Rumor detecting

Masterarbeit
zur Erlangung des akademischen Grades
M.Sc. Internet Technologies and Information Sytems

CHENG LI

Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für verteilte Systeme
Fachgebiet Wissensbasierte Systeme
Forschungszentrum L3S

Hannover
13.11.2016

Examiner I	Prof. Dr. Wolfgang Nejdl
Examiner II	Dr. Eirini Ntoutsi

Abstract

In this thesis, we propose a approach of identifying rumors in Twitter.

Titter is a mircoblogiging service that which are used by millions users. Users can publish and exchange information with short tweets whatever when and where. This makes it a ideal media for spreading breaking news and false rumors.

So automatic detecting rumors on social media has become a trending topic. But early researches mostly focused on rumors during one or several (??) events like earthquake or terrorist attack. But in our work, we more focus on general rumors.

And most of previous work for rumor detection focused on static features like the content of tweets or propagation features, and they ignored that those features change during the information's propagation over time.

we use Dynamic Series-Time Structure (DSTS)(wenxian) to capture the temporal features. And we add Spike Model, SIS Model and SEIZ Model as time series features. To improve the time series model's performance at early stage of the event we develop a single tweet's credibility scoring model which only using features which can be extracted from single tweet on the Twitter interface.

Our experiments using the events from Twitter and our model demonstrates better performance on detecting rumors.

Contents

1 Introduction

Twitter is a mircoblogging service which are used by millions users. Users can publish and exchange information with short tweets within 140 characters. It is cheap and can be accessed though several like website, email or mobile phone. That makes it a ideal media for spreading breaking news and false rumors. A study by the Pew Research Center showed that the people in USA under age of 30 consider Internet as the major resource of news and Internet became the second important media overall(wenxian).

But these advantages make Twitter which became one of the most importance resources of breaking news, at the same time becomes into a ideal media for spreading unverified information. On Twitter everyone can be a journalist and publish news or rumors without any substantiation which must be done by traditional journalists before news' publishing.

Rumor could be defined as a statement whose truth value is unverified or deliberately false(wenxian). And they could be harmful to the government, market and society. One case is some hacked accounts spread a rumor about Obama had been injured in white house. The S&P crashed and wiped off 130 Billion dollars of stock value (wenxian).

So a method of detecting rumors on Twitter that could detect rumors in the early stage of propagation can be very useful.

1.0.1 Contributions

In this thesis, we make the following contributions:

- We develop a model of classification of single tweet with high credibility or low credibility. We call it single tweet's creditability scoring model. Considering it could be set up online for the early rumor events detection in the further and it should response as quick as possible. So we use only the features which can be extracted from one tweet in the Twitter's interface. We test 2 models. One is random decision forest with handcrafted features. Because the features are limited, it gets only 64% accuracy. Second model is two layer RNN model. The result of this model we called it credit score.

- We develop a time series model for detecting rumor events. We used Dynamic Series-Time Structure (DSTS)(wenxian) to capture the changes of features over time. And we tested 3 time series model as feature: modified Spike Model(wenxian), SIS model and SEIZ model(wenxian). We add the results of credit scoring model as a feature into time series model to improve the performance in the early stage. And we approved that credit score is one of the best feature in the rumor detection task.
- We study how features changes over time during the spreading of rumors.

1.0.2 Thesis Outline

(xiugai) The rest of this these is organized as follows: In Chapter **??** we try to understand the research practices of humanities scholars when working with web archives. We also seek to find problems faced by the scholars when accessing web archives using keyword search. Based on our findings we introduce Historical Query Intents(HQIs) in Chapter **??**. In the same chapter we also motivate and develop a novel retrieval model suited for HQIs. The experiments to test the performance of this model against its competitors is covered in the same chapter. The penultimate Chapter **??** details the architecture of the system that was built to demonstrate the efficacy of the retrieval models to users. Finally, in Chapter **??** we add some concluding remarks and describe future work.

2 Background and Related Work

2.1 Twitter

Twitter is a mircoblogging service. Now there are more than 140 million active users[1]. User can publish short messages within 140 characters aka tweets.

2.1.1 Retweet

A retweet is re-posting of a tweet by other users. One way of retweet is using RT at the beginning of the tweet which is retweeted. The other way is using "retweet button" which is officially launched by Twitter after 2015. The difference between these two retweet is tweets are retweeted by "retweet button" can't be searched by Twitter's searching interface, but manually retweets with RT keyword can. So in our work retweet means the tweets are retweeted manually. The number of how many times of this tweet has been retweeted is showed behind it.

2.1.2 Mentions

Mentions are in form like "@username" which are added in the text of tweet. The users are mentioned will receive the notification of this tweet on their homepage.

2.1.3 Hashtags

Mentions are in form like "#topic". It means this tweet belongs to some topics.

2.1.4 Favorite

Favorites means how many users like this tweet. It is showed on the interface of Twitter

[1] https://blog.Twitter.com/2012/Twitter-turns-six

2.1.5 Verified User

Verified User means this account of public interest is authentic by Twitter. It is showed by a blue icon behind the name of the poster.

2.1.6 Followers

The followers of a user are accounts who receive this user's posting. The total number of followers can been seen in the profile of poster.

2.1.7 Following

The following are other accounts who follow this user. The total number of following can been seen in the profile of poster.

2.1.8 Twitter API

Twitter API is provided by Twitter[2] for developer. But the search API only return a sampling of recent Tweets published in the past 7 days[3]. We need the full stories of the events, so we crawled the data directly from the searching interface[4].

2.2 Credibility of Tweets

Titter has been used for reporting breaking news when emergency events happen like disaster (wenxian). But the people are likely to trust the news which post on traditional news website more than the news with same headline but posted on twitter. And Thomson's work shows that different crowds of people trust tweets basing on different kinds of sources.[23].

2.3 Definition of Rumor

The definition of rumor in our work is unverified information spreading on Twitter over time. It is a set of tweets including the the sources, spreaders, retweets, doubting tweets and the denying tweet.

2.3.1 Rumor Event

If a rumor didn't widely spread and it could be harmless. So we more focused on the rumors which are widely spread and contain one or more bursty pikes during propagation. We call it "rumor event".

