2014 IEEE Joint Intelligence and Security Informatics Conference

Predicting Popularity of Forum Threads for Public Events Security

Qingchao Kong[1] Wenji Mao[1] Daniel Zeng[12] Lei Wang[3]
[1]State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[2]Department of Management Information Systems, University of Arizona, Tucson, AZ 85721, USA
[3]College of Management and Economics, Tianjin University, Tianjin 300072, China
{qingchao.kong, wenji.mao, dajun.zeng, l.wang}@ia.ac.cn

Abstract—**Web user's online interactive behavior with others often makes some user generated contents popular. The modeling and prediction of the popularity of online content are an important research issue for many key application domains. In this paper, we focus on one form of user generated content, forum threads, and their popularity prediction for public events security. To predict the popularity of forum threads, we first define the popularity prediction problem, and identify the dynamic factors that affect the popularity of forum threads. Based on the information of dynamic evolution at the early stage, we propose a popularity prediction algorithm which makes use of the locality property and combines multiple dynamic factors. The proposed algorithm is further evaluated using the Tianya forum dataset on the discussions of various public events. The experimental results show that, compared to the baseline methods, our method achieves relatively better performance in predicting the popularity of forum threads on public events security.**

Keywords—popularity of online content, modeling and prediction of popularity evolution, social media analytics

I. INTRODUCTION

Web users are enjoying multiple ways of online interactions nowadays. They can discuss public events and express their opinions conveniently via writing personal blogs, posting threads in forums, or broadcasting tweets in social networking sites like Twitter and Facebook. Pervasive online content, especially user generated content such as thread posts, tweets and blogs, have become rich sources for information collection and sharing, and dissemination of public opinions [1, 2].

Web user's online interactive behavior with others often makes some user generated contents popular. The modeling and prediction of the popularity of online content are an important research issue and can facilitate many key application domains. It can provide crucial information of the public attitude and attention towards a certain online content, as well as the public tendency at a future time. It is particularly important in the field of security informatics. For both public and private sectors, modeling and prediction of the popularity of online content can effectively support security-related early warning, decision making and emergency response [3]. In other business applications, accurate prediction of product popularity based on online content manifests users' needs and preferences, which can promote product recommendation and ad placement [4].

To study the problem of popularity modeling and prediction, we focus on one media form, online forum. Online forum is a representative form of online content with diverse topics and a large user base. It has become an active and important platform to express public opinions and discuss hot topics in practice. In addition, because of the real-time and interactivity characteristics of the Internet, online contents are usually dynamically evolved over time [5, 6], which results in the dynamic property of popularity evolution. Thus, another focus of our work is the dynamic aspect of the popularity of online content.

In this paper, we focus on the modeling and analysis of dynamic evolution of forum threads and propose the computational method to predict popularity at a future time. We first explicitly define the problem of popularity prediction, and identify the dynamic factors that affect the popularity evolution of forum threads. Based on the dynamic information of forum threads at the early stage, we combine multiple dynamic factors and develop a kNN-based algorithm for popularity prediction. To evaluate the effectiveness of our proposed algorithm, we use a real-world dataset on the discussions of various public events crawled from the Tianya forum. Compared to the related research and baseline methods, our proposed algorithm achieves relatively better prediction results.

The contributions of this paper are three-fold: 1) we are among the first to provide a clear definition of the popularity prediction problem for forum threads; 2) we identify the dynamic factors that affect the popularity evolution of forum threads, which include the structural properties of the comment tree and user reply network; 3) we propose a kNN-based popularity prediction algorithm that combines multiple dynamic factors and makes use of the locality property of dynamic factors as priors. The effectiveness of the prediction algorithm is empirically evaluated.

II. RELATED WORK

A. *Definition of Popularity*

Popularity related research has focused on online videos [4, 7–15], Twitter [16–19] or Weibo [20] in China, hashtags [21, 22], Digg shares [23, 24], images [25, 26], etc. The specific definition of popularity of online content depends on the application context. Past research usually measures "popularity" by "counting", for example, number of views (i.e., view count) of online videos, number of mentions (i.e., mention count) or retweets (i.e., retweet count) in Twitter, number of hashtag occurrences. Ma *et al.* [20] compared two popularity definitions of Weibo posts, that is, retweet count and view count, and found that the two measures are positively correlated, but the correlation is not very strong. Yin *et al.* [12] defined the popularity of Digg shares using both positive votes and negative votes, instead of only considering the total votes. Similarly, observing that the view count of images is constantly increasing over time, Khosla *et al.* [25] defined the popularity as the ratio of the view count and upload time.

