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 [32] [41][39]. But in our work, we study on the general rumors.

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

We build up a system with random forest and the Dynamic Series-Time Structure (DSTS) [27] with the temporal features and their varieties over time. And we test the Spike Model [22], SIS Model and SEIZ Model [15] as temporal features in our model. To improve the time series model's performance at early stage of the event we develop a single tweet's credibility scoring model with zhou's CNN+LSTM model [47] which can predict a single tweet whether is rumor related or not with 81.20%accuracy. Finally we got the accuracy of the time series rumor detecting model get 90% accuracy within 48 hours.

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

Zusammenfassung

Acknowledgment

I am dedicating this thesis to my parents. They have been strong and supportive of my choices during my masters and have pushed me to work harder everyday. I am very grateful to my mentor Dr. Avishek Anand for helping me across the finish line and helping me make defining career choices. My sincere gratitude to Prof. Dr. Wolfgang Nejdl for having faith in me and guiding me through my bachelors and masters. A big thank you to the LearnWeb team including Dr. Ivana Marenzi for giving me a project to contribute towards. I am especially grateful to M. Sc. Sergej Zerr for taking me under his wing when I first arrived and I owe a lot of what I have learned to him. A special mention to the ITIS program for providing international students like me a world-class education. Last but not the least, to my friends - Pragyan, Zeon, Saniya, Rohit, Irene and the others who have helped me through good times and bad.

Without you all my masters would not have been a success.

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 [19].

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 [35]. 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 [29].

So a method of detecting rumors on Twitter that could detect rumors in the early stage of propagation can be very useful. The structure is shown in figure 1.1. First 1) we crawled the data from Twitter interface, 2) we use beautiful soup and spark technologies to extract feature from the tweets, 3) we extract time series features and fit to the dynamic series-time structure 4) we use the text of tweets as feature to train the single tweet's credibility scoring model neural network and merge the its output to the time series features, 5) training the time series model with DSTS.

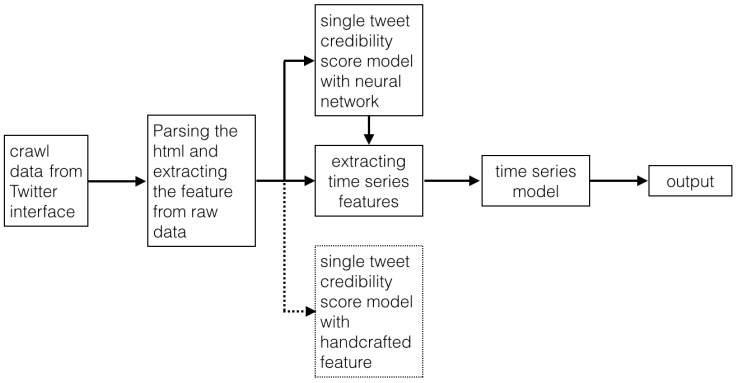


Figure 1.1: pipeline of the rumor detecting system

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 what we tested is based on zhou's[47] work, it is a hybrid CNN and LSTM model for text classification. The result of this model we called it credit score with 81% accuracy.

- We develop a time series model for detecting rumor events. We used Dynamic Series-Time Structure (DSTS)[25] to capture the changes of features over time. And we tested 3 time series model as feature: modified Spike Model [22], SIS model and SEIZ model[15]. 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. In 48 hours after the event's spreading we got 90% accuracy to detect the rumor events.

- 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 6 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 [20]. 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[14]. And Thomson's work shows that different crowds of people trust tweets basing on different kinds of sources [42].

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

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".

2.3.2 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.

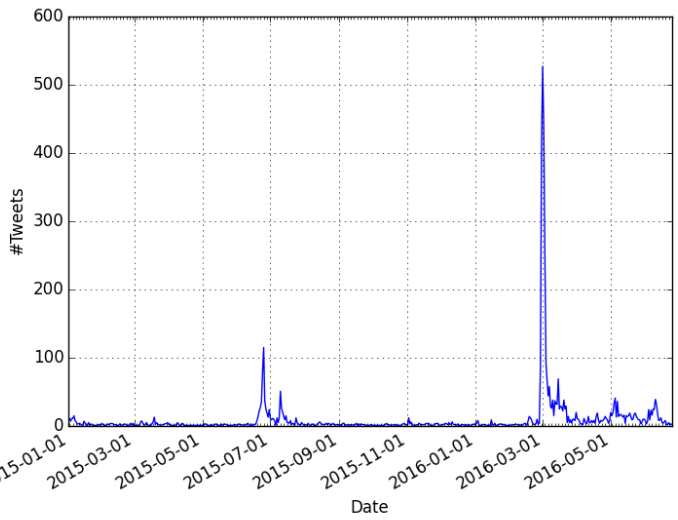


Figure 2.1: large scale tweet volume of event Robert Byrd

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,

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

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.

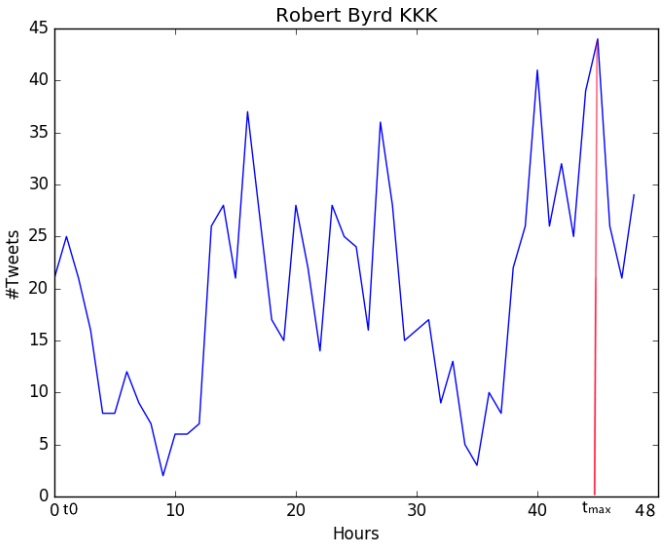


Figure 2.2: tweet volume of the rumor event of Robert Byrd after selected time period

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[37].

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

Classification is a supervised data mining technique. Our work can be considered as a classification task. Decision tree is a model we use for base line in the after work. A decision tree uses a tree model of decisions and their possible consequences as shown in figure 2.3. DT is an non-parametric supervised learning method used for classification and regression. Because the trees structure is very simple to visualize, the result of DT is easy to understand. But Decision trees are unstable [6]. But DT is the basic unit of random forest.

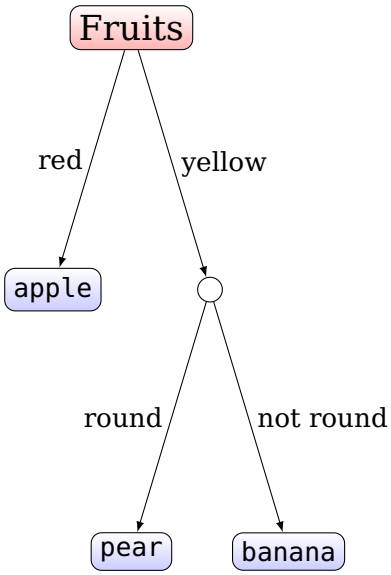


Figure 2.3: Decision tree

2.4.3 Random Forest (RF)

Another we use for testing is random forest.

RF is an algorithm of supervised learning which developed by Leo Breiman [7]. It's built by a set of classification trees [8]. 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 2.4. 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.

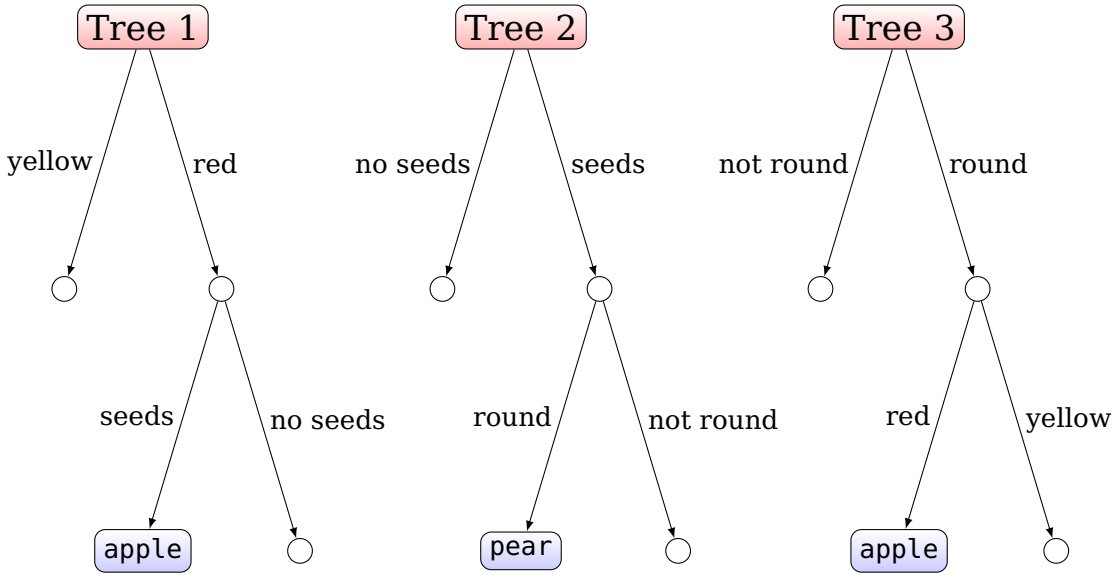


Figure 2.4: An example of random forest with 3 trees

Because RF uses a random subset of features instead of the best features in every node, so it can avoid the overfitting problem [7].

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. One more important benefit of RF is it can easily visualize, so it can convince people not only by the test accuracy but also by the giving us a good explanation, for example rumor containing more negative sentiment or the poster of real news more likely living in large city.

2.5 Neural Network

Figure 2.5 is a simple model of forwarding neural network which is a computing systems that processes information by their dynamic state response to external inputs [39]. In the figure 2.5 the round nodes are called neurons. Each neuron has a active

function like sigmod or tanh 2.6 Neurons are connected each others with weight. The process of training the model is actually the process learning the weights of the connections of the network. With the back-propagation algorithm we 1) compute the loss from the network output and 2) update the weights by passing the error backwards through the network [36].

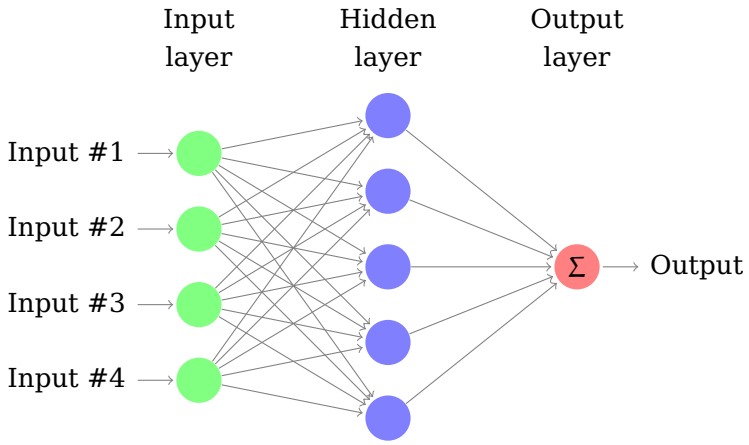


Figure 2.5: An example of 1 layer neural network

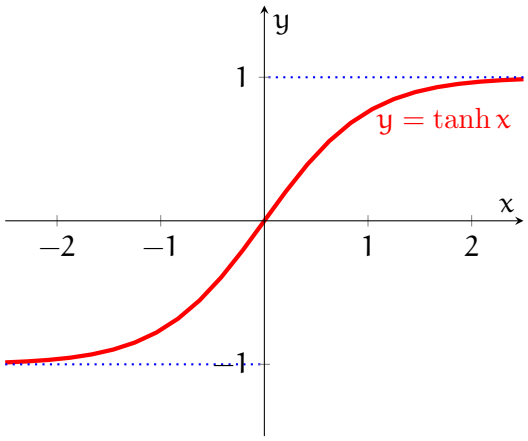


Figure 2.6: tanh function

2.5.1 Convolutional Neural Network

CNN is a kind of neural network which contains a convolutional layers. First CNN is developed by LeCun et al. developed CNNs [24] for computer vision to resolve the problem of imagines' shift or rotation. With the convolutional layer CNN make that is possible to learn from the local features. Figure 2.7 is an example p of CNN in application of text classification. The word "wait" and its following word "for" are mapped together into different classes though different convolutional filters. That makes convolutional neural network does not learn from the single word form sentence but from the word and its context. That makes sense because one word can in different context have different meaning. For example "Apple Tree" and "Apple Company", the same apple but contains different meaning. Pooling layer often follows after the convolutional layers in CNN models whose mainly task is to down-sample from the output of convolutional layers.

