Rumor Detection With Different Methods and Modified Methods

P.X. Huang
15307130283

G.Z. She
15307130224

C.G Zhang
15307130096

Abstract

Automatically identifying rumors from online social media especially microblogging websites is a hot issue in nature language processing. From different views, various theories have been proposed to realise it. In this paper, we try two different methods to identify rumors based on Sina Weibo data and Twitter data. On one hand, we implement a RNN model based on feature engineering, on another hand, we utilize the propogation structure to do this task.

1 Introduction

Rumor is a statement whose truth value is unverifiable or deliberately false. It could dramatically damage social stability and cause enormous losses. For instance, on April 23, 2013, a rumor on Twitter about two explosions in the white house injuring Barack Obama caused the stock market crash in the US. Therefore, it is crucial to track and debunk such rumors. Considering the cost of classifying rumors manually, automatically identifying rumors have become a hot issue.

Based on different characters of rumors, various methods have been proposed, for example, we could use content-based features to training machinery classifier to implement it. To realise and compare different models. Our work could be roughly divided into three part:

First, instead of extracting static features based on content, user and diffusion(C. Castillo and Poblete, 2011)(K. Wu and Zhu, 2015), we propose a novel time series model called Dynamic Series-Time Structure (DSTS)(Jing Ma and Wong, 2015) to capture the variation of a wide spectrum of social context information over time. In this part, we will build our SVM classifier using the refining DSTS-based features.

Second, To capture the dynamic temporal signals characteristic of rumor diffusion, we introduce deep learning method(RNN/LSTM) into our work. Going further than the work of Ma et al.(Jing Ma and Zhu, 2016), we modify our RNN model, and utilise the DSTS-based features extracted in previous part. In this part, We utilise two datasets from both Twitter and Sina Weibo which are the most popular blogs websites in English and Chinese, respectively.

Third, we transform our emphasis into utilising the propagation structure which was proposed by Ma et al.(Jing Ma, 2017), instead of extracting normal features, we introduce the kernel-based data-driven method called Propagation Tree Kernel(PTK) into our work to extract propagation-based features, and we also extend PTK into a context-enriched PTK(cPTK).

The first part could be seen as the fundamental of part two and a baseline of whole project. From a different perspective, the third method is introduced into our work, so we conduct comparison and analysis basing on the result of these three model.

2 Related Work

2.1 DSTS model

Our work bases on the Dynamic Series-Time Structure (DSTS) proposed by Ma, et al, and the details could be found in.(Jing Ma and Wong, 2015). Generally speaking, basing on the creating time of each post, we separate one sample(a serious of posts) into several events(i.e. posts groups), and one event contains all the posts in a certain time period. Then one sample can be transformed into a matrix: $V(E_i) = (\mathbf{F}^D_{i,0}, \mathbf{F}^D_{i,1} \ldots)$ where $\mathbf{F}^D_{i,k}$ is the feature of *time period* k in event i. After extracting features from these posts groups, we calculate the variate ratios ($\mathbf{S}^D_{i,t} = \frac{\mathbf{F}^D_{i,t+1} - \mathbf{F}^D_{i,t}}{Interval\{E_i\}}$) of every two events, where $\mathbf{S}^D_{i,t}$ is the slope of features between the t-th and (t+1)-th intervals, $Interval\{E_i\}$ is the interval length of E_i. Then a sample can be expressed as $V(E_i) = (\mathbf{F}^D_{i,0}, \mathbf{F}^D_{i,1} \ldots \mathbf{S}^D_{0,t}, \mathbf{S}^D_{1,t} \ldots)$. Then we can use these labelled samples to training classifiers.

2.2 RNN model

Our work based on the RNN models proposed by Jing Ma, 2016(C. Castillo and Poblete, 2011). As we all know that the performance of RNN model in deal with time series problem is well recognized. The propagation of an event are a time series problem, so we use the RNN model. First we can get variable length time series of an event, for the propagation of the events often focus on a period of time. If we use the centain time span, we will not only lose important information, but also have lower efficiency. Then from the content we can extract useful information by TF-IDF values. After that, we use word embedding methods to get the vector of the content which used as input of the LSTM and other GRU models. Finally use the RNN model to predict the rumor by training the model on training dataset.