To study the popularity of forum threads, instead of using view count, we use the number of thread comments (i.e., comment count) as popularity measurement in this paper, which can better reveal user's interest.

B. Popularity Prediction

Szabo and Huberman [23] conducted a large scale study about the popularity of Digg shares and Youtube videos. They found that, popularity at the early stage is approximately linearly correlated with popularity at the future stage (denote the linear coefficient as α). Based on this observation, the S-H model was proposed. However, there are obvious drawbacks of the S-H model. First, the S-H model only considers the popularity values at the early and future stage, and do not take other rich features into account. Second, some online contents have similar popularity at the early stage but evolve to distinct popularity at the future stage [15]. The S-H model could not handle this situation because it assumes all online contents share the same linear coefficient α. This leads to another drawback of the S-H model. Although the S-H model performs well with large real-world datasets, the popularity distributions in these datasets are all severely skewed. Consequently, the S-H model is mainly applicable to those datasets with constantly low (or high) popularity.

Recent research on popularity prediction utilizes richer features in specific problem context. Based on the type of features used, recent popularity prediction related work fall into two categories: methods based on static factors [4, 7, 8, 10, 12, 16–18, 20, 22, 24] and methods based on dynamic factors [13–15, 21].

Methods based on static factors usually explore popularity related factors and then train a logistic regression model using these factors as features. For example, in order to predict the future popularity of online videos, Figueiredo [4] considered content features, including video category, upload date and video sharing information, such as the time it is first shared and view count of video shares. In addition, generative models like topic models, are also used to predict the popularity of online content [8, 12, 16, 24].

As online content exhibits strong dynamic characteristics over time, compared to the methods based on static factors, methods based on dynamic factors are able to capture the dynamic property of popularity evolution and thus gaining increasing attention in recent years. Ahmed *et al.* [13] first defined two features: 1) the share of user attention that a content attracts with respect to all other observed contents and 2) the normalized rate of change in the attention attracted at some time interval. They designed a HMM-like approach to predict which popularity cluster the content may belong to.

Dynamic factors in related work, such as view count, comment count, number of users (i.e., user count), share ratio, can reveal how popularity evolves over time, however, structural properties related dynamic factors about online contents and users are largely unexplored. In addition, most current work only predict some "count" as popularity using datasets with skewed popularity distributions, which, as we have discussed, shares the same drawback with the S-H model. Therefore, a different problem definition of popularity prediction with respect to popularity distributions in real-world datasets is needed as well.

In this paper, we focus on modeling the popularity evolution of forum threads on public events. We first give the problem definition of popularity prediction of forum threads. As for the dynamic factors, we not only use the information of comments and users, but also the structural properties of comment tree and user reply network. With these dynamic factors, we view the popularity prediction problem as a time series classification problem. Based on kNN algorithm, which is known to be very effective in time series classification [27], we calculate how each dynamic factor influences each sample using the locality property, and then propose an algorithm that combines multiple dynamic factors to predict future popularity of forum threads. Our proposed algorithm is empirically evaluated by comparing with the related work and baseline methods, using the Tianya forum dataset on the discussions of public events.

III. Popularity Prediction Based on Dynamic Evolution

A. Problem Definition

In this section, we give the problem definition of popularity prediction. Given a forum thread, denote $C(t)$ as the comment count at time t, i.e. the popularity of this thread at time t. To be more specific, define r as:

$$r = \frac{C(t_r)}{C(t_t)} \tag{1}$$

where t_r and t_t satisfy $t_r < t_t$. See Figure 1 for a description of thread lifecycle and the definitions of t_r and t_t.

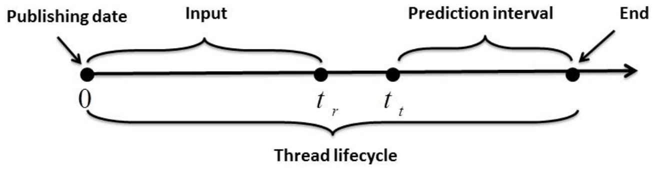

Figure 1 Thread lifecycle and definition of t_r and t_t

Let p be a pre-specified threshold, satisfying $0 < p < 1$. For a given thread, $r < p$ means there still will be a considerable amount of comments (relative to $C(t_t)$) after time t_r, and it is defined to be "popular" in the future (denoted as class L_1), otherwise it is considered to be "unpopular" (denoted as class L_2). The popularity prediction is defined as a binary classification problem, i.e. predicting whether $r < p$ or not, given the evolution history information of this thread before t_r.

