False Rumors Detection on Sina Weibo by Propagation Structures

Ke Wu [1], Song Yang [2], Kenny Q. Zhu [3]

Department of Computer Science & Engineering
Shanghai Jiao Tong University, Shanghai, China
[1]keer236@gmail.com, [2]blue-snow@sjtu.edu.cn, [3]kzhu@cs.sjtu.edu.cn

***Abstract*—This paper studies the problem of automatic detection of false rumors on Sina Weibo, the popular Chinese microblogging social network. Traditional feature-based approaches extract features from the false rumor message, its author, as well as the statistics of its responses to form a flat feature vector. This ignores the propagation structure of the messages and has not achieved very good results. We propose a graph-kernel based hybrid SVM classifier which captures the high-order propagation patterns in addition to semantic features such as topics and sentiments. The new model achieves a classification accuracy of 91.3% on randomly selected Weibo dataset, significantly higher than state-of-the-art approaches. Moreover, our approach can be applied at the early stage of rumor propagation and is 88% confident in detecting an average false rumor just 24 hours after the initial broadcast.** [1]

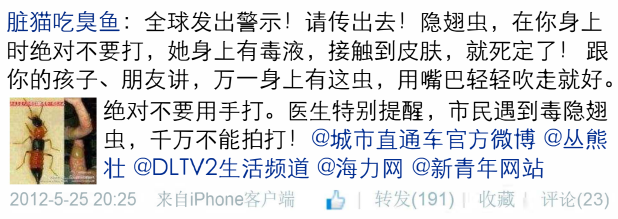
I. Introduction

Microblogging has taken over the Internet as one of the most popular forms of online social networking. Twitter [1], or the Chinese equivalent, Sina Weibo [2], allows users to instantly broadcast short messages of up to 140 characters with optional images and other meta data to all their followers. Such messages maybe reposted with or without comments by the followers and thus get transmitted throughout the social network. Many of the microblogs carry unconfirmed and uncertain information, and when they get spread around quickly, they become rumors.

There is no uniform definition of rumors according to social scientists. In our work, a *rumor* is defined as an unverified or unconfirmed statement or report circulating in a community [3]. By this definition, a stand-alone microblog which has not been spread around is not a rumor. Breaking news such as celebrity getting married that gets spread wildly online is not a rumor either, because it is factual information. Discussions about personality by horoscope signs are considered rumors because there is no way to confirm their truthfulness. Some rumors eventually are verified to be false; these are called *false rumors* in this paper. These rumors are created either intentionally or unintentionally, but carry false or even malicious information and spread widely in the community. For instance, some "urban legends" such as "Coke can dissolve a tooth overnight" are considered false rumors here because they are spread widely in the community and proven to be false eventually. Recent past has witnessed many incidents of online false rumors causing massive public panic and social unrest. On April 23, 2013, a false rumor on Twitter about an explosion in Whitehouse injurying President Obama sent the Dow down 140 points within a few minutes. In March, 2011, after the Japanese Fukushima nuclear disaster, a Sina Weibo post claimed that increased consumption of iodized salt can protect human from nuclear radiation and this message caused millions of people to raid the supermarkets around China to buy salt and even soy sauce. Because of these and many other similar damaging incidents, automatic detection of false rumors on social network has recently gathered substantial research interest and is also the goal of this paper.

[1]Kenny Q. Zhu, the corresponding author, is partially supported by NSFC Grants 61033002, 61100050 and 61373031, and Google Facult Research Award.

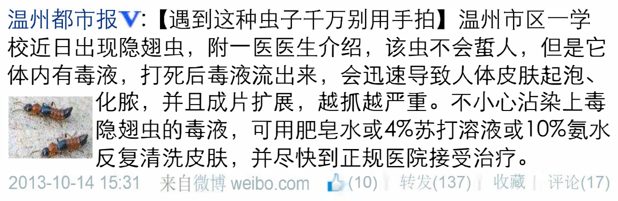
While some researchers have worked previously on Twitter (see Section V), this paper focuses on false rumor detection on Sina Weibo, for which only limited research has been done [4], [5], [6]. Automatic detection of false rumors is generally a hard problem, because without proper background knowledge or concrete, official evidence against, even human being cannot distinguish between the false rumor and other messages.



脏猫吃臭鱼：全球发出警示！请传出去！隐翅虫，在你身上时绝对不要打，她身上有毒液，接触到皮肤，就死定了！ 跟你的孩子、朋友讲，万一身上有这虫，用嘴巴轻轻吹走就好。绝对不要用手打。医生特别提醒，市民遇到毒隐翅虫，千万不能拍打！@城市直通车官方微博 @丛熊壮 @DLTV2生活频道 @海力网 @新青年网站

2012-5-25 20:25 来自iPhone客户端 | 转发(191) | 收藏 | 评论(23)

Fig. 1: False rumor about Rove Beetle



温州都市报V:【遇到这种虫子千万别用手拍】温州市区一学校近日出现隐翅虫，附一医医生介绍，该虫不会蜇人，但是它体内有毒液，打死后毒液流出来，会迅速导致人体皮肤起泡、化脓，并且成片扩展，越抓越严重。不小心沾染上毒隐翅虫的毒液，可用肥皂水或4%苏打溶液或10%氨水反复清洗皮肤，并尽快到正规医院接受治疗。

2013-10-14 15:31 来自微博 weibo.com (10) | 转发(137) | 收藏 | 评论(17)

Fig. 2: Normal message about Rove Beetle

Consider the following two postings from Sina Weibo (translated from the original messages in Figure 1 and Figure 2):

- **Dirty Cat Eats Foul Fish**: *Global warning! Please repost! You should never crush a rove beetle if it lands on your skin. Rove beetles contain venom which kills people for sure if it contacts skin! Tell your children and friends that it's better to just gently blow the rove beetle away if it lands on your skin. Doctors specially warn that if you come across* *a rove beetle, never smash it with hands! @AccessCity @CongXiongzhuang @DLTVLivingChannel @hiliziWebsite @NewYouthWeb*

- **Wenzhou City News**: *[Don't crush this beetle with hands when you see it] Rove beetles have been discovered in a school of Wenzhou. According to a doctor, rove beetle doesn't bite human beings, but once the venom in its body comes in contact with human skin, it can quickly cause blistering and festering, spreading to other areas and getting worse if you scratch it. If you ever touch the venom of rove beetle, please clean the skin repeatedly with soap water, 4% baking soda solution or 10% ammonia, and seek help at a local hospital immediately.*

Both messages quote the words of doctors, include vivid photos, and have comparable number of reposts, which makes them hardly distinguishable. The first message even *mentioned* a few prominent organizational users with the "@" sign, which appears to be official. However, the first posting is a false rumor while the second is not. The truth is, venom of rove beetle does not kill people but can result in skin infection. Such knowledge is hard to come by for average people, which explains why some innocent people repost false rumors and inadvertently help their spread. The detection task is even harder for computers, because the two messages contain very similar keywords such as "rove beetle", "crush", "venom", "skin" and "doctor". Existing natural language processing techniques can easily confuse the two messages due to their similarities.