[2] https://dev.twitter.com/overview/api
[3] https://dev.twitter.com/rest/public/search
[4] https://twitter.com/search-home

2.3.2 Time Period of an Event

The beginning of a rumor event is hard to define. As far as I know there is related work which give us a approach to define the beginning of one rumor event. One reason is a rumor may be created serval years ago and kept exiting in Twitter, but it didn't attract the people's attention. However it can be triggered by other events and suddenly quickly spread as a bursty event.

For example, a rumor[5] claimed that Robert Byrd was member of KKK. This rumor has been circulating in Internet for a while, as shown in figure 2.1 that almost every day there are several tweets talking about this rumor. But this rumor was triggered by a picture about Robert Byrd kissing Hillary Clinton in 2016 and twitter users suddenly notice this rumor and this rumor was bursty spread. So what we are really interested in is the hours around the bursty pike.

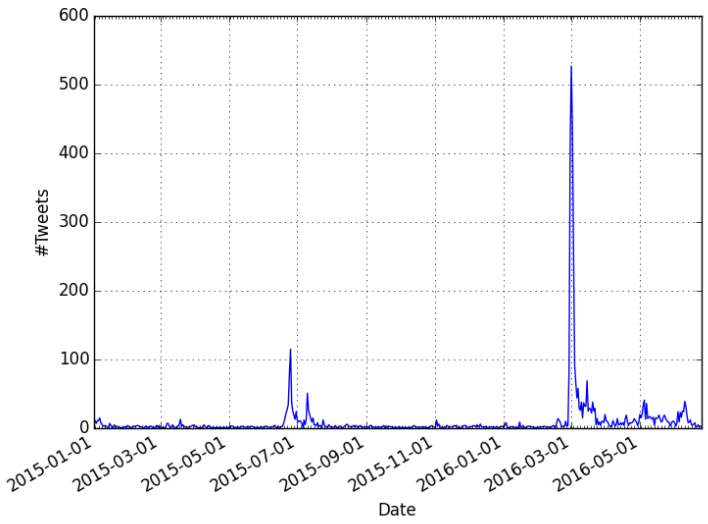


Figure 2.1: large scale tweet volume of event Robert Byrd

We defined the hour which has the most tweet's volume as t_{max} and we want to detect the rumor event as soon as possible before its burst, so we define the time of the first tweet before t_{max} within 48 hours as the beginning of this rumor event, marked as t_0. And the end time of events we defined as $t_{end} = t_0 + 48$. We show the tweet volume in figure 2.2 of the above rumor events after defined 48 hours time period.

[5] http://www.snopes.com/robert-byrd-kkk-photo/

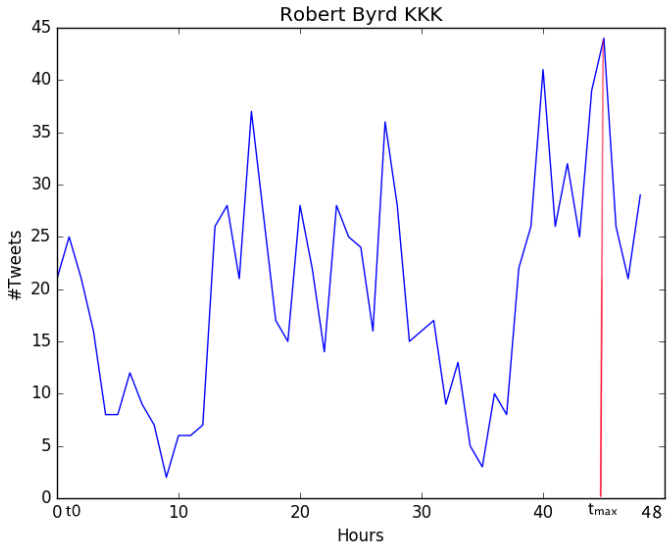


Figure 2.2: tweet volume of the rumor event of Robert Byrd after selected time period

2.4 Machine learning

2.4.1 Machine learning Overview

Machine learning covers vast numbers of algorithms and has been successful applied in different field. The challenges of ML are finding the best model which is suitable for this task, fitting the parameters and selecting the features.

Normally we split the ML methods into Supervised learning, Unsupervised learning and Reinforcement learning[19].

The supervised learning is the most popular method of ML. It needs a set of inputs and a set of desired outputs. And the algorithm will learn to produce the correct output based on the new input.

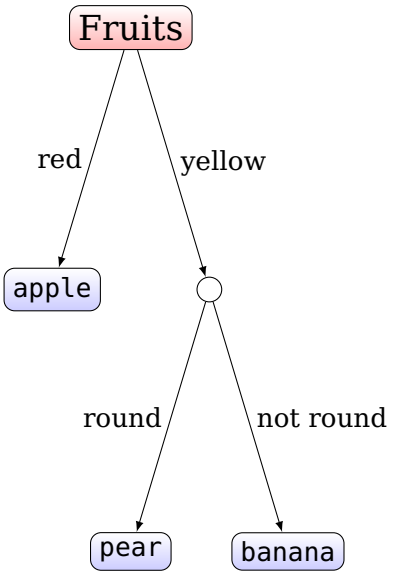
The unsupervised learning needs a set of inputs but no outputs. The algorithm will generate the outputs like clusters or patterns. The unsupervised learning task can be used for example when people can't label the outputs.

2.4.2 Decision tree

2.4.3 Random Forest (RF)

Classification is a supervised data mining technique. Our work can be considered as a classification task. And the classification model is random forest.

RF is an algorithm of supervised learning which developed by Leo Breiman [4]. It's built by a set of classification trees [5]. Each tree is trained by a small bootstrap sample of training set and while prediction each tree votes one single candidate. RF generates the result of prediction By taking the majority vote.



For example we got task to classify pears and apples. We have the features are whether the fruit round, whether the fruit has seeds, whether the fruit red and whether the fruit is juicy. We build up a 3 trees random forest as the graph 2.3. Tree 1 randomly takes the subset of the features red and seeds and votes to apple. Tree 2 randomly takes the subset of the features red and seeds and votes to pear. Tree 3 randomly takes the subset of the features round and red and votes to apple. The most vote is apple, so the output of RF is apple.