In practical applications, two restrictions can be applied: 1) threads should have at least $min\_C(t_r)$ comments at time t_r, which ensures enough information input for the prediction algorithm; 2) one thread should have at least $min\_C(t_t)$ comments at time t_t to be considered popular.

B. *Constructing Dynamic Factors*

A thread will become popular or not mainly depends on two types of factors, static factors and dynamic factors. Static factors include thread content, publishing date, author, which rarely change during the whole thread lifecycle. In contrast, dynamic factors are the primary drive of the threads' lifecycle and decisive in affecting future popularity. As comments, which are our research object, and users, who create those comments, are the two key aspects in modeling popularity evolution, we will design dynamic factors based on them. These factors capture the number and structural information of comments and users, including comment count, user count, as well as comment tree and user reply network. Here we elaborate them in detail.

- *Comment count* and *user count*. Note that in each sampling interval, only the number of newly added comments and commenting users compared to the last interval are recorded. In addition, the resulting time series are transformed as in [21].

- Structural properties of the *comment tree*. We extract structural properties of the comment tree as dynamic factors, which include max depth and average path length. Specifically, the comment tree is constructed as follows (see Figure 2 for an example of a comment tree): 1) the original post is the Root node; 2) if comment A directly replies to the original post, add a new node A and a new link from A to the Root node; 3) if comment A replies to other comment B (B is not the Root node), add a new node A and a new link from A to B.

- Structural properties of the *user reply network*. Different from the comment tree, nodes in reply network are commenting users. Dynamic factors for the reply network include link density and mean degree. Specifically, the reply network is constructed as follows: 1) the thread author is the first node in reply network, denoted as FN; 2) if user A replies to the original post, add a new node A and a new link from A to FN; 3) if user A replies a comment posted by user B, add a new node A and a new link from A to B.

As the comment count and user count grow, the structure of the comment tree and reply network change accordingly, so the dynamic factors change over time.

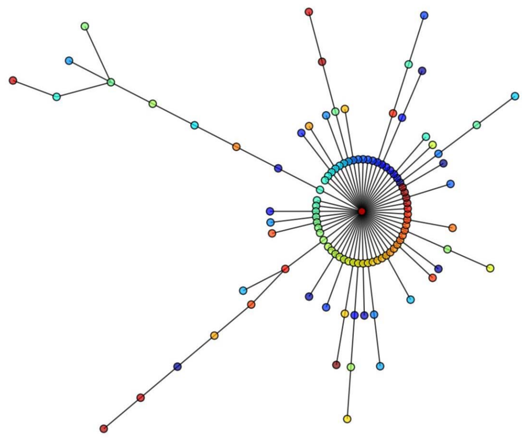

Figure 2 Example of a comment tree

After a thread was created, we record the values of the dynamic factors once every fixed time interval (i.e. sampling interval), such as from several hours to one day. The recording procedure will result in a time series corresponding to each dynamic factor. When multiple dynamic factors are recorded, a multi-dimensional time series will be generated.

C. *IPW Algorithm*

As we have discussed, if only one dynamic factor is considered, each forum thread corresponds to a time series of the dynamic factor values. Now the popularity prediction problem becomes a classical time series classification problem. Based on the analysis in the "Related Work" section, kNN algorithm can be adopted, in which Euclidean distance is used to measure the similarity between time series.

Since single dynamic factor only captures one aspect of the dynamic evolution of threads, our goal is to try to combine multiple dynamic factors to improve the prediction accuracy. The analysis of prediction results with single dynamic factor shows that, different dynamic factors affect different types of thread samples, i.e. for different thread samples, dynamic factors have different classification performances. Based on the above analysis, we improve the kNN algorithm and propose the IPW (Instance Prior Weighting) algorithm (see Table I). The basic idea of the IPW algorithm is as follows: for a thread sample in test set, using each dynamic factor, the algorithm first finds k nearest neighbors for this test sample, and calculates classification confidence scores for each class (L_1 and L_2) based on the classification results using these neighboring samples. Then classification confidence scores for every dynamic factor are summed up with respect to each class, and the final prediction result is the class which holds the maximum confidence score.

