of a social system,” *Proc. Nat. Acad. Sci. USA*, vol. 105, no. 41, pp. 15649–15653, Oct. 2008.

[2] J. Yang and J. Leskovec, “Patterns of temporal variation in online media,” in *Proc. 4th ACM Int. Conf. Web Search Data Min.*, Hong Kong, 2011, pp. 177–186.

[3] W. van Osch, C. K. Coursaris, and B. A. Balogh, “Informing brand messaging strategies via social media analytics,” *Online Inf. Rev.*, vol. 40, no. 1, pp. 6–24, Feb. 2016.

[4] L. Wu *et al.*, “Product adoption rate prediction in a competitive market,” *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 2, pp. 325–338, Feb. 2018.

[5] A. Tatar, M. D. de Amorim, S. Fdida, and P. Antoniadis, “A survey on predicting the popularity of Web content,” *J. Internet Services Appl.*, vol. 5, no. 1, pp. 1–20, Dec. 2014.

[6] G. Szabo and B. A. Huberman, “Predicting the popularity of online content,” *Commun. ACM*, vol. 53, no. 8, pp. 80–88, Aug. 2010.

[7] L. Hong, O. Dan, and B. D. Davison, “Predicting popular messages in Twitter,” in *Proc. 20th Int. Conf. Companion World Wide Web*, Hyderabad, India, 2011, pp. 57–58.

[8] Y. Borghol, S. Ardon, N. Carlsson, D. Eager, and A. Mahanti, “The untold story of the clones: Content-agnostic factors that impact YouTube video popularity,” in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Beijing, China, 2012, pp. 1186–1194.

[9] Z. Ma, A. Sun, and G. Cong, “On predicting the popularity of newly emerging hashtags in Twitter,” *J. Amer. Soc. Inf. Sci. Technol.*, vol. 64, no. 7, pp. 1399–1410, Jul. 2013.

[10] S. H. Doong, “Predicting Twitter hashtags popularity level,” in *Proc. 49th Hawaii Int. Conf. Syst. Sci. (HICSS)*, Koloa, HI, USA, 2016, pp. 1959–1968.

[11] J. Kleinberg, “Bursty and hierarchical structure in streams,” in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Edmonton, AB, Canada, 2002, pp. 91–101.

[12] A. Sun, D. D. Zeng, and H. Chen, “Burst detection from multiple data streams: A network-based approach,” *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 40, no. 3, pp. 258–267, May 2010.

[13] S. Kong, Q. Mei, L. Feng, F. Ye, and Z. Zhao, “Predicting bursts and popularity of hashtags in real-time,” in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Gold Coast, QLD, Australia, 2014, pp. 927–930.

[14] S. Wang, Z. Yan, X. Hu, P. S. Yu, and Z. Li, “Burst time prediction in cascades,” in *Proc. 29th AAAI Conf. Artif. Intell.*, Austin, TX, USA, 2015, pp. 325–331.

[15] W. X. Zhao *et al.*, “Identifying event-related bursts via social media activities,” in *Proc. Joint Conf. Empirical Methods Nat. Lang. Process. Comput. Nat. Lang. Learn.*, Stroudsburg, PA, USA, 2012, pp. 1466–1477.

[16] Y. Jiang and J. C. Jiang, “Diffusion in social networks: A multiagent perspective,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 2, pp. 198–213, Feb. 2015.

[17] A. Kupavskii, A. Umnov, G. Gusev, and P. Serdyukov, “Predicting the audience size of a Tweet,” in *Proc. 7th Int. Conf. Weblogs Soc. Media*, Cambridge, MA, USA, 2013, pp. 693–696.

[18] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec, “Can cascades be predicted?” in *Proc. 23rd Int. Conf. World Wide Web*, Seoul, South Korea, 2014, pp. 925–936.

[19] Q. Kong, W. Mao, D. Zeng, and L. Wang, “Predicting popularity of forum threads for public events security,” in *Proc. IEEE Joint Intell. Security Informat. Conf.*, The Hague, The Netherlands, 2014, pp. 99–106.

[20] P. Cui *et al.*, “Cascading outbreak prediction in networks: A data-driven approach,” in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Chicago, IL, USA, 2013, pp. 901–909.

[21] M. Vasconcelos, J. M. Almeida, and M. A. Gonçalves, “Predicting the popularity of micro-reviews: A foursquare case study,” *Inf. Sci.*, vol. 325, pp. 355–374, Dec. 2015.

[22] F. Figueiredo, J. M. Almeida, M. A. Gonçalves, and F. Benevenuto, “TrendLearner: Early prediction of popularity trends of user generated content,” *Inf. Sci.*, vols. 349–350, pp. 172–187, Jul. 2016.

[23] J. Ugander, L. Backstrom, C. Marlow, and J. Kleinberg, “Structural diversity in social contagion,” *Proc. Nat. Acad. Sci. USA*, vol. 109, no. 16, pp. 5962–5966, 2012.

[24] T. Zaman, E. B. Fox, and E. T. Bradlow, “A Bayesian approach for predicting the popularity of Tweets,” *Ann. Appl. Stat.*, vol. 8, no. 3, pp. 1583–1611, Sep. 2014.