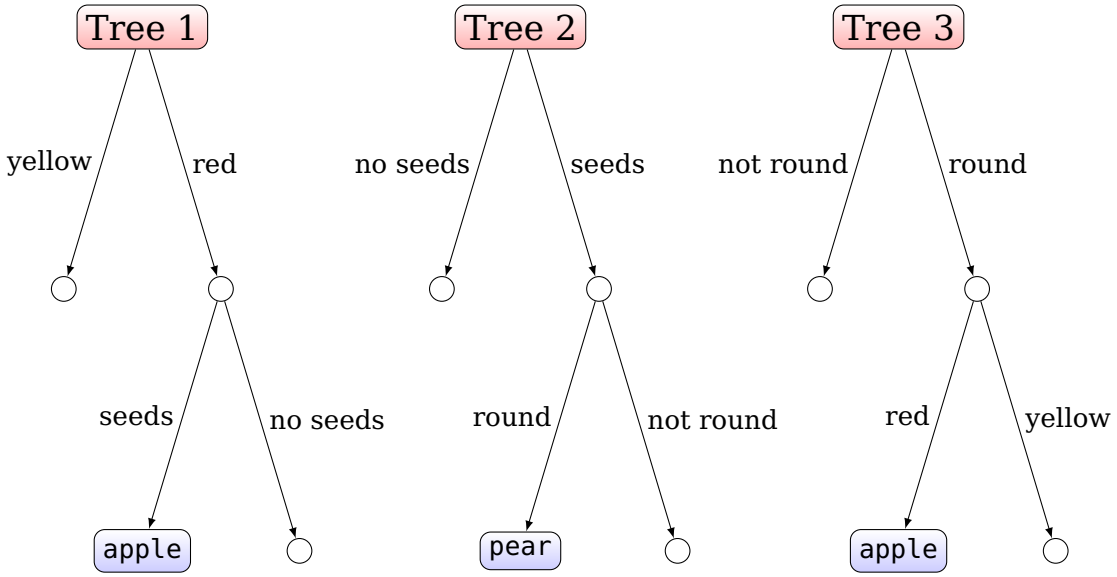


Figure 2.3: An example of random forest with 3 trees

Because RF uses a random subset of features instead of the best features in every node, so it can avoid the overfitting[4].

Another benefit of RF is that it can return the **features' importance**. RF is built

up by a subset of training data. If there are N trees, RF will take N times bootstrap sampling. So some features we didn't select to use for training the model. The data we didn't selected we call it out-of-bag (OOB) data. In the above example the feature whether the fruit is juicy didn't join the construction the Trees, so it is a OOB data. These data can be use for validating the model and the output is OBB error $E_{oob}(G) = \frac{1}{N}\sum_{n=1}^{N} err(y_n, G_n^-(X_n))$ with X_n are features only in OOB. At last we get the importance of feature by permuting one feature to random numbers. $importance(i) = E_{oob}(G) - E_{oob}^p(G)$ where $E_{oob}^p(G)$ is the OBB error after permuting a feature. The more performance drop down, that means the more important this feature is. We use this method to measure the performance of features and evolute the models later.

2.4.4 Recurrent neural network

A Recurrent neural network (RNN) is one of of class of neural network 2.4. The different between RNN and other neural network is there are connections between the hidden layers **??**, that makes the sequence of the input can also influence the network. For example people understand the words in one sentence based on the previous words, the normal neural network will ignore the previous inputs.

