Personalized Recommendation for Online Social Networks Information: Personal Preferences and Location-Based Community Trends

Shaymaa Khater, Denis Gračanin, *Senior Member, IEEE*, and Hicham G. Elmongui

***Abstract*— Microblogs, such as Twitter, are a way for users to express their opinions or share pieces of interesting news by posting relatively short messages (corpus) compared with the regular blogs. The volume of corpus updates that users receive daily is overwhelming. Also, as information diffuses from one user to another, some topics become of interest to only small groups of users, thus do not become widely adopted, and could fade away quickly. This paper proposes a framework to enhance user's interaction and experience in social networks. It first introduces a model that provides better subscription to the user through a dynamic personalized recommendation system that provides the user with the most important tweets. This paper also presents TrendFusion, an innovative model used to enhance the suggestions provided by the social media to the users. It analyzes, predicts the localized diffusion of trends in social networks, and recommends the most interesting trends to the user. Our performance evaluation demonstrates the effectiveness of the proposed recommendation system and shows that it improves the precision and recall of identifying important tweets by up to 36% and 80%, respectively. Results also show that TrendFusion accurately predicts places in which a trend will appear, with 98% recall and 80% precision.**

***Index Terms*— Recommendation systems, social networks, topics modeling, trending topics.**

I. Introduction

ONLINE social networks are experiencing an explosive growth in recent years in both the number of users and the amount of information shared. Through these message streams, the users can connect with each other, share, find content, and disseminate information. Some of these sites provide social links (e.g., Twitter and Facebook). Others are used to share content (e.g., Youtube and Flicker). Understanding users' behavior in these sites is one of the important research challenges.

We use Twitter social network as our case study. Twitter, one of the most popular microblogging social media platforms, was launched in July 2006 and has about 328 million monthly active users and about 500 million postings per day [1]. Twitter poses a question to its users, "what is happening?" and the answer to this question is restricted to 140 characters. Twitter users receive information feeds, either by subscriptions: where subscription is related to the updates that the user request from his online community (i.e., by following friends and groups). or by suggestions through which the social media send the users information that they might be interested in. One of the important suggestions by the social media is the trending topics suggested to the user based on geographic location.

However, the online social media face different challenges in providing these streams of information to the user. While tweets may contain valuable information, many are not interesting to the users. A large number of tweets can overwhelm users, since they interact with many other users and they have to read ever-increasing content volume on their timeline [2]. Thus, the difficulty in recommending content that was of interest to users became a key challenge for social networks sites. Moreover, on the suggested stream level, current research focuses on providing the trending topics based on the current popularity of the topics within the geolocated communities. This, in turn, might miss the topics of interest that can affect these locations in the future.

The goal of this paper is to provide the user with personalized recommendation for online social network information. This is based on both the individual level and the geolocated community level. We are building on our previous work [3] to add recommendation on the geolocated communities' level. The general structure is shown in Fig. 1. On the individual level, the proposed approach provides better subscriptions view for the user (a tweets analysis system in Fig. 1). Accordingly, we propose a new model of a dynamic personalized recommendation system that provides the user with the most important tweets. The proposed model captures the user's interests, which change over the time, and shows the messages that correspond to such dynamic interests.

Moreover, on the geolocated community level, this paper focuses on improving the suggestions provided by the social media to the users (a trends analysis system in Fig. 1). Our assumption is that enhancing the trending topics suggested by the social media to user based on their location will reflect positively on their online experience. We approached this point by investigating the interplay between local community interests and public trends. We developed a model for predicting localized trends diffusion from one localized community of users to other geographically separated communities of users. We show that observing the local trends in some locations (e.g., cities) can be used to predict where these trends

Manuscript received July 9, 2015; accepted June 9, 2017. Date of publication July 21, 2017; date of current version August 28, 2017. *(Corresponding author: Denis Gračanin.)*

S. Khater is with Solebrity, Inc., Ashburn, VA 20147 USA (e-mail: shkater@mena.vt.edu).

D. Gračanin is with the Department of Computer Science, Virginia Tech, Blacksburg, VA 24060 USA (e-mail: gracanin@vt.edu).

H. G. Elmongui is with the Department of Computer and Systems Engineering, Alexandria University, Alexandria 21544, Egypt (e-mail: elmongui@alexu.edu.eg).

Digital Object Identifier 10.1109/TCSS.2017.2720632

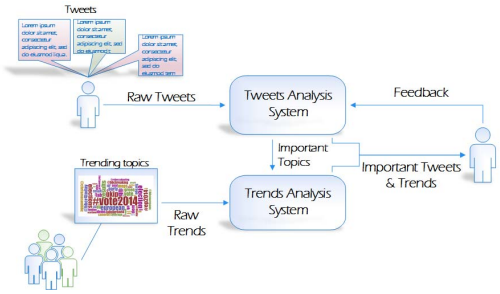


Fig. 1. General Framework.

will appear next. Finally, the interesting topics for the user discovered by the tweets analysis system are then used by the trends analysis system to personalize the trends suggested to the user.

The most important aspect of our model is the prediction of trends that will appear in a location, before even users in that location start mentioning that topic. This is extremely useful in many cases, such as building a proactive localized recommendation system for topics or for early prediction of social events (e.g., protests).

Our contributions are as follows.

1) The notion of dynamic level of interest (LoI) for the recommendation made to the Twitter users. We build a user-specific-time-variant (dynamic) LoI graphs for each topic constituting the tweets. This is based on utilizing the weights of topics in the user's tweets to determine its level of importance to the user.
2) A model incorporates the dynamic change in users' interests in tweets topics, along with other social features and tweets-related features to recommend interesting tweets for the user.
3) A new information diffusion model [*snowball cascade (SC) model*] in online social networks is suitable to model the diffusion between geographically separated communities, rather than relying on the users' social network structure.
4) TrendFusion, a predictive model that can predict whether the trending topics will appear in a certain city in the future, along with the activeness time, i.e., the time it will appear.
5) A Web application that recommends to a user the most interesting tweets according to past interests, along with predicting the trending topics and their related tweets that will appear in a location selected by the user.

II. Related Work

Since this paper focuses on the interplay between personal interests and public trends, we study the information propagation process in social networks and trending topics in geolocated communities.

A. Recommendation Systems in Twitter

Recommendation systems had first emerged as an independent research area in the mid-1990s when researchers started focusing on recommendation problems that depend on the ratings structure. These recommendation problems were reduced to the problem of estimating ratings for the items that have not been seen by a user [4]. Intuitively, this estimation is usually based on the ratings given by this user to other items. Once we can estimate ratings for the yet unrated items, we can recommend to the user the item(s) with the highest estimated rating(s).

Moreover, the recent popularity of online social network sites and the overwhelming amount of information available today made it difficult for users to find useful information. As a solution to this problem, many recommender systems were introduced to help users find interesting information. Some of these systems were conducted to study the social network structure and recommend friends to the user based on the similarities of interests. One study proposed Twittomender that recommends Twitter users based on the relationships of their Twitter social graphs [5]. Kwak *et al.* [6] estimated the influence of users on Twitter by proposing three methods: the number of followers, PageRank, and the number of retweets.

Other systems studied the personalized recommendation systems to recommend only useful content to the users. Chen *et al.* [7] proposed a collaborative filtering method to generate personalized recommendations in Twitter through a collaborative ranking procedure. A URL recommendation model demonstrated the utility of various combinations of tweet content and social graph information during recommendation [8]. These methods are different from this paper as they miss the dynamic change of interests of the user.

A limited work has been done in dynamically personalized tweet recommendation. The study done by Abel *et al.* [9] is the most relevant to our problem. They analyzed how users' profiles changes over time, and how to recommend news articles for topic-based profiles. Our model is different in that it tries to capture the change of each user's interest in different topics over time, and recommend interesting tweets based on this interest. Uysal and Croft [10] explored user-publisher and user-tweet features to rank the Twitter feed for each user based on their probability of being retweeted. Compared with our model, it just uses the explicit features of the tweets without considering the personalized features for each user.

B. Topic Modeling

Topic Modeling is a rapidly growing field of research in the area of text mining and statistical modeling. As text comprises about 85% of data worldwide [11], topic models have been widely used to address the problem of "information overload" associated with this huge collection of text and corpuses. They are also extended in many ways to be used in social media to identify text patterns in the content and to facilitate many applications, such as sentiment analysis and content filtering [12].

Although topic models, such as latent Dirichlet allocation (LDA), had been applied successfully on articles and documents, their application on microblog contents, such as Twitter, faces different challenges: 1) the posts are short, 140 characters; 2) the use of informal language and nonstandard abbreviations (e.g., LOL and WOW); and 3) the

text contains other context that may act as noise as the URL, Twitter names, and tags. To overcome these difficulties, some studies proposed to aggregate all the tweets of a user in a single document [13]. This can be regarded as an application of the author-topic model [14] to tweets, where each document (tweet) has a single author. Another modification to the author-topic model was introduced by Zhao *et al.* [15]. They introduced a model, Twitter-LDA, which assumes a single-topic assignment for an entire tweet. The model is based on the following assumptions. There is a set of topics T in Twitter, each represented by a word distribution. Each user has topic interests, modeled by a distribution over the topics. When a user wants to write a tweet, the user first chooses a topic based on the user's topic distribution. Then, the user chooses a bag of words one by one based on the chosen topic. However, these treatments assume that the user's interests in topics will not change over time, which contradicts our assumption that the user's interest in topics changes over time. We are proposing a complementary approach to the LDA model that can help in extracting better topics from microblogs.

C. Information and Influence Propagation in Social Networks

In recent years, information propagation on social networks has been attracting much attention in academic and industrial circles [16]. Understanding the mechanisms of information propagation is vital to finding the factors affecting the information propagation process. These factors, in turn, provide a better explanation for predicting information popularity [17].

Two factors affect the information propagation process: the importance of the information and the level of interactions between the users. The studies of the first factor mainly consider the analysis of the messages propagation and the decay with respect to the time since the posting of the message [18]. Most of these approaches are descriptive. However, our approach is predictive.

For the second factor, the level of interactions between the users, the current research efforts focus on the interactions between the users, along with the geographic, demographic, topical, and contextual features that affect the propagation between the users [19]. For example, Galuba *et al.* [20] proposed a propagation model that predicts which users will tweet about which URL based on the history of past user activity. Agarwal *et al.* [21] studied the problem of identifying influential bloggers in the blogosphere.

As our model analyzes and predicts the localized diffusion of trends in social networks between locations, this paper is different in that it does not take into account the social structure of the social networks. It is prohibitively complex to include the social structure connections relating the locations. Another point is that the location information posted by the user is not always available or accurate. This paper is also different from the research that studies relationship between geography and information diffusion [22], as our model considers other nongeographical parameters.