[25] Y. Wang, A. V. Vasilakos, J. Ma, and N. Xiong, "On studying the impact of uncertainty on behavior diffusion in social networks," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 2, pp. 185–197, Feb. 2015.

[26] F. Zhou, R. J. Jiao, and B. Lei, "Bilevel game-theoretic optimization for product adoption maximization incorporating social network effects," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 8, pp. 1047–1060, Aug. 2016.

[27] A. H. Zadeh and R. Sharda, "Modeling brand post popularity dynamics in online social networks," *Decis. Support Syst.*, vol. 65, pp. 59–68, Sep. 2014.

[28] L. Yu, P. Cui, F. Wang, C. Song, and S. Yang, "From micro to macro: Uncovering and predicting information cascading process with behavioral dynamics," in *Proc. IEEE Int. Conf. Data Min. (ICDM)*, Atlantic City, NJ, USA, 2015, pp. 559–568.

[29] Y. Hu, C. Hu, S. Fu, P. Shi, and B. Ning, "Predicting the popularity of viral topics based on time series forecasting," *Neurocomputing*, vol. 210, pp. 55–65, Oct. 2016.

[30] H. Zhu, C. Liu, Y. Ge, H. Xiong, and E. Chen, "Popularity modeling for mobile apps: A sequential approach," *IEEE Trans. Cybern.*, vol. 45, no. 7, pp. 1303–1314, Jul. 2015.

[31] C. Li, J. Ma, X. Guo, and Q. Mei, "DeepCas: An end-to-end predictor of information cascades," in *Proc. 26th Int. Conf. World Wide Web*, Perth, WA, Australia, 2017, pp. 577–586.

[32] Q. Cao, H. Shen, K. Cen, W. Ouyang, and X. Cheng, "DeepHawkes: Bridging the gap between prediction and understanding of information cascades," in *Proc. ACM Conf. Inf. Knowl. Manag.*, Singapore, 2017, pp. 1149–1158.

[33] Y.-Q. Zhang, X. Li, J. Xu, and A. V. Vasilakos, "Human interactive patterns in temporal networks," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 2, pp. 214–222, Feb. 2015.

[34] J. Lehmann, B. Gonçalves, J. J. Ramasco, and C. Cattuto, "Dynamical classes of collective attention in Twitter," in *Proc. 21st Int. Conf. World Wide Web*, Lyon, France, 2012, pp. 251–260.

[35] S. Wang *et al.*, "CPB: A classification-based approach for burst time prediction in cascades," *Knowl. Inf. Syst.*, vol. 49, no. 1, pp. 243–271, Dec. 2015.

[36] Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos, "Rise and fall patterns of information diffusion: Model and implications," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Beijing, China, 2012, pp. 6–14.

[37] J. Wu, Y. Zhou, D. M. Chiu, and Z. Zhu, "Modeling dynamics of online video popularity," *IEEE Trans. Multimedia*, vol. 18, no. 9, pp. 1882–1895, Sep. 2016.

[38] G. H. Chen, S. Nikolov, and D. Shah, "A latent source model for non-parametric time series classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 1088–1096.

[39] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*, 1st ed. Boca Raton, FL, USA: CRC Press, 2012.

[40] J. Zhang, B. Liu, J. Tang, T. Chen, and J. Li, "Social influence locality for modeling retweeting behaviors," in *Proc. 23rd Int. Joint Conf. Artif. Intell.*, Beijing, China, 2013, pp. 2761–2767.

[41] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intell. Data Anal.*, vol. 6, no. 5, pp. 429–449, Oct. 2002.

[42] W. Mao and F. Wang, *Advances in Intelligence and Security Informatics*. Oxford, U.K.: Academic, 2012.

[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

Qingchao Kong received the B.S. degree in computer science and technology from the Harbin Institute of Technology, Weihai, China, in 2011 and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, China.

He is an Assistant Professor with the Institute of Automation, Chinese Academy of Sciences. His current research interests include popularity modeling and prediction, user behavior modeling, social media analytics, and data mining.

Wenji Mao received the Ph.D. degree in computer science from the University of Southern California, Los Angeles, CA, USA, in 2006.

She is a Professor with the Institute of Automation, Chinese Academy of Sciences, Beijing, China, and the School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing. Her current research interests include artificial intelligence, intelligent agents, and social modeling.

Guandan Chen received the B.S. degree in automation from Xiamen University, Xiamen, China. He is currently pursuing the Ph.D. degree in social computing with the Institute of Automation, Chinese Academy of Sciences, Beijing, China.

His current research interests include deep neural network, social media analytics, and data mining.

Daniel Zeng (M'04–SM'07–F'16) received the Ph.D. degree in industrial administration from Carnegie Mellon University, Pittsburgh, PA, USA, in 1998.

He is a Professor with the Institute of Automation, Chinese Academy of Sciences, Beijing, China, and the School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing. His current research interests include software agents and multiagent systems, intelligence and security informatics, social computing, and recommendation systems.