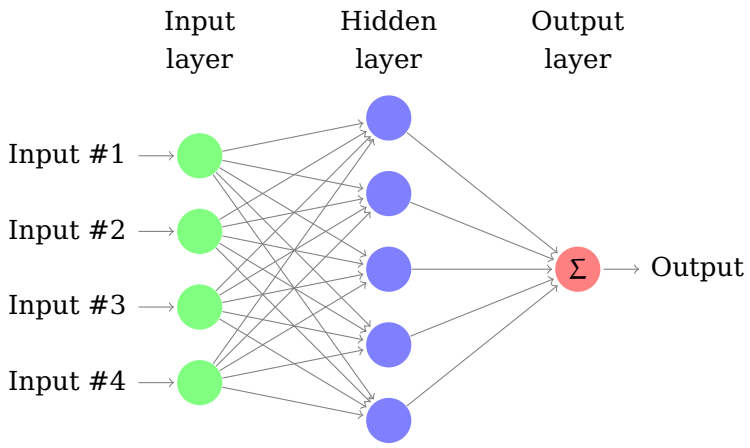


Figure 2.4: An example of 1 layer neural network

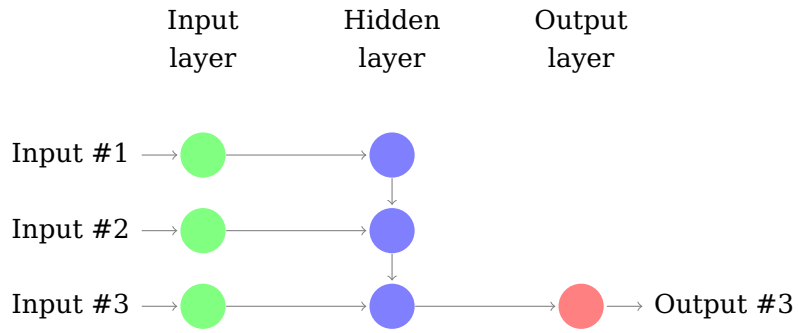


Figure 2.5: multi-input single-output Recurrent neural network

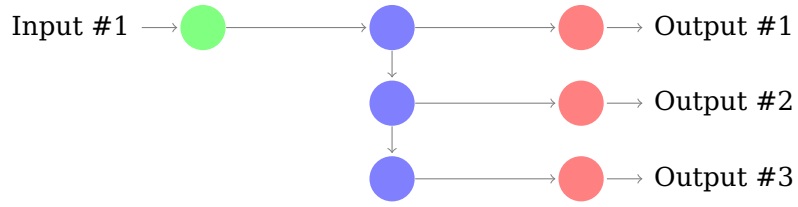


Figure 2.6: single-input multi-output Recurrent neural network

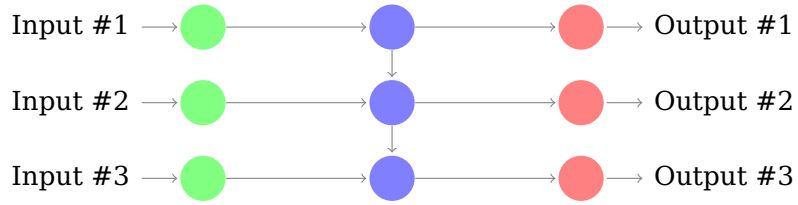


Figure 2.7: multi-input multi-output Recurrent neural network

2.5 Spark

2.6 tensorflow

2.7 LM fitting

2.8 Related Work

People research on Twitter for a long time and there are a lots directions to study this complex social network like event detection[9], spam detection [1] [24] or sentiment detection[3]. These work are similar with our task rumor detection but still contain many differences.

People first studied on rumors in psychology area for years [2] [21]. But Twitter becomes an important platform for people sharing and exchanging information including rumors, the detection of misinformation on twitter turns to be a trending researching topic.

Researcher first began from studying rumor spreading during several special events

like natural disasters[18] [22] or terrorist attacks [20]. But these results are not general enough to other types of rumors. Our data includes sports, politic, entertainment, also natural disasters and other kinds of rumors.

Carlos Castillo researched the information credibility on Twitter[17] [6][7]. But his work is more about people opinion (trustful or not) to a tweet not the credibility of tweet itself. But it still a good start of resolving problem. Lots of other works are based on Castillo's work [26] [13] and used different set of features.

There are rencent researches based on the propagation of rumor on Sina Weibo[26] [25], but twitter doesn't contain such detail of propagation like Weibo, so these work are useless for us.

Liu's work [25] focused on one type of rumor "the false photos" on social network not the general type of rumors.

Xiaomo claimed their system is the first real time rumor debunking system on Twitter [13]. But their work are similar with above other previous works, they use the static features of rumors and ignored the feature's changing over time during the event spreading on the social network which can be seen in figure 2.8 and 2.9. The fraction of tweets containing url with top 5000 domain and the fraction of poster living in large city which are both constantly changing. But one feature called "crowd wisdom" is an new idea and we use it in our work.

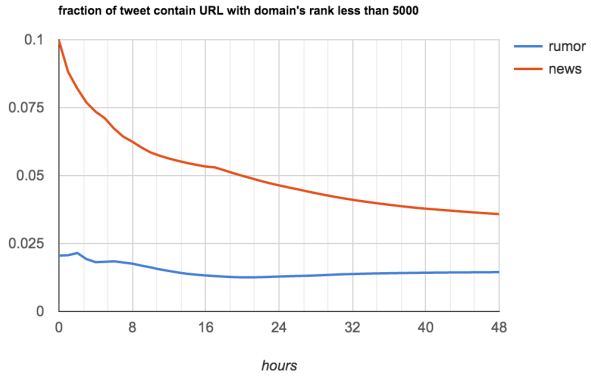


Figure 2.8: The fraction of tweets containing url with top 5000 domain

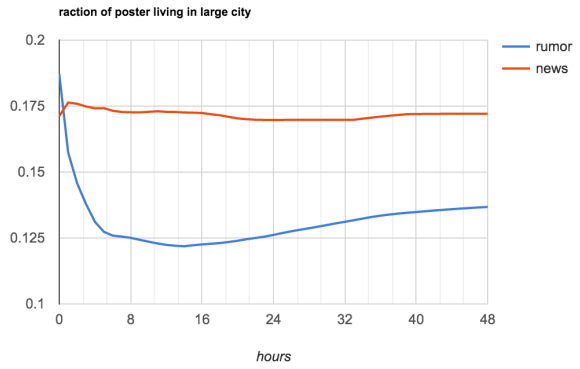


Figure 2.9: The fraction of poster living in large city

Other researchers used propagation models like SpikeM [11] and SEIZ models [8] to capture the tweet's volume changing over time, but they didn't use other features into time series model. And these features can't show the effect in the early stage of rumor spreading. We can provide it in the last section(wenxian).

J Ma used Recurrent Neural Networks for rumor detection [14], as the same disadvantage of the other Neural Networks models, the process of model's training is black box to us, so we can't know way we got this result. In our case we can't get the evidences to convince people why the detected event is more likely to be a rumor or rather than a breaking news.

Another work from J Ma used a time series model called Dynamic Series-Time Structure [15] to capture the variation of features. But they only used the features of social content and they didn't further explain how the performances of features change over time or how do they improve the performance of the model in first few hours after the events beginning. We use their Series-Time Structure with our extended data and we will show how do these features change during time.

3

Data Collection

We collected rumor stories from a rumor tracking website **snopes.com**. We crawled 4300 stories from the website and we manually constructed 130 queries. The approach of constructing queries is mainly following the work(wenxian). The regular expression of a query is $(Object\&Subject\&Description(Description1||Description2||...))$ for example a story is about Obama removing a flag in pearl harbor. Object is Obama, subject is flag and its synonym like flags, flagpole. Description is remove and its synonym removes, removed, removal, removing, "token down" or a url about this rumor "Departed.co". In this case there is a proper noun "pearl harbor" is also useful. Finally we transfer the regex to Twitter's query: obama (flag OR flags OR flagpole) (remove OR removes OR removed OR removal OR removing OR "token down" OR "Departed.co") pearl harbor.

We collect news stories for the data of (wenxian). We crawled average 50% of them and we selected events with the most coverage and minimal 100 tweets.

After we crawled and parsed the whole timeline of an events. We detect the 48 hours time period of the burst in the way we mentioned in section 2.3.2. We crawled the homepage of poster of the tweets in the event time period total 133,396 users. We also extracted 11,038 domains which are contained in the tweets in the 48 hours time period and we crawled these domains' catalogs in bluecoat.com [1], ranks in alexa.com[2] and WOT score in [3].

We use Beautiful Soup as the html parsing library to parse the Twitter timeline pages and the users' homepage [4]. Beautiful Soup is a Python library for pulling data out of HTML and XML files. For increasing the speed of parsing html and extracting features from raw data, we use Spark[5] technology, because it can simply manage multithread and its MapReduce in memory technology makes the calculation much quicker.