D. Trends in Social Networks

Trending topics in Twitter are words and phrases, appearing on the main page of Twitter, that are currently popular in users' tweets, and are identified for the past hour, day, and week. They represent the popular topics of conversations among the Twitter users [6]. Trends in social networks have recently been a focus of interest for many researchers.

Some works studied trends from a temporal view. Leskovec *et al.* [23] studied the temporal properties of information shared in social networks by tracking memes across the blogosphere. Other works studied the structural nature of the social graph that leads to creating the trends [24].

Other studies focused on studying the dynamics, the growth, and the decay of the trending topics [25], [26]. Asur *et al.* [25] studied the trending topics on Twitter and provided a theoretical basis for the factors affecting the formation, persistence, and decay of trends.

A limited work has been done to analyze the relation between trends and geography. Kamath *et al.* [27] modeled the social media spread from location to location by trying to predict the top K cities in which a topic will be trending. Ferrara *et al.* [28] investigated the spatial and geographic dynamics that govern trending topics in Twitter. However, their goal was different, as they aimed at studying what dynamics underlie the production and consumption of trends in different geographic areas. In other words, they wanted to know if trends travel through the Internet, or by people physically traveling across cities.

III. Tweets Analysis: Personalized Recommendation System Model

We now describe the first model we developed to provide better subscriptions view for the user.

A. Problem Description

When users log on Twitter, they see a stream of tweets sent by friends, which composes their timeline. Many of these tweets are conversational tweets and/or are not of personal interest to the user. The goal of our model is to decide for each user the tweets that might be of interest from the user's timeline. Beside being able to post their own tweets, users can also interact with their timeline by replying, retweeting, or favoring the tweets. As there are no explicit means to extract the user's LoI in a tweet from Twitter, we relied on these actions to predict the user's interests. Hence, the retweets, replies, and favorites can be used as an indication of the interest of the user in the corresponding tweets. We define a tweet as a tuple $\langle \mathbf{u}, \mathbf{p}, \mathbf{e}, \mathbf{o}^e, t, \text{int}_{\text{tu}} \rangle$ where (vectors are in boldface) $\mathbf{u}$ is a vector describing the features of the user u receiving the tweet, $\mathbf{p}$ is a vector describing the features of the publisher p of the tweet, $\mathbf{e}$ is a vector describing the features of the tweet e, $\mathbf{o}^e$ is a vector that holds the distribution of probabilities of the tweet text e across different topics, t is the time window in which the tweet is posted, and int_{tu} is a binary value indicating whether this tweet is of interest to the user u or not.

Given these tuples for the tweets in the user history, our goal is to predict int_{tu} for each new tweet.

We now describe in more detail our approach and discuss its main components (Fig. 2).

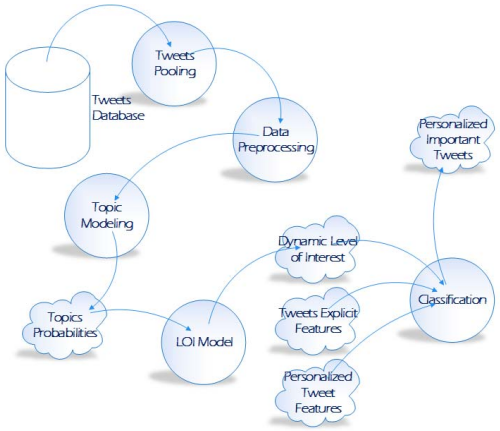


Fig. 2. Tweets analysis structure.

B. Topic Extraction

In order to predict the user's interest in a corpus, we based our prediction on the user's interest in the topic(s) covered in that corpus, alongside with other features. Consequently, we needed to build a topic model of our tweets. Topic models, such as LDA [29], are well known for exploratory and predictive analysis of text. Generally, topic models define topics as distributions over the words in a vocabulary and documents as being generated by mixtures of these topics. Topic models represent document words in a bag-of-words format without considering word order to be of any particular importance. According to the frequencies of different words appearing together in each document, the model then determines the most relevant set of words to each topic. After training a topic model, it can be used later to infer the topic(s) available in new documents.

Formally, the LDA model can be described as follows.

Given: A set of e posts denoted by $\mathbf{E} = \{e_1, \ldots, e_n\}$, the LDA algorithm generates a set of k topics denoted by $\mathbf{L} = \{l_1, \ldots, l_k\}$. Each topic is a probability distribution over m words denoted by $l_i = \{w_1^i, \ldots, w_m^i\}$, where w_j^i is a probability value of word j assigned to topic l_i. The post can then be represent as $\mathbf{o}^e = \{o_1^e, \ldots, o_k^e\}$, where o_j^e is the percentage of topic l_j in the post e composition.

C. Tweets Pooling

The short length of tweets might result in a poor topic model. Thus, to help get around the problems associated with the analysis of numerous small documents, we construct large documents out of the tweets. Therefore, instead of looking at each tweet individually, we group together tweets that are similar in some sense (the same semantics, the same hashtags, and so on) in a process called *pooling*. In our model, we present some schemes that we used to aggregate tweets into a larger document from which a better topic model can be trained. These pooling schemes can be described as follows.

1) *Hashtag pooling:* In Twitter, the hashtag is a string of characters preceded by the hash # character. They are used as identifiers for tweets discussing the same topic [30]. By including hashtags in a message, users indicate to which conversations and topic their message is related to. Using these hashtags can be a good indicator for tweets relatedness, and so can be used in the aggregation process of tweets. For the hashtag-based pooling scheme, we aggregated documents sharing each hashtag in one pool, as in [31]. If a tweet has more than one hashtag, this tweet will be added to the tweet pool of each of those hashtags.
2) *Replies pooling:* We used replies for tweets as another way for aggregation. In general, a reply is a string preceded with the @ character. It is used as a comment on another tweet posted by you or by anybody in the social networks. As the tweets and their replies might share the same topics discussed, aggregating them in one pool can be a good indication for the tweet relatedness.
3) *URLs pooling:* We also aggregated tweets that include the same URLs in their text. Tweets sharing the same URLs might be discussing the same topic and, hence, can be aggregated.

In our model, we consider each aggregated pool of tweets a document, and the words in the pool the vocabulary. We use these documents and the vocabulary to extract the topics that form the corpus.

D. Dynamic Level of Interest

In this section, we study how the interests of individual users about a certain topic change over time. Getting the dynamic LoI in a tweet takes place through some steps.

1) First, we get the per topic activity in each day d for the user, denoted by $\mathbf{A}_d = \{a_1^d, \ldots, a_k^d\}$, where a_i^d is the level of activity of the user in topic l_i on day d. $\mathbf{A}_d$ is calculated by adding the vectors $\mathbf{o}^e$ in that day [see (1)]. The details of this step are shown in Algorithm 1

$$\mathbf{A}_d[i] = a_i^d = \sum_{\forall e \in E : e_{\text{date}} = d} \mathbf{o}^e[i] = \sum_{\forall e \in E : e_{\text{date}} = d} o_i^e. \quad (1)$$

2) Given a new tweet e_{new}, the user's LoI in the tweet can be calculated using (2). Basically, the equation sums up the user activity vectors in the window of the last seven activity instances prior to the tweet creation day d. Each of these instances corresponds to user's actions done in one day. For a user who is active (posting a tweet, replying, retweeting, or favoring another tweet) every day, this window will span one week period. For less active users (not active every day), this window will be longer to cover the last seven active days in which the user was active. We only consider the last seven instances, as considering intervals longer than seven days will introduce irrelevant noisy tweets as discussed in [32]. This step is illustrated in Algorithm 2.

$$\text{Level Of Interest}(u, e_{\text{new}}) = \sum_{l \in L} \left(\mathbf{o}^{e_{\text{new}}}[l] \cdot \sum_{d \to d-7} \mathbf{A}_d[l] \right) \sum_{l \in L} \left(o_l^{e_{\text{new}}} \cdot \sum_{d \to d-7} a_l^d \right). \quad (2)$$

Algorithm 1 Users Level of Activity per Topic

```
Procedure CalculateDailyActivityVectors
Input Set of all users users
begin
  L ← List of all topics
  for each User u in users do
    Days ← All Days in which u was active
    for each Day d in Days do
      Tweets ← All tweets by u in d
      // Initialize vector for user u in day d
      A_d^u ← [0, ..., 0]
      for each Tweet e in Tweets do
        o^e ← Percentages of topics in e
        for each Topic l in L do
          A_d^u[l] = A_d^u[l] + o^e[l]
        end for
      end for
    end for
  end for
end
```

Algorithm 2 LoI in a New Tweet

```
Function CalculateLevelOfInterest
Input User u
      Tweet e
begin
  o^e ← Percentages of topics in e from LDA
        model
  d ← e posting date
  LoI ← 0
  for each Topic l in L do
    val ← 0
    for i = 1 to 7 do
      val ← val + A_{d-i}^u[l]
    end for
    val ← val * o^e[l]
    LoI ← LoI + val
  end for

  return LoI
end
```

E. Personalized Tweet Recommendation

In addition to measuring the dynamic LoI for each user, some other static features can affect his interests. Some of these features represent the personalized interests of the user, and others are general features that are related to the tweet's quality or the publisher's authority that can affect the tweet's degree of interest to the user. Sections III-E1 and III-E2 describe the personalized features and other explicit features that might affect user's interest.

1) Personalized Social Features: Social features are the features that represent the social relationship between the user and the publisher. This relation can be friendship, neighborhood who posts tweets about events happening in the neighborhood or celebrities who have interests in common with the user.

A user-publisher similarity feature measures the similarity between the activity level of the user and the publisher on all topics. This is measured as the cosine similarity between vectors formed by the summation of the LoI in a topic for the user over time [see (3)]. Generally, the cosine similarity measure yields a value between -1 and 1. The value of 1 means the exact distribution match, i.e., activities of both the users are distributed in the same proportions on different topics, though one of them might be generally more active than the other. The value of 0 means that the users have nothing in common

$$\text{Cosine Similarity}(U_t, P_t) = \frac{U_t \cdot P_t}{\|U_t\| \cdot \|P_t\|} = \frac{\sum_{t=1}^{T} U_t \times P_t}{\sqrt{\sum_{t=1}^{T} {U_t}^2} \times \sqrt{\sum_{t=1}^{T} {P_t}^2}}. \quad (3)$$

2) Explicit Features: Besides the personalized social features, we analyzed other explicit features that can affect the user's interests. These features appear or can be inferred from the user profile. This includes the following.