2.3 PTK and cPTK model

2.3.1 Intuition

Representation

Each event and its propogation can be interpreted as a tree, the node represent the post, the directed link between two nodes represent one propogation route, this model can be visualized by us in picture 1. **Similarity**

According to our visualization, we can not tell a certain pattern from it, so the problem is attributed to how to quantify the pattern of the tree(propogation). The idea comes in this way, since the rumor is dynamically generated, so in this aspect we can not set a static basic pattern for our the propogation. So the idea should be attributed to comparison. so we can define the similarity between different trees

2.3.2 Modeling SVM

PTK model is easy to implement but is a little confused for readers. The rough idea is that the connnection of all the extant messages are not linearally correlated, and we can resolve this connection by nonlinear machine learning method like SVM.And we extract the similarity of trees as the features for each propogation tree, and then we seperate the training and testing data into SVM training, The main difference of the following two models are the definition of the similarity between two trees

PTK

Algorithm 1 The PTK Algorithm to caculate the similarity of two trees

```
1:  Input:Tree A, Tree B
2:  SETUP:
3:  $J(c_i, c_j) = \frac{Ngram(c_i) \cap Ngram(c_j)}{Ngram(c_j) \cup Ngram(c_j)}$
4:  ch(v, k): the kth child of the node v
5:  $\sigma(u_i, u_j)$:similarity(distance) of two users(encoded as word vectors)
6:  function $f(v_i, v_j)$: similarity of two nodes $v_i, v_j$in two trees
    $f(v_i, v_j) = e^{-t}(\alpha\sigma(u_i, u_j) + (1-\alpha)J(c_i, c_j))$
7:  $A(v, v') = f(v, v') \prod_{k=1}^{min(nc(v), nc(v'))} (1 + A(ch(v, k), ch(v', k)))$
8:  Initialization: sum = 0
9:  for node $v_i$ in tree A do
10:     Find the most proper node $v'_j$ of Tree B corresponding to tree A, which has the max similarity
11:     sum = sum + A($v_i, v'_j$), pay attention that this function is recursively called.
12: end for
13: for node $v_j$ in tree B do
14:     Find the most proper node $v'_i$ of Tree B corresponding to tree B, which has the max similarity
15:     sum = sum + A($v_j, v'_i$),
16: end for
17: Output: sum
```

cPTK

The slight difference between cPTK and PTK is the cPTK take the context of a node into consideration, to be more specific, we need to modify the f function to $\Sigma_{x=0}^{min(depth(i), depth(j'))} A(v_i[x], v'_j[x])$, where v[x] means the xth ancestor of v, in this way, we accumulate the effect of the pre-propogation.

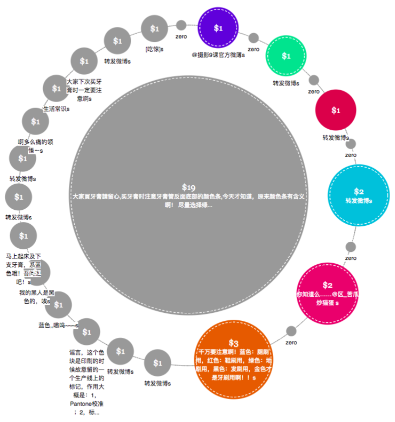


Figure 1: picture 1

3 Modified Methods

3.1 Modified the RNN model

The RNN models proposed before are already get good performace both on weibo and twitter, but there still some shortcomings:

- Just use the TF-IDF value to construct the information of posts
- Use information of the content of posts, but the content of some posts are do may not make sense especially the data from weibo
- May be try the Bi-LSTM or other RNN models

According to the structure of the recurrent neural networks(RNNs), we can improve the model from the following two aspects. First we can change the structure of RNN model, use different neural networks, such as BiLSTM. Second we can promote the input of the RNN model. As for the input, we can add other information to the input vector which extract from the feature engineering, on the other hand, we use a filter to choose the important content of the posts and tweets.

3.1.1 BiLSTM

In this part, we have tried the BiLSTM model to modify the model. BiLSTM contains a layer that consist from two LSTM models of opposite directions. That is because the GRU-2 model behave better than the GRU model, so we try the BiLSTM and LSTM-2 to find whether the performance will be better. And the BiLSTM model can use the information obtained from the later post to fix the bias.