One way to understand the difference between these two messages is by looking at the message propagation structures. Figure 3 and Figure 4 illustrate the initial parts of the propagation graphs for the above examples. Each node in the graph (denoted by rounded box) represents either an *opinion leader* or a *normal user*. We will define *opinion leader* and *normal user* in III-A. Each node includes the username and his or her message (either the original message or the repost message). Furthermore, the node for an opinion leader also carries the number of followers and friends, number of reposts. It can be seen that the false rumor (Figure 3) is first posted by a normal user, then reposted and supported by some opinion leaders and finally reposted by a large number of normal users. On the contrary, the normal message (Figure 4) is posted by an opinion leader and reposted directly by many normal users. This subtle difference of propagation structure is the primary inspiration of this paper. Besides the overall propagation structure, there are other hints in Figure 3 that suggest it is a false rumor: some normal users expressed their doubts or dispproval in their responses (highlighed in red color). Such signals can be picked up by high level semantic analysis of each individial response.

Much of the previous work on false rumor detection focuses on extracting large number of lexical and semantic features from the original message and responses, and learn a model from labeled data [7], [8]. While they do consider the relationships among a thread of messages, they limit themselves to a flat summary of statistics about the message propagation patterns, such as the total number of reposts, depths and degrees of the propagation tree, etc. They do this for the convenience of constructing feature vectors for machine learning. Such an approach is over-simplistic because it ignores the internal graphical structure of the message transmission as well as the differences among the users along that structure. Our key insight is that most false rumors can be identified not only by what the false rumor says, but also by the way people respond to it and who these people are. The propagation patterns, combined with the topics of the thread and the sentiment of the responses, can give strong indications whether the original message is the truth or fiction.

The main contributions of this paper are summarized below:

- We model the pattern of message propagation as a tree, which not only reflects the relation among reposts and their authors but also the temporal behavior and the sentiment of reposts. The tree can be simplified to adapt to the space and time requirement of the system (Section III-A).
- We propose a random walk graph kernel to model the similarity of propagation trees. Results suggest that the propagation of false rumors can be distinguished from other messages (Section III-B).
- We combine the graph kernel and a radial basis function kernel, together with other novel features to build a hybrid SVM classifier (Section III-C and Section III-D). Our experiments show the hybrid model achieves significant better classification accuracy of 91.3% than those reported in previous work and can be used for the early detection of false rumors (Section IV).

II. Problem Definition

Sina Weibo is a social network in which a user can follow some other users (called *friends*), and be followed by some other users (called *followers*). A follower receives messages posted by his or her friends and can respond by reposting to the messages.

We model the Weibo data as a forest W of *message propagation trees*. A propagation tree $T = \langle \text{V, E} \rangle$ is akin to a message thread on forums or bulletin boards. Each node m in V represents a text message posted on Weibo which contains 140 Chinese or Latin characters. m is associated with meta data $\langle u,\ t,\ c,\ i \rangle$, where u is the creator of the message, t is the time stamp of the message, c is the type of client from which the message is sent (e.g., web, mobile, etc.), and i is the set of optional images which are posted along with the text. The user information u contains additional attributes of the user such as gender, number of friends and followers, number of messages posted in the past, last time of post, etc. The root node of a tree is called "original message", while all the other nodes in the tree are called "reposts", as they are the responses to either the original message or other reposts. If there is a directed edge from m_1 to m_2, then m_2 is a response to m_1. For example, the following is the initial part of a thread where m_1 is the original message, and "//@user1" means a response to user1.

- m_1 (user1): When a rove beetle is on your skin, don't crush it or your skin will fester.
- m_2 (user2): Really? I never saw it before! //@user1
- m_3 (user3): Thanks for the warning. //@user1

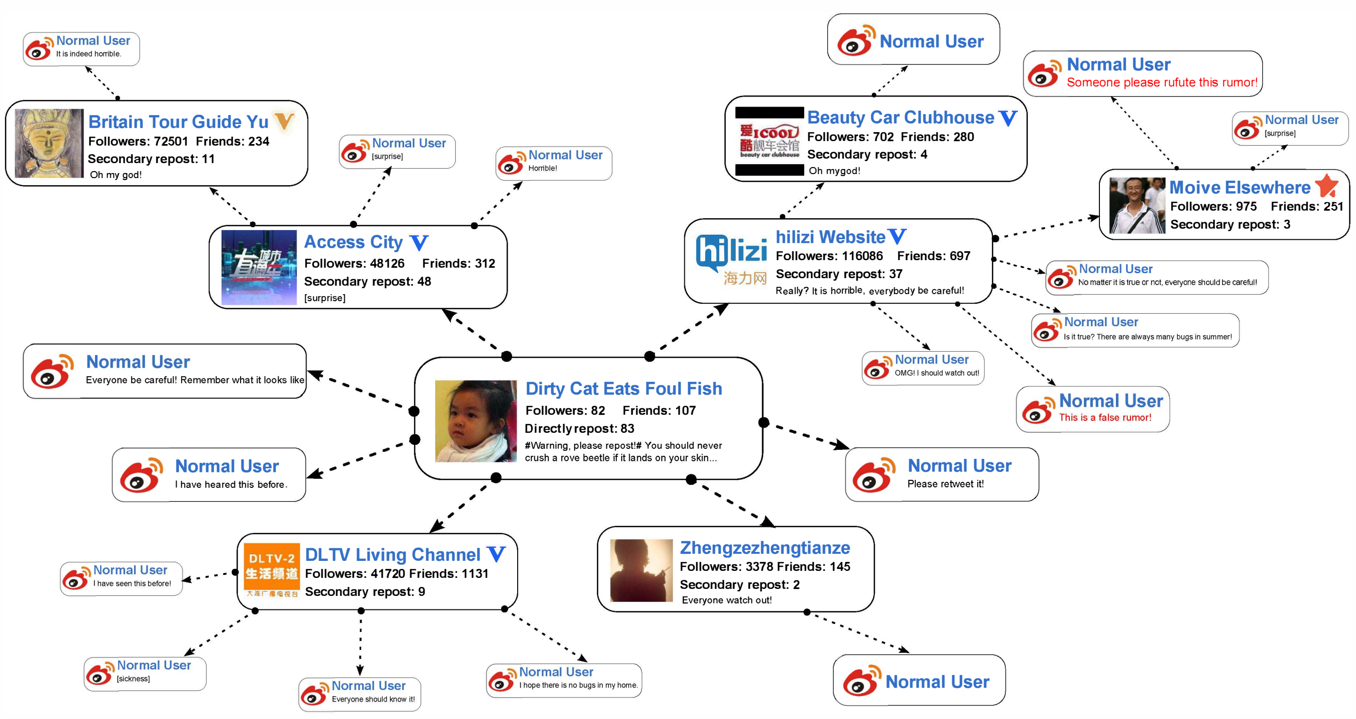


Fig. 3: Fragment of false rumor propagation graph

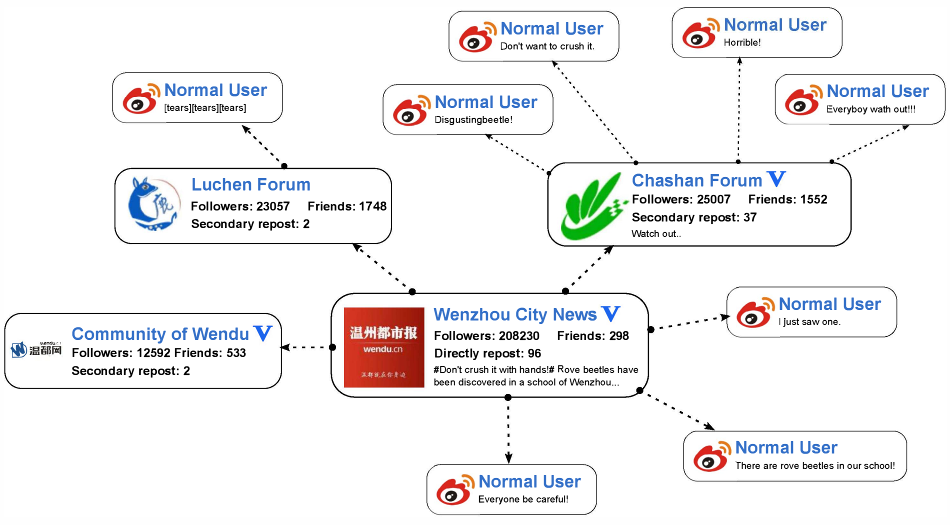


Fig. 4: Fragment of normal message propagation graph

- m_4 (user4): That sounds awful! What is rove beetle? //@user2

- m_5 (user5): It's true. I've been bitten once. //@user2

Figure 5 illustrates the corresponding propagation tree.