1) *Publisher-Based Features:* Related to the tweets' publisher. These features are used as an indicator for the activity of the publisher.
   a) *Publisher Followers:* The number of followers for each publisher. High authoritative publishers are likely to have more followers than others.
   b) *Publisher Tweets Count:* The number of tweets posted by the publisher since the opening of the account. This feature is an indicator for how active the publisher is.
   c) *Mention Count:* The number of times a publisher's name is mentioned in all the tweets. If a publisher is frequently mentioned, the publisher is more likely to be popular and has more interactions than other publishers.
2) *Tweet-Based Features:* It describe the tweets contents as follows.
   a) *Retweet Count:* The number of times the tweet got retweeted. It is a way of estimating the popularity of the tweet. A tweet retweeted more times is more likely to be a useful one.
   b) *HasURL and HasHashtag:* Sometimes a publisher includes supplement to their tweets with URL or hashtags. Hashtags can sometimes be an indication of the tweet's topic.
3) *Location Feature:* It represents the cities or countries found in the publishers' profiles. This feature is used to capture the spatial neighborhood effects. If a publisher posted a tweet about local events, and this publisher is the user's neighbor, then, most probably, the user will be interested in this post.

F. Experimental Results

In this section, we describe our data sets and the preprocessing steps followed by the experimental results for each step in our model.

1) Data Set and preprocessing: For our experiments, we created a Twitter data set containing 5 million tweets and 20 000 users that were seeded by first selecting 100 active users from the Virtual Town Square blog [33]. We used Twitter REST API [34] to facilitate the data collection. The majority of the tweets collected were published in a three-month period from April 2013 to June 2013. We then expanded the user base by following their followers and friends. We were able to include 20 000 users with all their posts.

As Twitter APIs does not allow access to the timeline of the user directly without authorization, we build each user's timeline by first getting the posts for each of the base users and then following the tweets posted by their friends, and consider them the scanned tweets by the user. All the favored tweets by the base users are also retrieved.

We build our model from a repository of more than 5 million tweets. To eliminate incomplete and noisy data, we preprocessed the tweets by discarding tweets with non-English words. We also removed meaningless words, such as stop words, Twitter specific stop words, user names, and special characters, and stemmed the remaining words.

Usually, users do not have time to see all the tweets posted on their timeline. Also, users can be away or inactive (i.e., no posts, retweets, or favorites) for long periods of time. Using this period in our data set will make the number of negative examples much bigger than the number of positive examples.

To overcome this, we filtered the browsing history by considering a window made up of a set of 20 tweets. The number 20 is chosen due to the fact that Twitter limits the number of tweets to be retrieved to 20 tweets each time the user browse his timeline. The window's sliding scheme depends mainly on the user's action in time of browsing the history tweets. As in the case of retweeting or reply, the window of interest will be 15 tweets before the original tweet till 5 tweets after the user's action. When the user is just posting a tweet without referencing any history ones, the window of interest will be considered 15 tweets before and 5 tweets after the user's action, respectively. Twitter API does not reveal the exact date of favoring a tweet. In the case of favoring a certain tweet, the window of interest is chosen to be 15 tweets before and 5 tweets after the original favored tweet.

Fig. 3 shows the different cases for choosing the window of interest in the user's timeline. This filtration step is applied to the tweets before the classification step in both training and testing. The filtered out tweets are still used in training the topic model and calculating the user activities.

2) Tweets Pooling: The tweets pooling process aggregates semantically similar tweets into one pool. Each pool is treated as a document. We first began by aggregating each tweet and their replies into one pool. Then, for each hashtag, we aggregated all the tweets that are sharing the same hashtag. Finally, we aggregated the tweets that contain the same URLs in their content. This pooling process decreases the number of documents and increases the document size to be the size of the aggregated tweets.

3) Evaluating Topic Models: The unsupervised nature of topic modeling methods makes choosing one topic model over another a challenging task. Topic model quality tends to be evaluated by performance in a specific application. Topic models can be evaluated based on perplexity [35] as a quantitative method. Perplexity is a well-known standard metric used in information retrieval field. It tries to quantify the accuracy of a model by measuring how well the trained model deals with an unobserved test data as in (4). Perplexity is defined as a reciprocal geometric mean of per word likelihood of a test corpus. A lower perplexity indicates a better generalization performance

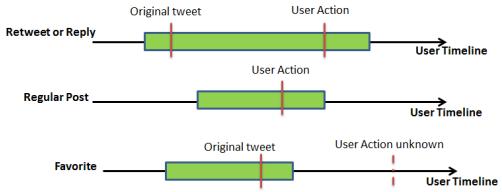


Fig. 3. Timeline window for the user.

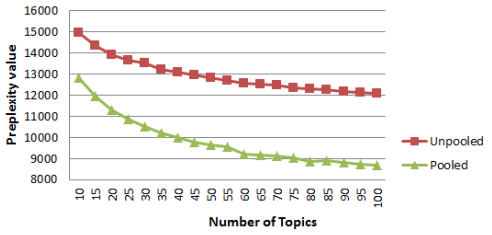


Fig. 4. Perplexity for LDA and our model.

$$\text{Perplexity}(D_{\text{test}}|M) = e^{\frac{-\sum_{d\in D_{\text{test}}} \log P(w_d|M)}{\sum_{d\in D_{\text{test}}} N_d}} \tag{4}$$

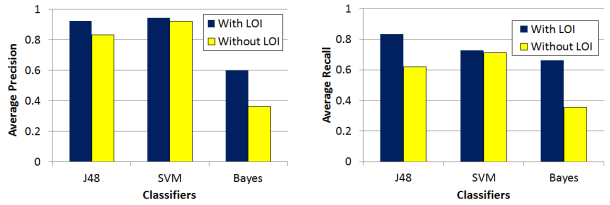
where w_d represents the words of test document d, M is the topic model, and N_d is the number of words in document d. The perplexity results of LDA with unpooled data and our model are shown in Fig. 4. The perplexity of the proposed method is better than LDA without pooling the data. We conducted our experiments using 35 topics, as the improvement in perplexity was low compared with the increase in the runtime. More details are provided in Section III-G2.

For topic extraction, we used the MAchine Learning for LanguagE Toolkit (MALLET) [36]. MALLET is a Java-based package that implements the LDA model. For example, top ten words for sports topic are browns, game, nfl, cleveland, football, coach, team, eagles, win, and season [3].

After the model is trained, it can be used to predict the topics in unseen corpuses. Thus, we can now predict topics distribution for every corpus in our database.

4) Calculating the Dynamic Level of Interest: In real life, the degrees of popularity of the topics are not constant. There are topics that attract more users than the others. Also, the user's interest in one topic can change from one time to another. The dynamic LoI is calculated using (2). The user's dynamic LoI is based on the dynamic level of activity of the user in each topic.

5) Personalized Recommender Model: Using the above-described features, a feature vector was created for all the



(a) (b)

Fig. 5. Average precision and recall for the three classifiers. (a) Average precision. (b) Average recall.

tweets in the activity windows of the users, as described in Section III-D. Each of the feature vectors is augmented by a class value. We considered the only possible class values are *interesting* or *not interesting*. The class value is set to be *interesting* if the user replied, retweeted, or favored the corresponding tweet. Otherwise, the class value is set to be *not interesting*. We used the feature vectors for each user individually to train three classifiers: 1) J48, a Java implementation of the C4.5 tree-based classifier [37]; 2) supervised support vector machine (SVM), a function-based classifier; and 3) naive Bayes classifiers [38]. The three classifiers are used to predict whether the tweets of the timeline are interesting (the user will most likely interact with) or noninteresting.

6) Dynamic LoI and Other Features Effect: We recorded two quality measures in our experiments: precision [$P = \mathrm{TP}/(\mathrm{TP} + \mathrm{FP})$] and recall [$R = \mathrm{TP}/(\mathrm{TP} + \mathrm{FN})$], where TP, FP, and FN are the number of true positive, false positive, and false negative examples, respectively.

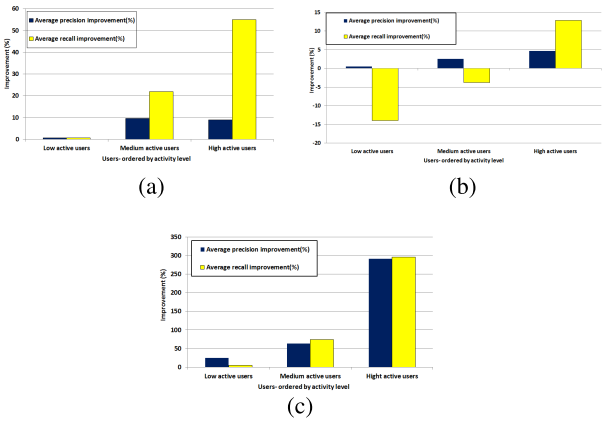
Fig. 5(a) shows the average precision values for the three classifiers. Using the *dynamic LoI* feature improved the average precision of J48 by 8% and improved that of the SVM and naive Bayes with about 2% and 36%, respectively. SVM is performing better than J48 and naive Bayes classifiers, when not relying on the *dynamic LoI*. The best results are achieved when using *dynamic LoI* feature with either J48 or SVM classifiers. When using the *dynamic LoI* feature, the J48 and SVM performed equally. The naive Bayes classifier performed better with *dynamic LoI*, but not as good as the other two classifiers.

Fig. 5(b) shows an improvement in J48, SVM, and naive Bayes average recall by 33%, 3%, and 80%, respectively. J48 is also outperforming SVM and the naive Bayes classifiers when using *dynamic LoI*.

G. Discussion

We had to accurately judge the gain from including the *dynamic LoI* feature and to determine the influence of users who have few tweets, For that, we used the concept of "active users" from the traditional media research [39] and focused on those users with some minimum level of activity. We sorted the users by their activity level in posting, retweeting, or favoring the others posts. The users are then divided into three categories according to their activity level (high active, medium active, and low active users).

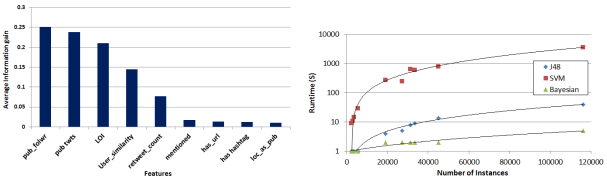
Fig. 6(a)–(c) shows the average gain in precision and recall values in J48, Bayes, and SVM classifiers, respectively, when



(a) (b)

(c)

Fig. 6. Average gain in precision and recall with including dynamic LOI feature. (a) J48—activity level effect. (b) SVM—activity level effect. (c) Bayes—activity level effect.



(a) (b)

Fig. 7. Classification analysis. (a) Average feature information gains across all users. (b) Classification runtime.

including the *dynamic LoI* feature. A positive value means that including the feature improved the classification, whereas a negative value means that including the feature worsen the classification. The gain from including the *dynamic LoI* feature is higher when considering users with high activity. This makes our model more important for highly active users. The only negative gain is with the SVM classifier for users with less activity. This is intuitive since less active users do not show their interest in the topics as they do not retweet or reply on the tweets.