3.1.2 Promote Input

In this part, when we go through the original data, we can find that the original model do not make full use of the data, for example, we have user's information do not use in the model, such as user id, followers number and so on. Besides we have a lot of tweets and weibo which content do not make sense, although we have used the TF-IDF value to extract the information that matters, it is better we filtered the useless information before the embedding and TF-IDF calculation.

Add Feature Add the features extract from the post using the method that used in the feature engineering. We use the feature vector and the content vector by connecting them together, then use the LSTM or GRU model to train the model. For example, we get feature vector F, and content vector C , then the input vector will be $I = [F, C]$In this way, we use the information of the users and other vital information. Noted that, we just use the 15 features that choose from the 28 features mention in (Jing Ma and Wong, 2015), the extract methods will be mentioned in the experiment part, that is because we can just improve the performance for a little degree, but the compute cost will increase a lot if we use the all the features.

Filter We preprocess the data by use some of the rules such as do not use the information like retweet, URL. So that we can get more useful information from the original data.

The structure of final modified model is as follows:

4 Experiment

4.1 Database and Setup

We use a database comes form the Sina Weibo and Twitter, the details of this database are showed bellow: To conduct our experiment, we randomly selected 80% of all samples as train set and 20% of

Database	Sina Weibo	Twitter
Posts	3752459	644123
Events	4664	992
Rumors	2313	498
Non-Rumors	2351	494

Table 1: Details of the datasets

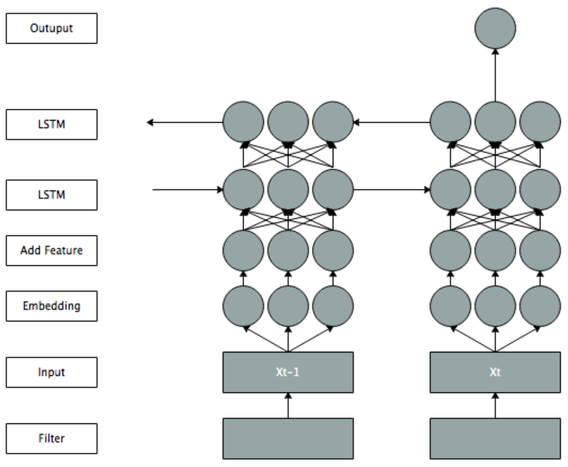


Figure 2: Structrure of modified model(the BiLSTM part can be replace by GRU-2)

them as test set.

4.2 Feature Engineering

According to origin paper, we should extract 29 features from each period, and they could be divided into three types: content-based, user-based and propagation-based features. Details could be fount in (Jing Ma and Wong, 2015)

In practice, considering the characteristics of our data, there are some features we should mention:

1. City size: Weather the city is provincial capital or not

2. Verified type: Weather the user is a normal user or not

3. Sentiment dictionary: *Boson sentiment lexicon*[1]

4. Post sentiment: We use $Snownlp$[2] to do the sentiment analysis

5. **LDA-based topic distribution:** In the origin paper, the model trains LDA model on all training posts and recognizes the LDA-based feature the same as other features (extract these features from every posts in a period and use the average of them as a feature of this period/event). However, we find most posts except the first one make no sense, for example, many of them are just 'retweet','hhhhhh' and so on, so we changed this feature: we only extract the topic distribution feature on the first post and set it an independent feature.

6. We use $gensim$[3] and *stopwores list*[4] to do the LDA

4.2.1 Pre-experiment:

At the Pre-experimental stage, we set the number of time periods as 100 and the features dimension of each event is 12. As a result, the accuracy of the model could reach to **83.1066%**. Through multiple experiment, in this stage, we also fixed the 'penalty coefficient' of our SVM model as $1 * 10^{-7}$. In the further experiment, we used this model as the baseline.

4.2.2 Feature refining:

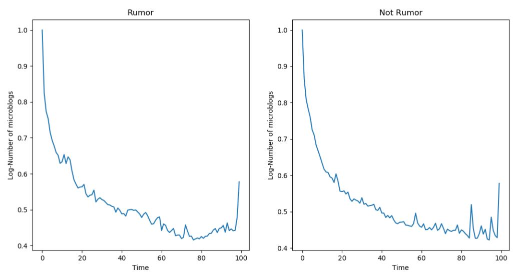
After increasing the feature dimension to 29, we find the accuracy of our model dropped dramatically to 67.2366%.