[1] http://sitereview.bluecoat.com/sitereview.jsp#/?search=bbc.com
[2] http://www.alexa.com/siteinfo/bbc.com
[3] https://www.mywot.com/en/api
[4] https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[5] http://spark.apache.org/

4 Single Tweet's Creditability Scoring Model

Most of the pervious researches are focusing on event level rumors and claims taht the task of classification for individual tweet is not reliable [13] [15] [27], because a single tweet is to short to contains engough features. But Carlos Castillo designed a single tweet's creditability rank system Tweetcred [7]. So we think it is still enable to build up a single tweet creditability model. And we were inspired by the (wenxian) work that we may use nn() without handcrafted features to build up the single tweet's creditability model which in later experiments outputs a better performance.

4.1 Single Tweet's Creditability Model with handcrafted features

First we follow Castillo's [7] idea to implement a single tweet's creditability model with handcrafted features and we test it with decision trees, decision forest and SVM. But the performance of each model are not good enough.

4.1.1 Features

We use a collection of features major from Castillo's Tweetcred system[7] totally 27 features in table 4.1. These features can be extracted directly from Twitter interface without third part website.

Text Features

The Text features capture the content of the text of the tweets. There are 16 Text features. **Sentiment Features** are included in text features. We used the python natural language Toolkit (nlTK) [1] to analyze the tweets' sentiment and extract the features: the NumPositiveWords, NumNegativeWords and Polarityscores. Polarity scores is a float for sentiment strength of one tweet[2] $Polarity\_scores = \frac{1}{N}\sum_{0}^{n} Polarity(token_n)$.

[1] http://www.nltk.org/
[2] http://www.nltk.org/api/nltk.sentiment.html

User Features

We selected total 5 features of the poster. These features are extracted directly from the twitter interface as in figure 4.1. ReputationScore is defined as the ratio between #Friends over # Followers. $ReputationScore = \frac{\#Friends}{\#Friends+\#Followers}$.

Figure 4.1: Sample of Users' Information on Twitter Interface

Twitter Features

Twitter Features are the features of twitter's special functions. It includes Hashtag, Mention, Number of URLs, Number of retweeted and whether this tweet is retweet (contains RT as keywords or "*QuoteTweet-innerContainer u-cf js-permalink js-media-container*" as the CSS class of the tweet in html).

Category	Feature	Description
Twitter Features	Hashtag	Whether the tweet contains #hashtag
	Mention	Whether the tweet mentions others @user
	NumUrls	# url in the tweet
	Retweets	How many times this tweet has been retweeted
	Isretweet	Whether this tweet is retweeted from others
Text Features	LengthofTweet	The length of tweet
	NumofChar	# of individual characters
	Capital	Fraction of characters in Uppercase
	Smile	Whether this tweet contains :->, :-), ;->, ;-)
	Sad	Whether this tweet contains :-<, :-(, ;->, ;-(
	NumPositiveWords	# of positive words
	NumNegativeWords	# of negative words
	Polarityscores	polarity scores of the Tweet
	Via	Whether this tweet contains via
	Stock	Whether this tweet contains $
	Question	Whether this tweet contains ?
	Exclamation	Whether this tweet contains !
	QuestionExclamation	Whether this tweet contains multi Question or Exclamation mark
	I	Whether this tweet contains first pronoun like I, my, mine, we, our
	You	Whether this tweet contains second pronoun like U, you, your, yours
	HeShe	Whether this tweet contains third pronoun like he, she, they, his, etc.
User Features	Followers	# of followers
	Friends	# of friends
	Tweets	# of tweets which are posted by this user
	Description	Whether this user has description
	Verified	Whether this user is a verified user
	ReputationScore	Ratio between #Friends over (# Followers + #Friends)

Table 4.1: Features for Single Tweet's Creditability Scoring Model

4.1.2 Classification Methods

Most of existing researches use SVM, Random Forest and DT. We test our data also with these 3 models. We implement these 3 model with scikit-learn library[3]. We shuffled the 260 events and split them into 10 subset, we uses them for 10 times cross-validation. We show the parameters after optimization for each model them in table 4.2.

Model	Parameters	Value
Random Forest	Number of Trees	200
SVM	kernel	radial basis function
	penalty parameter of the error term	2.0
	gamma	$\frac{1}{27}$
Decision Trees	criterion	gini

Table 4.2: Parameters of Classification models

We show the results in the table 4.3. The best model with the highest accuracy is RF, but it reaches only 64.87% and the other two models are even worse, so it is clear to see with manually handcrafted features, one single tweet is difficulty to be

[3] scikit-learn.org/

classified.

Model	**Accuracy**
Random Forest	**0.6487**
SVM	0.5802
Decision Trees	0.5774

Table 4.3: Prediction Accuracy of Different Single Tweet's Creditability Scoring Models

And we rank the features using the features importance which we mentioned in section 2.4.3, showing in table 4.4. The best feature is polarity scores of sentiment. It means that there is a big bias between the rumors tweets and the tweets real events. It was mentioned by previous work [2] where he gathered a large rumors collection during WW2 which are printed in the Boston Heralds Rumor Clinic. He summarized rumors as several types: pipe-dream, fear and aggression. The most researches believe that rumors mostly contain negative sentiment and polarity [21][10]. In our study average polarity score of news event is -0.066 and average polarity score of rumors is -0.1393 ,it means that rumors contain more negative sentiment.

And we usually think the verified users may have less possibility to be involved in the rumors' spreading, but the result shows that the verified users may be not really trustful like we thought. And "Isreweet" feature is neither a good feature which means the possibility of people retweeting the rumors or true events are similar.