We had also evaluated the relative importance of other features used in the classification process. We used the information gain feature selection method to measure the dependence between features and the class labels [40]. Fig. 7(a) shows the features ranked according to their average information gain. It is clear that the LOI feature is considered one of the relatively highest important features in the classification process as compared with other features. We analyzed the classification runtime for each classifier. Fig. 7(b) shows the runtime for the three classifiers. It is clear that the SVM has the longest runtime compared with J48 and Bayes classifiers (note the logarithmic scale on the Y-axis).

1) Tweets Pooling Effect: Since changing the number of words in the documents can greatly affect the output of the topic modeling step, we repeated the experiments after applying the pooling step. The experiments show that the average recall was improved by more than 6% without loss in precision.

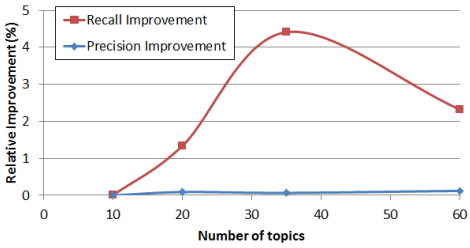


Fig. 8. Number of topics variation effect.

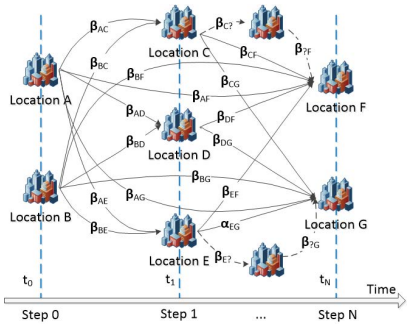


Fig. 9. Information cascade represented by a DAG.

2) Number of Topics Variation Effect: Besides our previous experiments, we analyzed the effect of varying the number of topics on the classification process. For example, a user might be interested in a certain topic, but the classifier would only recognize the user's interest in a subtopic. We demonstrated this by reconducting the experiments with the J48 classifier while varying the number of topics (Fig. 8). Generally, the small number of topics results in very broad topics. This results in poor classification.

On the other hand, a large number of topics will result in many very specific topics, as subtopics become the main categories. This also leads to poor classification in our case. Again, the variation of the number of topics has a minor effect on precision, but the recall was improved by 4% by having around 35 topics. The recall value dropped again, when raising the number of topics to 60. Therefore, although the perplexity value was better at 60 topics than 35 topics, the overspecification did not help in our case.

To clarify, assume that we have tweets about different sports. Ideally, in the generated topics by LDA, there will be a general topic for all sports, such as football and basketball. Using a small number of topics will lead to very broad topics. Continuing in the same example, the sports might be merged with other nonrelated topics to form a bigger topic. For example, the topic modeling system (LDA) might merge sports with movies for instance to generate an entertainment topic. Therefore, assume there is a user that is only interested in sports. This will make our system mistakenly recommend tweets about movies to the user.

On the other hand, a large number of topics mean that the sports will be split into more specific topics, for example, football, basketball, and so on. According to the user timeline, our system might detect his interest in one of these sports, such as his interest in football, but miss his interest in basketball. This explains the rise and drop of performance of our system when the number of topics is varying from small to large.

IV. Trends Analysis: TrendFusion Model

This section describes TrendFusion, our proposed model for improving the suggestions provided by the social media to the users. The model is used for predicting localized trends diffusion from one localized community of users (location) to other geographically separated communities of users.

The TrendFusion model relies on the information cascade concept to represent the flow of a piece of information, usually called the contagion, through a social network [41]. The cascade is usually represented as a directed acyclic graph (DAG). Fig. 9 shows an example of information cascade, where it has the following.

1) *Nodes:* The entities (such as users, groups, or cities) that represent locations in our model.
2) *Edges:* They represent the information propagation between entities.
3) *Seeds:* The vertices initiate the cascade.
4) *An Activation Step (or a Step):* Every time a given trend appears at the same time at one or more entities corresponds to an activation *step*, or simply a step, in the cascade.
5) *A Cascade:* A sequence of activation steps generated by a contagion process. The weights on the edges represent the influence of an active entity on an inactive one. The way to calculate these influences and how an inactive node responds to them are specific to each model.

A. TrendFusion Framework

The two main objectives of TrendFusion are as follows.

1) Predict whether a trend will appear for some location based on its diffusion in other locations.
2) Predict when the trend will appear.

The problem we are trying to solve can be defined as follows.

1) *Given:* A history of spatially and temporally tagged trending topics in a number of locations.
2) *Processing:* Define a model that can extract and capture the dependence relations between locations.
3) *Output:* When a topic is trending in some locations, use the model to predict where and when this topic will be trending next.

B. TrendFusion Model

Generally, most information diffusion models assume that the considered entities (such as users and groups) are connected by a social graph and that the graph structure is known beforehand. In our case, there is no such social graph connecting the locations together. Thus, before applying any known diffusion model, we need first to infer the underlying *hypothetical* graph that describes the influence between the

locations. Fortunately, several network inference models have been developed recently [41]–[43]. These algorithms estimate the underlying network structure given past activation times.

In the TrendFusion model, we assume a fully connected graph and estimate the transmission rates along the edges using the NetRate algorithm [44]. We based our assumption of the fully connected graph on the first law of geography by Tobler [45], "*Everything is related to everything else, but near things are more related than distant things.*" We start with a fully connected graph of the locations and estimate the transmission rate between each pair of locations using NetRate. The lowest transmission rates are then omitted reducing the edges (connections) between the locations.

The NetRate algorithm estimates the transmission rates, not just a binary ON/OFF value. The algorithm takes the input in the form of information cascades. The NetRate algorithm relies on the survival theory and the concept of *hazard rate* that will be explained shortly [46].

C. Generating the Hazard Rate Graph

After converting the activations into different cascades of trends between the locations, we compute the pairwise hazard function between these locations. The hazard rate is mostly related to the survival theory [46] and can be described as the instantaneous activation rate between two locations i and j [44], i.e., how likely is it that location j will adopt a trend at time t_j, if location i adopted that trend at time t_i [see (5)]

$$H(t_j|t_i;\lambda_{i,j}) = \frac{f(t_j|t_i;\lambda_{i,j})}{S(t_j|t_i;\lambda_{i,j})} \tag{5}$$

where $f(t_j|t_i)$ is the conditional likelihood of transmission from location i to location j. Likelihood depends on the activation times t_i and t_j (i.e., the time the trend first appears in location i and location j) and a pairwise transmission rate $\lambda_{i,j}$. The transmission rate $\lambda_{i,j}$ models the strength of an edge (i, j) and determines how frequently information spreads from location i to location j. The most commonly used parametric models for the shape of the conditional transmission likelihood are the exponential, power-law, and Rayleigh distributions models [46]. $S(t_j|t_i)$ in (5) refers to the survival function computed for the edge connecting the locations i and j. It is computed as the probability that location i does not cause location j to activate by time t_j as

$$S(t_j|t_i;\lambda_{i,j}) = 1 - F(t_j|t_i;\lambda_{i,j}) \tag{6}$$

where $F(t_j|t_i)$ denotes the cumulative density function computed from the transmission likelihoods.

D. TrendFusion Stages

TrendFusion consists of five stages (Fig. 10). The first three stages can be shared across the locations of interest. Stages 4 and 5 should be repeated for each location.

1) Stage 1: Collect Trends From Locations: Trends should be collected from all the locations of interest. The trends are collected every Δt time unit. If social media do not reveal the localized trending topics, an extra step of monitoring user activities and extracting the trending topics is needed.

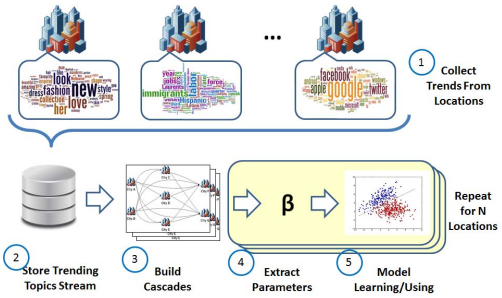


Fig. 10. Stages of TrendFusion model.

Algorithm 3 Build Cascades

Procedure BuildCascadesFromActivations
Input ActivationsList al
begin
  // An activation a is a record $a = (trend, location, time)$
  ActivationsList $alo \leftarrow$ Order al by time
  for all Activation a **in** alo **do**
    if $a.trend$ appeared in ($a.time$ - 24 hours) **then**
      $cas \leftarrow$ last cascade of $a.trend$
      if $a.location$ appeared in cas **then**
        Add $a.time$ to instances of $a.location$ in cas
      else if $a.time$ equals time of last step in cas **then**
        Add a to last step of cas
      else
        Add new step to cas containing $a.location$
      end if
    else
      Create new cascade cas
      Add new step to cas containing $a.location$
    end if
  end for
end

2) Stage 2: Store Trending Topics Stream: As the stream of the trending topics is received, they are labeled by the location/time and they were received from/at. The trending topics are stored for further analysis.

3) Stage 3: Build Cascades: Since trending topics are continuously polled every fixed time step, it is not always clear if a trend is a beginning of a new cascade or a continuation of an old one. Therefore, a process is needed to build cascades from trending topics that are retrieved every Δt. Algorithm 3 provides the details of the cascades building process. The process begins by chronological ordering of all received spatially and temporally tagged trends (activations list), where one activation represents a record of (*trend*, *location*, and *time*). The algorithm first determines if an activation should be part of an earlier cascade or it should be considered as a seed for a new cascade. Ferrara *et al.* [28] state that the life time of almost all trends does not exceed 24 h. Thus, we consider a trend to be a seed for a new cascade if it was not trending for more than 24 h. The algorithm then determines whether or not to consider this activation as a new step. If the location did not appear before in the cascade, then this is a new step.

Otherwise, this is considered an update to the location activity times.

4) Stage 4: Extract Parameters: This stage is done for each location. In a given cascade, every location that appears in that cascade will have a distinctive set of parameters. The parameters are calculated mainly based on the diffusion model used, as will be explained in Section IV-E. For example, an average distance parameter will be calculated between a given location and all its parents or ancestors depending on the diffusion model. There are four main classes of the parameters.

