Rumor Detection on Sina Weibo by Propagation Structures

Ke Wu Shanghai Jiao Tong University Shanghai, China keer236@gmail.com

Song Yang Shanghai Jiao Tong University Shanghai, China blue-snow@sjtu.edu.cn

Kenny Q. Zhu Shanghai Jiao Tong University Shanghai, China kzhu@cs.sjtu.edu.cn

Abstract

This paper studies the problem of automatic detection of rumors on Sina Weibo, the popular Chinese social network. Traditional feature-based approaches extract features from the 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 outstanding classification accuracy of 91.3% on randomly selected Weibo dataset.

1 Introduction

Microblogging has taken over the Internet as one of the most popular forms of online social networking. Many of the microblogs carry unconfirmed, false, or even malicious 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 a widely circulated statement about an object, event or issue, which is inconsistent with facts or known scientific evidences. By this definition, a stand-alone microblog which has not been spread around is not a rumor. Some "urban legends" such as "Coke can dissolve a tooth overnight" are considered rumors here because they are proven to be false. But discussions about personality by horoscope signs are not rumors because they can not be easily refuted by existing sciences. Recent past has witnessed many incidents of online rumors causing massive public fear and social unrest. Consequently, automatic detection of rumors on social network has gathered substantial research interest.

While some researchers have worked previously on Twitter[1] (see Section 5), this paper focuses on rumor detection for Sina Weibo[2], for which only limited research has been done (Yang et al., 2012; Sun et al., 2013; Bao et al., 2013). Automatic detection of rumors is generally a hard problem, because without proper background knowledge or concrete, official evidence against, even human being cannot distinguish between the truth and the rumor. Consider the following two postings from Sina Weibo (translated from original messages in Figure 1 and Figure 2):

- *#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! Tell your children and friends that it's better to just gently blow it away. Doctors remind you that if you come across a rove beetle, never smash it with hands! @UrbanExpressOfficialWeibo @DLTV2LifeChannel @NewYouthWeb ...*
- *#Don't crush it with hands!# 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 skin infection especially if you scratch it. If you ever touch the venom of rove beetle, please clean the skin with soap water or 4% baking soda solution and seek help at local hospital immediately.*

Both messages quote the word of doctors, include vivid photos, and have comparable number of reposts, which makes them hardly distinguishable. However, the first posting is a 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

[1] http://www.twitter.com
[2] http://www.weibo.com

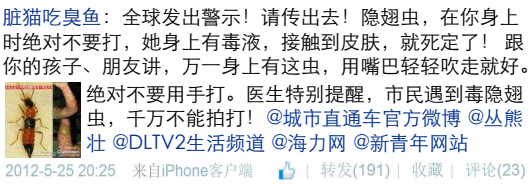

Figure 1: Rumor about Rove Beetle

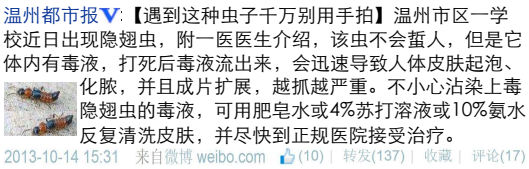

Figure 2: Non-rumor about Rove Beetle

average people, which explains why some innocent people repost 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.

Much of the previous work on 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 (Castillo et al., 2011; Qazvinian et al., 2011). 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 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 rumors can be identified not only by what the 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 3.1).
- We propose a random walk graph kernel to model the similarity of propagation trees. Results suggest that the propagation of rumors can be distinguished from non-rumors (Section 3.2).
- We combine graph kernel and radial basis function kernel, together with other novel features to build a hybrid SVM classifier (Section 3.3 and Section 3.4). Our experiments show the hybrid model achieves superior classification accuracy of 91.3% vs. those reported in previous work (Section 4).

2 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 $T = \langle \mathrm{V}, \mathrm{E} \rangle$ 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

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

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

Figure 3 illustrates the corresponding propagation tree.

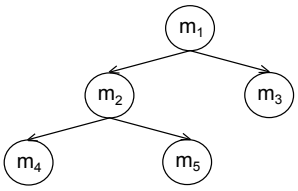

Figure 3: A Partial Propagation Tree

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

3 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)(Buhmann, 2003). 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.

3.1 Propagation tree

For the purpose of the random walk graph kernel, we enrich the propagation tree in Figure 3 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[3] will be used in the graph kernel computation.