In the IPW algorithm, the "IP matrix" is used to depict how each dynamic factor influences the classification result

TABLE I. IPW ALGORITHM

```
Input:
Target test sample $o$
Training set $T$
Dynamic factor set $D$
Number of nearest neighbors $k$

Output:
Prediction result for test sample $o$

Algorithm:
1. Call the IPCreate algorithm (see Table II) and get $IP\_matrix$
2. Vector $score$ is used to accumulate confidence scores for each class by all dynamic factors and
   initialized to [0, 0]
3. FOR each dynamic factor $m \in D$
      3.1 Using dynamic factor $m$, calculate distances between $o$ and every sample in training set $T$
      3.2 Denote $K\_list$ as the top-$k$ nearest neighbors
      3.3 FOR sample $i \in K\_list$
            3.3.1 Using dynamic factor $m$, denote $d_i$ as the distance between sample $i$ and test sample $o$
            3.3.2 Calculate weight $w_i$ for sample $i$ using $d_i$: $w_i = exp(-1 * d_i)$
            3.3.3 Get $p_{mi}$ from $IP\_matrix$ corresponding to dynamic factor $m$ and sample $i$
            3.3.4 Denote the true class of sample $i$ as $L_j$, accumulate confidence score for this class: $score[j]$
                  $= score[j] + w_i * p_{mi}$
         END-FOR
   END-FOR
4. The prediction result for test sample $o$ is the class which holds the maximum confidence score value.
```

TABLE II. IPCREATE ALGORITHM

```
Input:
Training samples set $T$
Dynamic factor set $D$
Number of nearest neighbors $k$
Output:
$IP\_matrix$

Algorithm:
1. FOR each sample $i \in T$
      1.1 FOR each dynamic factor $m \in D$
         1.1.1 Using dynamic factor $m$, find $k$ nearest neighbors for sample $i$ using weighted kNN
               algorithm. Among the $k$ nearest neighbors, denote $n_1$ as the number of samples that belong to
               class $L_1$ and $n_2$ as the number of samples that belong to $L_2$.
         1.1.2 Let $diff$ be the difference between $n_1$ and $n_2$ ($diff$ is non-negative)
         1.1.3 If $n_1 > n_2$, sample $i$ will be classified as $L_1$, and $L_2$ otherwise. Denote the classification result
               as $pred$.
         1.1.4 IF the true class of sample $i$ is the same as $pred$
         1.1.5 THEN Set the corresponding entry in $IP\_matrix$ with respect to $m$ and $i$ with $exp(+1 * diff)$
         1.1.6 ELSE Set the corresponding entry in $IP\_matrix$ with respect to $m$ and $i$ with $exp(-1 * diff)$
           END-FOR
   END-FOR
2. Return $IP\_matrix$
```

on each thread sample and each matrix entry shows how strong the influence is, as shown in Figure 3.

$$\begin{array}{c|cccc|}
 & \textbf{Sample 1} & \textbf{Sample 2} & \dots & \textbf{Sample } N \\
\textbf{Dynamic factor 1} & p_{11} & p_{12} & \cdots & p_{1N} \\
\textbf{Dynamic factor 2} & p_{21} & p_{22} & \cdots & p_{2N} \\
\dots & \cdots & \cdots & \cdots & \cdots \\
\textbf{Dynamic factor } M & p_{M1} & p_{M2} & \cdots & p_{MN}
\end{array}$$

Figure 3 IP matrix

Table I shows pseudo-code of the IPW algorithm. The IPW algorithm is as follows: first call IPCreate algorithm (see Table II) to construct *IP_matrix* (Step 1). For target test sample *o*, vector *score* is used to accumulate the sum of confidence scores for each class by all dynamic factors and initialized to be [0, 0] (Step 2). For each dynamic factor *m* in dynamic factor set *D* (Step 3), calculate distances between *o* and every sample in training set *T*, and get the *k* nearest neighbors (Step 3.1 and Step 3.2). Note that mutual nearest neighbor strategy is adopted here and in the IPCreate algorithm. For each nearest neighbor *i* (Step 3.3): calculate weight w_i for *i* using the distance between sample *i* and *o* fordynamic factor *m* (Step 3.3.2), and get p_{mi} from *IP_matrix* corresponding to dynamic factor *m* and sample *i* (Step 3.3.3). Based on the true class sample *i* belongs to, accumulate confidence score for this class (Step 3.3.4). The prediction result for test sample *o* is the class which holds the maximum confidence score (Step 4).

Note that in Step 3.3.5 of the IPW algorithm, if p_{mi} equals to 1, which means ignore the influence of dynamic factors, the IPW algorithm will be degenerated to weighted kNN; if both p_{mi} and w_i equal to 1, the IPW algorithm will be degenerated to classical kNN.