In this paper, false rumors and normal messages (which are not false rumors) only refer to the original Weibo messages and not reposts. An original message is either a false rumor or a normal message. Our problem is, given a message propagation tree $T = \langle V, E \rangle$ as well as the meta data associated with V, return whether $root(T)$ is a false rumor or not.

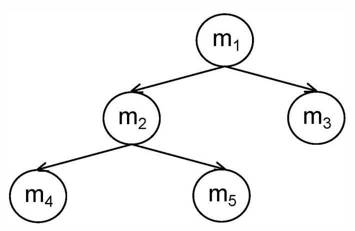


Fig. 5: A Partial Propagation Tree

III. Approach

Our general approach is based on an SVM classifier using a hybrid kernel function which combines a novel random walk graph kernel and a normal radial basis function (RBF) [9]. The graph kernel assesses the similarity between different propagation trees while the RBF kernel computes the distance between two vectors of both traditional and high level semantic features. In this section, we will first introduce the labeled propagation tree structure as well as the random walk kernel, then present 8 new features used in the RBF kernel, and finally show how to combined the two kernels into a hybrid SVM kernel.

A. Propagation tree

For the purpose of the random walk graph kernel, we enrich the propagation tree in Figure 5 by adding additional information which represents the type of user of each message and the opinion and sentiment toward the original message. The resulting propagation tree[2] will be used in the graph kernel computation.

We divide the users into two types: *opinion leaders* and *normal users*. Opinion leaders are those influential users whose opinions dominate their followers[10]. A user is considered an opinion leader if

$$\frac{\#\text{ of followers}}{\#\text{ of friends}} > \alpha \tag{1}$$

where $\alpha > 1$ and # of followers ≥ 1000. We thus label each node of the tree as p if it comes from an opinion leader and n otherwise.

We label the edge from m_i to m_j, called e_j[3], with a triple $\boldsymbol{v_j} = (\theta(a), \theta(d), \theta(s))$, where a is the *approval score* of m_j, which indicates approval or agreement, d is the *doubt score* of m_j which indicate doubts and suspicion, and s is the overall sentiment score in m_j. We defer the computation of a, d and s to Section III-C. θ is a damping function defined by

$$\theta(x) = 2^{-\rho t} x \tag{2}$$

where t is the time difference in days between the original message and m_j, and ρ is a parameter between 0 and 1. The idea is: the sooner a user gives a response, the more intense the response is. Figure 6 shows such a labeled propagation tree in which all reposts are sent in the same day as the original message. Once the triple is extracted from the message, message id m_i can be removed from the nodes for simplicity.

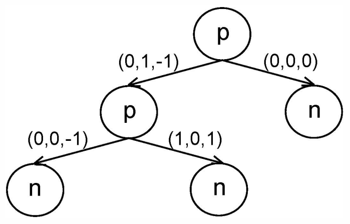


Fig. 6: Example of Labeled Propagation Tree

[2]Note that this data structure contains only part of the information contained in the message propagation tree model in Section II. The remaining information will be used in the RBF kernel.

[3]Since the parent of m_j is unique, e_j uniquely identifies the edge between m_j and its parent.

Our intuition is that patterns can be discovered from the labeled propagation tree, which helps distinguish false rumors from others. For example, Figure 7 compares the partial labeled trees rooted from the two messages about rove beetles in Figure 1 and Figure 2. Despite the lexical similarity of the two original messages, the false rumor (a) is reposted and supported by many opinion leaders at first before normal users take over the propagation; conversely the normal message (b) is initially reposted by a majority of normal users. This shows that the influence of multiple opinion leaders can quickly create a "hype" which is followed by ordinary users.

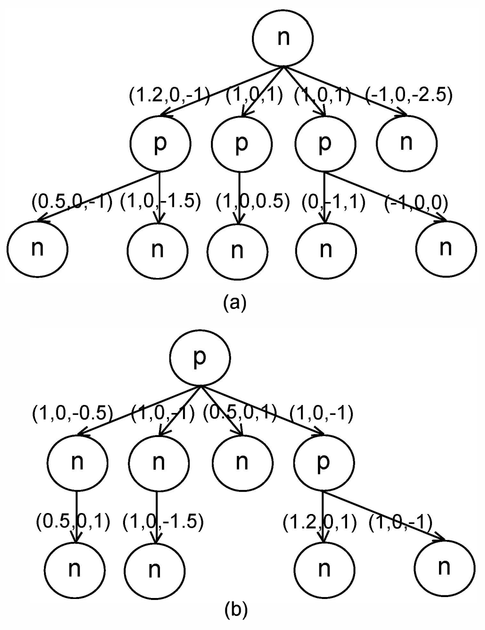


Fig. 7: Tree of False Rumors and Others

In a social network with 50 million active users, a popular message can be reposted thousands of times and the propagation tree thus gets extremely large. To reduce the computation complexity of graph kernel function, we develop the following rules to simplify a tree by lumping adjacent normal user nodes together to form one *super node*, and thus reduce Figure 7(b) to Figure 8:

1) If m_i is the parent of m_j, and both are labeled as n, then m_i, m_j merge into one node m_{ij}, whose parent is the parent of m_i and whose children are the children of m_i or m_j;
2) If m_i is sibling of m_j and both are labeled as n then m_i, m_j merge into one node m_{ij}, whose parent is the parent of m_i and m_j, and whose children are the children of m_i or m_j;
3) The merged node m_{ij} has label n and the label of incoming edge e_{ij} is $\boldsymbol{v_{ij}} = \boldsymbol{v_i} + \boldsymbol{v_j}$;
4) Do not merge the root with any other nodes;
5) Repeat the above rules until no pair of nodes can be merged;
6) For each super node, normalize the vector on incoming edge by the number of ordinary nodes merged into this super node.

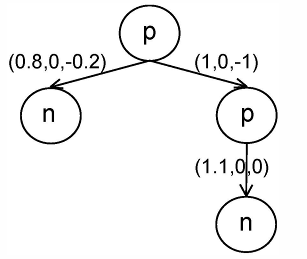


Fig. 8: Simplified Propagation Tree

B. Random walk graph kernel

To classify different propagation trees by SVM, we need to calculate the similarity between trees. There are several tree kernel functions that can be used to calculate the similarity of trees such as the subset tree (SST) kernel[11] and subtree (ST) kernel[12]. While these kernels prove to be useful in natural language processing[13], they can not be used for our problem because they consider two nodes to be similar only when they have the same number of children. Whereas in this paper, if m_i is reposted a times and m_j is reposted b times we would like to consider them similar to some extent.