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

We divide the users into two types: *opinion leaders* and *normal users*. Opinion leaders are those influential users whose opinions dominate their followers(Bodendorf and Kaiser, 2009). A user is considered an opinion leader if

$$\frac{\#\text{ of followers}}{\#\text{ of friends}} > \alpha \qquad (1)$$

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[4], with a triple $\boldsymbol{v_j} = (\theta(s), \theta(d), \theta(t))$, where s is approval score of m_j which indicate approval or agreement, while d is the doubt score of m_j which indicate doubts and suspicion, and t is the overall sentiment score in m_j. We defer the computation of s, d and t to Section 3.3. θ is a damping function defined by

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

where t is the time stamp difference in days between the original message and m_j, and ρ is a parameter between 0 and 1. The shorter the time period, the more intense the response is. Figure 4 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.

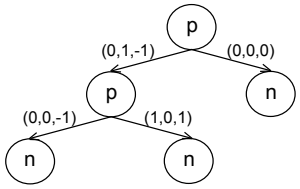

Figure 4: Example of Labeled Propagation Tree

Our intuition is that patterns can be discovered from the labeled propagation tree which helps distinguish rumors from non-rumors. For example, Figure 5 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 rumor (a) is reposted and supported by many opinion leaders at first before normal users take over the propagation; conversely the non-rumor (b) is initially reposted by a majority of normal users. This shows

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

that the influence of multiple opinion leaders can quickly create a "hype" which is followed by ordinary users.

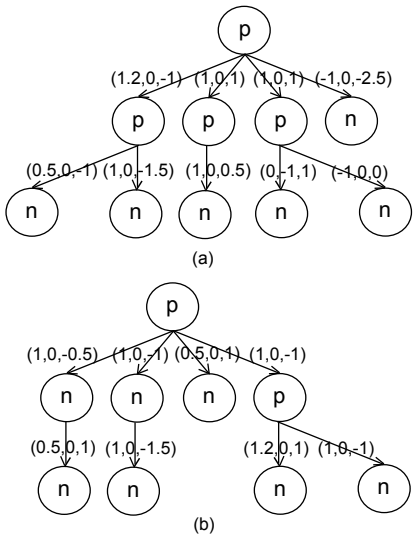

Figure 5: Tree of Rumor and non-Rumor

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 5(b) to Figure 6:

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.

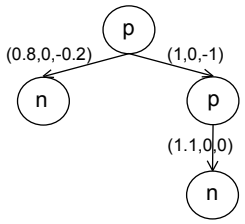

Figure 6: Simplified Propagation Tree

3.2 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 calculate the similarity of trees based on the subset tree (SST)(Collins and Duffy, 2002) or subtree (ST)(Vishwanathan and Smola, 2002). While these kernels prove to be useful in natural language processing(Moschitti, 2006), they can not be used for our problem because they consider two nodes are 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 (Gärtner et al., 2003) to calculate the similarity of trees. Because the labels of edges in propagation tree are not discrete values but continuous vectors, we modified original random walk kernel so that the kernel is applicable to graphs with continuously labeled edges(Neuhaus and Bunke, 2006).

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

$$V_\times = \{(v, v') \in V \times V' : label(v) = label(v')\}$$
$$E_\times = \{((u, u'), (v, v')) \in V_\times^2 : (u, v) \in E \wedge (u', v') \in E'\}$$

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

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

The kernel function k measuring the similarity of nodes (u, u') and (v, v'), which is given by

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

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

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

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

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

for x, then we compute $e^t x$. In the first step, we use conjugate gradient (CG) method to solve the linear system (Vishwanathan et al., 2006). 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, so the CG solver can be sped up significantly (Wright and Nocedal, 1999). To solve the linear system, CG takes $O(n^4 i)$ where i is the number of iterations.

3.3 Features

We extract a total of 23 features from Sina Weibo data to build a vector for RBF kernel. Some of the features have been proposed previously (Yang et al., 2012; Castillo et al., 2011; Qazvinian et al., 2011) 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. 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 1 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 detail.

Topic Type feature refers to the topic types of the original message. We assume that the message belongs to one or more topics. Since Sina Weibo has an official classification of 18 topics[5], we train a Latent Dirichlet Allocation (LDA) (Blei et al., 2003) model which returns an 18-topic distribution for message m_i:

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

where s_j is the probability of m_i belonging to topic j.