1) *Diffusion Parameters (Hazard Rate):* It is the value representing the activation rate between any two locations calculated over all cascades:
   a) maximum hazard (max_hazard);
   b) sum of hazards (sum_hazard).
2) *Geographical Parameters:* It is used to examine the geospatial properties of the trending topics spread.
   a) *Geographical Distance Between Locations ($shrt\_dist$):* It indicates the shortest distance between locations and whether these distances affect the appearance of trends in these locations. For this, we have used the Haversine distance, which is commonly used to measure the distance between the locations based on the spherical shape of the Earth (as compared with Euclidian distance) [47]. An average distance between the locations (avg_dist) is also calculated.
   b) *Coverage (cvr):* A spread over geographical area of a trend S at time t. The area, which the trend covers, is determined by getting the area of bounding box in which the trend appeared. For the bounding box area, we determined the bounding locations (north east, north west, south east, and south west) in which each trend appears. We then calculated the area using the Haversine distance between the boundaries.
3) *Historical Parameters:* These parameters describe the path characteristic of each trend through all locations. Their values are based on previous cascades.
   a) *Trending Topics Similarity Between Locations (sim_{tt}) [27]:* The similarity parameter is used to measure the trending topics similarity between the locations. We used the Jaccard coefficient between the sets of trends observed at each location, as shown in (7), where M_{location_i} is the set of trends appeared in location_i. A similarity score of 1 means that all trends are common between the two locations. A score of 0 means that no trends are in common between the two locations. An average similarity is calculated over all trends

$$\text{sim}_{tt}(\text{location}_i, \text{location}_j) = \frac{|M_{\text{location}_i} \cap M_{\text{location}_j}|}{|M_{\text{location}_i} \cup M_{\text{location}_j}|}. \quad (7)$$

   b) *Average Gap (avg_gap):* For each trend appearing between two locations, the gap is calculated as the time difference between the end time in location i and its appearance in location j. It is calculated over all trends.
   c) *Overlap Time:* For two locations i and j, the overlap time is calculated as the difference between trend's end time in location i and its appearance in location j, given that ($t_{i.end} > t_{j.start}$). An average overlap time is generated over all cascades.
   d) *Average Trend Age (avg_age):* The average time of trend's presence in the social network.

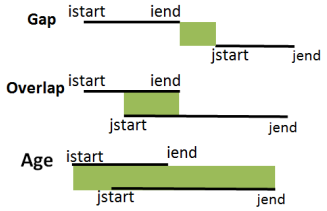


Fig. 11. Time tracking of trends' appearances in locations i and j.

Fig. 11 shows the calculating time differences between trends' appearances in locations.

4) *Trend Parameters:* Information about the relationship between locations based on the current cascade.
   a) *Trend's Rank (sum_rank):* As Twitter provides a trends box that contains the top ten trending topics, ranked according to their popularity. The trend's rank differs when the trend list is updated every 5 min.
      i) *Maximum Rank (sum_rank):* The highest rank reached by each trend in each cascade. The sum of trend's ranks over all cascades is also computed.
      ii) *Weighted Sum of Trend's Rank ($weighted\_sum\_rank$):* It indicates whether or not the trend's rank has an effect on the transmission rate. It is calculated as a sum of trend's ranks multiplied by the hazard rate between two locations.
   b) *Number of Parents/Ancestors ($num\_parents$ / $num\_ancestors$):* The number of parents and ancestors' locations for each location/cascade.

5) Stage 5: Model Learning/Using: As locations are different, a distinct predictive model is needed for each location. The model should learn the parameters extracted from the previous stage and should be used to predict if a new cascade will appear in that location. For this, we utilized two diffusion models. We first present our information diffusion model, the SC model. We then use the widely used general threshold (GT) model as our baseline. The differences between the models will be described in detail.

E. Snowball Cascade Model

The central part of TrendFusion is a new cascade model, SC Model. Conceptually, as any other information diffusion cascade model, the SC model tries to predict whether or not

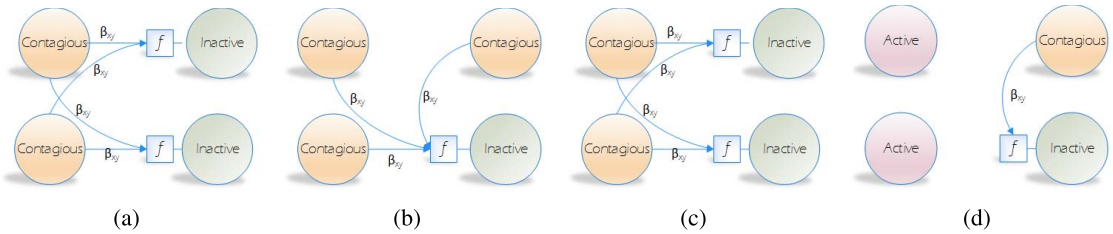


Fig. 12. Steps of the SC and GT models. (a) Step 1 in the SC model. (b) Step 2 in the SC model. (c) Step 1 in the GT model. (d) Step 2 in the GT model.

a certain piece of information will get adopted by different nodes in a social network. Generally, there are three types of nodes: active, contagious, and inactive. Given a piece of information, *inactive* nodes are those nodes that did not adopt that information yet, *active* nodes are the nodes that adopted it already, and the *contagious* nodes are the nodes that are trying to influence other nodes of adopting it. Initially, other than the seed nodes, all the other nodes are considered inactive. The seed nodes are those nodes that initially introduce that information to the network. At the beginning of the cascade, seed nodes are activated. Once a set of nodes is activated, they become contagious and will always be contagious, i.e., it will keep trying to influence other nodes. The rationale behind the continuous influence is simple: as long as a topic is trending in a location, this interest can affect other locations. Thus, in the SC model, the number of active nodes in the system that is trying to spread the influence will grow over time. Active nodes try to influence other nodes, which, if activated, become contagious and try to influence other nodes, and so on. This snowball effect is the reason behind the model name.

The SC model is different from the widely used GT cascade model [48], [49]. In the GT model, contagious nodes try to collectively influence other inactive nodes. But once they are done, they are no longer contagious, i.e., they will no longer try to influence other nodes.

Yet another difference between the two models is that in the SC model, the edge weights are vectors rather than scalars. The vector values change from one activation to the other. This is different from the GT model, where the edge weights are required only to be fixed scalars. The vectors on the edges represent the set of parameters that might affect the influence between a contagious location and inactive location at a given step of a cascade.

Fig. 12 shows an example of two steps for four nodes in SC and GT models, respectively. In the SC model [Fig. 12(a)], two nodes are contagious, both trying to influence the two inactive nodes. The $\boldsymbol{\beta}$ values on the edges represent vectors containing the influence rates along with other parameters that are described in TrendFusion Stage 4 (Section IV-D4). The function box in the SC model acts as a binary classifier that takes the $\boldsymbol{\beta}$ vector values as an input. In the second step [Fig. 12(b)], the contagious nodes remain contagious, and keep on trying to influence other inactive nodes till the end of the cascade.

However, in the GT model [Fig. 12(c)], two nodes start as contagious nodes, both trying to influence the two inactive nodes. The second step [Fig. 12(d)] shows that one of the inactive nodes got infected and became contagious itself, and the other one was not affected. The two contagious nodes in step 1 became active in step 2. This means that they are already infected but will not try to influence other node anymore. Another difference between the SC and GT models lies in calculating the influence rate. In the GT model, the β values on the edges are scalar values representing the influence rates between the corresponding nodes. The function box in the GT model is just a summation operation followed by a condition to check that the sum is below a certain threshold. The threshold is a specific property of each node, i.e., the threshold is different from a node to another node. If the sum exceeds that threshold, the node becomes contagious, as in the second step shown in Fig. 12(d).

1) SC Model Definition: Consider a directed graph $G = (V, E)$, where V is the set of vertices representing locations, and E is the set of weighted edges, with weights $\boldsymbol{\beta}^t_{\text{uv}}$ of edge $e_{\text{uv}} \in E$ representing the influence rate from location u to location v at time step t. Let N_v be the set of vertices with edges going into v, and S_t be a subset of N_v with the vertices that are active on or before time t. For every vertex v, there is an activation function $f()$, such that at time t, if $f(\boldsymbol{\beta}^t_{u_0 v}, \boldsymbol{\beta}^t_{u_1 v}, \ldots, \boldsymbol{\beta}^t_{u_n v}) > \theta_v \forall u_i \in S_t$, vertex v becomes active at time $t+1$. The value of θ_v can be learned for each location by a binary classifier.

2) GT Model Definition: Consider a directed graph $G = (V, E)$, where V is the set of vertices representing locations, and E is the set of weighted edges, with weights w_{uv} representing the influence rate of the edge $e_{\text{uv}} \in E$ from location u to location v. Let N_v be the set of vertices with edges going into v, and S_t be the subset of N_v active at time t. For every vertex v, there is an activation function $f()$, such that at time t, if $f(S_t) > \theta_v$, vertex v becomes active at time $t+1$. In the original model, the value of θ_v is randomly chosen from a uniform distribution in the interval [0, 1]. We rely on statistical classifiers to estimate the likelihood value of θ_v.

The GT model can be considered as a special case of the SC model, where the $\boldsymbol{\beta}$ vectors are reduced to a fixed scalar (influence rate), and the $\boldsymbol{\beta} = 0$, for all nodes that had already been active before time t.

F. Evaluation

We now describe the methodology used to generate our data set, and then, we describe in detail the results of every stage in our model.

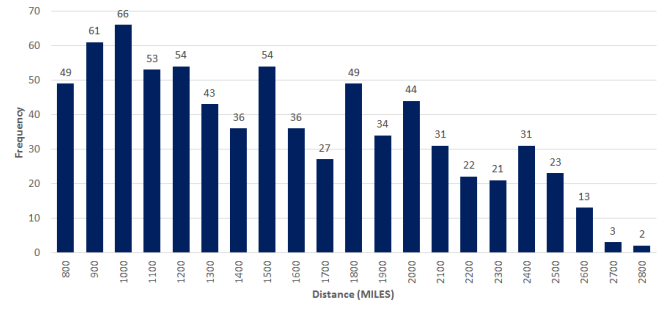


Fig. 13. Histogram of the distances between the 48 cities.

1) Trending Topics Data Set: We used Twitter APIs [1] to collect all trending topics appearing on Twitter for a period of 30 days, starting from August 2014 until September 2014, in 48 U.S. locations (cities). Twitter provides a trends box that contains the top ten trending hashtags or phrases at any given moment, ranked according to their popularity. These trending topics, along with their rank, are updated every 5 min. Each user can monitor the trends at the worldwide, country, or city level.

We deployed a crawler to get the trends every 5 min for the 48 cities. We also collected all trends reported by Twitter for the United States and the whole world. To mask the effect of global trends in our experiments, we filtered out the trends for the cities that appeared in the U.S. trends or the global trends. We ended up collecting more than 400k different trends. The data are stored as tuples of the form: (woeid, $trend_0$, $trend_1$, ... , $trend_9$, and date/time), where $woeid$ is Yahoo Where On Earth ID [50] and $trend_0, \ldots, trend_9$ are the top ten trends. Fig. 13 shows the histogram of the distances between the 48 cities, where the x-axis represent the upper limit of each distance bin in miles.