Instead of tree kernel, we use a random walk graph kernel [14] to calculate the similarity of trees. Because the labels of edges in the propagation tree are not discrete values but continuous vectors, we modify the original random walk kernel so that the kernel is applicable to graphs with continuously labeled edges[15].

Given two trees $T = (V, E)$ and $T' = (V', E')$, we first calculate the direct product graph of two trees. The direct product graph of two trees is $G_\times = (T \times T') = (V_\times, E_\times)$, where

$$\begin{aligned} V_\times &= \{(v, v') \in V \times V' : label(v) = label(v')\} \\ E_\times &= \{((u, u'), (v, v')) \in V_\times^2 : \\ &\qquad (u, v) \in E \wedge (u', v') \in E'\} \end{aligned} \tag{3}$$

The adjacency matrix of the direct product graph $G_\times$ is $A_\times$, which is defined as $[A_\times]_{(u,u'),(v,v')} = l$, where

$$l = \begin{cases} k((u, u'), (v, v')) & \text{if } ((u, u'), (v, v')) \in E_\times, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

The kernel function k measures the similarity of edges $e_{(u,v)}$ and $e_{(u',v')}$, and is given by

$$\begin{aligned} k((u, u'), (v, v')) &= k_{edge}((u, v), (u', v')) \\ &= e^{-\frac{\|\boldsymbol{v_1} - \boldsymbol{v_2}\|^2}{2\sigma^2}} \end{aligned} \tag{5}$$

where $\boldsymbol{v_1}$ is the vector label of $e_{(u,v)}$, $\boldsymbol{v_2}$ is the vector label of $e_{(u',v')}$, and σ is a parameter.

Given the adjacency matrix $A_\times$ and a weighting parameter $\lambda \geq 0$ we can define a random walk kernel on T and T' as

$$\begin{aligned} K_\times(T, T') &= \sum_{i,j=1}^{|V_\times|} \Big[\sum_{n=0}^{\infty} \lambda^n A_\times^n\Big]_{ij} \\ &= \boldsymbol{e}^T(\boldsymbol{I} - \lambda A_\times)^{-1}\boldsymbol{e} \end{aligned} \tag{6}$$

If $\lambda < 1$ and is sufficiently small then the sum will converge.

Assuming T and T' contain n vertexes, then $A_\times$ is a $n^2 \times n^2$ matrix. Thus computing $(\boldsymbol{I} - \lambda A_\times)^{-1}$ directly requires $O(n^6)$ time, which is too slow. In order to speed up, we compute the graph kernel in two steps. First, we solve the linear system

$$(\boldsymbol{I} - \lambda A_\times)x = e \tag{7}$$

for x, then we compute $e^T x$. In the first step, we use conjugate gradient (CG) method to solve the linear system [16]. CG is very efficient to solve the system of equations $Mx = b$ if the matrix M is rank deficient. In our work, the adjacency matrix is from the product of two trees, which means the matrix is sparse. As such, the CG solver can be sped up significantly [17]. To solve the linear system, CG takes $O(n^4 i)$ where i is the number of iterations.

C. Features

We extract a total of 23 features from each message propagation tree to build a vector for RBF kernel. Some of the features have been proposed previously [4], [7], [8] and shown to be effective. These features are largely based on the basic characteristics of the original message itself or its author. Besides, we propose 8 new features in this paper, which can boost the accuracy of classifier. Some of the features are specific to the Sina Weibo platform while others are generic. We divide these features into 3 categories: *message-based* and *user-based* features which are extracted from the original message and its author, and *repost-based* features which are calculated from the set of all reposts of an original message. Table I documents all 23 features and their brief descriptions. Features marked with * are new features proposed in this paper. Next we discuss the new features in more details.

Topic Type feature refers to the topics of the original message. We assume that a message can belong to one or more topics. Since Sina Weibo has an official classification of 18 topics [18], we train a Latent Dirichlet Allocation (LDA) [19] model which returns an 18-topic distribution for message m_i:

$$topic(m_i) = (s_1, \ldots, s_{18}) \tag{8}$$

where s_j is the probability of m_i belonging to topic j. We further convert $topic(m_i)$ into a binary vector by setting the k highest probability topics to 1 and the rest of the topics to 0. We select k such that the total probabilities of top k topics is above 0.5 while the total probabilities of top $k-1$ topics is below 0.5.

Search Engine feature refers to the number of results returned by searching for the original message and the keyword "false rumor" on a web search engine. Due to the limitation of query length imposed by search engines, we divide the message into word sequences of ql_{max} characters[4] each. Then each sequence q_i is searched by querying "intext:q_i intitle:false rumor". The final score for the message is obtained by averaging the numbers for all queries.

User Type feature refers to the verified type of author. Recently Sina Weibo not only classifies users into verified and unverified, but categorizes all users into 12 refined types.

[4]ql_{max} is 32 for Google.

TABLE I: Description of 23 Features

Category	Feature	Description
MESSAGE	HAS MULTIMEDIA	Whether the message includes pictures, videos or audios
	SENTIMENT	The average sentiment score of the message
	HAS URL	Whether the message contains URLs
	TIME SPAN	The time interval between user registration and posting
	CLIENT	The type of software client used to post the original message
	TOPIC TYPE*	The topic type of the message based on LDA
	SEARCH ENGINE*	The number of search results returned by Google
USER	IS VERIFIED	Whether the author is verified by Sina Weibo
	HAS DESCRIPTION	Whether the author has personal description
	GENDER	The author's gender: female or male
	LOCATION	Location where user was registered
	NUM OF FOLLOWERS	The number of people following the author at time of post
	NUM OF FRIENDS	The number of people the author is following at time of post
	NUM OF POSTED MESSAGES	The number of messages posted by the author at time of post
	REGISTRATION TIME	The time of author registraton
	USER TYPE*	The type of author based on the verified information
REPOST	NUM OF COMMENTS	The number of comments on the original message
	NUM OF REPOSTS	The number of reposts from the original message
	AVG SENTIMENT*	The average score of sentiment based on lexicon
	AVG DOUBT*	The average score of doubting based on lexicon
	AVG SURPRISE*	The average score of surprising based on lexicon
	AVG EMOTICON*	The average score of emoticon
	REPOST TIME SCORE*	The time interval between original message and repost

For example, -1 means not verified, 0 means verified media celebrities, 3 means verified official media, etc.

Avg Sentiment feature refers to the average sentiment score of all reposts of an original message. Each repost is first segmented into Chinese words by the toolkit NLPIR [20] with stop words removed. After that, we calculate the sentiment score of each message based on the sentiment lexicon of HowNet [21] and the basic emotion family of Ekman [22]. The average sentiment score is

$$\frac{1}{n}\sum_{i=1}^{n}\frac{NP_i - NN_i}{|m_i|} \tag{9}$$

where NP_i is the number of positive words and NN_i is the number of negative words in m_i, $|m_i|$ is the number of words in m_i, and n is the total number of reposts for that message. Note that a positive word can be negated by a preceding "not" or similar words (called negation word) in Chinese and hence become a negative word. Same goes for negative words. **Avg doubt**, **Avg surprise** and **Avg emoticon** features are calculated similarly except the lexicons used are specially for these categories. For example, when calculate **Avg doubt** of an original message, NP_i is the number of doubt words (based on a doubt word lexicon) and NN_i is the number of non-doubt words.