Search Engine feature refers to the number of results returned by web search engine when searching for the original message and the keyword "rumor". Due to the limitation of query length imposed by Google, we divide the message into word sequences of ql_{max} characters [6] each. Then each sequence q_i is searched in Google by querying "intext:q_i intitle: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 verified users into refined types. 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 and has stop words removed. After that, we calculate the sentiment score of each message based on the sentiment lexicon of HowNet (Dong, 2007). The average sentiment score is

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

where NP_i is the number of positive words and NP_i 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 in Chinese and hence becomes a negative word. Vice versa for negative words. **Avg doubt, Avg surprise** and **emoticon features** as well as the approval score s and doubt score d in Section 3.1 are calculated similarly except the lexicons used are specially for these categories.

[5] `http://huati.weibo.com`.

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

Table 1: 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 posting time and registration time
	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 posting time
	NUM OF FRIENDS	The number of people the author is following at posting time
	NUM OF POSTED MESSAGES	The number of messages posted by the author at posting time
	REGISTRATION TIME	The time of author registration
	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 score of reposts time

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

$$\frac{1}{n}\sum_{i=1}^{n} 2^{-(t_i - t_0)}$$

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.

3.4 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 rumor or non-rumor. The RBF kernel for this binary classifier is

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

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 section 3.2, we define the random walk kernel to calculate the similarity of two trees

$$K_\times(T, T') = \boldsymbol{e}^T(\boldsymbol{I} - \lambda A_\times)^{-1}\boldsymbol{e}$$

where $A_\times$ is the adjacency matrix of direct product of T and T'. In order to normalize the kernel function, we divide $K_\times(T, T')$ by nn' where n and n' are the numbers of nodes in T and T':

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

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

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

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

4 Evaluation

The evaluation of the hybrid SVM classifier is composed of 4 phases: collection of Weibo data and annotation of the original messages; tuning parameters of the SVM model; evaluation of different features; and finally comparison with competing methods on end-to-end results.

4.1 Dataset

We collect a set of known rumors from Sina community management center (`http://service.account.weibo.com`), which deals with reporting of issues including various

misinformation which we regard as certified rumors. There are 11466 reported rumors between 2012/05/28 and 2014/04/11. Out of these, we keep rumors that have at least 100 reposts which leaves us 2601 rumors along with all their reposts up to 2014/04/11. We also randomly select 5000 other Weibo original messages and their reposts from Sina Weibo API. We manually filtered out messages with fewer than 100 reposts as well as rumors to form a set of 2536 non-rumors. Each message or repost contains links to the author profile information such as age, gender, number of followers and friends which we also crawled using the Weibo API.

At the end of this phase, our labeled data set consists of 2601 rumors, 2536 non-rumors[7] and about 4 millions users involved in these messages. Of these 500 rumors and 500 non-rumors (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.

4.2 SVM parameter tuning

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 ($2\sigma^2 = 3, \rho = 0.1$) obtained through 10-fold cross validation. The SVM classifier is implemented based on LIBSVM(Chang and Lin, 2011). To suppress the effects of the vector kernel, we set $\beta = 1$ in this experiment. The results of experiment are shown in Figure 7.

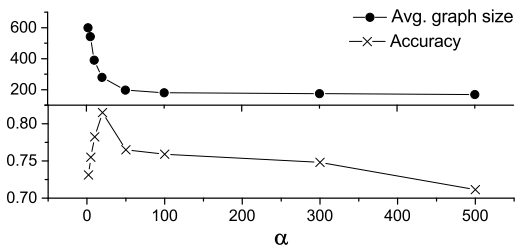

Figure 7: Average number of vertexes in product graph and classification accuracy vs. α

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 a SVM classifier ($\gamma = 2^{-11}, cost = 2^{13}, 2\sigma^2 = 3, \rho = 0.1$, obtained through 10-fold cross validation) and record its accuracy. The results of experiment are shown in Figure 8.