We use it for for single tweet's creditability score model in the later section 4.2

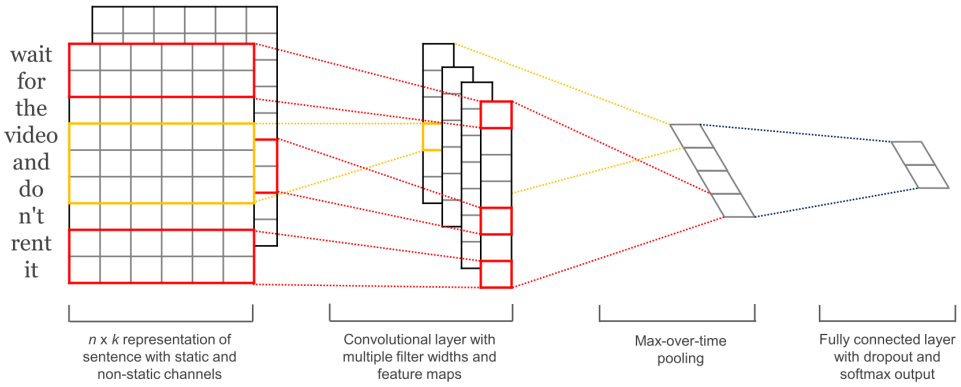


Figure 2.7: CNN for Text Classification (source from[17])

2.5.2 Recurrent Neural Network

A Recurrent neural network (RNN) is one of of class of neural network. The different between RNN and other neural network is there are feedback connections between the hidden layers showing as figure 2.8, that makes the input from the past also can influence the network. The activation status of the hidden layer neurons presents as an internal state which can be considered as a kind of memory. That makes recurrent neural network has the ability to process the sequence input. Human understand the words of a sentence often need the help of the previous words, but the feed forwarding neural network will ignore the previous inputs.

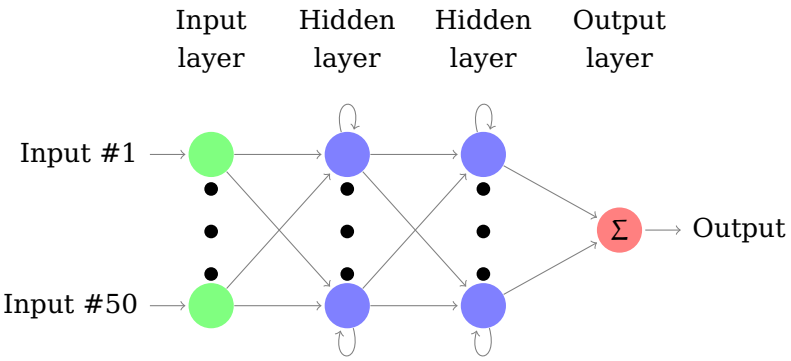


Figure 2.8: 2 layer Recurrent neural network

There are several types of applications of RNN. First one is multi-input and single-output 2.9, application in for example text classification. Second is single-input and multi-output 2.10, application in for example input an imagine and generate a sentence for description. Third is multi-input and multi-output 2.11, application in for example the translation system.

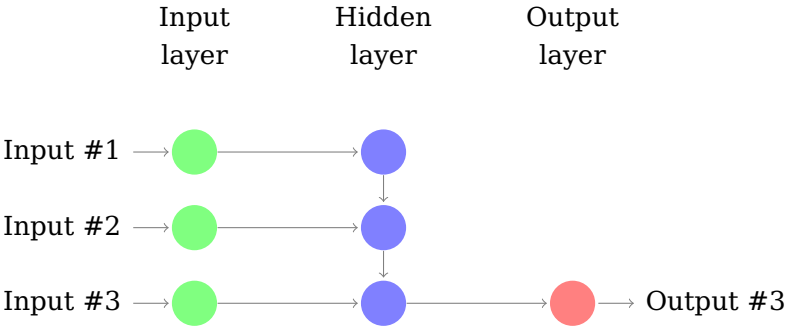


Figure 2.9: multi-input single-output Recurrent neural network

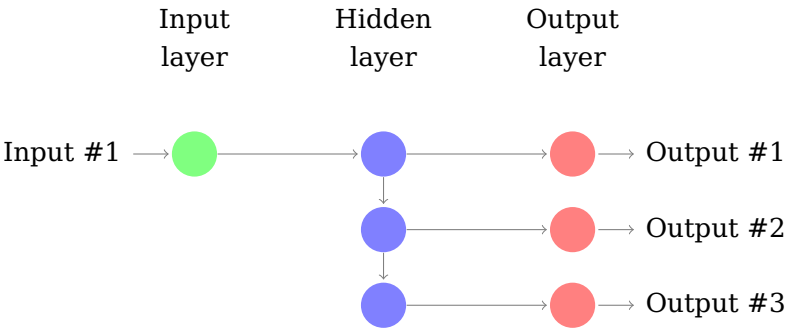


Figure 2.10: single-input multi-output Recurrent neural network

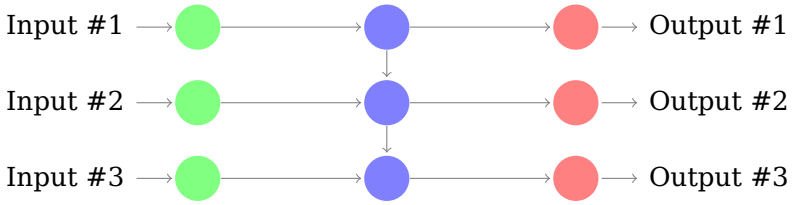
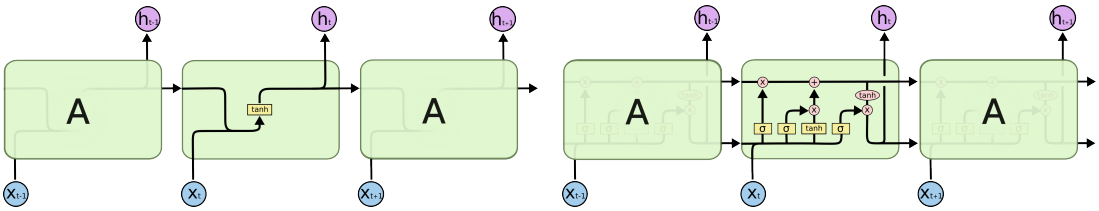


Figure 2.11: multi-input multi-output Recurrent neural network

2.5.3 Long Short-Term Memory

But recurrent neural networks are difficult to train for a long sequence input, because there are two problems **"gradient explode"** and **"gradient vanish"** [34]. Gradient vanish means that the gradient of the earlier inputs goes exponentially fast towards zero. That make recurrent neural networks can't learn the long-term dependencies in sequences. Gradient explode means the gradients become exponentially large while training.

People can change the active function like ReLU to avoid the vanishing gradient problem, but more popular solution is using Long Short-Term Memory which is first invented by Hochreiter and Schmidhuber [13]. An LSTM memory cell contains self-connected memory cell with its a memory cell, an input gate, an output gate and a forget gate units. LSTM uses a series gate units to control the cell receiving data flow, extracting data flow or forgetting the current state at each time step.



(a) Simple RNN model (b) LSTM RNN model

Figure 2.12: Two models of RNNs (a) Normal model of RNN with tanh Unit (b) RNN with LSTM Cells (source [33])

2.5.4 GRU Cell

GRU Cell is a special case of LSTM which is invited by Cho, et al [10]. The forget and input gates are merged into a update gate. The cell state and hidden state are combined as one state. These changes make the structure of GRU simpler than LSTM.

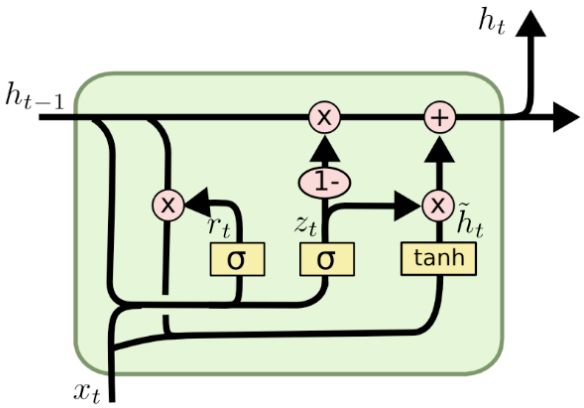
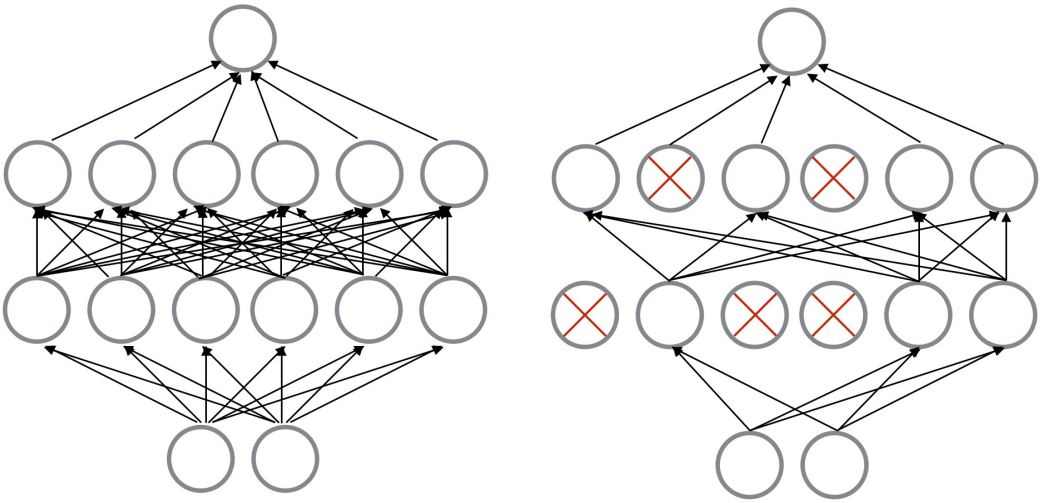


Figure 2.13: GRU Cell (source [33])

2.5.5 Dropout

To avoid the overfitting we used dropout methode[38]. The main idea of dropout is to dropout some units in hidden layer in neural network while training. When training the model some of the units are removed temporarily with probability p as figure 2.14. But while testing the dropout units still join the test. The aim of dropout is to avoid one best state keeping existing, by disabling some units time to time of the network dropout makes the model to be trained in different form every time. Dropout causes to the convergence speed slower than without it, the model needs more epochs to find the local best solution. But the benefit is it can avoid the overfitting [38].



(a) standard NN model (b) NN with dropout

Figure 2.14: Dropout Neural Net Model. (a) a standard neural network (b) after deploying dropout

2.6 K Fold Cross Validation

To limit the overfitting and improve the robustness of the model we often use cross validation technology. K fold cross validation is that the dataset was shuffled and split into k subsets. We train the model with some of subsets and others are used for testing.

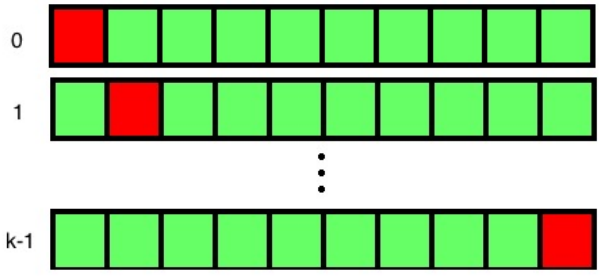


Figure 2.15: K Fold Cross Validation (green rectangle is the training set, red rectangle is the testing set)

In our work we split the data not equally, we consider an event as a unit. We use 10 times cross validation, so we split 260 event into 10 subsets. And we test all models with the same shuffled sequence, aka same training sets and test sets.

2.7 Levenberg-Marquardt Method

2.8 Spark

2.9 Tensorflow

2.10 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[18], spam detection [1] [43] or sentiment detection[4]. 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] [40]. But since 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[32] [41][30] or terrorist attacks [39]. 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[9][11]. 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 [45] [25] and used different set of features, for example yang [45] add the location of poster and the client type of user (web client or mobile phone) as two important features. And

There are rencent researches based on the propagation structure of rumor on Sina Weibo[44] [3], but Twitter doesn't give us such detail of propagation like Weibo, so these works are useless for us.

Liu's work [44] 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 [25]. 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 2.16 and 2.17. 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.

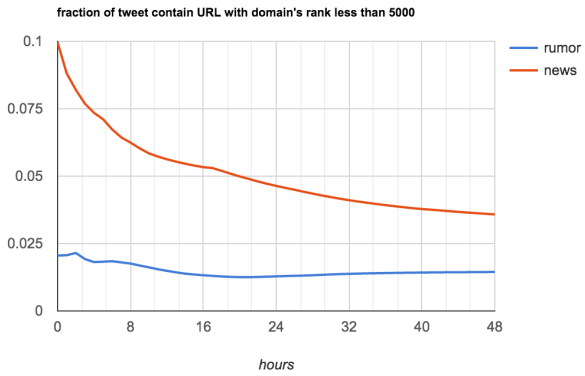


Figure 2.16: The fraction of tweets containing url with top 5000 domain

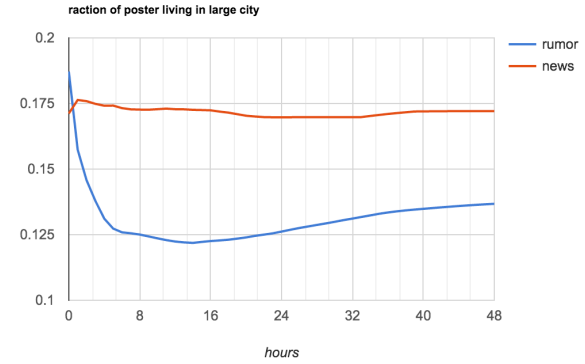


Figure 2.17: The fraction of poster living in large city

Other researchers used propagation models like SpikeM [22] and SEIZ models [15] 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 later section 5.4.2.