Similarly, approval score a or doubt score d of m_j (introduced in Section III-A) is computed as

$$\frac{NP_j - NN_j}{|m_j|} \tag{10}$$

where NP_j is the number of approval (or doubt) words and NN_j is the number of dispproval (or non-doubt) words in m_j, $|m_j|$ is the number of words in m_j.

Repost Time feature is calculated from the time difference in days between the original message and the repost:

$$\frac{1}{n}\sum_{i=1}^{n}2^{-(t_i - t_{\bullet})} \tag{11}$$

where n is the total number of reposts, t_i is the time stamp of repost m_i and t_o is the time stamp of the original message. This feature represents the timeliness of the responses.

D. Hybrid kernel

For traditional SVM, input data is represented as $\{\boldsymbol{X_i}, y_i\}$ where $\boldsymbol{X_i}$ is the feature vector. In this work, we use $\{\boldsymbol{X_i}, y_i\}$ to represent an original message m_i. $\boldsymbol{X_i}$ has 23 dimensions and y_i is the binary class label of false rumor or not. The RBF kernel for this binary classifier is

$$K(\boldsymbol{X_i}, \boldsymbol{X_j}) = \langle \phi(\boldsymbol{X_i}) \cdot \phi(\boldsymbol{X_j}) \rangle \tag{12}$$

where ϕ denotes the feature map from an input space to the high dimensional space associated with the kernel function.

Moreover, every original message m_i is associated with a propagation tree T_i. In order to normalize the kernel function of two propagation trees in Eq. (6), we divide $K_\times(T, T')$ by nn' where n and n' are the numbers of nodes in T and T':

$$K(T, T') = \frac{1}{nn'}K_\times(T, T') \tag{13}$$

Therefore, the kernel function of message m_i and m_j can be defined as

$$K(m_i, m_j) = \beta K(T_i, T_j) + (1-\beta)K(\boldsymbol{X_i}, \boldsymbol{X_j}) \tag{14}$$

where $0 < \beta < 1$, and β determines the proportional weight of random walk kernel versus the feature vector kernel. In the following experiments, we will train an SVM classifier based on this hybrid kernel.

E. General Applicability

Although this paper targets Sina Weibo, the methods developed can be applied to other micro-blogging platforms such as Twitter to detect false rumors there. Since most of the old features we use were proposed in Castillo[7], which targeted Twitter specifically, it is easy to extract these old features using the Twitter API. On the other hand, almost all new features belong to the message category and the reposting category, which are only related to the text of micro-blogs, hence can also be extracted using the Twitter API.

IV. Evaluation

The evaluation of the hybrid SVM classifier consists of the following phases: collection of Weibo data and annotation of the original messages; tuning parameters of the SVM model; evaluation of effectiveness of various features; comparison with competing methods on end-to-end results; and finally performance of the model on false rumor early detection.

A. Dataset

To train and evaluate our approach of detecting false rumors, a labeled data set is needed. We collect a set of known false rumors from Sina community management center [23], which deals with reporting of issues including various misinformation which we regard as certified false rumors. There are 11466 reported false rumors between 2012/05/28 and 2014/04/11. Since a rumor must have sufficient circulation, we only keep those false rumors that have at least 100 reposts, which leaves us with 2601 false rumors up to 2014/04/11. Sina Weibo API provides interfaces to capture the information of original messages as well as their repost messages. From Sina Weibo API, we captured the post time, post client and content of 2601 false rumors along with all their reposts.

In the real world, the number of false rumors on Sina Weibo is much smaller than the number of normal messages (1 out of 9 or less). Thus a "dummy" classifier that rules all messages as normal messages will achieve a very high accuracy (above 90%) on real-world data. To avoid this problem, we construct a data set with roughly equal number of false rumors and normal messages. Most studies in the past also use data sets which are either 50-50 split [7], [24] or close to that [4], [8]. Thus, we randomly select 5000 other Weibo original messages which are not proved to be false as well as their reposts using the Sina Weibo API. Then, we manually filtered out messages with fewer than 100 reposts as well as false rumors to form a set of 2536 normal messages. Each message or repost contains links to the author profile information such as age, gender, number of followers and friends, and can be crawled using the Weibo API.

At the end of this phase, our labeled data set [5] consists of 2601 false rumors, 2536 normal messages and with 4 million distinct users involved in these messages. Of these 500 false rumors and 500 other messages (called small data set) are used for SVM parameter tuning while the rest (called big data set) are used for end-to-end cross validation.

[5]The labeled data set of the original messages (without reposts) is available at http://adapt.seiee.sjtu.edu.cn/~kzhu/rumor/.

B. SVM parameter tuning

The SVM classifier is implemented using LIBSVM[25]. We first use the small data set and 10-fold cross validation to obtain the following parameters values that will be used in all subsequent experiments:

$$\begin{aligned} 2\sigma^2 &= 3 \\ \rho &= 0.1 \\ \gamma &= 2^{-11} \\ cost &= 2^{13} \end{aligned}$$

By Eq. (1), α controls the number of opinion leaders in the tree and hence affects the final simplified tree and the calculation of the kernel function. Here we analyze the impact of α on the size of the product graph in the graph kernel as well as the accuracy of the SVM through 10-fold cross validation. To suppress the effects of the vector kernel, we set $\beta = 1$ in this experiment. The results of experiment are shown in Figure 9 and Figure 10.

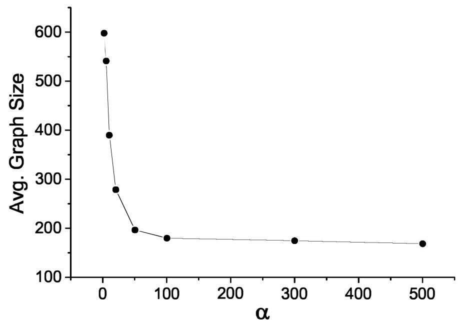


Fig. 9: Average number of vertexes in product graph vs. α

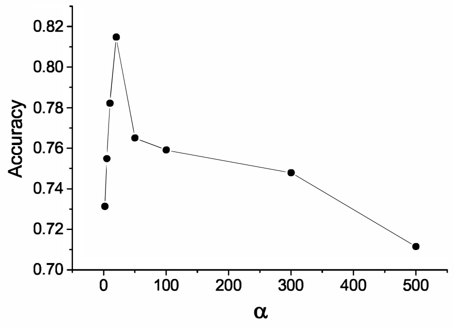


Fig. 10: Classification accuracy vs. α

Accuracy hits the maximum when $\alpha = 20$. It is also interesting to note that as α grows, the size of the graph converges, which indicates that when α is large, most ordinary users have been merged into super nodes and there are only small number of opinion leaders in the "long tail" who have extremely large fan base. In the following experiments, we set $\alpha = 20$.

Next we try to tune the best β value to balance the graph kernel and the RBF kernel. For each value of β, we train an SVM classifier and record its accuracy. The results of experiment are shown in Figure 11.

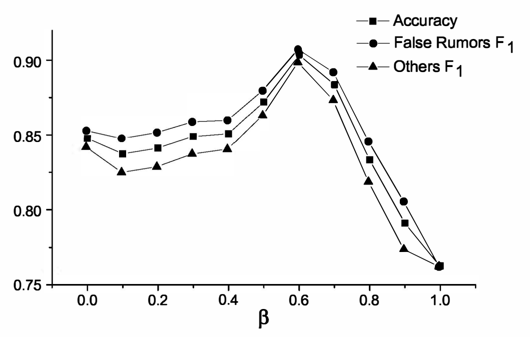


Fig. 11: Accuracy, F_1 for false rumors and for other messages vs. β

Results show that when $\beta = 0.6$, the hybrid kernel achieves the best accuracy and the combination of the two kernels performs better than each individual kernel (two ends of the graph). Therefore, in the following experiments, we set $\beta = 0.6$.