To construct *IP_matrix*, we propose the IPCreate algorithm (see Table II). The IPCreate algorithm is as follows: for each sample *i* in training set *T* and each *m* in dynamic factor set *D* (Step 1 and Step 1.1), using dynamic factor *m*, find *k* nearest neighbors for sample *i* using weighted kNN algorithm. Denote n_1 as the number of samples that belong to class L_1 and n_2 as the number of samples that belong to L_2 (Step 1.1.1). Let *diff* be the difference between n_1 and n_2 (Step 1.1.2). If $n_1 > n_2$, sample *i* will be classified as L_1, and L_2 otherwise (Step 1.1.3). Based the classification result and sample *i*'s true class, calculate the corresponding entry in *IP_matrix* with respect to dynamic factor *m* and sample *i* (Step 1.1.5 and Step 1.1.6): if sample *i* is correctly classified, corresponding entry in *IP_matrix* shall be larger than 1, and smaller than 1 otherwise. Note that the above entry value is related with *diff*: larger *diff* means dynamic factor *m* is more confident about the classification result and thus should have a bigger impact on sample *i*.

IV. Experiment

A. Tianya Forum Dataset

Founded in 1999, Tianya forum (http://bbs.tianya.cn/) is very popular in China with approximately 85 million registered users at the end of 2013. Tianya forum has 8 major sections and about 200 sub-sections, and covers a broach range of topics, including politics, stocks, news, sports, computers, and so on. Tianya forum users are highly active in participating public events by posting discussion threads. For example, in February 2007, one thread containing pictures about a forced demolition incident in Chongqing started a heated discussion in Tianya. The discussion quickly spread to other online forums, social media and finally the public media. It is considered to be a demolition landmark event in China.

We crawl threads from a major Tianya forum section discussing about various public events from Jan. to May 2014. To ensure enough input information when predicting, threads with less than 10 comments are filtered out, resulting in a total of 3293 forum thread samples. Each thread sample includes the thread author, thread publishing date, original post content, and the author, publishing date, content of each comment. Replying relation between comments are also extracted. Based on the user and comment information, we construct the comment tree and user reply network using the method discussed in Section III.

Figure 4 (left) shows the comment count distribution, the x-axis shows the comment count and the y-axis shows the thread count. Figure 4 (right) shows the thread lifecycle distribution, the x-axis show the lifecycle of threads (in days), and the y-axis shows the thread count.

B. Baseline and Related Methods

We compare our approach to popularity prediction with both baseline methods and methods from the related work. Besides the kNN algorithm with each single dynamic factor, we also choose voting methods [28] as the baseline method which combines the prediction results of kNN algorithm with each dynamic factor. Among the methods from related work, the S-H model [23] and ML methods in [15] are included as the comparing methods. We would not include methods in [13, 14] here because they use application specific features. Here is a more detailed description of the above baseline and related methods:

- *kNN algorithm*

- *Voting method*. First use kNN with each dynamic factor to classify the test sample, and take a vote based on the prediction result. The final prediction result is the class which has the maximum votes.

- *S-H model*. First calculate the linear coefficient α using the training set. Under this problem definition, based on the assumption of the S-H model, a test sample is classified as L1 if $\alpha > p$, and classified as L2 otherwise.

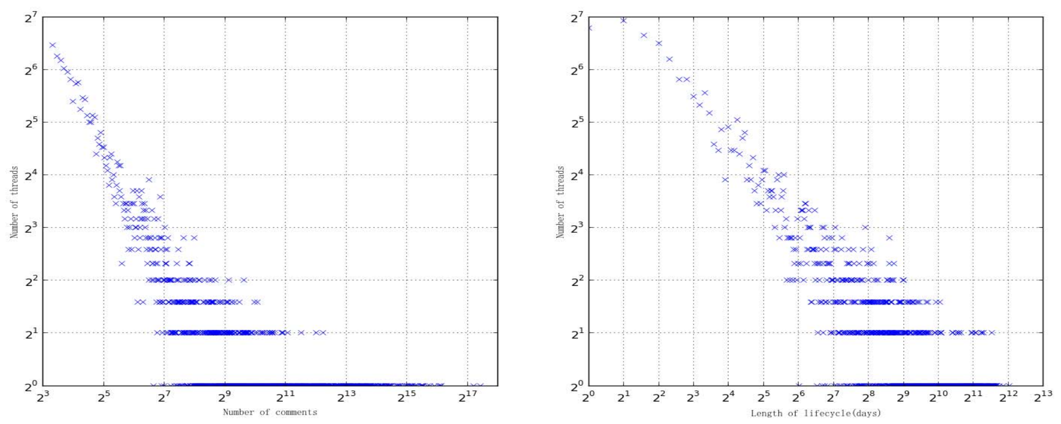

Figure 4 Distributions of threads

• *ML* (*Multivariate Linear*) *method.* For each training thread sample, ML method first records newly added comments in each sampling interval, denoted as $N(i)$, where i is the index of sample intervals. ML method extracts features as follows: $x(i) = N(i+1) - N(i)$. Please refer to [15] for more details.