2) Applying TrendFusion Framework Steps: As mentioned earlier, the steps presented in Algorithm 3 are used to convert the data collected from the previous step into cascades. We then use the MATLAB implementation of the NetRate algorithm [51] to build the influence graph for all locations. This implementation assumes linear DAG for cascades, i.e., it assumes that each step in the cascade consists only of one location. However, the SC model allows multiple locations per cascade step. Therefore, the algorithm was modified slightly to account for this difference. The modified NetRate is used to generate three graphs, one for each assumed distribution for the hazard rate. The graphs from NetRate are then used with the cascades to generate the training and testing examples for each location.

For each location, we generate the training file containing the examples for the first 22 days of the data and a testing file containing the remaining data. The extracted parameter was based on the SC model. We also used the GT model as a baseline, so training and testing data were also generated for it. Each of the parameter vectors is augmented by one class and one dependent variable.

Given a cascade, when generating the training examples for a location, an example is generated for each step in the cascade before that location appears in it. For example, if a location appeared in step n, we generate $n-1$ examples for each step before that location appeared. If the location does not appear in the cascade, then the number of examples generated will be equal to the number of steps in the cascade. The class values are set to be the *appearing* or *not appearing*, depending on whether or not the location appeared in the cascade. If the class value is appearing, then the dependent variable value is set based on the lag value between the time at the cascade step and the time the trend appeared in the location.

We used the parameter vectors for each cascade for individual trends to train three classifiers:

1) logistic regression (LR), a probabilistic statistical classification model [38];
2) stochastic gradient descent (SGD) classifier [52];
3) random forest (RF), ensemble learning method classifier [53].

We used Weka [54] and R [55] statistical packages to train the three classifiers and afterward use them to predict whether or not the trend will appear in the designated location.

3) Experiments: The evaluation includes six experiments.

1) Predict trends based on individual steps.
2) Predict trends considering each cascade as a whole.
3) Determine the effect of similarity of topic interest between the locations on the quality of prediction.
4) Determine the effect of each parameter on the classification process.
5) Determine the average time a topic can be predicted to be trending before it actually does.
6) Predict when a trend will appear.

G. Results and Discussion

We evaluated the performance of TrendFusion by running our training and testing examples through the three classifiers. Each example represents a step in a cascade. We used the widely adopted GT model as a baseline to compare its performance with TrendFusion. We recorded two quality measures in our experiments, $recall$ and $precision$. Here, $recall$ is the ratio of the number trends we were able to predict to the total number of actual trends. Similarly, $precision$ is the quality of our prediction, i.e., the ratio of the number of topics that actually become trending in our predictions to the total number of topics we predicted will be trending.

In the first experiment, we considered the output from each individual example. This means that at each step, we take a decision regardless of other steps in the cascades. Fig. 14(a) shows the recall and precision values obtained by TrendFusion and the GT models using the three classifiers. It is clear that TrendFusion was giving the same performance across the different classifiers with a recall value of around 0.71 and a precision value of around 0.84 (84% of the predicted trend will be actually trending).

On the other hand, the GT model recall values were in the range between 0.47, 0.48, and 0.5 for the LR, SGD, and RF classifiers, respectively, which means that it misses around half of the trends. The precision values range from 0.71 to 0.78, which means that the slight increase in the recall was accompanied with more false positive predictions. This

Algorithm 4 Classify Cascade

Procedure ClassifyCascade
Input Location l
Cascade cas
begin
// Determine the class for the whole cascade
$count_{true\_positive} \leftarrow 0$
$count_{false\_positive} \leftarrow 0$
$count_{true\_negative} \leftarrow 0$
$count_{false\_negative} \leftarrow 0$
if l appears in cas then **then**
$class \leftarrow appearing$
else
$class \leftarrow notappearing$
end if
// Collective classification for all steps
for all step s in cas **do**
$prediction \leftarrow$ classify_at(s)
if $prediction$ is $appearing$ **then**
if $class$ is $appearing$ **then**
$count_{true\_positive} \leftarrow count_{true\_positive} + 1$
else
$count_{false\_positive} \leftarrow count_{false\_positive} + 1$
end if
return
end if
end for
// $prediction$ should be $not\ appearing$
if $class$ is $not\ appearing$ **then**
$count_{true\_negative} \leftarrow count_{true\_negative} + 1$
else
$count_{false\_negative} \leftarrow count_{false\_negative} + 1$
end if
end

shows that the GT model is not suitable for modeling the diffusion of trending topics between the locations.

In the second experiment, we evaluated each cascade as a whole, getting one decision for the whole cascade. For a given location, we set the class value to be *appearing* for the cascades in which the location appeared, and *not appearing* for the cascades in which the location did not appear. The classification is performed on each step, and then, the predicted values are reduced to one value for the whole cascade. If the predicted value at any of the steps is *appearing*, we consider the combined prediction as *appearing*, as if doing a *logical* OR. The reason behind this way of classification is that the *class* is assigned at each step based on the fact whether or not the location appeared later in the cascade. So at an early step in reality, that might not have any influence on a given location that appeared later in the cascade, the class is still assigned as *appearing*. This is due to the fact that we do not have ground-truth data.

A false positive prediction is made in a cascade where a given location did not appear, if at any step an *appearing* class is predicted, as detailed in Algorithm 4.

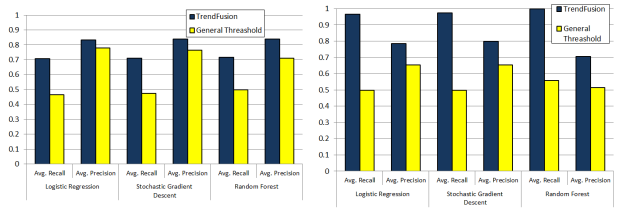


Fig. 14. Average precision and recall for TrendFusion and GT models using cascade steps and all cascades, respectively. (a) Precision and recall—cascade steps. (b) Precision and recall—all cascades.

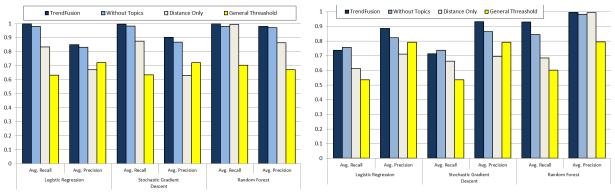
Fig. 14(b) shows the average recall and average precision values for the TrendFusion and GT models for the second experiment with the same three classifiers as before. The average recall values for TrendFusion improved greatly. The wrong *not appearing* predictions made in the first experiment at the beginning of the cascades are neutralized in this experiment by a later correct *appearing* prediction. Values for $recall$ are 0.96, 0.98, and 0.99 for LR, SGD, and RF classifiers, respectively.

On the other hand, $precision$ dropped slightly to around 0.8 for the LR and SGD classifiers and to 0.71 for the RF classifier. This also means that one wrong *appearing* prediction at any step of cascade in which a given location did not appear will cause the overall prediction to be considered wrong. Although the average recall is slightly improved for the GT model, it still in the range of 0.5–0.56 for the three classifiers. The average precision also dropped as expected to the values of 0.65, 0.65, and 0.51 for LR, SGD, and RF classifiers, respectively. This still point out that even though that the GT model was good in modeling information diffusion in a social graph at the users level, it is not suitable to model the trending topics diffusion between the locations.

These two experiments were conducted using the transmission rates generated by the modified NetRate algorithm assuming exponential distribution. We also examined the two distribution models (power law and Rayleigh) to decide the shape of the conditional transmission likelihood and to analyze the effect of changing them on the classification process. The experiments were repeated using the other two distributions. The results were very consistent with the results obtained for exponential distribution. The variation in the results obtained in all experiments did not exceed 1%.

The third experiment was conducted to measure the effect of similarity of users' topics of interest on the quality of prediction. The trending topic names may or may not be indicative of the kind of information people are tweeting about, so we wanted to measure the effect of applying topic modeling on the trends, and how this can affect the flow of trends between different locations. The steps were as follows.

1) *Collecting Tweets for Each Location:* For each location, all tweets posted during the examined period were collected.
2) *Applying Topic Modeling on Tweets:* Using the algorithm discussed in III-B, we get the distributions of topics in each tweet.



(a) (b)

Fig. 15. Average precision and recall for TrendFusion without and with adding topics, using only distance features and the GT model. (a) Precision and recall—cascade steps. (b) Precision and recall—all cascades.

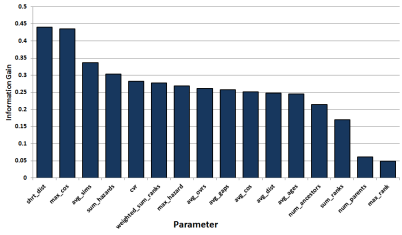


Fig. 16. Rank of each parameter in the classification process.

3) *Getting Related Tweets for Trends:* For each trend appearing, we get all the tweets that include the trend hashtag or word.
4) *Getting Related Topic for Trends:* Using the previous two steps, we measure topic interest in each location by getting topic distribution all over the trends.

After measuring the topic interest in each location, we included this as a parameter in the prediction model (Section IV-D4). Fig. 15(a) and (b) shows the recall and precision values obtained by TrendFusion after including the topics features, TrendFusion after including distance features only, and the GT models using the three classifiers considering the cascade steps and all the cascades, respectively. The results show that including the topics as parameter helped in improving the average recall and precision in all of the cases when considering cascades as a whole. Including the topics also helped the precision in all cases when considering individual steps, almost without affecting the recall. The results for using the distance as the only parameter to the models show that the distance is not the only factor that impacts the propagation of trends. This stresses the importance of the other suggested parameters, which will be detailed in the next experiment.

The fourth experiment was conducted to measure the effect of each parameter on the classification process. This is achieved by ranking all the parameters according to their average information gain. Fig. 16 shows the rank of each parameter used in the classification process. We observed that it has the following.

1) *Geography Matters:* It is clear from Fig. 16 that locations that are geographically near each other are most likely to influence each other in the social context.
2) The similarity in interests and diffusion parameters is of high importance: the locations similar in the trending topics in the past are more likely to have the same trends later on. The locations with high combined diffusion rate to a given location will affect it with high probability.
3) Trend parameters are the least important: although locations may be influencing each other, the rank of the trending topic in one location is not affecting its rank in the other location. This might be due to the fact that each location has different interests in topics. This also means that it does not really matter in how many locations did a topic appear in, to be influential to other locations, it might just give an indication of how globally important is that topic.
4) The remaining parameters were equally important.

The fifth experiment explored the average time a topic can be predicted to be trending before it actually becomes trending. Fig. 17(a) shows the average time before a trend can appear. The x-axis represents the lag time between the beginning of the cascade and the time a trend will occur. The y-axis represents the time before a trend is predicted as trending. This shows that we are able to predict the topics on average 3 h before they actually trend.