C. Feature selection

To investigate the effectiveness of our new features, we train several SVM classifiers using different subsets of the features. We also train a classifier without graph kernel function to show its usefulness. The small data set (1000 messages) is divided into training set and test set with a ratio of 2:1. For each subset of features, we train an SVM classifier on the training set and test the classifier on the test set. The results of experiment are shown in Table II. Here (-)X means the whole set of features except feature X. F stands for "false rumors" while O stands for "other messages".

The results show that the inclusion of the graph kernel is indeed very effective, improving the accuracy by 0.056, which is the largest single-feature improvement among all features. This clearly indicates that the explicit representation of propagation tree patterns better models the false rumors than others. Results also suggest that TOPIC TYPE is the most effective among all ordinary features. This is because false rumors tend to concentrate on a few sensitive topics, such as missing persons or health issues. The features about sentiments also have significant impact on the result, which means people's opinions, especially when they are doubtful or surprised, point to possible false rumors. Finally, when all new features and the graph kernel are removed, the accuracy of the classifier drops considerably, which again shows that the graph kernel and our new features, when combined together, provide substantial boost for the classifier.

D. End-to-end false rumor detection

We compare our hybrid SVM classifier with two other state-of-the-art false rumor detection algorithms[7], [4]. Castillo's J48 decision tree is implemented using the 15 best reported features under WEKA; Yang's SVM classifier was implemented using all 19 reported features except locations of the messages which are not available in our data. The location feature was shown to be not particularly useful in Yang's system anyway. We also train an SVM classifier with only graph kernel ($\beta = 1$) as baseline to evaluate the classification performance of graph kernel (Graph). Finally, we train an SVM classifier with all features except the graph kernel, plus 7 simple graph features proposed by Castillo[7] to examine if our graph kernel is indeed important versus just simple graph features extracted from propagation tree (Simple). For this experiment, we compute the accuracies, precisions, recalls and F_1 measures by 3-fold cross validation on the big data set.

TABLE III: Comparison of different methods

Methods	Hybrid	Castillo	Yang	Graph	Simple
Accuracy	**0.913**	0.854	0.772	0.770	0.856
F precision	**0.905**	0.853	0.773	0.773	0.846
F recall	**0.922**	0.854	0.776	0.763	0.871
F F_1	**0.913**	0.854	0.774	0.768	0.859
O precision	**0.920**	0.853	0.770	0.766	0.866
O recall	**0.903**	0.854	0.768	0.776	0.840
O F_1	**0.912**	0.854	0.769	0.771	0.853

Table III shows the result. Overall, the table demonstrates that our approach outperforms the other competitions by large margins across all measures. The hybrid SVM classifier achieve a higher accuracy than the classifier with simple graph features. This is because hybrid SVM classifier could store much information of the propagation tree while simple graph features lose a lot of such structural information. Besides, the graph kernel alone has a comparable performance to the baseline of Yang. These results indicate that propagation tree pattern is a critically important high-order feature for distinguishing false rumors from others.

E. False rumor early detection

In this subsection, we evaluate the ability of the hybrid SVM classifier in detecting false rumors in the early stage of propagation. Given a detection deadline, we assume all reposts and related information published after this deadline are invisible when testing the model of the classifier. For example, if the detection deadline is 24 hours, then we only use the data that was generated during the first day after the original message is posted. The sooner the deadline, the less data of reposts and hence the propagation structures can be used. A deadline of 0 hours means that we will not use any repost data except the original messages themselves. Such detection deadlines affects the graph kernel as well as all features in REPOST category.

Figure 12 shows the average accuracy of detection for various deadlines. The experiment was performed on the small data set. The data set was divided into two parts evenly, one part was used to learn the model of classifier, another part was used to test the model. When learning the model, all training data is used, while only the test data before the deadline is used when testing the model. Our model (Hybrid) achieves 72% accuracy when deadline is 0. This accuracy is low because there is no information of propagation structure but only some static features from the original messages. As deadline

TABLE II: Impact of features

	Accuracy	F precision	F recall	F F_1	O precision	O recall	O F_1
(-)TOPIC TYPE	0.865	0.836	0.911	0.872	0.900	0.818	0.857
(-)SEARCH ENGINE	0.880	0.872	0.912	0.892	0.906	0.863	0.884
(-)USER TYPE	0.894	0.877	0.919	0.897	0.912	0.868	0.890
(-)AVG SENTIMENT (-)AVG DOUBT (-)AVG SURPRISE	0.872	0.846	0.912	0.878	0.903	0.830	0.865
(-)AVG EMOTICON	0.887	0.874	0.908	0.891	0.902	0.866	0.884
(-)REPOST TIME SCORE	0.892	0.874	0.919	0.896	0.912	0.865	0.888
(-)Graph Kernel	0.848	0.851	0.846	0.849	0.844	0.849	0.846
(-)All New Features	0.796	0.806	0.782	0.794	0.786	0.810	0.798
(-)All New Features & Graph Kernel	0.761	0.748	0.792	0.769	0.777	0.730	0.753
With All Features & Graph Kernel	**0.904**	**0.886**	**0.929**	**0.907**	**0.924**	**0.877**	**0.900**

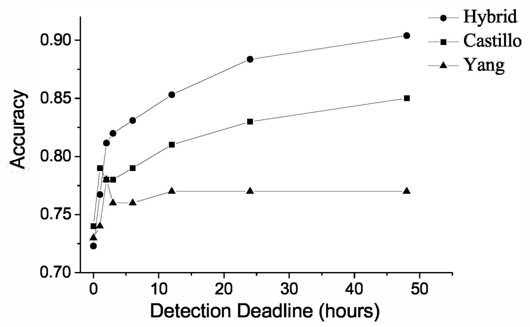


Fig. 12: False rumor early detection

is lengthened, the accuracy climbs rapidly, which indicates that the features from the responses and the propagation graph weigh in. By 24 hours after the initial posting, the detection accuracy is 88%, which suggests that the model is 88% confident to detect an average false rumors within the first day of the posting of original messages. We also experimented with rumor early detection on Yang's[4] and Castillo's[7] algorithms, and compare their results with ours in Figure 12. The accuracies of the three methods at time 0 are mostly the same because three methods share similar basic features when there is no repost. However, after 24 hours, Yang's result improves a little, Castillo's improves substantially, while ours improves the most. This is because Yang's method uses only simple features related to reposting (e.g., "IS RETWEETED", "NUM OF COMMENTS", "NUM OF RETWEETES"), Castillo's method includes some simple but effective features of reposting such as "AVG SENTIMENT SCORE" and "PROPAGATION MAX LEVEL", while our method considers the propagation structure as well as many other reposting features and thus benefits most from the signals of the reposts. Among the three methods, ours converges most quickly in terms of early false rumor detection.

V. Related Work

Previous work on false rumor detection for microblogging service (either on Twitter or Weibo) has largely modeled the problem as a binary classification problem. Hence the primary focus has been feature selection. In the following, we will first discuss the definition of rumor. Then, we will discuss features ever used in false rumor detection in the literature. After that, we will compare various classification methods and discuss some graph kernels. We conclude this section with rumor source detection, which is a related problem.