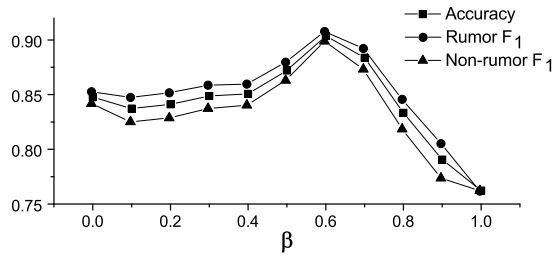

Figure 8: Accuracy, F_1 measure for rumors and non-rumors vs. β

Results show 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 graphs). In the following experiments, we set $\beta = 0.6$.

4.3 Feature selection

To investigate the effectiveness of our new features, we train several SVM classifiers using different subset of the features. We also train a classifier without graph kernel function to show its usefulness. The small data set (4137 messages) is divided into training set and test set with a ratio of 2:1. For each subset of features, we train a SVM classifier ($\gamma = 2^{-11}, cost = 2^{13}, 2\sigma^2 = 3, \rho = 0.1$) on the training set and test the classifier on the test set. The results of experiment are shown in Table 2. Here (-)X means whole set of features except feature X, and ALL means all features with hybrid kernel.

The results show that the inclusion of the graph kernel is the deal changer here, improving the accuracy by 0.056 which is the largest among all features. This clearly indicates that the explicit representation of propagation tree patterns better models the rumors and non-rumors. Results also suggest that TOPIC TYPE is the most effective among all ordinary features. This is because rumors tend

[7]The labeled data set of the original messages only is available at `http://adapt.seiee.sjtu.edu.cn/~kzhu/rumor/`.

Table 2: Impact of features

	Accuracy	R F_1	NR F_1
(-)TOPIC TYPE	0.865	0.872	0.857
(-)SEARCH ENGINE	0.880	0.892	0.884
(-)USER TYPE	0.894	0.897	0.890
(-)AVG SENTIMENT (-)AVG DOUBT (-)AVG SURPRISE	0.872	0.878	0.865
(-)AVG EMOTICON	0.887	0.891	0.884
(-)REPOST TIME SCORE	0.892	0.896	0.888
(-)GRAPH KERNEL	0.848	0.849	0.846
ALL	**0.904**	**0.907**	**0.900**

to concentrate on a few sensitive topics, such as missing persons or healthy 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 rumors.

4.4 End-to-end comparison

We compare our hybrid SVM classifier ($\gamma = 2^{-11}, cost = 2^{13}, 2\sigma^2 = 3, \rho = 0.1$) with two other state-of-the-art rumor detection algorithms(Castillo et al., 2011; Yang et al., 2012). Castillo's J48 decision tree is implemented using the 16 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. Besides, we also train a SVM classifier with only graph kernel ($\beta = 1$) as baseline to evaluate the classification performance of graph kernel (Graph). For this comparison, we compute the accuracies, precisions, recalls and F_1 measures by 3-fold cross validation on the big data set.

Table 3: Comparison of different methods

Methods	Hybrid	Castillo	Yang	Graph
Accuracy	**0.913**	0.854	0.772	0.770
R precision	**0.905**	0.853	0.773	0.773
R recall	**0.922**	0.854	0.776	0.763
R F_1	**0.913**	0.854	0.774	0.768
N precision	**0.920**	0.853	0.770	0.766
N recall	**0.903**	0.854	0.768	0.776
N F_1	**0.912**	0.854	0.769	0.771

Table 3 shows the result. Overall, the table demonstrates that our approach outperforms the other competitions by large margins across all measures. 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 rumors from non-rumors.

5 Related Work

Previous works on rumor detection for microblogging service (either on Twitter or Weibo) have 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 features explored in the literature, and then compare various classification methods.

5.1 Features

We divide existing features for rumor detection into 4 types.

Linguistic Features pertain to the microblog message text (Qazvinian et al., 2011). They range from simple features such as message length, punctuations, letter case, whether URLs or hashtags are included (Ratkiewicz et al., 2010), types of emoticons used and POS tags (Hassan et al., 2010) to more advanced semantic features such as sentiment scores (Castillo et al., 2011; Qazvinian et al., 2011) and opinion words (Kwon et al., 2013). Previous research (Yang et al., 2012; Kwon et al., 2013) shows that not all these features are effective for 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 3.3). 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 (Morris et al., 2012), 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. (Castillo et al., 2011) 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 works (Castillo et al., 2011; Qazvinian et al., 2011; Mendoza et al., 2010) focus on the numeric summary of such graph struc-

tures, 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 works treat each node (either message or user) equally and hence only derive such generic statistics. Recently, some researchers (Jin et al., 2013; Bao et al., 2013) adapted the epidemiological models to 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 (Castillo et al., 2011). More advanced models use these times to detect sudden spikes in the volume of responses or periodicity of such spikes (Kwon et al., 2013). Researchers also use time to calculate rates of population change among the compartments in epidemiological models (Jin et al., 2013; Kwon et al., 2013). 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 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.