C. Parameter Settings

The difference between t_r and t_t, denoted as Δt, is fixed to be 24 hours, i.e. the IPW algorithm predicts the threads' popularity one day after t_r. Sampling interval is set to be 3 hours. The number of nearest neighbors is set to be 5. Now there are only two free parameters left: the threshold p and t_r.

Other experiment details include: 1) the ratio between training set size and test set size is set to be 4; 2) to avoid the skewness of the dataset, down sampling [29] strategy is adopted to keep the two classes balanced; 3) the thread will not be included in test set if it is already dead before time t_r.

D. Experimental Results and Analysis

In this section, we first compare baseline methods, methods from related work, and the IPW algorithm. We also design experiment to see how threads' evolution stages affect the prediction results when t_r varies and how threads' popularity growth type affect the prediction results when p varies.

1) Comparison of prediction results

In this experiment, p and t_r are set to be 0.5 and 48 hours (relative to the thread publishing date and hereinafter), respectively. So t_t is 72 hours as the difference between t_r and t_t is 24 hours. Table III shows the prediction results.

As we have observed in Table III, single dynamic factors have different prediction results, showing that they have different classification capabilities. In addition, compared to single dynamic factor method, voting method does improve the prediction result.

TABLE III. PREDICTION RESULTS

Methods		**Prediction accuracy**
kNN algorithm with single dynamic factors	Comment count	57%
	User count	54%
	Depth of comment tree	56%
	Average path of comment tree	54%
	Link density of reply network	58%
	Average degree of reply network	57%
Voting method		59%
S-H model		50%
ML method		55%
IPW algorithm		**61%**

As for the S-H model, in this problem definition setting, all test samples can only be classified to one class, because all samples share the same linear coefficient α.

The ML method uses the comment count as features, and has relatively good performance in predicting whether the threads will have higher or lower popularity growth. Similar to the single factor methods, the ML methods only uses one factor, i.e. the comment count, as features, without takes advantage of other dynamic factor information.

Compared to the baseline and other related methods, the proposed IPW algorithm achieves relatively better prediction performance.

2) Influence of evolution stages on prediction results

In this experiment, we fix the threshold p and vary t_r, observing how different evolution stages of threads influence popularity prediction results for each single dynamic factor method and the IPW algorithm.

The prediction results are shown in Table IV (the highest accuracies are in bold in each column). Table IV shows that, as t_r increases, similar to the results from Table

IV, prediction results of single dynamic factors are not stable neither, while the IPW algorithm's performance is relatively stable and always achieves the highest accuracy. The results also suggest that, as t_r increases, i.e. when the algorithm has more input, prediction results are not improved accordingly. One possible reason is that, as t_r increases, although the algorithms have more input, but more early stage evolution information also brings more noise.

3) Influence of popularity growth type on prediction results

Based on the popularity prediction problem definition, when threshold p approaches 0, larger portion of threads in class L_1 tend to have small r (see Eq. (1)), i.e. have high popularity growth between t_r and t_t, while when threshold p approaches 1, larger portion of threads in class L_2 tend to have large r (see Eq. (1)), i.e. have low popularity growth between t_r and t_t. In this experiment, we fix t_r and t_t to be 48 hours and 72 hours, respectively and vary p, and find how different popularity growth types of threads influence popularity prediction results for each single dynamic factor method and the IPW algorithm.

The prediction results are shown in TABLE V. TABLE V shows that, as threshold p increases, the prediction accuracy of single dynamic factors is not stable and relative performance ranking is also changing. One possible reason is that, different dynamic factors capture different levels of popularity growth and thus have different popularity prediction performance. Compared to kNN methods using single dynamic factors, the proposed IPW algorithm has more stable prediction results and achieves relatively higher accuracy under p's most settings.