Feature	Feature Importance
PolarityScores	0.1460686474
Capital	0.09638447209
LengthofTweet	0.09283739724
Tweets	0.08750049577
Friends	0.08065591431
ReputationScore	0.08002109553
Followers	0.07938657292
NumOfChar	0.07659755102
Stock	0.04920394972
NumNegativeWords	0.03068379335
Exclamation	0.02304551015
numUrls	0.02124370609
NumPositiveWords	0.01976939973
Hashtag	0.01851408745
Mention	0.01596532677
Question	0.01486070376
Retweets	0.01349486577
I	0.0109471116
You	0.00998103276
HeShe	0.00774915859
UserDescription	0.007402174886
Via	0.005545157727
QuestionExclamation	0.005422123705
Isretweet	0.003240079497
Verified	0.003081752983
Smile	0.0003979192278
Sad	0

Table 4.4: Features Importance

4.2 Single Tweet's Creditability Model without handcrafted features

As we showed in last section. The result of Single Tweet's Creditability Model with handcrafted features no matter with SVM, DT or RF is not so convincible. We may miss some important features we didn't mine out of the tweet. Inspired by the Lai and J Ma's works [12] [14] we test neural networks as the classifier to build up this system. It doesn't need to extract features from the raw data. So now we can think this task as a normal text classification task. Based on the previous work we tested it with (wenxian) models: Basic tanh-RNN, 1-layer GRU-RNN as figure 4.2 and 2-layer GRU-RNN as figure4.3 model.

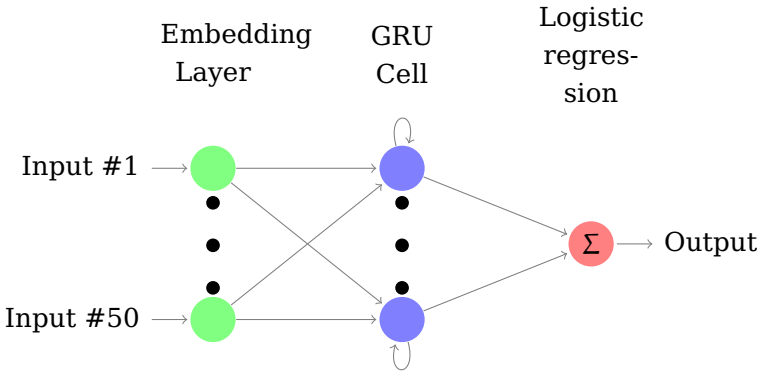


Figure 4.2: 1 layer GRU Recurrent neural network

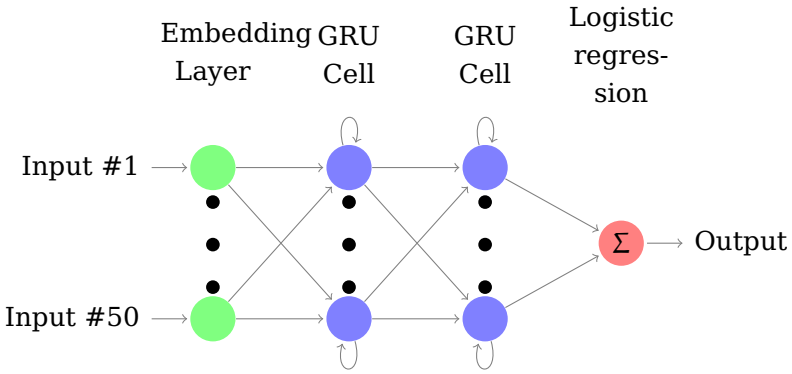


Figure 4.3: 2 layer GRU Recurrent neural network

4.2.1 Data preparing

We use the same dataset as the above models and same shuffled sequence. The difference is the we feed only the text of each tweet to our neural network models. And we randomly take 10% data from the training set to be the valid set for optimizing the parameters.

4.2.2 First Layer: Embedding Layer

The first layer is embedding layer which is same to both models. The size of embedding we set up as 50. The output of the layer will be a vector presenting the words.

4.2.3 Hidden Layer: GRU and LSTM

5 Time Seriers Rumor Detection Model

As we showed in figures 2.8 and 2.9, the features change during the the events' spreading over time. In order to capture these changes of the each features we use Dynamic Series-Time Structure (DSTS) which was presented in Ma's work [15]. In the an Event E_i there is a set of tweets tw_{ij} and we split them into different time intervals according to the creation time so that we can analyze the features in time series. We test the different classifiers with this model, we compare it with static features and in the end we rank all features and show their performance over time.

5.1 Dynamic Series-Time Structure (DSTS)

5.1.1 Time Stamps Generation

For an event E_i we define $timeFirst_i$ as the start time of the event, $timeFirst_i$ as the time of last tweet of the event. We split the each tweet tw_{ij} into N time intervals according to the creation time. The length of each time interval we define as follow:

$$Interval(E_i) = \frac{\lceil (timeLast_i - timeFirst_i) \rceil}{N} \tag{5.1}$$

And the index of time interval $TS(t_{ij})$ where a tweet tw_{ij} which is created in time t_{ij} should fall into, we define as follow :

$$TS(t_{ij}) = \frac{\lfloor (t_{ij} - timeFirst_i) \rfloor}{Interval(E_i)} \tag{5.2}$$

In our work $Interval(E_i)$ as we defined in section 2.3.2 is one hour and N is constant 48 hours for each event.

5.1.2 Dynamic Series-Time Structure (DSTS)

Now we have all the time intervals of an event E_i and we can generate a vector $V(E_i)$ of features in each time interval. And in order to capture the changes of feature over time we should no only model the features in individual time intervals but also

we should model their difference between two time intervals. So the model of DSTS is represented as:

$$V(E_i) = (\mathbf{F}^D_{i,0}, \mathbf{F}^D_{i,1}, ..., \mathbf{F}^D_{i,N}, \mathbf{S}^D_{i,1}, ..., \mathbf{S}^D_{i,N}) \tag{5.3}$$

where the $\mathbf{F}^D_{i,t}$ is the feature vector in time interval t of event E_i. $\mathbf{S}^D_{i,t}$ is the difference between $\mathbf{F}^D_{i,t}$ and $\mathbf{F}^D_{i,t+1}$. $V(E_i)$ is the time series feature vector of the event E_i.

$$\mathbf{F}^D_{i,t} = (\widetilde{f}_{i,t,1}, \widetilde{f}_{i,t,2}, ..., \widetilde{f}_{i,t,D}) \tag{5.4}$$

$$\mathbf{S}^D_{i,t} = \frac{\mathbf{F}^D_{i,t+1} - \mathbf{F}^D_{i,t}}{Interval(E_i)} \tag{5.5}$$

We use Z-score to normalize feature values which is implemented by sklearn.

$$\widetilde{f}_{i,t,k} = \frac{f_{i,t+1,k} - \bar{f}_{i,k}}{\sigma(f_{i,k})} \tag{5.6}$$

where$f_{i,t,k}$ is the k-th feature in time interval t of the event E_i in time interval t. $\bar{f}_{i,k}$ is the mean of the feature k of the event E_i and $sigma(f_{i,k})$ is the standard deviation of the feature k over all time intervals. We skip this step when we use random forest because RF doesn't need feature normalization.