J Ma used Recurrent Neural Networks for rumor detection [26], 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 [27] 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 within 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], their ranks in alexa.com[2] and WOT score in wot.com [3].

As we told in the section 2.1.8, the Twitter limits its API with the return result only in recent 7 days. So we have to crawl the data from the web interface. 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

[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/

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 process much faster.

[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 [25] [27] [46], because a single tweet is to short to contains engough features. But Carlos Castillo designed a single tweet's creditability rank system Tweetcred [11]. 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 [11] 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[11] 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] $Polarity\_scores = \frac{1}{N}\sum_{0}^{n} Polarity(token_n)$.

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 retweets 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).

[1] http://www.nltk.org/
[2] http://www.nltk.org/api/nltk.sentiment.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	UserFollowers	# of followers
	UserFriends	# of friends
	UserTweets	# of tweets which are posted by this user
	UserDescription	Whether this user has description
	UserVerified	Whether this user is a verified user
	UserReputationScore	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

[3] scikit-learn.org/

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 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 [40][21]. 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 probability of people retweeting the rumors or true events are similar.

Feature	Feature Importance
PolarityScores	0.1460686474
Capital	0.09638447209
LengthOfTweet	0.09283739724
UserTweets	0.08750049577
UserFriends	0.08065591431
UserReputationScore	0.08002109553
UserFollowers	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
UserVerified	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 [23] [26] we test neural networks as the classifier which does not need to extract features from the data. Based on the previous work we tested it with 6 models: Basic tanh-RNN 4.2(a), 1-layer GRU-RNN 4.2(b) ,1-layer LSTM 4.2(c), 2-layer GRU-RNN 4.2(d), Fasttext 4.2(e) and CNN+LSTM 4.2(f) model as figure 4.2(d) model. Basic tanh-RNN, 1-layer GRU-RNN, 1-layer LSTM-RNN and 2-layer GRU-RNN models are based on the work of J Ma's works [26] . Fasttext comes from joulin's work [16] which is a fast text classification model no need of GPU acceleration. The hybrid model of CNN and LSTM (C-LSTM) is zhou idea [47] which has the best performance in out experiments.

4.2.1 Data Preparing

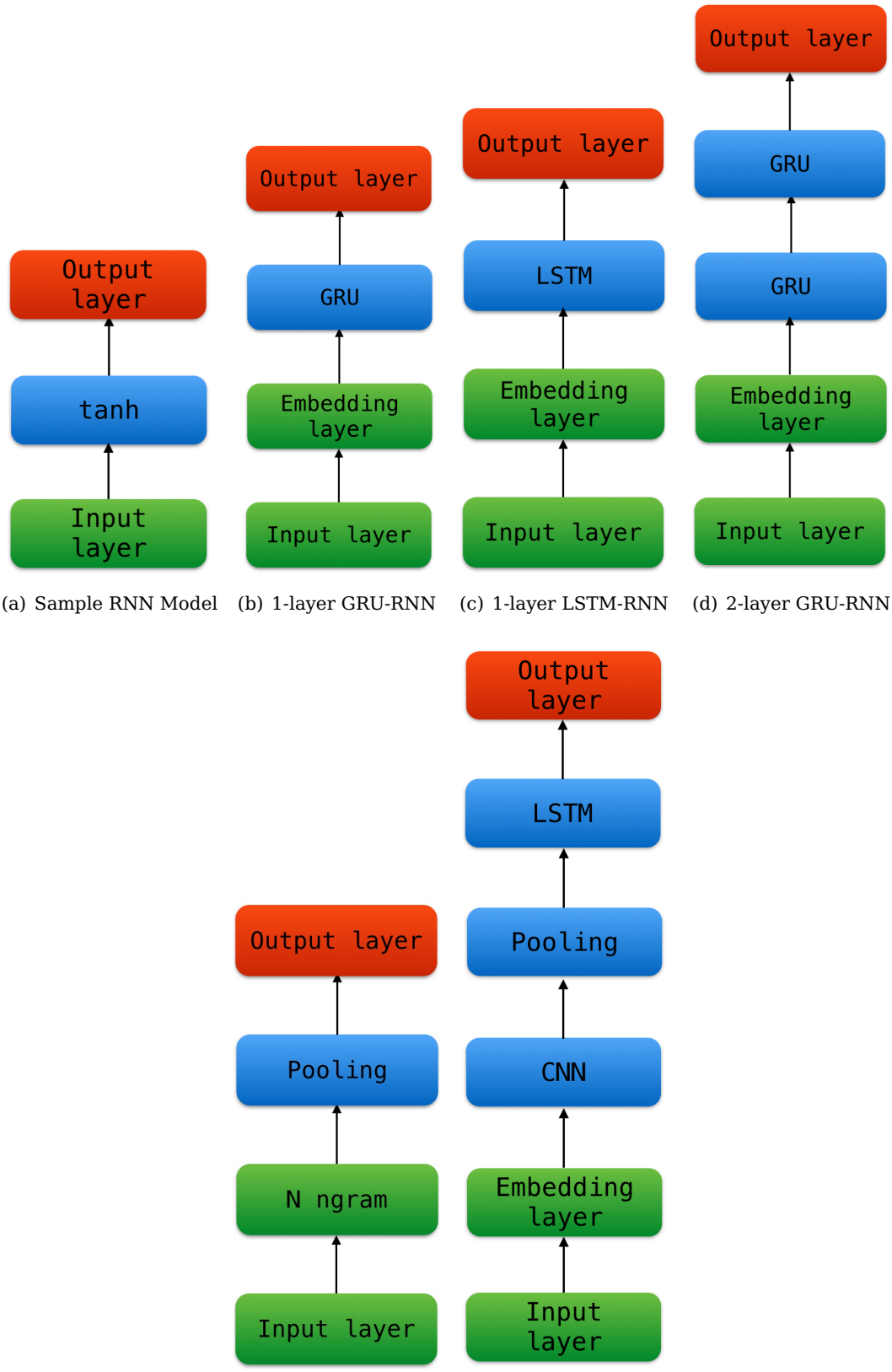
We test the above models with same dataset and same shuffled sequence as the RF, SVM model. The difference is the we feed only the text of each tweet to the neural network models. And we randomly take 20% data from the training set to be the valid set for optimizing the parameters.

4.2.2 Embedding Layer

The first layer is embedding layer which is set up the same to all models.The embedding size is 50. The output of the embedding layer is the vectors presenting the words.

4.2.3 Tested Results

The best accuracy result is C-LSTM model as shown in table 4.5. To avoid overfitting we use 10-fold cross validation and dropout technology.



(a) Sample RNN Model (b) 1-layer GRU-RNN (c) 1-layer LSTM-RNN (d) 2-layer GRU-RNN

(e) FastText (f) Hybrid CNN + LSTM (C-LSTM)

Figure 4.2: neural network model for single tweet classification 1

Model	Accuracy
CNN+RNN	**0.8119**
2-layer GRU	0.7891
GRU	0.7644
LSTM	0.7493
Basic RNN with tanh	0.7291
FastText	0.6602

Table 4.5: Prediction Accuracy of Different Single Tweet's Creditability Scoring Models

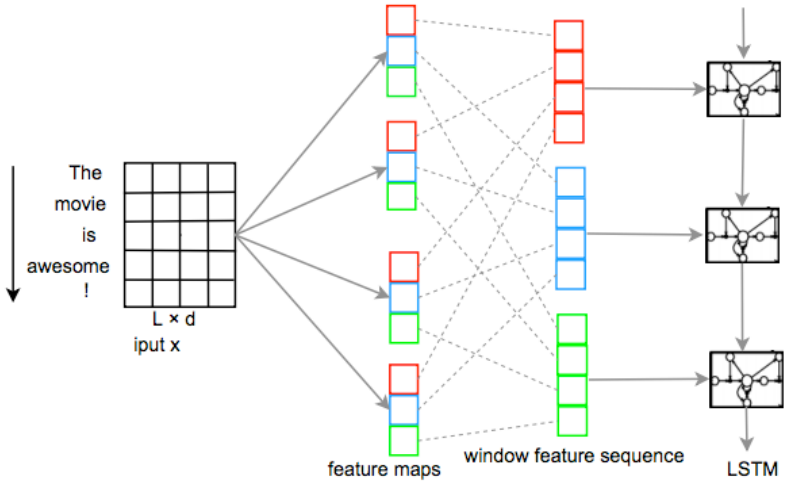


Figure 4.3: The architecture of C-LSTM Interface (source: [47])

4.2.4 CNN + LSTM (C-LSTM) Model

The best accuracy result is C-LSTM model which is invented by zhou [47]. The architecture of the model is shown in figure 4.3. Front hidden layer is a CNN which can split the text to different features and the pooling layer group the same type of feature together then the last hidden layer is a LSTM layer. According to his paper C-LSTM achieved the best result for a 2-classification task with 87.8% accuracy. And in our test it gets 81% accuracy.

5 Time Seriers Rumor Detection Model

As we showed in figures 2.16 and 2.17, 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 [27]. 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 and Decision Trees because they do not need feature normalization.

5.2 Features

We use a collection of features based on previous works [9][11] [45][25][26][30][27][44][15]. We extracted totally 50 features in table 5.4. These features are not only extracted from Twitter interface but also other website like bluecoat.com we mentioned them in section 3.

5.2.1 Text Features

5.2.2 Twitter Features

5.2.3 User Features

The list of large city [1]

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

5.2.4 Epidemiological Modeling Features

Jin's work is as far as we know the first people using epidemiological model to analyze rumors' prorogation on twitter [15]. 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)** infected, a user who didn't we define as **(S)** susceptible. But unlikely as a normal epidemiological scenario 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:

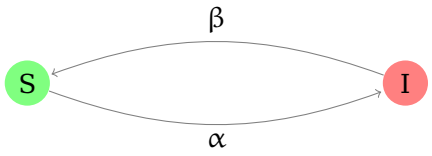


Figure 5.1: SIS Model

- 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 recover. 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.

Because of the Shortcomings of above two model, they test another model called SEIZ which reference from [5] . To adapt to Twitter context, the compartments of

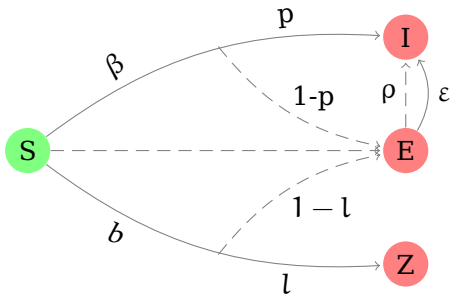


Figure 5.2: SEIZ Model

the SEIZ model can be mapped like this: **(S)** Susceptible is a user who has not been exposed to the event aka he didn't see any tweets about the certain event yet, **(I)** infected means a user has posted tweets about the certain events, **(Z)** skeptic is a user who has been exposed to the certain event but he decides to ignore it and **(E)** exposed is a user who been exposed to the certain event but he will post the tweets after some delay.

We show the model in figure 5.2. And the SEIZ works as follow:

- People recruit from **(S)** Susceptible compartment to Skeptics with rate b. But with probability l some of them directly deny the events and turn to **(Z)** skeptic compartments. Others with probability 1-l probability turn to **(E)** exposed compartment.
- People recruit from **(S)** Susceptible compartment to Infected with rate β. But with probability p some of them directly believe the events and repost it and turn them to be **(Z)** skeptic compartments. Others with probability 1-p probability turn to **(E)** exposed compartment.
- People from **(E)** exposed compartment have ρ probability contacting again with the Infected and turn them to **(I)** infected compartment. And others have ε probability turn into **(I)** infected compartment by themselves for example external shock.

And we show the model mathematical like: we show SEIZ model mathematical as follow:

$$\frac{d[S]}{dt} = -\beta S\frac{I}{N} - bS\frac{Z}{N} \tag{5.9}$$

$$\frac{d[E]}{dt} = (1-p)\beta S\frac{I}{N} + (1-l)bS\frac{I}{N} - \rho S\frac{Z}{N} - \varepsilon E \tag{5.10}$$

$$\frac{d[I]}{dt} = p\beta S\frac{I}{N} + \rho S\frac{Z}{N} + \varepsilon E \tag{5.11}$$

$$\frac{d[Z]}{dt} = lbS\frac{Z}{N} \tag{5.12}$$

Symbol	**Definition**
β	S-I contact rate
b	S-Z contact rate
ρ	E-I contact rate
ε	Incubation rate
$1/\varepsilon$	Average Incubation Time
bl	Effective rate of S -> Z
$\beta\rho$	Effective rate of S -> I
b(1-l)	Effective rate of S -> E via contact with Z
$\beta(1-p)$	Effective rate of S -> E via contact with I
l	S->Z Probability given contact with skeptics
1-l	S->E Probability given contact with skeptics
p	S->I Probability given contact with adopters
1-p	S->E Probability given contact with adopters

Table 5.1: Parameters of SEIZ

The author presents an index of SEIZ called R_{SI} as equation 5.13. It contains all rate values of SEIZ and related to the flux ratio of the **(E)** exposed compartment, the ratio of entering **(E)** to leaving **(E)**. If R_{SI} is bigger than 1 means the influx of exposed compartment is bigger than the efflux. This index may be a good candidate of feature to analyze rumor spreading on Twitter.

$$R_{SI} = \frac{(1-p)\beta + (1-l)b}{\rho + \varepsilon} \tag{5.13}$$

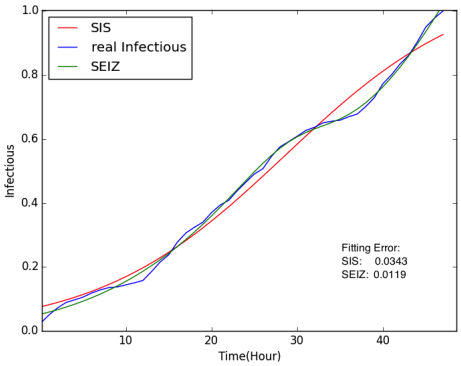
We use Levenberg-Marquard algorithm which we present in section 2.7 to learn the parameters of the SIS and SPEI. The fitting data is the tweet volume of the 260 events (130 rumors and 130 news). In time each interval from t_0 to t_n, we fit the sequenced tweet volume from the beginning time the t_0 to the current time interval

t_n of an event to SIS and SEIZ model and learn. From SIS we get two feature β_n, α_n and from SEIZ we get 7 features $\beta_n, b_n, l_n, p_n, \varepsilon_n, \rho_n, RSI_n$. We add them into our DSTS.