A. *Definition of Rumor*

Although there is no commonly accepted definition of rumor, in many dictionaries [3] and previous literatures [26], an important character of rumor is uncertainty. In the previous work on false rumor detection for microblogging service, there are different definitions of rumor and false rumor. Some previous work [8] defines rumor according to social psychology, where a rumor is defined as a statement whose truth-value is unverifiable or deliberately false. In their research, they use the word "false rumor," "misinformation" or "disinformation" to distinctively refer to rumors that turn out to be false eventually [7], [27]. In our work, we follow a similar definition.

However, some other research [4], [28] does not make a distinction between "rumor", "false rumor", and "misinformation," and use these terms interchangeably to mean false statements. Some literatures [29], [24] use the word "rumor" as the opposite term of "news" where they consider "news" to be always true while "rumor" always false. Most of these researchers come from Asia where "rumor" is generally carries a negative connotation. Such cultural differences may be the reason why there is no universally agreed upon definition for this term.

B. *Features for False Rumor Detection*

We divide existing features for false rumor detection into 4 types.

Linguistic Features pertain to the microblog message text [8]. They range from simple features such as message length, punctuations, letter case, whether URLs or hashtags are included [30], types of emoticons used and POS tags [31] to more advanced semantic features such as sentiment scores [7], [8] and opinion words [32]. Previous research [4], [32] shows that not all these features are effective for false rumor detection. The most significant features among them are emoticons, opinion words and sentiment scores (positive or negative). In this paper, we used all the effective features, plus a topic model feature and a search engine feature (see Section III-C). These new semantic features were not previously attempted and they turn out to be very useful.

User Features describe the characteristics of an individual user. These include the time and location of the account registration, gender and age of the user, username and avatar [33], whether this is a verified account, number of friends, number of followers, the description and the personal home page of the user, number of messages post in the past, etc. [7] These features are associated with the original message to be classified in this paper. Furthermore, we utilize a more refined user type than verification status.

Structural Features pertain to either the message propagation tree or the user friendship network. All existing work [7], [8], [28] focuses on the numeric summary of such graph structures, e.g., the total number of nodes (i.e., messages or users) in the graph, maximum or average depth of the graph, the degree of the root and the maximum or average degree of the graph. Most of the work treats each node (either message or user) equally and hence only derives such generic statistics. Recently, some researchers [24], [6] adapted the epidemiological models to false rumor detection, and group users into population compartments such as susceptible (S), infected (I) and skeptic (Z), etc. Users transit from one compartment to another as they choose to or not to repost a topical message. Structural features under these models are slightly more refined as the they keep the counts for each compartment separately. Our approach adopts some of these features but we advocate that the actual graph structure of the message propagation is more explicit thus important than the summary statistics. But since the graph can be very big, we distinguish the messages posted by opinion leaders or normal users and propose a way to simplify the graph so it can be used efficiently in a graph kernel.

Temporal Features look at the time stamps of the messages and compare them with the time of the original post or the time when the author was first registered [7]. More advanced models use these times to detect sudden spikes in the volume of responses or periodicity of such spikes [32]. Researchers also use time to calculate rates of population change among the compartments in epidemiological models [24], [32]. We use the time between a repost and the original message as a damping factor to indicate the strength of the sentiments in the response. Thus responses which are posted long after the original message have little effect on false rumor identification.

In addition to the above 4 types, there are also miscellaneous features like type of software client used to post a message, location from which a message is posted, etc.

C. Classification Methods

Most of existing research uses common supervised learning approaches such as decision tree, random forest, Bayes networks and support vector machine (SVM). Castillo et al. [7] reported that different methods produce comparable results but decision tree is the best for 608 topics (equivalent to our original messages), with a classification accuracy of 89% under 3-fold cross validation. More recently, Kwon et al. [32] considered random forest to outperform other methods with 11 features on 102 topics each with at least 60 tweets. Although they reported 90% accuracy under 2-fold cross validation, their data set is relatively small. Our paper proposes a hybrid SVM classifier which combines a random walk graph kernel with normal RBF kernel using 23 features including 8 new features. Our experiments show that its performance is superior against the state-of-the-art methods and features used by Castillo et al. and Yang et al.

Okazaki et al.[27] instead used unsupervised approach for extracting false information after the 2011 Japan earthquake and tsunami. They designed a set of linguistic patterns for correction or refutation statements, extracted the text passages that match the correction patterns and clustered them into different topics. At last, they selected a representative passage for each topic as the rumor.

Besides that, research that adapts epidemiological models [24], [6] to false rumor detection generally define ordinary differential equations (ODEs) on the rate of user population changes and fit non-linear functions to the data. By observation of the function curves of different population compartment, they then manually design a classification function to tell false rumors from others.

D. Graph Kernels

Traditional SVM classifiers are based on the data that can be represented as simple, flat vectors. However, it is not always reasonable as many objects in the real world are structured by nature[34]. For this reason, people have developed ways to incorporate complex structures such as trees and graphs as the kernels of SVM. In this paper, as proposed in Section III-B, although the propagation pattern of a message is tree-structured, we use graph kernel to calculate the similarity of two propagation trees.

One kind of kernel that can be used in graph is convolution kernel [35]. Convolution kernel assumes that one object can be decomposed into different parts and the similarity of two objects can be computed by combining the similarity of parts from two objects. Convolution kernel provides a generic way to construct kernel for discrete structured data and can be used in many different problems. One application of convolution kernel on tree structures is subset tree (SST) kernel[11]. SST decomposes syntax tree into different subset trees and the similarity of two trees can be computed by the similarity of their subset trees. Although convolution kernel is very general, it remains a difficult problem to find a reasonable method to decompose an object into different parts.

Graph kernel can be defined on all paths or shortest paths [36]. The all-path kernel is defined as the sum over all kernels on all pairs of paths from two graphs. However, computing the all-path kernel is time-consuming because finding all paths in a graph is NP-hard. Conversely, computing shortest path in a graph can be solved in polynomial time using classic algorithms such as Dijkstra [37] and Floyed-Warshall [38], [39]. Given a pair of graphs, we can first compute the shortest-path graph for both graphs. Then we compute pair-wise kernels from all pairs of edge walks of length 1, each coming from one of the short-path graphs. The shortest-path graph kernel of the pair of graphs is defined as the sum of all these pair-wise kernels. The shortest-path graph kernel is positive definite and can be computed efficiently ($O(n^4)$). However, it is inappropriate for computing the similarity of propagation trees in our problem because the shortest-path graph kernel only consider the shortest path of one graph, which loses information in the propagation tree.

Random walk graph kernel was first proposed by Gärtner [14]. The idea of random walk graph kernel is, given a pair of graphs, to first perform random walks on both simultaneously, then to count the number of matching walk paths. This procedure is equivalent to doing random walks on the direct product of two graphs [40]. Random walk graph kernel is applicable to graphs with labeled nodes and edges and have considered the whole graph when computing similarity, which is appropriate for propagation trees. However, the original random walk graph kernel can not deal with continuously labeled graphs [15]. For this reason, an extension of the original random walk kernel has been proposed [41]. The idea is to calculate the similarity rather than equality between two walks. The extended random walk graph kernel can then be applied to our problem.

E. Rumor Source Detection

Some previous work [42], [43] focuses on rumor propagation through social network. They try to use graph theory to detect rumors and find the source of rumors. A social network is modeled as a directed graph where each vertex represents an individual person and each edge represents information flow between two individuals. Some of the nodes are designated as "monitor nodes" where data that they receive may be observed. Given that messages are sent from some of the nodes and get propagated through the network, rumors can be detected and their sources can be recognized by observing the data received at the monitor nodes.