5.1.3 Features

We use a collection of features based on previous works [8] [17][6][7][26][13][14][25][15]. We extracted totally 49 features in table 5.2. These features are not only extracted from Twitter interface but also other website like bluecoat.com we mentioned them in section 3.

Text Features

Twitter Features

User Features

The list of large city [1]

Epidemiological Modeling Features

Jin's work is as far as we know the first people using epidemiological model to analyze rumors' prorogation on twitter [8]. They fits the volume of the rumors and news events into two models SIS (Susceptible, Infected, Susceptible) and SEIZ (susceptible, exposed, infected, skeptic).

SIS is one of the most popular epidemiological model. To adapt to the scenario of Twitter, we define a user who posts a tweet of relevant event as **I**nfected, a user who didn't we define as **S**usceptible. But unlikely as a normal epidemiological scenario

[1] http://www.demographia.com/db-worldua.pdf

infected nodes can be cured and return to be susceptible, the user once posts a tweet of the certain events, he will be classified into the infected component forever. He can't be return susceptible class. At time t the total number of population is $\Delta N(t) = I(t) + S(t)$ where $I(t)$ is the size of infected population and $S(t)$ is the size of susceptible population. As shown in Figure 5.1, SIS model works as follow:

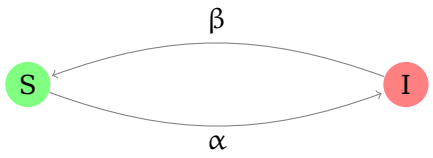


Figure 5.1: SIS Model

- A user who posts tweets about the certain event is regarded as infected
- A susceptible user has not tweeted about the certain event
- A susceptible user may see the a tweet about the certain event from a infected users and he immediately retweets or posts a tweet about this events, and in that he turns himself to infected.
- Susceptible user will remain susceptible until he contacts (via tweet) with infected person.

we show SIS model mathematical as follow:

$$\frac{d[S]}{dt} = -\beta SI + \alpha I \tag{5.7}$$

$$\frac{d[I]}{dt} = \beta SI - \alpha I \tag{5.8}$$

SIS model assumes that a susceptible user once exposed to a infected user turns to infected immediately. That is one reason of this model why it didn't fit to Twitter. If fact when twitter users see a tweet they have their normal senesce to judgment the truth of the information and they can decide wether further spreading the tweet or ignoring them.

Another popular model is SIR which contains one more term than the SIS. The definitions of S and I are the same of SIS but the term R stands for **r**ecover. Once a susceptible user is recover, he will be removed from the susceptible component and he can't be infected again. But we can't get a reasonable explanation of the term R if we model the an event spreading on Twitter.

So they use another model called SEIZ which reference from Bettencourt . We show the model as figure 5.2.

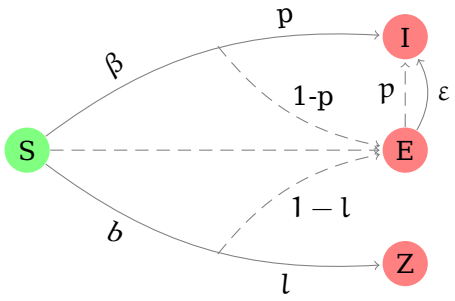


Figure 5.2: SEIZ Model

SpikeM model Features

Kwon is another approach [11] discussing differences between the rumors' propagation pattern and the news events' propagation pattern on twitter.

SpikeM first was introduced by Yasuko Matsubara [16] which are used to describe the information diffusion. We present it as follow:

$$\Delta B(n+1) = p(n+1) \cdot (U(n) \cdot \sum_{t=n_b}^{n} (\Delta B(t) + S(t)) \cdot f(n+1-t) + \varepsilon) \tag{5.9}$$

$$p(n) = 1 - \frac{1}{2} P_a(\sin(\frac{2\pi}{P_p}(n + P_s))) + 1) \tag{5.10}$$

$$U(n+1) = U(n) - \Delta B(n+1) \tag{5.11}$$

where

$$f(\tau) = \beta \cdot \tau^{-1.5} \tag{5.12}$$

and initial conditions:

$$\Delta B(0) = 0, U(0) = N \tag{5.13}$$

In addition, adding an external shock $S(n)$, a spike generated at beginning time n_b. Mathematically, it is defined as follows:

$$S(n) = \begin{cases} 0 & (n \neq n_b) \\ S_b & (n = n_b) \end{cases} \tag{5.14}$$

As the definition:

$$B(n) + U(n) = N \tag{5.15}$$

The term of $\sum_{t=n_b}^{n} (\Delta B(t) + S(t))$ is the total number of informed blogers at time n,

Crowd Wisdom Features

CreditScore Features

Symbol	Definition
N	total population of available bloggers
n_d	duration of sequence
n	time-tick (n=0, . . . , n_d)
U(n)	count of **u**n-informed bloggers
B(n)	count of informed **b**loggers
$\Delta B(n)$	count of informed **b**loggers at time n
f(n)	in**f**ectiveness of a blog-post, at age n
β	strength of infection
$\beta \cdot N$	"first-burst" size of infection
S(n)	volume of external **s**hock at time n
n_b	starting time of **b**reaking news
S_b	strength of external shock at birth (time n_b)
ε	background noise
P_a	strength of periodicity
P_p	period
P_s	phase shift of periodicity