From the above prediction results we can see that the IPW algorithm performs relatively better than the other methods it compared with. However, its prediction accuracies are still not very high in general. One possible reason for this is that when the structure of comment tree and user reply network of some threads in the dataset is trivial, and the structural information will be of little help, so the prediction performances are degraded. Another possible improvement of the IPW algorithm is to assign different weights to comments published at different time, because intuitively newer comments would contribute more to the thread's popularity than older comments.

TABLE IV. PREDICTION RESULTS WHEN VARYING t_r

t_r		24	36	48	60	72	84	96
kNN algorithm with single dynamic factors	Comment count	62%	60%	52%	53%	54%	57%	54%
	User count	60%	61%	49%	50%	50%	57%	52%
	Depth of comment tree	57%	56%	57%	59%	57%	55%	58%
	Average path of comment tree	56%	55%	53%	59%	57%	57%	60%
	Link density of reply network	59%	58%	57%	60%	60%	59%	**63%**
	Mean degree of reply network	59%	58%	**58%**	59%	59%	54%	61%
IPW		**65%**	**62%**	**58%**	**62%**	**62%**	**62%**	**63%**

TABLE V. PREDICTION RESULTS WHEN VARYING p

Threshold p		0.2	0.3	0.4	0.5	0.6	0.7	0.8
kNN algorithm with single dynamic factors	Comment count	54%	54%	56%	55%	52%	58%	58%
	User count	54%	55%	54%	52%	51%	58%	53%
	Depth of comment tree	59%	59%	58%	59%	50%	55%	**60%**
	Average path of comment tree	58%	58%	53%	57%	50%	53%	52%
	Link density of reply network	55%	57%	57%	59%	53%	59%	59%
	Average degree of reply network	55%	54%	59%	57%	50%	57%	56%
IPW		**60%**	**61%**	**60%**	**62%**	**56%**	**59%**	58%

V. CONCLUSIONS

Online contents are rich sources for collecting, disseminating and sharing of public opinions and it is becoming increasingly important in Web-enabled security research. In this paper, we study the popularity prediction problem of forum threads on public events security, which cover a broad range of topics and large user base. Based on the early stage evolution process of forum threads, we construct dynamic factors which affect the future popularity of threads, and propose the instance prior weighting (IPW) algorithm that combines multiple dynamic factors to improve the prediction results. To evaluate the proposed algorithm, we use real-world forum threads on the discussions of public events. The experiment results show that, compared to the related work and baseline methods, our proposed algorithm achieves relatively better performance in predicting popularity evolution of forum threads.

Due to the characteristics of the current problem context, our work in this paper utilizes the forum dataset and adopts binary classification for problem definition. Nonetheless, our work can be easily extended to multi-classification problem definition and other similar media forms like Weibo. Future work includes further analysis on how these dynamic factors interact, how they affect threads' popularity, and use both static factors and dynamic factors to improve the prediction results.

ACKNOWLEDGMENTS

This work is supported in part by NNSFC grants #61175040, #71025001, #71103180, #91124001 and #91124002, MOH grant #2012ZX10004801, and grant #2013A127.

REFERENCES

[1] W. Mao, A. Tuzhilin, and J. Gratch, "Social and Economic Computing," *IEEE Intell. Syst.*, vol. 26, no. 6, pp. 19–21, Nov. 2011.

[2] D. Zeng, H. Chen, R. Lusch, and S.-H. Li, "Social Media Analytics and Intelligence," *IEEE Intell. Syst.*, vol. 25, no. 6, pp. 13–16, Nov. 2010.

[3] Q. Kong, W. Mao, and D. Zeng, "Predicting user participation in social networking sites," in *2013 IEEE International Conference on Intelligence and Security Informatics (ISI)*, Seattle, WA, USA, 2013, pp. 154–156.

[4] F. Figueiredo, "On the prediction of popularity of trends and hits for user generated videos," in *Proceedings of the sixth ACM international conference on Web search and data mining*, New York, NY, USA, 2013, pp. 741–746.

[5] J. Ugander, L. Backstrom, C. Marlow, and J. Kleinberg, "Structural diversity in social contagion," *Proc. Natl. Acad. Sci.*, vol. 109, no. 16, pp. 5962–5966, 2012.

[6] J. Yang and J. Leskovec, "Patterns of temporal variation in online media," in *Proceedings of the fourth ACM international conference on Web search and data mining*, New York, NY, USA, 2011, pp. 177–186.