At first, we think phenomenon is due to the increasing complexity of the structure of data. So we tried to train a SVM_{RBF} classifier on our data. However, no matter what hyper-parameter (*Dimension* & *penalty coefficient*) we set, SVM_{RBF} classifier will recognize all samples as Nonrumors. It is obvious that there are some noise introduced by these new features.

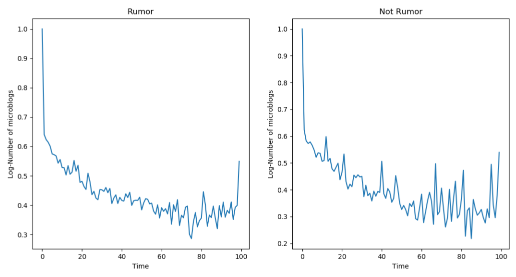
To remove the noise, we did PCA on our data first. Since the rank of the *spectrum matrix* of the training data is 563, we set the output dimension of PCA as 5/200/563. No matter what dimension of PCA we choose, the result is still bad (the accuracy is no higher than 67.2366%). Because we cannot clearly know the structure of our data, in the next step, we manually added feature one by one to the baseline model.

Finally, we figured out the bug feature: the number of posts in one event(time period). From following figure, we could find that compared with *Number of comments*, the distribution of *Number of posts* are similar between Rumor and Nonrumors. To be specific, the distribution of *Number of comments* are more various between Rumors and Nonrumors especially in these later period.

[1]https://bosonnlp.com/dev/resource
[2]https://github.com/isnowfy/snownlp
[3]https://github.com/RaRe-Technologies/gensim
[4]http://download.csdn.net/download/jdbc/9843227



(a) Number of posts



(b) Number of comments

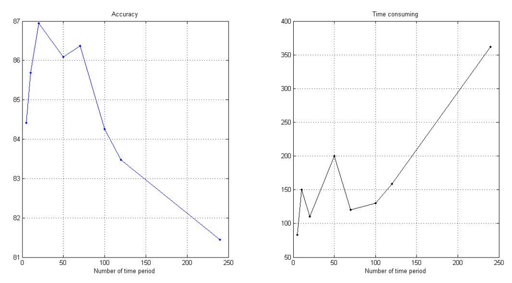
After moving out *Number of posts* from feature space, we extracted 28 features in each event, and the accuracy of our model could reach to 84.33%.

4.2.3 Number of time period:

We have discussed that the number of time periods will influence the efficiency of our model, and in this part, we will try to figure out a more exact relationship between them. We did multiple experiments, details as follow:

Number of Time period	Accuracy	Time(s)
240	84.15%	361.6
120	83.48%	159
100	84.26%	130.2
70	86.37%	120
50	86.09%	200
20	**86.94%**	110
10	85.69%	150
5	84.41%	80

Table 2: Experiment details



(a) Number of Time period

Analysis:

1. Because most posts of one event distribute on the starting period, when the number of time periods is small, most of posts are compressed into just a few periods, and many valuable features will loss during the progress of averaging. As the number of time periods increases, more information could be extracted from starting parts, so the efficiency of model will increase.

2. The side effect of increasing time periods is more zero-data will be introduced into feature space, for the reason that the features of middle and later periods, which have only few posts will also increase. It will also increase the complexity of model and bring noise into the model.

3. Generally speaking, there is a positive correlation between *Time consuming* and *Feature dimension*, because the increase of feature dimensions will increase the model complexity and more parameters needed to be fixed.

4. There is a negative correlation between *Time consuming* and *Accuracy*. From the point of data characters, when these samples are easy to classify, the margin will be easy to find and less time is request.

4.3 Confirm Modified Methods of RNN

In the experiment, we use the twitter data to confirm the hypothesis we proposed about the RNN model. First we compare the LSTM with the BiLSTM, then we compare the modified GRU-2 with GRU-2.

we can find that the BiLSTM perform not very well on test data, but we can find that the loss of the BiLSTM lower than LSTM model, so we can conclude that the poor performance of BiLSTM on test data may because the information of the later posts make the bias increase, and the model is overfitting much more easily when the epoch number increase.