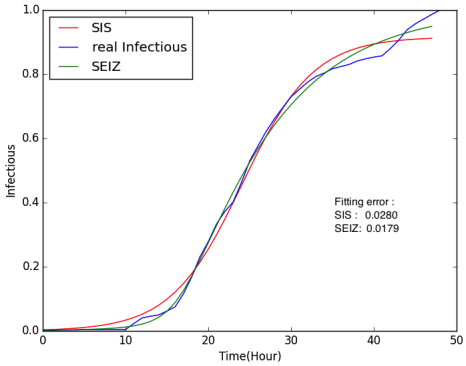
$FittingFunction_{SIS}(TweetVolume_0, ..., TweetVolume_n)$->$\beta_n, \alpha_n$

$FittingFunction_{SEIZ}(TweetVolume_0, ..., TweetVolume_n)$->$\beta_n, b_n, l_n, p_n, \varepsilon_n, \rho_n, RSI_n$

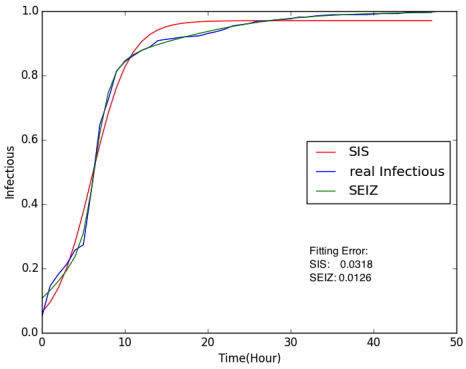
We show 4 examples as following two rumors in figure 5.4(a) 5.4(b) and two news in figure 5.4(c) 5.4(d). It is obvious that SEIZ is more appropriate than SIS to model in our Twitter application, because the fitting error of SPEI is less than SIS.



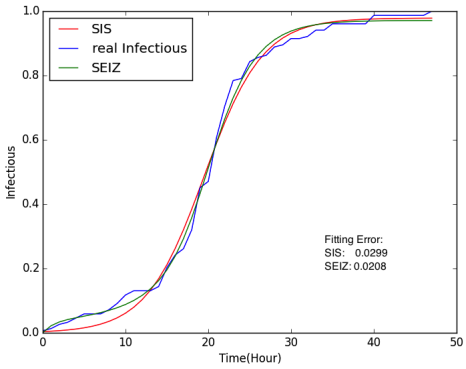
(a) SIS and SEIZ model for rumor 1



(b) SIS and SEIZ model for rumor 2



(c) SIS and SEIZ model for news 1

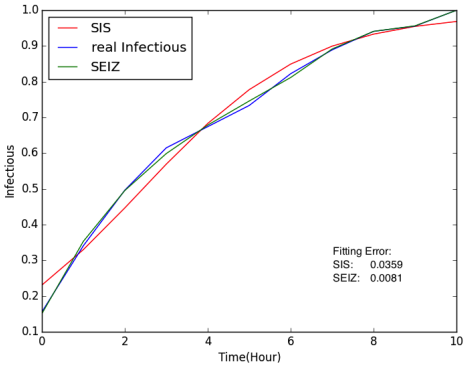


(d) SIS and SEIZ model for news 2

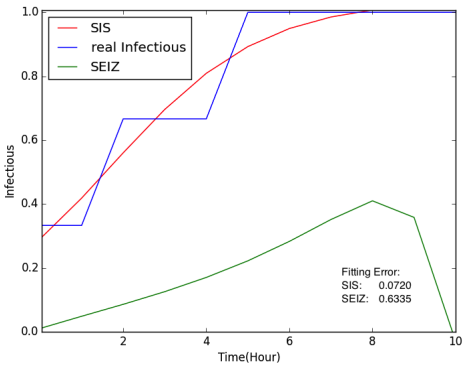
Figure 5.3: Fitting results of SIS and SEIZ model of (a) Rumor: Robert Byrd was a member of KKK (b) Rumor: CNN altered a photograph of a shooter making him look white (c) News: Doctor announces Michael Schumacher is making process (d) News: Two U.S. sailors are arrested over an alleged rape of a Japanese woman on Okinawa

But fitting the models needs enough input data. We don't have enough data to

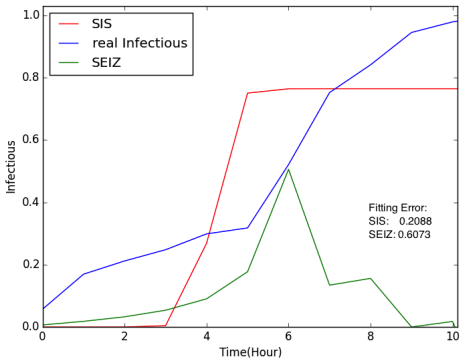
learn the parameters at the first few hours. We show the performance of fitting these two model with only the first 10 hours tweet volume in figure 5.4. As we can see excepting the first one, the fitting result of other three is not good enough.



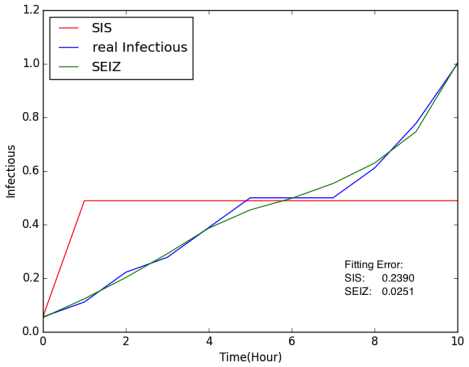
(a) SIS and SEIZ model for rumor 1 with 10 hours data



(b) SIS and SEIZ model for rumor 2 with 10 hours data



(c) SIS and SEIZ model for news 1 with 10 hours data



(d) SIS and SEIZ model for news 2 with 10 hours data

Figure 5.4: Fitting results of SIS and SEIZ model with only first 10 hours tweet volume data (same 4 stories as above)

5.2.5 SpikeM model Features

Kwon showed us another approach [22] for finding the differences between the rumors' propagation pattern and the news events' propagation pattern on twitter. He adjusted the SpikeM Model and also used the parameters as features.

SpikeM first was introduced by Yasuko Matsubara [28] which cab describe the pattern of information diffusion. We present it as follow:

$$\Delta B(n+1) = p(n+1) \cdot (U(n) \cdot \sum_{t=n_b}^{n} (\Delta B(t) + S(t)) \cdot f(n+1-t) + \varepsilon) \tag{5.14}$$

$$p(n) = 1 - \frac{1}{2} P_a(\sin(\frac{2\pi}{P_p}(n + P_s))) + 1) \tag{5.15}$$

$$U(n+1) = U(n) - \Delta B(n+1) \tag{5.16}$$

where

$$f(\tau) = \beta \cdot \tau^{-1.5} \tag{5.17}$$

and initial conditions:

$$\Delta B(0) = 0, U(0) = N \tag{5.18}$$

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

$$S(n) = \begin{cases} 0 & (n \neq n_b) \\ S_b & (n = n_b) \end{cases} \tag{5.19}$$

As the definition:

$$B(n) + U(n) = N \tag{5.20}$$

The term of $\sum_{t=n_b}^{n} (\Delta B(t) + S(t))$ is the total number of informed users at time n, so $\Delta B(n+1) = p(n+1) \cdot (U(n) \cdot \sum_{t=n_b}^{n} (\Delta B(t) + S(t)) \cdot f(n+1-t) + \varepsilon)$ means that at time n+1 an infected node n randomly select a node m of all nodes and if the node m is susceptible the probability of m turning to infected is β, so it is a standard SI model. SpikeM extends the SI model from

- a power-law decay term $f(\tau) = \beta \cdot \tau^{-1.5}$ in equation 5.24. So the earlier infected nodes has less strength of infection in a power-law decay pattern.
- a periodic interaction function in equation 5.22. It stands for that people have a periodic interaction patterns, like people go to sleep at night so they post much more tweets in the day. Parameters P_p, P_a, and P_s are the period, strength, and shift of the periodic interaction function.
- ε is the background noise term.

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 of periodicity
P_s	phase shift of periodicity

Table 5.2: Parameters of SpikeM

But the SpikeM can't fit to the events with multi-pike like the figure 2.2. So the author think the term external shock $S(n)$ in equation 5.25 should not occur once but more. So they extend the SpikeM model by adding a periodic interaction function the term external shock $S(n)$.

$$\Delta B(n+1) = p(n+1) \cdot (U(n) \cdot \sum_{t=n_b}^{n} (\Delta B(t) + \bar{S}(t)) \cdot f(n+1-t) + \varepsilon) \tag{5.21}$$

$$p(n) = 1 - \frac{1}{2} P_a(\sin(\frac{2\pi}{P_p}(n + P_s))) + 1) \tag{5.22}$$

$$U(n+1) = U(n) - \Delta B(n+1) \tag{5.23}$$

$$f(\tau) = \beta \cdot \tau^{-1.5} \tag{5.24}$$

The external shock $S(n)$ is added a a periodic interaction function

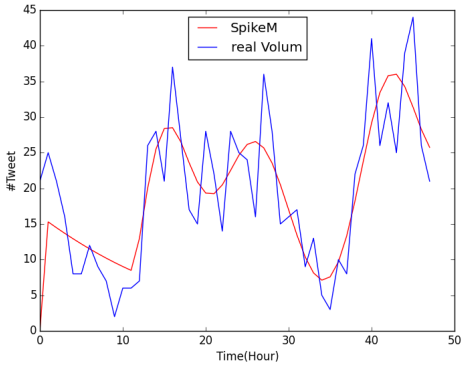
$$\bar{S}(t) = S(t) + q(t) \tag{5.25}$$

$$q(t) = q_a(\sin(\frac{2\pi}{q_p}(t + q_s))) + 1) \tag{5.26}$$

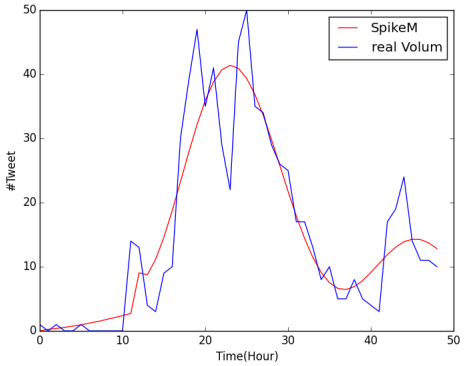
Symbol	Definition
q_a	strength of periodicity of the external shock
q_p	period of periodicity of the external shock
q_s	phase shift of periodicity of the external shock

Table 5.3: New Parameters of extended SpikeM

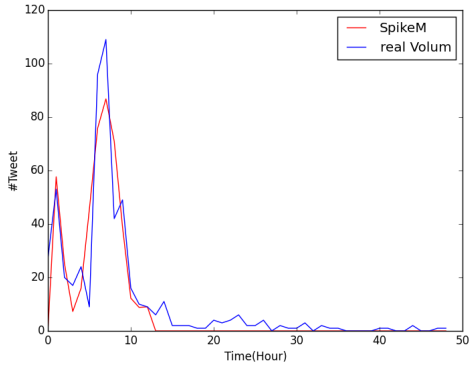
As the same approach of fitting SIS model, we learn the parameters of SpikeM model with Levenberg-Marquard algorithm. We fit the sequenced tweet volume from the beginning time the t_0 to the current time interval t_n of an event to the model and use the output parameters as the features adding into DSTS. We use the p_a, p_p, p_s and q_a, q_p, q_s as feature. We show 4 examples of the SpikeM fitting result in figure 5.5. But same problem as fitting SIS or SEIZ, if we test only within 10 hours data the result seems much worse than the result with full 48 hours showing in figure 5.6.