We noticed a drop at 17 h of time to trend [Fig. 17(a)]. We investigated the possible reasons for this drop. We found that the number of trends that appeared in new locations after 17 h is relatively much less than different hours. To find out the reason for that, we used the facts presented by Upbin [56] that shows the average Twitter activity by hour [56]. Upbin [56] showed that the user activity is highest between 9 A.M. and 2 P.M., and lowest between 1 A.M. and 6 A.M.. According to this, we assumed that most trends are formed during the high activity intervals. The first four horizontal lines in Fig. 17(b) represent different time zones in the United States, namely, Eastern (EST), Central (CST), Mountain (MST), and Pacific (PST) standard time. The red peaks represent high activity time at each time zone. The blue troughs represent low activity intervals. The lower line represents the combined activities, and it shows that the highest activity in the United States happens around 1 P.M. Eastern, and the lowest activity happens around 6 A.M. Eastern. The difference between these numbers is 17 h; thus, the trends will not be trended within this gap, and hence, the drop in a number of trends happens after 17 h.

In the sixth experiment, we tried to predict when a trend will appear. The training and testing examples in this case are labeled by the time lag between each step and the step at which the trend appeared in a given city. We trained a linear regression model and used it to try to predict when will the trend happen. Fig. 17(c) shows a histogram where the bins (x-axis) represent the error in prediction in hours. The results show that most of the predictions were around zero error. The bimodal peaks are probably due to the activity windows described in Fig. 17(b), where the high activity interval makes the trends travel faster, and the low activity window makes the trends be delayed in traveling.

V. TrendFusion System

To further show the applicability of the concepts presented in this paper, we created "TrendFusion" Web application to provide a user with personalized timeline that suits his/her interests. The site analyzes the user's Twitter feed, according

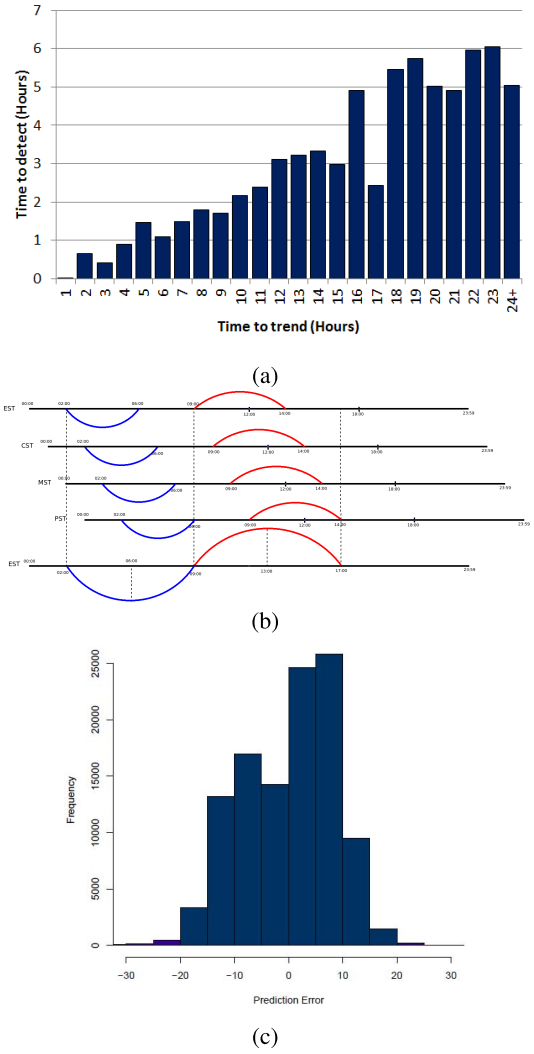


Fig. 17. Predicted trends analysis. (a) Lag analysis for predicted trends. (b) Activity times over 24 h. Red line: highly active window. Blue line: low active window.

to the techniques and theories presented in this paper, and predicts the tweets interesting to the user. Fig. 18 shows the general framework components and how they map onto the TrendFusion system.

After registration, users are redirected to Twitter to give read-only permission to TrendFusion to access their Twitter account. This is essential for retrieving the timeline information of the user. When given the permission from the user, TrendFusion retrieves up to 800 of the past tweets in the user's timeline. This is the maximum limit allowed by Twitter that can be retrieved from a user timeline. TrendFusion extracts the tweets with user's actions using the window system described in Section III-F1. As described earlier in Section III, the system relies on the user's actions, such as retweets, replies, favorites, and posts, to assume the user's interest in a past tweet. The extracted tweets are thus used to train a classifier. TrendFusion will build a unique personalized model for each registered user. The implementation is still relying on Weka package [54] to train a J48 classifier. The choice of J48 is based on its relatively high accuracy and at the

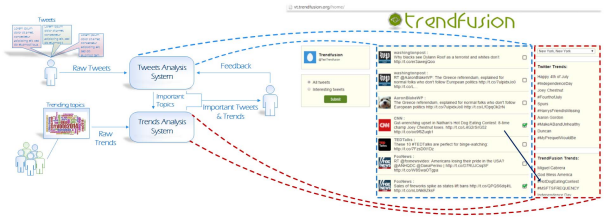


Fig. 18. TrendFusion System.

same time small overhead, according to the results presented in Section III-G.

After signing into TrendFusion, the user is given the option to view all the timeline, or to view the interesting tweets only. To keep the user timeline as current as possible, TrendFusion will try to pull new tweets from Twitter every 5 min. When a new tweet is retrieved, TrendFusion will extract all the relevant features as described in Section III-E. TrendFusion will then load the user classifier model to predict if the tweet is important to the user or not.

The user can also choose to set a specific tweet as important to him or not by clicking on the check box that exists at the right of each tweet. A checked box means that the tweet is important to the user, and an empty box means that the tweet is not important. TrendFusion will update the user models once every day. The user can thus guide the system by checking the important tweets that the system did not recognize as important, and unchecking the unimportant tweets that the system recognized as important.

The system allows the user to view the Twitter trending topics for 48 U.S. cities and to view the predicted trending topics that will be appearing in the user's chosen location. The suggested trends are also personalized according to the user's interests discovered from the tweets marked as important by the tweets analysis system. So for each user in the system, his/her interesting topics are passed from the tweets analysis system to the trends analysis system. As we also build a topic model for the trends, we use the user interesting topics to filter and rank the suggested trends by the trends analysis system. For example, in Fig. 18, an important tweet to the user is reflected in a suggest trending topic that is predicted to appear at the user's city, as marked by the blue arrow.

Before launching TrendFusion Web application, trends were retrieved for each of the 48 cities for about a month. These trends are then used to create the hazard rate graph using the NetRate algorithm [44] as described in Section IV-C. The collected history trends were also use to build a localized classifier for every city in the 48 cities. The model was build using an SGD classifier with the features described in Section IV-D4. The features are extracted based on the SC model described in Section IV-E.

TrendFusion will retrieve Twitter trends for the 48 cities every 5 min. This is because Twitter caches trends for 5 min. If a shorter interval is used, the duplicated sets of trends will be retrieved. Once a set of trends is retrieved for a given city, they are added to the cascades following the steps in Algorithm 3. For the cities that did not appear in a cascade, run the city localized classifier on the features extracted from that cascade.

The user can choose to view the worldwide trending topics, with no predicted trends, or can select one city to see Twitter trends and predicted trends according to TrendFusion.

VI. Conclusion and Future Work

In this paper, we introduced the concept of dynamic LoI for microblogs users. To determine the LoI of the user in a new corpus, we proposed a novel model that is based on topics in that corpus and the history of the user activity in each topic. The goal of the model is to identify the important tweets to a user in the user's timeline.

To illustrate the effectiveness of our model, we used a Twitter APIs to build a data set with more than 5 million tweets, and more than 20 000 users. We demonstrated the importance of using the *dynamic LoI* feature, by showing the improvement of the average precision and the average recall for the three classifiers used (J48, naive Bayes, and SVM). Using our approach, we were able to improve the precision and recall of identifying important tweets by up to 36% and 80%, respectively. The model analysis showed that the model has higher gain for users with high activity level.

We analyzed the behavior of the LDA topic model to identify the key factors that can affect its performance. We demonstrated that by choosing a proper number of topics and applying pooling techniques to the tweets, an additional 10% improvement can be achieved.

We also developed TrendFusion, a model for predicting the localized trends diffusion in social networks. Our goal was to develop a model that will allow us to predict whether a trend will be appearing in a certain city in the future, and if it will appear, when it would appear. We also demonstrated that the diffusion models that are designed for modeling information spread between users, and are not suitable for modeling trends diffusion across cities, where no real friendship relations exist. The main aspect of TrendFusion is a new information cascade model, SC model. The model assumes that an activated node in a graph will always be contagious.

We illustrated the effectiveness of our model using the trending topics obtained from Twitter for 48 of biggest U.S. cities. We demonstrated the effectiveness of our model comparing our model with the GT model, a widely accepted diffusion model. TrendFusion outperformed the GT model by achieving the recall and precision of prediction of trends by 98% and 80%, respectively.

TrendFusion is also capable of predicting the time at which the trend will appear. TrendFusion successfully predicted trends before they actually become trending by up to 24 h. The root mean squared error in TrendFusion time prediction is less than 6 h. However, more analysis is needed to understand how to identify locations that can be influential for the spread of a given topic.

References

[1] *Twitter Home Page*, accessed on Jun. 23, 2017. [Online]. Available: http://twitter.com

[2] B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas, "Short text classification in Twitter to improve information filtering," in *Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, New York, NY, USA, 2010, pp. 841–842.

[3] S. Khater, H. G. Elmongui, and D. Gračanin, "Tweets you like: Personalized tweets recommendation based on dynamic users interests," in *Proc. Int. Conf. Social Comput. (SocialCom)*, Dec. 2014, pp. 14–15.

[4] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[5] J. Hannon, M. Bennett, and B. Smyth, "Recommending Twitter users to follow using content and collaborative filtering approaches," in *Proc. 4th ACM Conf. Recommender Syst.*, New York, NY, USA, Sep. 2010, pp. 199–206.

[6] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?" in *Proc. 19th Int. Conf. World Wide Web*, New York, NY, USA, 2010, pp. 591–600.

[7] K. Chen, T. Chen, G. Zheng, O. Jin, E. Yao, and Y. Yu, "Collaborative personalized tweet recommendation," in *Proc. 35th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, New York, NY, USA, 2012, pp. 661–670.

[8] J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. H. Chi, "Short and tweet: Experiments on recommending content from information streams," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, New York, NY, USA, 2010, pp. 1185–1194.

[9] F. Abel, Q. Gao, G.-J. Houben, and K. Tao, "Analyzing user modeling on Twitter for personalized news recommendations," in *Proc. 19th Int. Conf. User Modeling Adaption, Pers.*, Berlin, Germany, 2011, pp. 1–12.