Table 5.1: Parameters of SpikeM

Category	**Feature**	**Description**
Twitter Features	Hashtag	% of the tweets containing #hashtag
	Mention	% of the tweets mentioning others @user
	NumUrls	# of url in the tweet
	Retweets	average times of tweets have been retweeted
	Isretweet	% of tweets are retweeted from others
	ContainNEWSWeb	% of tweets containing URL and its domain's catalogue is News
	WotScore	average WOT score of domain in URL
	Isretweet	% of tweets being retweeted from others
Text Features	LengthofTweet	average length of tweets
	NumofChar	average number of individual characters of tweets
	Capital	average fraction of characters in Uppercase of tweets
	Smile	% of tweets containing :->, :-), ;->, ;-)
	Sad	% of tweets containing :-<, :-(, ;->, ;-(
	NumPositiveWords	average number of positive words
	NumNegativeWords	average number of negative words
	Polarityscores	average polarity scores of the Tweets
	Via	% of tweets containing via
	Stock	% of tweets containing $
	Question	% of tweets containing ?
	Exclamation	% of tweets containing !
	QuestionExclamation	% of tweets containing multi Question or Exclamation mark
	I	% of tweets containing first pronoun like I, my, mine, we, our
	You	% of tweets containing second pronoun like U, you, your, yours
	HeShe	% of tweets containing third pronoun like he, she, they, his, etc.
User Features	Followers	average number of followers
	Friends	average number of friends
	Tweets	average number of tweets being posted
	Numphoto	average number of photos being posted
	IsInLargeCity	% of user living in large city
	Join_date	average days since user joining Twitter
	Description	% of user having description
	Verified	% of user being a verified user
	ReputationScore	average ratio of #Friends over (#Followers + #Friends)
Epidemiological Features	β_{SIS}	average β of Model SIS
	α_{SIS}	average α of Model SIS
	β_{SEIZ}	average β of Model SEIZ
	b_{SEIZ}	average b of Model SEIZ
	l_{SEIZ}	average l of Model SEIZ
	ε	average ε of Model SEIZ
	ρ	average ρ of Model SEIZ
	SEIZIndex	average SEIZIndex of Model SEIZ
SpikeM Model Features	P_s	average P_s of Model Spike
	P_α	average P_α of Model SpikeM
	P_p	average P_p of Model SpikeM
	Q_s	average Q_s of Model SpikeM
	Q_α	average Q_α of Model SpikeM
	Q_p	average Q_p of Model SpikeM
Crowd Wisdom Features	Crowdwisdom	% of tweets containing "Debunking Words"
Credit Score Features	CreditScore	average CreditScore

Table 5.2: Features of Time Series Rumor Detection Model

Bibliography

[1] F. Ahmed and M. Abulaish. An mcl-based approach for spam profile detection in online social networks. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 602–608. IEEE, 2012.

[2] G. W. Allport and L. Postman. The psychology of rumor. 1947.

[3] L. Barbosa and J. Feng. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 36–44. Association for Computational Linguistics, 2010.

[4] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[5] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

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

[7] A. Gupta, P. Kumaraguru, C. Castillo, and P. Meier. Tweetcred: Real-time credibility assessment of content on twitter. In *International Conference on Social Informatics*, pages 228–243. Springer, 2014.

[8] 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*, page 8. ACM, 2013.

[9] D. Kimmey. Twitter event detection. 2015.

[10] S. Kwon, M. Cha, K. Jung, W. Chen, and Y. Wang. Aspects of rumor spreading on a microblog network. In *International Conference on Social Informatics*, pages 299–308. Springer, 2013.

[11] S. Kwon, M. Cha, K. Jung, W. Chen, and Y. Wang. Prominent features of rumor propagation in online social media. In *2013 IEEE 13th International Conference on Data Mining*, pages 1103–1108. IEEE, 2013.

[12] S. Lai, L. Xu, K. Liu, and J. Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, pages 2267–2273, 2015.

[13] X. Liu, A. Nourbakhsh, Q. Li, R. Fang, and S. Shah. Real-time rumor debunking on twitter. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1867–1870. ACM, 2015.

[14] J. Ma, W. Gao, P. Mitra, S. Kwon, B. J. Jansen, K.-F. Wong, and M. Cha. Detecting rumors from microblogs with recurrent neural networks.

[15] J. Ma, W. Gao, Z. Wei, Y. Lu, and K.-F. Wong. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1751–1754. ACM, 2015.

[16] Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos. Rise and fall patterns of information diffusion: model and implications. In Q. Yang, D. Agarwal, and J. Pei, editors, *KDD*, pages 6–14. ACM, 2012.

[17] 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*, pages 71–79. ACM, 2010.

[18] O. Oh, K. H. Kwon, and H. R. Rao. An exploration of social media in extreme events: Rumor theory and twitter during the haiti earthquake 2010. In *ICIS*, page 231, 2010.

[19] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River, 2003.

[20] K. Starbird, J. Maddock, M. Orand, P. Achterman, and R. M. Mason. Rumors, false flags, and digital vigilantes: Misinformation on twitter after the 2013 boston marathon bombing. *iConference 2014 Proceedings*, 2014.

[21] C. R. Sunstein. *On rumors: How falsehoods spread, why we believe them, and what can be done*. Princeton University Press, 2014.

[22] Y. Tanaka, Y. Sakamoto, and T. Matsuka. Transmission of rumor and criticism in twitter after the great japan earthquake. In *Annual Meeting of the Cognitive Science Society*, page 2387, 2012.

[23] R. Thomson, N. Ito, H. Suda, F. Lin, Y. Liu, R. Hayasaka, R. Isochi, and Z. Wang. Trusting tweets: The fukushima disaster and information source credibility on twitter. *Proc. of ISCRAM*, 10, 2012.

[24] A. H. Wang. Don't follow me: Spam detection in twitter. In *Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on*, pages 1–10. IEEE, 2010.

[25] K. Wu, S. Yang, and K. Q. Zhu. False rumors detection on sina weibo by propagation structures. In *2015 IEEE 31st International Conference on Data Engineering*, pages 651–662. IEEE, 2015.

[26] 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*, page 13. ACM, 2012.

[27] Z. Zhao, P. Resnick, and Q. Mei. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1395–1405. ACM, 2015.

List of Figures

List of Tables