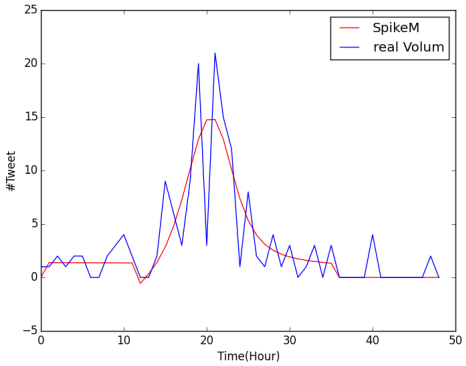
(a) SIS and SEIZ model for rumor 1



(b) SIS and SEIZ model for rumor 2

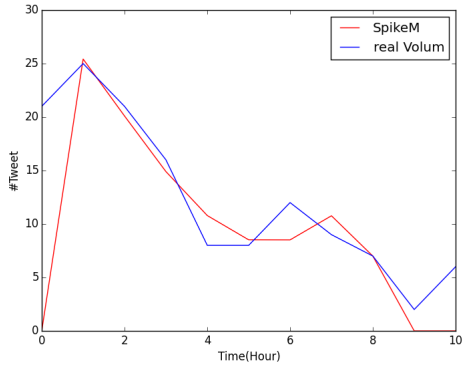


(c) SIS and SEIZ model for news 1

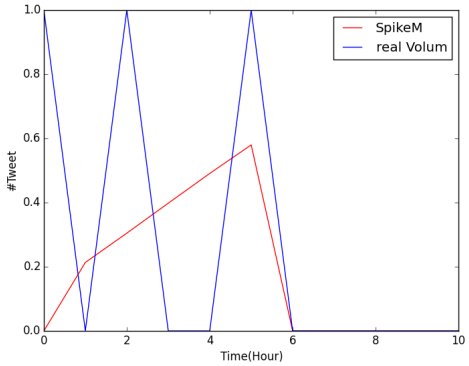


(d) SIS and SEIZ model for news 2

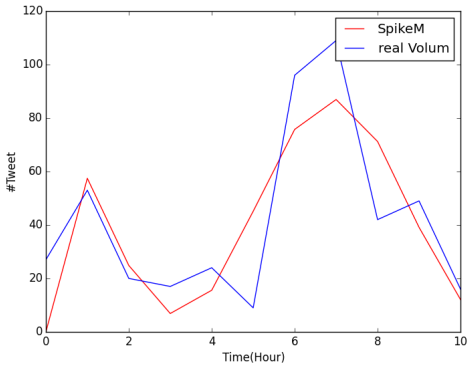
Figure 5.5: Fitting results of SpikeM model of (a) Rumor: Robert Byrd was a member of KKK (b) Rumor: CNN altered a photograph of a shooter making him look white (c) News: Doctor announces Michael Schumacher is making process (d) News: Two U.S. sailors are arrested over an alleged rape of a Japanese woman on Okinawa



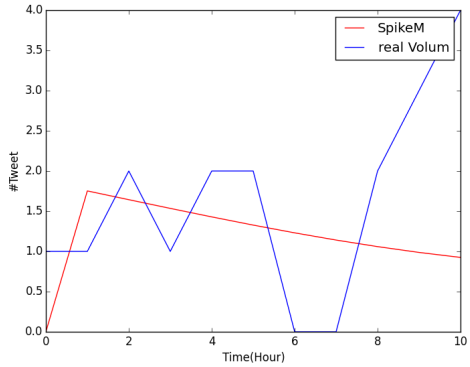
(a) SIS and SEIZ model for rumor 1 with 10 hours data



(b) SIS and SEIZ model for rumor 2 with 10 hours data



(c) SIS and SEIZ model for news 1 with 10 hours data



(d) SIS and SEIZ model for news 2 with 10 hours data

Figure 5.6: Fitting results of SpikeM model with first 10 hours data (same stories as above)

5.2.6 Crowd Wisdom Features

The idea come from Liu's work [25] but not same. The core idea is using the public's common sense to detecting the rumors. If there are more people denying or doubting the truth of an event, this event are more likely to be a rumor. In the Liu's work he uses a extensive list of positive, negative and negation keywords and a set of rules like âĂIJnegative words without negation words means the poster denies the event". And he uses the ratio number of positive poster (supporter) to the negative poster (deny the events).

Our work is simpler than than his work. We have only a set of negative words, we call it "debunking words" like hoax, rumor, not true, etc. In our test, this is a good feature, but it needs 17 hours to "warm up". It is logical because crowds can debunk rumors but crowds use time to react the event.

5.2.7 CreditScore Features

This feature is new feature. We our trained single tweet's creditability model the predict the tweets of the events. If the output is rumor we label it 1 otherwise 0 and we calculate the average score of the events in certain hour. We call this feature creditsScore. We will show it late, this is the best of our dataset, it improves the performance of time series model especially in the first 12 hours. When the event begin bursting in the early stage, people can only rely on the information from the single tweet itself. Because there is no clear propagation structure (pattern) or the wise men or journalists deny the event yet. Our neural network model "sees and check" the text of a single tweet and "give us the advise" when we have no other features in the first few hours since the beginning of the events.

The result shows this feature is the best feature.

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
	ContainNEWS	% 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	UserNumFollowers	average number of followers
	UserNumFriends	average number of friends
	UserNumTweets	average number of users posted tweets
	UserNumPhotos	average number of users posted photos
	UserIsInLargeCity	% of users living in large city
	UserJoinDate	average days since users joining Twitter
	UserDescription	% of user having description
	UserVerified	% of user being a verified user
	UserReputationScore	average ratio of #Friends over (#Followers + #Friends)
Epidemiological Features	β_{SIS}	Parameter β of Model SIS
	α_{SIS}	Parameter α of Model SIS
	β_{SEIZ}	Parameter β of Model SEIZ
	b_{SEIZ}	Parameter b of Model SEIZ
	l_{SEIZ}	Parameter l of Model SEIZ
	p_{SEIZ}	Parameter p of Model SEIZ
	ε_{SEIZ}	Parameter ε of Model SEIZ
	ρ_{SEIZ}	Parameter ρ of Model SEIZ
	R_{SI}	Parameter R_{SI} of Model SEIZ
SpikeM Model Features	P_s	Parameter P_s of Model Spike
	P_a	Parameter P_a of Model SpikeM
	P_p	Parameter P_p of Model SpikeM
	Q_s	Parameter Q_s of Model SpikeM
	Q_a	Parameter Q_a of Model SpikeM
	Q_p	Parameter Q_p of Model SpikeM
Crowd Wisdom Features	CrowdWisdom	% of tweets containing "Debunking Words"
Credit Score Features	CreditScore	average CreditScore

Table 5.4: Features of Time Series Rumor Detection Model

5.3 Classification Models

Same reason as the single tweet's Creditability we test the time series model also with 3 popular models Decision Trees, SVM, Random Forest and one more model the multilayer perceptron (MLP). We show the optimized parameters in the table 5.5. After 10-fold cross validation with same shuffled sequence testing above models, the result is shown the figure 5.6. The random forest is still the best model for our task, so we use RF as the test model for the further features' Evolution.

Model	**Parameters**	**Value**
Random Forest	Number of Trees	350
SVM	kernel penalty parameter of the error term gamma	radial basis function 3.0 $\frac{1}{50}$
Decision Trees	criterion	gini
MLP	alpha activation function hidden layer sizes weight optimization	0.0001 ReLU 2 layer(50 nodes each layer) adam

Table 5.5: Parameters of Classification models

Model	Accuracy in hours			
	6	12	24	48
Random Forest	**0.8615**	**0.8615**	**0.8692**	**0.9076**
MLP	0.7423	0.7423	0.7692	0.8192
SVM	0.7423	0.7884	0.7769	0.7538
Decision Trees	0.7807	0.8115	0.75	0.7385

Table 5.6: Prediction Accuracy of Different Single Tweet's Creditability Scoring Models

5.4 Experiment Result

5.4.1 Time Series VS Static Feature

First we compare our time series model and the normal static feature model. We show the result is in table 5.7 the full 48 hours details in Appendix table .2 and in figure 5.7. As we can see from the result that the accuracy of time Series in most of time is better than the static model. But after 24 hours the advantage of the time series is very limited. The reason may be after 24 hours the static model already has enough data to ignore the offset of features at the different time points. But the time series model still has the benefit of detecting rumors at the beginning of spreading.

Hour	Time series model	Static model	Difference
1	0.82	0.78	0.03
6	0.86	0.8	0.06
12	0.87	0.83	0.04
18	0.88	0.83	0.05
24	0.87	0.85	0.01
30	0.88	0.85	0.03
36	0.87	0.86	0.01
42	0.88	0.88	0
48	0.89	0.87	0.02

Table 5.7: Accuracy: Time series VS static Features

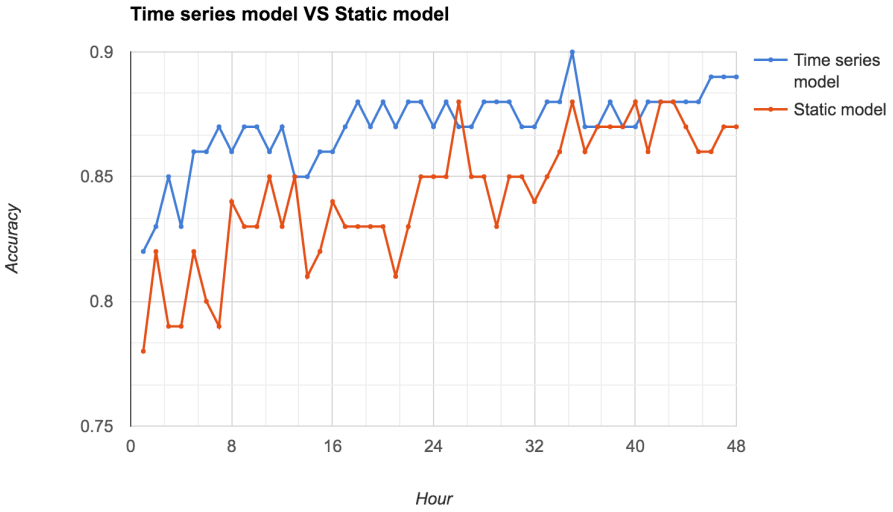


Figure 5.7: Accuracy: Time series VS static Features

5.4.2 Feature Analyzing Over Time

We rank the features' importance using the method we introduced in section 2.4.3, the full result is shown in appendix table .1. First we split the features in 7 catalogues as in table 5.4: Tweet_Feature, User_Feature,Text_Feature, Credit Score, SpikeM Features, Epidemiological Features, CrowdWisdom and the BestSet. The BestSet is a combination of the 28 top best average rank of 48 hours features, they are shown in table 5.8. The results over 48 hours are in figure 5.9 .

Best Feature set	
CreditScore	ContainNEWS
NumChar	UserTweetsPerDays
QuestionExclamation	UserReputationScore
WotScore	Question
UserJoin_date	Mention
LengthOfTweet	DebunkingWords
UserFollowers	Exclamation
UserVerified	Hashtag
Capital	You
UserNumPhoto	numUrls
UserFriends	NumPositiveWords
Via	P_a
UserIsInLargeCity	PolarityScores
UserDescription	R_{SI}

Table 5.8: Best Features

As we can see in figure 5.9 the best result on average over 48 hours in the *BestSet* with top 28 features. Second is the *All features*. Except those two the best group feature is *Text feature*. One reason is the text feature set has the largest group of feature with totally 16 features. But if look into each feature in text feature group we can see the best and the worst features are all in this set. *User feature* and *Twitter feature* are stable over time around 82%. The performances of 3 different models (SIS, SEIZ and SpikeM) describing the propagation pattern of rumors and news are not so well especially within 24 hours. *Crowd Wisdom* and *Credit Score* both contain only one feature but they already have impressive result comparing with the *User feature* and *Twitter feature*.

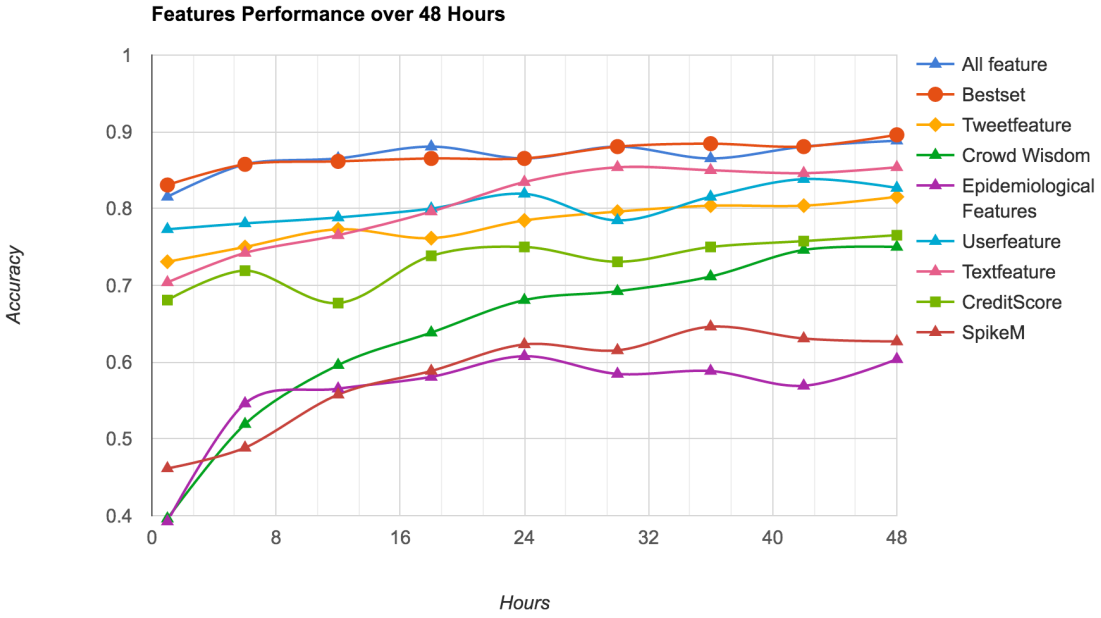


Figure 5.8: Accuracy: Time series VS static Features

5.4.3 Text Features

Text feature set contains 16 features. The ranks of feature as shown in table 5.10. The best one is *NumChar* which is the average number of different characters in tweets. It is quite hard to explain why it is like that.