VI. Conclusion

This paper studies the problem of automatically detecting false rumors on the popular Chinese microblogging service, Sina Weibo. We develop a graph-kernel-based SVM classifier which combines the features from the topics of the original message, the sentiments of the responses, the message propagation patterns, and the profiles of the users who transmit this message around. Message propagation patterns have been used as high order features for the first time. Our results show that the repost patterns of false rumors and others are very different, which makes the random walk graph kernel very useful in detecting false rumors. The combination of random walk kernel and RBF kernel performs better than each of them alone, as well as recent state-of-the-art approaches, with an accuracy of 0.913. More importantly, our model can be used for the early detection of false rumors. Results show that our algorithm is almost 90% confident when detecting false rumors just one day after their initial broadcast.

References

[1] "Twitter," http://www.twitter.com/.

[2] "Sina weibo," http://www.weibo.com/.

[3] "Oxford english dictionary," http://www.oed.com/view/Entry/168836/.

[4] F. Yang, Y. Liu, X. Yu, and M. Yang, "Automatic detection of rumor on sina weibo," in *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics.* ACM, 2012, p. 13.

[5] S. Sun, H. Liu, J. He, and X. Du, "Detecting event rumors on sina weibo automatically," in *Web Technologies and Applications.* Springer, 2013, pp. 120–131.

[6] Y. Bao, C. Yi, Y. Xue, and Y. Dong, "A new rumor propagation model and control strategy on social networks," in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining.* ACM, 2013, pp. 1472–1473.

[7] C. Castillo, M. Mendoza, and B. Poblete, "Information credibility on twitter," in *Proceedings of the 20th international conference on World wide web.* ACM, 2011, pp. 675–684.

[8] V. Qazvinian, E. Rosengren, D. R. Radev, and Q. Mei, "Rumor has it: Identifying misinformation in microblogs," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, 2011, pp. 1589–1599.

[9] M. D. Buhmann, *Radial Basis Functions: Theory and Implementations.* Cambridge University Press, 2003.

[10] F. Bodendorf and C. Kaiser, "Detecting opinion leaders and trends in online social networks," in *Proceedings of the 2nd ACM workshop on Social web search and mining.* ACM, 2009, pp. 65–68.

[11] M. Collins and N. Duffy, "New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron," in *Proceedings of the 40th annual meeting on association for computational linguistics.* Association for Computational Linguistics, 2002, pp. 263–270.

[12] S. Vishwanathan and A. Smola, "Fast kernels on strings and trees." in *Proceedings of Neural Information Processing Systems*, 2002.

[13] A. Moschitti, "Making tree kernels practical for natural language learning." in *Proceedings of the European Chapter of the Association for Computational Linguistics*, 2006, pp. 113–120.

[14] T. Gärtner, P. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," in *Learning Theory and Kernel Machines.* Springer, 2003, pp. 129–143.

[15] M. Neuhaus and H. Bunke, "A random walk kernel derived from graph edit distance," in *Structural, syntactic, and statistical pattern recognition.* Springer, 2006, pp. 191–199.

[16] S. Vishwanathan, K. M. Borgwardt, and N. N. Schraudolph, "Fast computation of graph kernels," in *Proceedings of Neural Information Processing Systems*, vol. 19, 2006, pp. 131–138.

[17] S. Wright and J. Nocedal, *Numerical optimization.* Springer New York, 1999, vol. 2.

[18] "Sina weibo topic," http://huati.weibo.com/.

[19] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[20] K. Zhang, "NLPIR: Natural language processing and information retrieval sharing platform," 2014, http://ictclas.nlpir.org/.

[21] Z. Dong, "HowNet knowledge database," 2007, http://www.keenage.com/.

[22] P. Ekman, "An argument for basic emotions," *Cognition & Emotion*, vol. 6, no. 3-4, pp. 169–200, 1992.

[23] "Sina weibo community managing center," http://service.account.weibo.com/.

[24] F. Jin, E. Dougherty, P. Saraf, Y. Cao, and N. Ramakrishnan, "Epidemiological modeling of news and rumors on twitter," in *Proceedings of the 7th Workshop on Social Network Mining and Analysis.* ACM, 2013, p. 8.

[25] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[26] N. DiFonzo and P. Bordia, *Rumor psychology: Social and organizational approaches.* American Psychological Association, 2007.

[27] N. Okazaki, K. Nabeshima, K. Watanabe, J. Mizuno, and K. Inui, "Extracting and aggregating false information from microblogs," in *Proceedings of the Workshop on Language Processing and Crisis Information*, 2013, pp. 36–43.

[28] M. Mendoza, B. Poblete, and C. Castillo, "Twitter under crisis: can we trust what we RT?" in *Proceedings of the first workshop on social media analytics.* ACM, 2010, pp. 71–79.

[29] T. Takahashi and N. Igata, “Rumor detection on twitter,” in *Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS), 2012 Joint 6th International Conference on.* IEEE, 2012, pp. 452–457.

[30] J. Ratkiewicz, M. Conover, M. Meiss, B. Goncalves, S. Patil, A. Flammini, and F. Menczer, “Detecting and tracking the spread of astroturf memes in microblog streams. arxiv preprint,” *arXiv preprint arXiv:1011.3768*, 2010.

[31] A. Hassan, V. Qazvinian, and D. Radev, “What’s with the attitude?: identifying sentences with attitude in online discussions,” in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, 2010, pp. 1245–1255.

[32] S. Kwon, M. Cha, K. Jung, W. Chen, and Y. Wang, “Prominent features of rumor propagation in online social media,” in *Proceedings of International Conference on Data Mining*, 2013, pp. 1103–1108.

[33] M. R. Morris, S. Counts, A. Roseway, A. Hoff, and J. Schwarz, “Tweeting is believing?: understanding microblog credibility perceptions,” in *Proceedings of Computer-Supported Cooperative Work and Social Computing*, 2012, pp. 441–450.

[34] T. Gärtner, “A survey of kernels for structured data,” *ACM SIGKDD Explorations Newsletter*, vol. 5, no. 1, pp. 49–58, 2003.

[35] D. Haussler, “Convolution kernels on discrete structures,” Technical report, Department of Computer Science, University of California at Santa Cruz, Tech. Rep., 1999.

[36] K. M. Borgwardt and H.-P. Kriegel, “Shortest-path kernels on graphs,” in *Data Mining, Fifth IEEE International Conference on.* IEEE, 2005, pp. 8–pp.

[37] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[38] R. W. Floyd, “Algorithm 97: shortest path,” *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[39] S. Warshall, “A theorem on boolean matrices,” *Journal of the ACM (JACM)*, vol. 9, no. 1, pp. 11–12, 1962.

[40] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, “Graph kernels,” *The Journal of Machine Learning Research*, vol. 11, pp. 1201–1242, 2010.

[41] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, “Protein function prediction via graph kernels,” *Bioinformatics*, vol. 21, no. suppl 1, pp. i47–i56, 2005.

[42] D. Shah and T. Zaman, “Rumors in a network: Who’s the culprit?” *Information Theory, IEEE Transactions on*, vol. 57, no. 8, pp. 5163–5181, 2011.

[43] E. Seo, P. Mohapatra, and T. Abdelzaher, “Identifying rumors and their sources in social networks,” in *SPIE Defense, Security, and Sensing.* International Society for Optics and Photonics, 2012, pp. 83 891I–83 891I.