[10] I. Uysal and W. B. Croft, "User oriented tweet ranking: A filtering approach to microblogs," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage.*, New York, NY, USA, 2011, pp. 2261–2264.

[11] H. Baars and H.-G. Kemper, "Management support with structured and unstructured data—An integrated business intelligence framework," *Inf. Syst. Manage.*, vol. 25, no. 2, pp. 132–148, Mar. 2008.

[12] J. Martinez-Romo and L. Araujo, "Detecting malicious tweets in trending topics using a statistical analysis of language," *Expert Syst. Appl.*, vol. 40, no. 8, pp. 2992–3000, 2013.

[13] J. Weng, E.-P. Lim, J. Jiang, and Q. He, "TwitterRank: Finding topic-sensitive influential twitterers," in *Proc. 3rd ACM Int. Conf. Web Search Data Mining*, New York, NY, USA, 2010, pp. 261–270.

[14] M. Steyvers, P. Smyth, M. Rosen-Zvi, and T. Griffiths, "Probabilistic author-topic models for information discovery," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, 2004, pp. 306–315.

[15] W. X. Zhao *et al.*, "Comparing Twitter and traditional media using topic models," in *Proc. Eur. Conf. Inf. Retr. (ECIR)*, Berlin, Germany, 2011, pp. 338–349.

[16] J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst, "Patterns of cascading behavior in large blog graphs," in *Proc. SIAM Int. Conf. Data Mining (SDM)*, 2007, pp. 551–556.

[17] P. Bao, H.-W. Shen, J. Huang, and X.-Q. Cheng, "Popularity prediction in microblogging network: A case study on sina weibo," in *Proc. 22nd Int. Conf. World Wide Web*, New York, NY, USA, 2013, pp. 177–178.

[18] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "Learning influence probabilities in social networks," in *Proc. 3rd ACM Int. Conf. Web Search Data Mining*. New York, NY, USA, 2010, pp. 241–250.

[19] S. Aral and D. Walker, "Identifying influential and susceptible members of social networks," *Science*, vol. 337, no. 6092, pp. 337–341, 2012.

[20] W. Galuba, K. Aberer, D. Chakraborty, Z. Despotovic, and W. Kellerer, "Outtweeting the Twitterers—Predicting information cascades in microblogs," in *Proc. 3rd Conf. Online Social Netw.*, Berkeley, CA, USA, 2010, p. 3.

[21] N. Agarwal, H. Liu, L. Tang, and P. S. Yu, "Identifying the influential bloggers in a community," in *Proc. Int. Conf. Web Search Data Mining*, New York, NY, USA, 2008, pp. 207–218.

[22] E. Casetti, "Innovation diffusion as a spatial process, by Torsten Hägerstrand," *Geograph. Anal.*, vol. 1, no. 3, pp. 318–320, 1969.

[23] J. Leskovec, L. Backstrom, and J. Kleinberg, "Meme-tracking and the dynamics of the news cycle," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, 2009, pp. 497–506.

[24] S. Bharathi, D. Kempe, and M. Salek, "Competitive influence maximization in social networks," in *Proc. 3rd Int. Conf. Internet Netw. Econ.*, Berlin, Germany, 2007, pp. 306–311.

[25] S. Asur, B. A. Huberman, G. Szabo, and C. Wang, "Trends in social media: Persistence and decay," in *Proc. 5th Int. AAAI Conf. Web Social Media*, 2011, pp. 434–437.

[26] C. Wang and B. A. Huberman, "Long trend dynamics in social media," *EPJ Data Sci.*, vol. 1, no. 2, p. 2, 2012.

[27] K. Y. Kamath, J. Caverlee, Z. Cheng, and D. Z. Sui, "Spatial influence vs. community influence: Modeling the global spread of social media," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, New York, NY, USA, 2012, pp. 962–971.
[28] E. Ferrara, O. Varol, F. Menczer, and A. Flammini, "Traveling trends: Social butterflies or frequent fliers?" in *Proc. 1st ACM Conf. Online Social Netw.*, New York, NY, USA, 2013, pp. 213–222.
[29] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.
[30] D. Laniado and P. Mika, "Making sense of Twitter," in *Proc. 9th Int. Semantic Web Conf. Semantic Web*, vol. 1. Berlin, Germany, 2010, pp. 470–485.
[31] R. Mehrotra, S. Sanner, W. Buntine, and L. Xie, "Improving LDA topic models for microblogs via tweet pooling and automatic labeling," in *Proc. 36th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, New York, NY, USA, 2013, pp. 889–892.
[32] Z. Ren, S. Liang, E. Meij, and M. de Rijke, "Personalized time-aware tweets summarization," in *Proc. 36th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, New York, NY, USA, 2013, pp. 513–522.
[33] *Virtual Time Square*, accessed on Mar. 10, 2014. [Online]. Available: http://vts.cs.vt.edu
[34] *Twitter Documentation*, accessed on Jun. 23, 2017. [Online]. Available: https://dev.twitter.com/docs
[35] H. M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno, "Evaluation methods for topic models," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, New York, NY, USA, 2009, pp. 1105–1112.
[36] (2002). A. K. McCallum. *MALLET: A Machine Learning for Language Toolkit*, accessed on Jun. 23, 2017. [Online]. Available: http://mallet.cs.umass.edu/
[37] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann, 1992.
[38] T. M. Mitchell, *Machine Learning*. Boston, MA, USA: McGraw-Hill, 1997.
[39] M. R. Levy and S. Windhak, "The concept of audience activity," in *Media Gratifications Research: Current Perspectives*, K. E. Rosengren, L. A. Wenner, and P. Palmgreen, Eds. Beverly Hills, CA, USA: Sage Publications, 1985, pp. 109–122.
[40] M. Dash and H. Liu, "Feature selection for classification," *Intell. Data Anal.*, vol. 1, no. 1, pp. 131–156, 1997.
[41] W. Chen, L. V. S. Lakshmanan, and C. Castillo, *Information and Influence Propagation in Social Networks* (Synthesis Lectures on Data Management). London, U.K.: Morgan & Claypool Publishers, 2013.
[42] M. Gomez-Rodriguez, J. Leskovec, and A. Krause, "Inferring networks of diffusion and influence," *ACM Trans. Knowl. Discovery Data*, vol. 5, no. 4, pp. 21:1–21:37, Feb. 2012.
[43] P. Netrapalli and S. Sanghavi, "Learning the graph of epidemic cascades," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 1, pp. 211–222, Jun. 2012.
[44] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf, "Uncovering the temporal dynamics of diffusion networks," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, L. Getoor and T. Scheffer, Eds. New York, NY, USA, 2011, pp. 561–568.
[45] B. Hecht and E. Moxley, "Terabytes of Tobler: Evaluating the first law in a massive, domain-neutral representation of world knowledge," in *Spatial Information Theory* (Lecture Notes in Computer Science), vol. 5756, K. S. Hornsby, C. Claramunt, M. Denis, and G. Ligozat, Eds. Berlin, Germany: Springer, 2009, pp. 88–105.
[46] M. Gomez-Rodriguez, J. Leskovec, and B. Schölkopf, "Modeling information propagation with survival theory," in *Proc. 30th Int. Conf. Int. Conf. Mach. Learn.*, vol. 28. 2013, pp. III-666–III-674.
[47] R. W. Sinnott, "Virtues of the haversine," *Sky Telescope*, vol. 68, no. 2, p. 158, 1984.
[48] N. Pathak, A. Banerjee, and J. Srivastava, "A generalized linear threshold model for multiple cascades," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2010, pp. 965–970.
[49] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, 2003, pp. 137–146.
[50] *Yahoo!-GeoPlanet*, accessed on Jun. 23, 2017. http://developer.yahoo.com/geo/geoplanet/
[51] *NETRATE*, accessed on Jun. 23, 2017. [Online]. Available: http://people.tuebingen.mpg.de/manuelgr/netrate/
[52] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, 2011, pp. 69–77.
[53] G. Biau, "Analysis of a random forests model," *J. Mach. Learn. Res.*, vol. 13, pp. 1063–1095, Apr. 2012.
[54] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.
[55] *The R Project for Statistical Computing*, accessed on Jun. 23, 2017. [Online]. Available: http://www.r-project.org
[56] B. Upbin. *What are the Best Times to Share on Facebook and Twitter?* accessed on Jun. 23, 2017. [Online]. Available: http://www.forbes.com/sites/bruceupbin/2012/05/09/when-to-make-stuff-go%-viral-online/

Shaymaa Khater received the B.Sc. and M.S. degrees in computer science from Ain Shams University, Cairo, Egypt, and the Ph.D. degree in computer science from Virginia Tech, Blacksburg, VA, USA.

She is currently a Senior Data Scientist with Solebrity, Inc., Ashburn, VA, USA. Her current research interests include deep learning, machine learning, social networks analysis, and graph data mining.

Denis Gračanin (M'87–SM'03) received the B.S. and M.S. degrees in electrical engineering from the University of Zagreb, Zagreb, Croatia, in 1985 and 1988, respectively, and the M.S. and Ph.D. degrees in computer science from the University of Louisiana at Lafayette, Lafayette, LA, USA, in 1992 and 1994, respectively.

He is currently an Associate Professor with the Department of Computer Science, Virginia Tech, Blacksburg, VA, USA. His current research interests include human–computer interactions and distributed simulation.

Dr. Gračanin is also a Senior Member of ACM and a member of the Association for the Advancement of Artificial Intelligence, American Physical Society, American Society for Engineering Education, and Society for Industrial and Applied Mathematics.

Hicham G. Elmongui was born in Alexandria, Egypt, in 1976. He received the B.S. (Hons.) and M.S. degrees in computer engineering from Alexandria University, Alexandria, in 1998 and 2001, respectively, and the M.S. and Ph.D. degrees in computer science from Purdue University, West Lafayette, IN, USA, in 2003 and 2009, respectively.

From 2001 to 2009, he was a Doctoral Fellow, a Teaching Assistant, and a Research Assistant with Purdue University. He was a Research Intern with Microsoft Research, Redmond, WA, USA, every summer from 2006 to 2008. In 2010, he was a Software Development Engineer with Amazon Web Services, Ashburn, VA, USA. Since 2010, he has been an Assistant Professor in computer and systems engineering with Alexandria University and an Adjunct Professor in computer science and electrical and computer engineering with Virginia Tech, Blacksburg, VA, USA. He has authored or co-authored over 30 papers and two patents.

Dr. Elmongui is a member of the Microsoft Research Alumni Network, CERIAS Alumni, and Upsilon Pi Epsilon. He has been recognized through many awards and honors, including the Outstanding Teaching Award and the Outstanding Service Award from Purdue University. He has served on the Technical Program Committee of several conferences.