PolarityScores is the best feature when we tested the single tweets model, but its rank in time series model is not so good. It is true that rumor contains more negative sentiment, but in an event (rumor or news) people can show their different views about this event [30] [39] like discussing or denying. *PolarityScores*'s performance becomes worse and worse over time. In table 5.9 we reference the result from Thomas Heverin's work [12]. He analyzed the tweets which responded to the 2009 Violent Crisis. The number of tweets containing original information and emotion are less and less over time. In the other hand more and more people start to share their different opinions even technology problems after 24 hours. That may make difference of sentiment features between rumors and news less and less in first 12 hours and it becomes useless after 24 hours.

Text feature overall is the the best feature set.

Time Period	Information	Opinion	Technology	Emotion	Action	Other
0-12hours	90.0%	6.8%	1.1%	5.6%	1.1%	0.0%
12-24 hours	86.6%	13.0%	3.1%	4.5%	1.3%	0.7%
24-36 hours	73.9%	18.3%	7.0%	2.7%	0.5%	3.7%
26-48 hours	74.6%	21.3%	1.0%	3.8%	0.5%	2.8%

Table 5.9: . Percentage of tweet type (non-exclusive) per 12 hour time period (source: [12])

Features	**Ranks**									
Hours	1	6	12	18	24	30	36	42	48	AVG
NumChar	3	3	4	3	3	8	7	6	4	4.29
QuestionExclamation	25	16	2	1	1	1	3	7	5	4.79
Question	15	11	13	7	5	4	8	5	8	8.29
LengthOfTweet	6	6	9	6	14	16	13	16	13	11.96
PolarityScores	12	15	23	28	33	33	34	31	32	28
Stock	34	44	47	47	47	47	47	47	48	46.15
Smile	35	45	45	48	48	48	48	48	48	47.06
Sad	36	46	46	49	49	49	49	49	49	47.9

Table 5.10: Rank of Part of Text Feature

5.4.4 Twitter Features

Twitter feature is stable over time from the beginning to the end.

The 3 best of *Tweet Features* are all the features about the contained ULR in tweet: *ContainNEWS, UrlRankIn5000, WotScore* showing in table 5.11. It it quite logical that the news will have higher probability reported by news websites or higher ranked website. And it is clear to see the that their performance significantly improve after 24 hours. In Alexander's work [31], he also shows similar phenomenon that credibility of the information from twitter is higher than external website in the first 24 hours.

But the other original twitter functions like the retweets or mention do not contribute much.

5.4.5 User Features

The performance of user features is similar with the twitter, they are both quite stable from the first hour to the last hour. But one difference is in the first few user feature is the best feature set except all feature set.

As in table 5.12 shown, the best feature of user feature is UserTweetsPerDays. And it is the best feature overall in the first 4 hours, but it drops down over time.

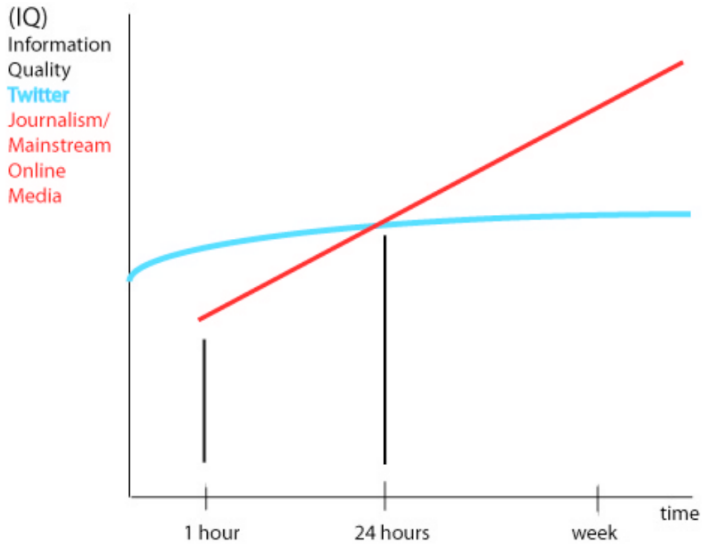


Figure 5.9: Information Quality over time (source: [31])

Features	**Ranks**									
Hours	1	6	12	18	24	30	36	42	48	AVG
ContainNEWS	8	4	5	4	4	2	2	2	2	3.48
UrlRankIn5000	14	13	7	11	7	3	1	4	6	5.96
WotScore	4	10	6	10	10	6	9	8	7	7.63
Mention	13	5	10	14	13	10	12	12	10	10.98
Hashtag	20	20	15	18	16	13	15	17	17	17.46
Retweets	21	21	27	38	42	35	31	37	34	33.25

Table 5.11: Rank of Part of Twitter Feature

Others user features like Reputation Score and Join date have also better performance at the first fews hours.

That means the sources (the poster in the first few hours) of news and rumors are quite different with each other, but more and more users join in the discussion, so the bias of two groups of user is less and less. After 6 hours we distinguish the rumors basing on the content of the tweet(text features) already better basing on the feature of the poster.

Features	Ranks									
Hours	1	6	12	18	24	30	36	42	48	AVG
UserTweetsPerDays	0	1	1	2	2	9	5	10	14	4.63
UserReputationScore	1	2	3	5	6	5	6	3	3	5.06
UserJoin_date	5	8	8	8	12	14	16	11	9	10.58
UserVerified	24	17	12	16	17	12	11	14	19	16.25

Table 5.12: Rank of Part of User Feature

5.4.6 SpikeM Features and Epidemiological Features

The two feature sets of these two models seem not so well. Only one feature P_α from the SpikeM is added in the BestSet features. As the problem of these two models which we have already figured out in section 5.2.4 is Both of these models need enough data to fit the parameters. Only after 24 hours these two models' features can reach 60% accuracy. In other words before 24 hour these is no clear propagation pattern of these events. In the work of Kwon [22], the durations of dataset which he uses are more than (wenxian). In the work of Jin[15], he uses () to fit the SEIZ models. Their data's durations are far larger than ours 48 hours.

P_α parameter from SpikeM is the only one feature in the top 26 feature set. It stands for the strength of periodicity. Kwon add 3 more parameters to explain the periodicity of the external shock. They are not so functional because of short time period.

Features	Ranks									
Hours	1	6	12	18	24	30	36	42	48	AVG
P_α	29	28	34	30	33	24	23	21	23	25.75
R_{SI}	47	24	30	23	36	39	38	24	30	29.56
β_{SIS}	49	30	33	28	31	36	28	33	25	30.15
Q_α	44	47	47	21	38	40	44	41	33	38.04

Table 5.13: Rank of Part of SpikeM Features and Epidemiological Features

5.4.7 Credit Score

Credit Score is the pre-trained Single Tweet Credibility Scoring model's output in section 4.2. As shown in table 5.15, excepting the first 4 hours Credit Score is the best feature overall. In figure 5.10 we show the result of model without Credit Score feature and model with full features set. Before the first 24 hours the model without credit score has worse performance than the full feature set, so the credit score feature contributes much for the early stage of rumor detecting.

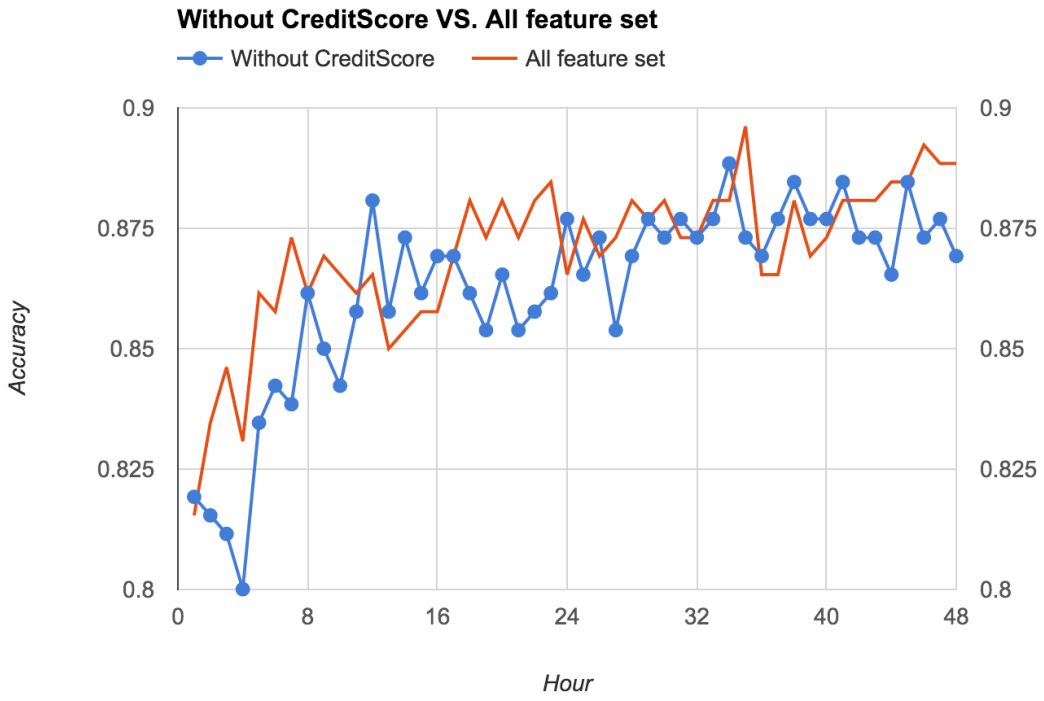


Figure 5.10: Accuracy: Time series VS static Features

Hour	Rank
1	2
2	1
3	1
4	1
5	0
..	0
48	0

Table 5.14: Ranks of CreditScore

5.4.8 Crowd Wisdom

Crowd Wisdom is also a good feature which can get 75.8% accuracy as a single feature. But its performance is very poor (less than 70%) in the first 32 hours. The crowds need time to unify their views to the event after absorbing all kind of information. That is also one reason we need this automatic detecting system.

5.4.9 Machine vs Human

Our system is meaningless if the system detecting the rumors after the human recognize them. In this section we will compare our system with the human rumor debunking website snopes.com.

Snopes.com has their own twitter account [2]. They will post tweets via this account about rumors. We consider the creation time of the first tweet which contains the keyword "snopes" in the tweets or URL of "snopes.com" is the time stamp of human confirming rumors.

But some of the rumors have a long duration, the website may report it several months ago or later. We think that doesn't refer to the same rumor affair. So we set up a threshold 72 hours. We only consider the time stamps of human confirming rumors 72 hours before or after the beginning time of the events which we defined in section 2.3.2. The result is in table. On average the editors of "snopes.com" need 25 hours to verify the rumors and post it. On hour 25 our system achieves 87% accuracy.

	Hours
Least	71
Earliest	-55
Average	25.49

Table 5.15: Time of Human Confirming Rumors

[2] https://twitter.com/snopes

6 Outlook

6.1 Conclusion

With more nations working actively to preserve the web due to the importance it plays in our lives, scholars from various fields have started looking to web archives as a data source waiting to be analyzed. We find that scholarly research with web archives carried out until today can be broadly classified as link based and content based studies. The link structure of web archives has been studied on a quantitative scale by Internet research institutes and computer scientists. They either study the link structure over time to analyze the evolution of the web or try to correlate the linking structure to real world phenomena. In this thesis we first looked at the usage of web archives by scholars. According to Brügger there are 4 major steps when conducting any type of research on web archives: corpus creation, analysis, dissemination and storage. An ideal system for web archive research should support all 4 steps however we find that adequate steps are only now being made to help researchers with the first step: corpus creation. Humanities scholars are usually interested in doing small scale qualitative and quantitative studies for which they need to explore web archives for relevant material. But the access to web archives with current web archive access systems like the wayback machine and rudimentary keyword search is not satisfactory. There are many other factors as well that have led to the lack of humanities studies on web archives. We attempted to illicit these through literature surveys and group discussions with humanities scholars conducting web archive research. Some of the interesting outcomes of this study were:

- The absence of abilities to explore the web archive discourage scholars from pursuing their ideas. Search engines are the predominant way of exploring the web today whereas archives seem to be left behind.
- The wayback machine is not an effective tool for building a corpus unless you

know the exact set of URLs you are interested in. Full text search should be provided for researchers who want to explore the archive.

- More details of the crawl should be exposed so that scholars can motivate their corpus of study better. The details of the crawl strategy should be added to the meta data of all documents.
- Scholars are used to working with well curated archives. Web archives pose greater burden for scholars in corpus creation since they need to curate the materials themselves.
- Due to the inconsistent nature of web archives, an error margin should be defined and an approximation of confidence as well even for small scale qualitative studies.
- Algorithms and features used to help scholars find new documents should be made transparent to users so that they can effectively document the corpus creation process.
- A web archive search system should allow for the corpus and its history of creation to be exported in standard formats researchers are used to working with.
- Scholars are interested in leveraging digital methods to analyze their data but find it hard to find tools flexible enough for their needs.