[7] G. Chatzopoulou, C. Sheng, and M. Faloutsos, "A first step towards understanding popularity in YouTube," in *INFOCOM IEEE Conference on Computer Communications Workshops*, San Diego, CA, USA, 2010, pp. 1–6.

[8] Y. Borghol, S. Mitra, S. Ardon, N. Carlsson, D. Eager, and A. Mahanti, "Characterizing and modelling popularity of user-generated videos," *Perform. Eval.*, vol. 68, no. 11, pp. 1037–1055, 2011.

[9] F. Figueiredo, F. Benevenuto, and J. M. Almeida, "The tube over time: characterizing popularity growth of youtube videos," in *Proceedings of the fourth ACM international conference on Web search and data mining*, New York, NY, USA, 2011, pp. 745–754.

[10] 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 *Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining*, New York, NY, USA, 2012, pp. 1186–1194.

[11] A. Brodersen, S. Scellato, and M. Wattenhofer, "YouTube around the world: geographic popularity of videos," in *Proceedings of the 21st international conference on World Wide Web*, New York, NY, USA, 2012, pp. 241–250.

[12] P. Yin, P. Luo, M. Wang, and W.-C. Lee, "A straw shows which way the wind blows: ranking potentially popular items from early votes," in *Proceedings of the fifth ACM international conference on Web search and data mining*, New York, NY, USA, 2012, pp. 623–632.

[13] M. Ahmed, S. Spagna, F. Huici, and S. Niccolini, "A peek into the future: predicting the evolution of popularity in user generated content," in *Proceedings of the sixth ACM international conference on Web search and data mining*, New York, NY, USA, 2013, pp. 607–616.

[14] H. Li, X. Ma, F. Wang, J. Liu, and K. Xu, "On popularity prediction of videos shared in online social networks," in *Proceedings of the 22nd ACM international conference on conference on information & knowledge management*, New York, NY, USA, 2013, pp. 169–178.

[15] H. Pinto, J. M. Almeida, and 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*, New York, NY, USA, 2013, pp. 365–374.

[16] T. Zaman, E. B. Fox, and E. T. Bradlow, "A bayesian approach for predicting the popularity of Tweets," arXiv e-print 1304.6777, Apr. 2013.

[17] A. Kupavskii, A. Umnov, G. Gusev, and P. Serdyukov, "Predicting the audience size of a Tweet," in *Proceedings of the seventh International Conference on Weblogs and Social Media*, Cambridge, Massachusetts, USA, 2013.

[18] M. Jenders, G. Kasneci, and F. Naumann, "Analyzing and predicting viral tweets," in *Proceedings of the 22nd international conference on World Wide Web companion*, Republic and Canton of Geneva, Switzerland, 2013, pp. 657–664.

[19] L. Hong, O. Dan, and B. D. Davison, "Predicting popular messages in Twitter," in *Proceedings of the 20th international conference companion on World Wide Web*, New York, NY, USA, 2011, pp. 57–58.

[20] H. Ma, W. Qian, F. Xia, X. He, J. Xu, and A. Zhou, "Towards modeling popularity of microblogs," *Front. Comput. Sci.*, vol. 7, no. 2, pp. 171–184, Apr. 2013.

[21] G. H. Chen, S. Nikolov, and D. Shah, "A latent source model for nonparametric time series classification," in *Advances in Neural Information Processing Systems*, 2013, pp. 1088–1096.

[22] Z. Ma, A. Sun, and G. Cong, "Will this #hashtag be popular tomorrow?," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2012, pp. 1173–1174.

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

[24] K. Lerman and T. Hogg, "Using a model of social dynamics to predict popularity of news," in *Proceedings of the 19th international conference on World Wide Web*, New York, NY, USA, 2010, pp. 621–630.

[25] A. Khosla, A. Sarma, and R. Hamid, "What makes an image popular?," in *Proceedings of the 23rd international conference on World Wide Web companion*, Seoul, Korea, 2014.

[26] J. Cheng, L. A. Adamic, and P. Alex Dow, "Can cascades be predicted?," in *Proceedings of the 23rd international conference on World Wide Web companion*, Seoul, Korea, 2014.

[27] L. Ye and E. Keogh, "Time series shapelets: a new primitive for data mining," in *Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining*, New York, NY, USA, 2009, pp. 947–956.

[28] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*, 1st ed. Chapman & Hall/CRC, 2012.

[29] N. Japkowicz and S. Stephen, "The Class Imbalance Problem: A Systematic Study," *Intell. Data Anal.*, vol. 6, no. 5, pp. 429–449, Oct. 2002.