5.2 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. 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. (Kwon et al., 2013) 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.(Okazaki et al., 2013) 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.

Finally, research that adapts epidemiological models (Jin et al., 2013; Bao et al., 2013) to 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 rumors from non-rumors.

6 Conclusion

This paper studies the problem of automatically detecting rumors on China's popular 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. Our results show that the repost pattern of rumor and non-rumor is very different, which makes the random walk graph kernel very useful in detecting rumors. The combination of random walk kernel and RBF kernel performs better than each of them alone, with a superb accuracy of 0.913.

References

Yuanyuan Bao, Chengqi Yi, Yibo Xue, and Yingfei Dong. 2013. 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*, pages 1472–1473. ACM.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Freimut Bodendorf and Carolin Kaiser. 2009. Detecting opinion leaders and trends in online social networks. In *Proceedings of the 2nd ACM workshop on Social web search and mining*, pages 65–68. ACM.

Martin D. Buhmann. 2003. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684. ACM.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Michael Collins and Nigel Duffy. 2002. 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*, pages 263–270. Association for Computational Linguistics.

Zhendong Dong. 2007. HowNet knowledge database. http://www.keenage.com/.

Thomas Gärtner, Peter Flach, and Stefan Wrobel. 2003. On graph kernels: Hardness results and efficient alternatives. In *Learning Theory and Kernel Machines*, pages 129–143. Springer.

Ahmed Hassan, Vahed Qazvinian, and Dragomir Radev. 2010. 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*, pages 1245–1255. Association for Computational Linguistics.

Fang Jin, Edward Dougherty, Parang Saraf, Yang Cao, and Naren Ramakrishnan. 2013. Epidemiological modeling of news and rumors on twitter. In *Proceedings of the 7th Workshop on Social Network Mining and Analysis*, page 8. ACM.

Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. 2013. Prominent features of rumor propagation in online social media. In *ICDM*, pages 1103–1108.

Marcelo Mendoza, Barbara Poblete, and Carlos Castillo. 2010. Twitter under crisis: can we trust what we RT? In *Proceedings of the first workshop on social media analytics*, pages 71–79. ACM.

Meredith Ringel Morris, Scott Counts, Asta Roseway, Aaron Hoff, and Julia Schwarz. 2012. Tweeting is believing?: understanding microblog credibility perceptions. In *CSCW*, pages 441–450.

Alessandro Moschitti. 2006. Making tree kernels practical for natural language learning. In *EACL*.

Michel Neuhaus and Horst Bunke. 2006. A random walk kernel derived from graph edit distance. In *Structural, syntactic, and statistical pattern recognition*, pages 191–199. Springer.

Naoaki Okazaki, Keita Nabeshima, Kento Watanabe, Junta Mizuno, and Kentaro Inui. 2013. Extracting and aggregating false information from microblogs. In *Proceedings of the Workshop on Language Processing and Crisis Information*, pages 36–43.

Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1589–1599. Association for Computational Linguistics.

Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Goncalves, Snehal Patil, Alessandro Flammini, and Filippo Menczer. 2010. Detecting and tracking the spread of astroturf memes in microblog streams. arxiv preprint. *arXiv preprint arXiv:1011.3768*.

Shengyun Sun, Hongyan Liu, Jun He, and Xiaoyong Du. 2013. Detecting event rumors on sina weibo automatically. In *Web Technologies and Applications*, pages 120–131. Springer.

S.V.N. Vishwanathan and A.J. Smola. 2002. Fast kernels on strings and trees. In *Proceedings of Neural Information Processing Systems*.

S.V.N. Vishwanathan, Karsten M Borgwardt, and Nicol N Schraudolph. 2006. Fast computation of graph kernels. In *NIPS*, volume 19, pages 131–138.

SJ Wright and J Nocedal. 1999. *Numerical optimization*, volume 2. Springer New York.

Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. 2012. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 13. ACM.