Scholars also saw potential in the usage of web archives as a playground for combining qualitative and quantitative studies. Based on our discussions with the scholars who attended the summer school in Aarhus University, we found that many would be interested in the ability to scale results up. They also bemoaned the lack of analysis tools currently available to work with web archives. The lack of technical expertise was clearly affecting a scholar's ability to conduct his research even though they knew what kind of analysis they wanted to do. We proposed that computer scientist should bridge this gap by either providing tools to support scholars or work with them on scaling up their hypothesis using pattern recognition and data mining techniques.

Initiatives are being made kick-start humanities research in web archives but to do the necessary infrastructure to first access web archives has to be in place. These initiatives are currently being carried out by national libraries and institutions like the Internet Archive and the IIPC. The BUDDAH project, a joint effort between the British Library and leading Universities in the UK, is working towards building a web archive search system by working in tandem with humanities scholars working with the UK web archive. We held discussions with members of the BUDDAH project and scholars to identify the pitfalls with the current web archive search system

they were working with. The British Library web archive search prototype already provides its users with keyword search and filtering based on domain and crawl date. As noted by Costa et. al. current IR techniques like keyword search do not fare well in web archive search. We observed this first hand with the British Library's choice to rank search results by crawl date. This makes it very difficult for users to easily comprehend a large result set. We decided to better understand a scholar's search intent by analyzing written descriptions of proposed studies on web archives by humanities scholars involved in the BUDDAH project. We found that scholars are interested in studying results over time and covering a variety of aspect for their topic.

To this end we introduced the notion of a historical query intent over longitudinal collections like web archives. Historical query intent was modeled as a 2 dimension diversification problem where one dimension is time and the other aspects. In this thesis we limit ourselves to a subset of web archives: the news archive. We proposed a new evaluation metric Tia-SBR to evaluate the performance of retrieval models for this task. We also adapted existing diversity metrics to take time into account. To evaluate the retrieval models we built a new temporal test collection based on 20 years of the *New York Times* collection. We strengthened existing 1 dimensional diversity methods by linearizing the aspect-time space into a single set. We also consider temporal diversity retrieval models in our experiment. We introduced HistDiv which shows improvements over temporal and non-temporal methods for most of the time-aware diversification methods. We also outperform all competitors in Tia-SBR showing the suitability of our approach for historical query intents. We observe that HistDiv works well for topics which have aspects that span across multiple time intervals and have fluctuating importance at different times. It trades-off nicely between important aspects and important times which we perceive as important in historical search. HistDiv does not perform quite as well for queries which only one dominant aspect at a certain time window. We also described the architecture of the History Search system which consists of: a REST API to access the retrieval models used in our experiments, a user interface designed specifically to show results as newspaper articles and a time line to provide an overview and ability to filter.

6.2 Future Work

The performance of the HistDiv algorithm encourages us to look beyond the traditional approaches and devise methods more suited to archives. HistDiv currently considers time as a set of disjointed windows. It will be interesting to see if bursts rather than time windows are more effective for historical query intents. In our current implementation, subtopics are mined using wikiminer which is a relatively naive approach. In the future, we want to experiment with different types of aspect

mining like LDA for instance. We also want to add more dimensions to the model like geographic locations. Just like aspect utility decays based on time, we can also decay it based on location. The absolute distance between lat long coordinates can be used or a taxonomy based approach.

The results also indicate where HistDiv performs badly and other algorithms perform well. We believe that if the user is presented with a choice of algorithms, she can effectively select the algorithm best suited to her need. A step beyond this would be to train a classifier that selects the appropriate retrieval model based on features from the query's aspect and temporal profile.

The test collection we built is still relatively small when compared to collections by TREC. We must strive to add to this test collection, by increasing the workload, or explore different avenues for evaluating retrieval models meant for archive search. For our test collection it will also be interesting to get actual historians to provide topics. More research is also needed to construct test collections for other query intents for archives.

Having performed admirably in a news archive test collection, it will be interesting to test HistDiv and the other methods on a web archive or at least a subset which is not just news. Existing web crawl datasets only span a small period of time but it might be interesting nonetheless to experiment with smaller time window sizes for these collections. Another important temporal feature that we want to consider is the temporal references within an article. It is reasonable to assume that major bursts are usually covered by summary articles after the end of the burst. These articles are valuable but may get neglected because they do not occur within the burst based on our current algorithm. HistDiv's ability to find documents from important time windows and aspects can also be applied in Temporlia's query classification task to interesting effect.

There is also much to be done to help scholars better engage with web archives. In this thesis we have only scratched the surface of the issues surrounding intelligent access to web archives. We must work together with scholars in order to develop effective algorithms and user friendly systems so that the researchers of tomorrow can conduct their studies with far greater ease than today.

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] 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*, pages 1472–1473. ACM, 2013.

[4] 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.

[5] L. M. Bettencourt, A. Cintrón-Arias, D. I. Kaiser, and C. Castillo-Chávez. The power of a good idea: Quantitative modeling of the spread of ideas from epidemiological models. *Physica A: Statistical Mechanics and its Applications*, 364:513–536, 2006.

[6] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

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

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

[9] 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.

[10] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[11] 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.

[12] T. Heverin and L. Zach. *Microblogging for Crisis Communication: Examination of Twitter Use in Response to a 2009 Violent Crisis in the Seattle-Tacoma, Washington, Area*. ISCRAM, 2010.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65. ACM, 2007.

[15] 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.

[16] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[17] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

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

[19] A. Kohut and M. Remez. Internet overtakes newspapers as news outlet. *Pew Research Centre*, 2008.

[20] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM, 2010.

[21] 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.

[22] 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.

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

[24] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] C. Matthews. How does one fake tweet cause a stock market crash. *Wall Street & Markets: Time*, 2013.

[30] 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.

[31] A. Mills, R. Chen, J. Lee, and H. Raghav Rao. Web 2.0 emergency applications: How useful can twitter be for emergency response? *Journal of Information Privacy and Security*, 5(3):3–26, 2009.

[32] 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.

[33] C. Olah. Understanding lstm networks. *Net: http://colah. github. io/posts/2015-08-Understanding-LSTMs*, 2015.

[34] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.

[35] 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*, pages 1589–1599. Association for Computational Linguistics, 2011.

[36] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[37] 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.

[38] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[39] 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.

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

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] C. Zhou, C. Sun, Z. Liu, and F. Lau. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*, 2015.

Appendix

time	CreditScore	ContainNEWS	NumChar	UserTweetsPerDays	QuestionExclamation	UserReputationScore	UrlRankIn5000	WotScore	Question	UserJoin_date	Mention	LengthOfTweet	DebunkingWords	UserFollowers	Exclamation	UserVerified	Hashtag	Capital	You	UserNumPhoto	numUrls	UserFriends	NumPositiveWords	Via	P_a	UserIsInLargeCity	PolarityScores	UserDescription	R_{SI}	β_{SIS}	ξ_{SEIZ}	NumNegativeWords	P_s	b_{SEIZ}	β_{SEIZ}	Retweets	p_{SEIZ}	α_{SIS}	Q_s	P_p	ρ_{SEIZ}	I	Qp	l_{SEIZ}	Q_a	HeShe	IsRetweet	Stock	Smile	Sad
1	2	8	3	**0**	25	1	14	4	15	5	13	6	33	7	27	24	20	10	28	9	22	11	18	30	29	19	12	23	45	42	39	17	31	43	46	21	40	41	47	44	49	16	38	37	48	26	32	34	35	36
2	1	5	2	**0**	18	3	11	4	12	7	15	6	43	8	36	22	21	14	40	9	30	10	23	34	25	24	13	38	31	28	41	16	29	27	20	19	39	32	48	26	35	17	47	33	49	37	42	44	45	46
3	1	4	2	**0**	17	5	12	3	10	8	11	6	44	9	38	24	23	15	41	7	30	13	20	32	25	22	14	39	21	26	27	18	36	37	35	19	29	33	48	28	40	16	47	31	49	34	42	43	45	46
4	1	5	3	**0**	16	2	13	4	14	7	11	9	43	6	39	34	21	12	30	8	37	10	23	32	26	25	15	41	28	33	31	17	19	22	27	24	38	40	48	18	29	20	47	35	49	36	42	44	45	46
5	**0**	4	3	1	16	2	13	10	11	8	5	6	42	7	38	17	20	14	33	9	36	12	22	28	35	30	15	41	31	24	23	18	32	25	34	21	29	19	48	37	40	27	47	26	49	39	43	44	45	46
6	**0**	4	2	1	14	3	11	9	8	10	7	6	41	5	35	29	22	15	34	12	40	16	17	21	28	25	18	42	24	30	13	19	36	20	27	26	23	37	48	33	31	32	47	39	49	38	43	44	45	46
7	**0**	6	2	1	7	3	11	12	10	8	4	5	23	9	22	28	21	15	39	13	37	14	20	19	27	26	18	42	24	38	17	16	34	35	25	31	29	41	48	30	33	32	47	40	49	36	43	44	45	46
8	**0**	4	2	1	5	3	7	9	8	10	13	6	24	11	18	15	21	14	41	12	34	19	20	31	28	17	22	40	23	29	27	16	25	35	33	26	32	37	48	30	36	38	47	39	49	42	43	44	45	46
9	**0**	8	2	1	4	3	7	5	10	6	13	9	25	11	15	14	23	12	32	16	28	19	22	24	26	33	20	37	31	30	18	17	41	35	29	21	27	36	48	39	34	38	47	40	49	42	43	44	45	46
10	**0**	8	3	1	2	4	5	10	11	6	12	7	18	9	17	19	20	13	32	14	26	24	23	27	29	36	16	39	41	30	21	15	35	28	34	31	22	25	48	38	37	33	47	40	49	43	42	44	45	46
11	**0**	5	4	1	2	3	7	6	13	8	10	9	19	11	17	12	15	14	24	16	18	20	25	31	33	32	23	40	26	29	36	21	37	39	22	27	38	30	48	35	42	28	47	34	49	41	43	44	45	46
12	**0**	6	3	1	2	4	7	5	13	8	9	11	18	12	16	15	17	10	21	20	23	14	28	27	34	25	19	41	30	33	36	31	40	26	37	24	32	22	48	35	29	39	47	38	49	42	43	44	45	46
13	**0**	7	3	1	2	4	9	5	12	6	8	11	15	13	22	14	17	10	35	18	20	16	31	27	36	24	23	43	40	30	26	34	37	28	32	21	29	19	42	33	25	38	39	41	44	45	46	47	48	49
14	**0**	6	3	1	2	4	5	9	12	7	8	11	15	13	17	14	22	10	26	19	18	16	27	36	34	29	21	39	24	23	25	33	41	28	32	20	40	42	31	35	30	44	38	37	43	46	45	47	48	49
15	**0**	5	4	1	2	3	6	13	8	7	11	10	12	16	9	15	27	14	20	26	19	18	22	25	32	34	30	40	24	37	17	21	38	33	41	28	35	44	29	39	23	43	36	42	31	46	45	47	48	49
16	**0**	4	3	2	1	5	7	10	11	9	12	6	13	17	8	15	21	14	20	22	23	24	43	36	18	30	29	39	33	37	16	28	41	26	38	32	31	34	35	44	19	42	40	27	25	45	46	48	47	49
17	**0**	4	3	2	1	5	11	10	7	8	14	6	12	17	9	16	18	13	15	21	20	19	44	26	35	43	28	31	22	32	36	34	37	39	40	38	30	23	25	41	27	33	24	29	42	45	46	47	48	49
18	**0**	5	3	2	1	4	9	8	7	10	12	11	14	16	6	15	17	13	18	24	20	19	40	26	30	32	29	35	23	28	25	31	41	27	38	37	33	36	34	43	39	44	22	42	21	45	46	47	48	49
19	**0**	4	3	2	1	7	8	9	6	11	12	10	13	15	5	18	17	14	16	24	19	20	36	26	22	38	29	43	28	33	35	30	23	32	25	41	27	39	34	40	37	42	21	31	44	45	46	47	48	49
20	**0**	3	4	2	1	7	5	6	8	13	12	11	10	16	9	17	18	15	14	21	20	23	35	33	28	30	41	29	22	27	39	36	37	43	32	31	25	24	38	40	42	46	19	26	34	45	44	48	47	49
21	**0**	4	3	2	1	7	6	8	10	14	12	13	5	17	9	18	15	16	11	27	20	19	30	35	24	33	29	28	25	22	44	37	26	31	36	38	40	21	39	32	42	43	23	41	34	46	45	47	48	49
22	**0**	4	3	2	1	7	6	8	11	14	12	13	5	18	9	17	15	16	10	20	21	22	26	33	24	36	30	28	35	25	38	27	23	37	41	43	31	19	40	42	29	39	34	44	32	46	45	47	48	49
23	**0**	4	3	2	1	6	7	10	5	12	13	14	9	15	8	17	16	18	11	19	24	23	34	35	25	43	33	32	27	38	29	39	21	41	28	42	37	40	22	30	36	46	20	26	31	44	45	47	48	49
24	**0**	3	5	2	1	6	4	14	7	13	10	11	9	17	8	16	15	19	12	20	18	26	25	24	33	32	30	22	36	31	35	37	23	39	41	40	42	38	21	27	34	45	29	43	28	44	46	47	48	49
25	**0**	3	4	2	1	10	7	9	5	13	11	14	6	17	8	19	18	15	12	22	16	30	23	20	37	26	29	24	21	32	31	40	41	28	39	42	36	35	27	38	43	44	25	34	33	45	46	47	48	49
26	**0**	4	6	2	1	8	3	9	7	12	14	13	5	18	11	16	15	17	10	22	20	24	19	32	23	28	36	21	26	37	31	34	27	33	43	44	29	35	25	39	41	38	40	42	30	46	45	47	48	49
27	**0**	3	8	4	1	7	2	9	6	15	12	13	5	17	14	16	11	18	10	22	21	29	19	25	23	24	33	20	26	30	36	31	34	42	27	44	38	40	28	32	41	37	45	39	35	43	46	47	48	49
28	**0**	1	7	9	2	8	3	6	4	14	10	15	5	17	13	16	11	21	12	23	19	27	20	26	35	24	29	18	32	30	22	38	25	28	41	42	44	31	36	37	43	40	34	39	33	46	45	47	48	49
29	**0**	2	8	9	1	5	3	6	4	14	10	16	7	17	15	12	13	22	11	23	21	24	19	25	18	32	33	20	31	34	27	36	28	44	39	35	41	38	26	29	42	40	37	43	30	46	45	47	48	49
30	**0**	3	8	7	1	9	2	5	4	14	10	16	6	17	13	15	12	20	11	22	21	31	18	23	24	25	29	19	39	26	27	32	36	38	33	34	43	37	35	28	44	30	42	41	40	45	46	47	48	49
31	**0**	3	9	4	2	7	1	5	6	16	10	15	8	20	13	14	12	21	11	22	18	24	23	29	17	34	28	19	25	26	30	37	31	36	35	32	44	39	27	46	43	38	41	40	33	45	42	47	48	49
32	**0**	2	6	4	3	7	1	9	5	16	10	15	8	17	12	14	13	21	11	22	18	24	23	30	19	26	33	20	28	25	32	42	31	38	34	27	36	44	40	41	43	39	35	29	37	45	46	47	48	49
33	**0**	2	4	7	3	8	1	6	11	13	10	15	5	17	12	9	14	19	16	22	20	30	21	24	25	26	42	18	29	23	36	37	33	35	32	27	43	31	28	40	34	39	41	38	46	44	45	47	48	49
34	**0**	1	5	3	4	6	2	8	12	16	9	15	7	13	11	10	17	23	14	24	20	27	21	25	19	28	36	18	37	26	32	33	22	39	34	31	45	29	38	40	35	42	43	41	30	44	46	47	48	49
35	**0**	2	7	5	3	6	1	9	8	16	12	13	4	10	14	11	15	24	17	21	19	38	20	22	27	23	34	18	32	30	37	33	28	25	26	31	40	45	29	42	36	35	44	43	39	41	46	47	48	49
36	**0**	2	6	7	5	4	1	8	9	14	12	15	3	13	10	11	18	25	16	24	19	30	21	20	23	22	32	17	38	28	31	37	27	39	26	36	42	45	33	29	40	35	34	43	44	46	41	47	48	49
37	**0**	1	6	9	3	5	2	7	8	12	11	14	4	16	10	13	18	27	15	23	17	26	21	24	20	22	32	19	39	25	28	34	30	43	29	41	45	33	38	31	35	40	36	44	37	46	42	47	49	48
38	**0**	1	6	8	4	7	3	5	9	10	14	16	2	15	12	11	18	26	13	22	23	30	24	19	21	20	35	17	29	31	38	44	28	25	32	37	46	36	33	27	34	40	41	45	39	42	43	48	47	49
39	**0**	1	5	10	4	7	3	6	9	13	14	16	2	8	12	11	17	27	15	23	24	32	21	20	22	19	39	18	25	26	33	37	28	29	36	41	34	35	38	30	31	45	40	46	42	43	44	47	48	49
40	**0**	1	7	13	4	6	3	8	5	11	12	17	2	9	14	10	16	27	18	26	24	28	23	20	22	19	33	15	45	31	38	40	25	36	39	32	34	41	21	30	35	46	37	43	29	42	44	47	49	48
41	**0**	2	6	10	7	3	4	8	5	11	12	16	1	9	15	14	17	29	13	21	23	26	25	18	22	19	31	20	32	35	40	42	27	33	28	37	44	46	24	30	41	39	36	34	38	45	43	47	48	49
42	**0**	1	4	11	8	3	6	7	5	9	10	16	2	12	14	13	19	27	15	20	25	29	23	18	21	17	35	22	24	33	40	45	26	30	32	43	36	31	39	37	38	42	34	46	41	44	28	47	49	48
43	**0**	2	3	11	7	4	6	8	5	9	10	15	1	16	12	14	17	28	13	21	23	30	24	18	22	19	32	26	25	31	35	45	36	27	20	44	37	42	29	43	41	38	33	34	39	46	40	47	48	49
44	**0**	2	4	10	5	6	7	8	3	9	11	15	1	14	12	16	18	26	13	23	22	27	24	20	19	17	31	21	25	29	37	40	36	32	30	44	38	43	35	39	28	46	41	33	34	45	42	47	49	48
45	**0**	2	3	10	6	4	7	8	5	9	11	16	1	14	12	15	19	28	13	22	20	30	24	18	23	17	31	21	26	33	41	36	42	25	29	39	27	46	35	43	45	40	34	37	32	44	38	47	49	48
46	**0**	2	5	11	4	3	6	7	8	9	12	17	1	16	10	18	15	25	13	23	21	35	26	14	24	19	29	20	30	32	40	44	33	22	34	38	27	46	28	42	43	45	39	41	31	37	36	48	49	47
47	**0**	2	4	13	3	6	5	7	8	9	11	15	1	19	10	18	16	25	12	22	20	37	24	17	21	14	33	23	29	27	41	36	32	26	30	45	38	44	28	43	42	35	34	40	31	46	39	48	47	49
48	**0**	2	4	14	5	3	6	7	8	9	10	13	1	18	11	19	17	24	12	22	20	35	27	15	23	16	32	21	30	25	43	40	38	29	26	34	28	42	31	45	36	37	39	46	33	41	44	48	47	49
AVG	0.10	3.48	4.29	4.63	4.79	5.06	5.96	7.63	8.29	10.58	10.98	11.96	12.35	13.63	15.15	16.25	17.46	18.65	19.15	19.65	22.85	23.54	24.73	25.42	25.75	26.13	28.00	28.06	29.52	30.00	31.67	31.69	31.81	32.46	32.65	33.35	35.27	35.54	35.75	35.83	36.50	37.31	37.60	37.96	38.58	42.81	43.08	46.15	47.06	47.90

Hour	Time series model	Static model	Difference
1	0.82	0.78	0.03
2	0.83	0.82	0.01
3	0.85	0.79	0.05
4	0.83	0.79	0.04
5	0.86	0.82	0.04
6	0.86	0.8	0.06
7	0.87	0.79	0.08
8	0.86	0.84	0.03
9	0.87	0.83	0.04
10	0.87	0.83	0.04
11	0.86	0.85	0.01
12	0.87	0.83	0.04
13	0.85	0.85	0
14	0.85	0.81	0.04
15	0.86	0.82	0.03
16	0.86	0.84	0.02
17	0.87	0.83	0.04
18	0.88	0.83	0.05
19	0.87	0.83	0.05
20	0.88	0.83	0.05
21	0.87	0.81	0.06
22	0.88	0.83	0.05
23	0.88	0.85	0.04
24	0.87	0.85	0.01
25	0.88	0.85	0.02
26	0.87	0.88	-0.01
27	0.87	0.85	0.02
28	0.88	0.85	0.04
29	0.88	0.83	0.04
30	0.88	0.85	0.03
31	0.87	0.85	0.02
32	0.87	0.84	0.03
33	0.88	0.85	0.03
34	0.88	0.86	0.02
35	0.9	0.88	0.02
36	0.87	0.86	0
37	0.87	0.87	0
38	0.88	0.87	0.02
39	0.87	0.87	0
40	0.87	0.88	0
41	0.88	0.86	0.02
42	0.88	0.88	0
43	0.88	0.88	0
44	0.88	0.87	0.01
45	0.88	0.86	0.02
46	0.89	0.86	0.03
47	0.89	0.87	0.02
48	0.89	0.87	0.02

Table .2: Time seriers VS static Features in detail

time	All Feature	BestSet	Tweet Feature	Crowd Wisdom	SIR	User Feature	Text Feature	CreditScore	SpikeM
1	0.815	**0.831**	0.731	0.396	0.392	0.773	0.704	0.681	0.462
2	0.835	**0.838**	0.727	0.396	0.562	0.762	0.685	0.712	0.515
3	0.846	**0.850**	0.769	0.423	0.565	0.758	0.708	0.708	0.492
4	0.831	**0.854**	0.762	0.477	0.538	0.765	0.738	0.712	0.550
5	**0.862**	0.854	0.758	0.508	0.554	0.781	0.746	0.692	0.500
6	**0.858**	**0.858**	0.750	0.519	0.546	0.781	0.742	0.719	0.488
7	**0.873**	0.854	0.754	0.573	0.554	0.769	0.754	0.700	0.554
8	0.862	0.862	0.773	0.573	0.542	0.754	0.758	0.692	0.542
9	**0.869**	0.858	0.754	0.573	0.565	0.785	0.754	0.700	0.588
10	0.865	**0.881**	0.758	0.588	0.565	0.788	0.746	0.708	0.573
11	0.862	0.869	0.769	0.585	0.538	0.781	0.735	0.708	0.542
12	**0.865**	0.862	0.773	0.596	0.565	0.788	0.765	0.677	0.558
13	0.850	**0.873**	0.754	0.588	0.554	0.796	0.785	0.692	0.550
14	0.854	**0.865**	0.777	0.600	0.573	0.800	0.762	0.708	0.542
15	0.858	**0.865**	0.777	0.619	0.546	0.804	0.777	0.731	0.550
16	0.858	**0.862**	0.773	0.627	0.565	0.792	0.788	0.754	0.565
17	**0.869**	0.858	0.777	0.635	0.538	0.792	0.788	0.735	0.577
18	**0.881**	0.865	0.762	0.638	0.581	0.800	0.796	0.738	0.588
19	**0.873**	0.862	0.773	0.646	0.585	0.808	0.819	0.742	0.608
20	**0.881**	0.862	0.773	0.662	0.604	0.812	0.815	0.746	0.596
21	**0.873**	**0.873**	0.785	0.669	0.592	0.808	0.838	0.742	0.600
22	**0.881**	0.873	0.788	0.685	0.600	0.819	0.827	0.746	0.588
23	**0.885**	0.865	0.785	0.677	0.592	0.815	0.838	0.731	0.608
24	**0.865**	0.865	0.785	0.681	0.608	0.819	0.835	0.750	0.623
25	**0.877**	**0.877**	0.769	0.692	0.600	0.819	0.842	0.746	0.623
26	0.869	**0.881**	0.785	0.696	0.581	0.800	0.846	0.735	0.600
27	0.873	**0.888**	0.788	0.696	0.612	0.804	0.854	0.735	0.600
28	**0.881**	0.881	0.788	0.692	0.596	0.796	0.846	0.735	0.612
29	**0.877**	**0.877**	0.781	0.688	0.596	0.792	0.854	0.727	0.608
30	**0.881**	**0.881**	0.796	0.692	0.585	0.785	0.854	0.731	0.615
31	0.873	**0.888**	0.800	0.692	0.604	0.785	0.850	0.735	0.588
32	0.873	**0.892**	0.808	0.696	0.588	0.792	0.850	0.746	0.631
33	0.881	**0.892**	0.808	0.700	0.581	0.812	0.846	0.762	0.642
34	0.881	**0.896**	0.804	0.704	0.558	0.808	0.854	0.735	0.658
35	**0.896**	0.881	0.815	0.712	0.585	0.804	0.854	0.731	0.658
36	0.865	**0.885**	0.804	0.712	0.588	0.815	0.850	0.750	0.646
37	0.865	**0.885**	0.796	0.715	0.577	0.812	0.862	0.727	0.646
38	**0.881**	0.873	0.804	0.738	0.577	0.808	0.846	0.754	0.646
39	0.869	**0.888**	0.800	0.731	0.573	0.827	0.850	0.742	0.635
40	0.873	**0.885**	0.808	0.735	0.588	0.819	0.854	0.758	0.646
41	0.881	**0.892**	0.804	0.738	0.577	0.812	0.854	0.758	0.631
42	**0.881**	**0.881**	0.804	0.746	0.569	0.838	0.846	0.758	0.631
43	0.881	**0.888**	0.812	0.750	0.600	0.838	0.854	0.742	0.627
44	0.885	**0.896**	0.815	0.754	0.577	0.835	0.858	0.746	0.600
45	0.885	**0.900**	0.804	0.754	0.577	0.835	0.858	0.738	0.646
46	**0.892**	0.888	0.804	0.754	0.612	0.835	0.854	0.731	0.662
47	0.888	**0.904**	0.819	0.754	0.596	0.827	0.858	0.750	0.619
48	0.888	**0.896**	0.815	0.750	0.604	0.827	0.854	0.765	0.627

Table .3: Accuracy of Different Feature Sets over 48 Hours

List of Figures

List of Tables