Then we compare the result of other models, for the DSTS based model, we resorted to liner-SVM as our major model, and we set the 'penalty coefficient' of our SVM model as $1 * 10^{-7}$. From the table above, we can conclude that the RNN model behave better than the traditional classifier models,

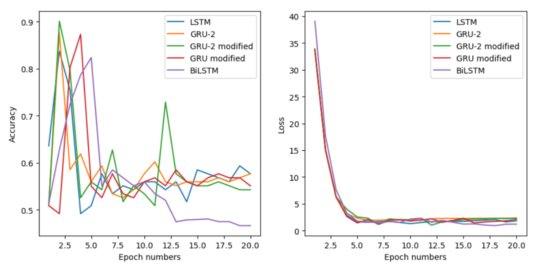


Figure 3: Accuracy on test data and loss on train data against epoch numbers

Model	Accuracy
GRU-2	90.26%
GRU-2(modified)	**91.12%**
SVM(DSTS)	86.94%
DT	86.34%
KNN	81.19%

Table 3: Performance of different models on weibo dataset

and the multi-layer GRU model behave better than the GRU model. The modified model even better than the behavior of original GRU model, which confirm that the promotion of input data(reduce the noise and add more useful information) enhance the model.

4.4 Implementation of PTK and cPTK

In this part, we will represent the implementation of the PTK and cPTK algorithm.

4.4.1 Data Structure

We use the array to store the whole tree, and add the index and relationship to this tree based on hash table,which will be convenient to the calculation of similarity.

4.4.2 Problem and Solution

Time-Consuming Caculation
When we measure the time when caculating similarity, we found that the 4000*4000 similarity matirx can not be caculated in a few months, since the 30*30 matrix requires 2 hours for a single caculation process. We use the following strategies for speeding up.
a.Hash Function
According to our analysis, a lot of calculation can be avoided, for example, when we find the most

Model	Accuracy
LSTM	83.68%
BiLSTM	82.34%
GRU-2	87.59%
GRU-2(modified)	**90.07%**
GRU(modified)	87.29%

Table 4: Performance of different models on twitter dataset

proper counterpart for each node, we have calculated a lot of similarity value between nodes, which will be recaculated in the following part of algorithm, so we can use a hash table to store this relationship. According to our test case, we can found that the average hit ratio is 0.64, and the time consumed has reduced by 0.4. b.Pseudo Distributed Calculation
From the algorithm we can see that the calculation of the similarity matrix is node-irrelevant, so we can split the task and dispath the missions to different CPUs. Since the similarity matrix is symetrical, we should split the calculation pieces like a wrapper, but for convenience for the intergration, and the storage of the whole matrix is not space-consuming, we just use the whole matrix to translate the calculation result.

Over(Under)flow Problem
The Overflow problem has two main reasons:1.The caculation precision of python is not enough.2.The parameter in formula is inproper and the function is not practical itself. a.Modify the Formula
The rationality of the modification of formulas can be guranteed by SVM method, as long as we assure the positive correlation between the b.Early stop
We can set the threshold for calculation, once reached it, we set it to the MAX and MIN, which do speed up the performance.

Pipeline Framework
When we take the reality into consideration, we should consider how this system can scale up if own goal is to predict the incoming result based on this system. Include the following part: 1.Split the tasks into pieces and send it to other process 2.Using the cron job to serealize the result data 3.Mantain a similarity matrix with proper size(UNDONE), which means that we need to drop out some former data due to the size of memory and the unvalid old data.

4.5 New Method KNN-Like

In this part, I will represent a simple and understandable methodwhich is similiar to KNN, so in a nutshell,we calculate the average similarity between the true cluster and false cluster, and select the category by the avg.

4.6 Data Analysis

4.6.1 Method

We spilt the data into training and testing set at ratio of 0.7, using svm provided by sklearn to do it.

4.6.2 Result and Conclusion

1.The small size dataset has high performance.
2.The cptk method has no distinct difference with ptk method, and the knn-like model is better on large dataset and with high interpretability

Datasize	PTK	cPTK	KNN-like
20	0.59	0.67	0.75
40	0.65	0.81	0.67
60	0.70	0.77	0.77
80	0.68	0.81	0.88

Table 5: Three Method Comparison

5 Conclusion and Future Work

From this project, we confirm that the RNN model do have its advantages to deal with the time series problem, besides the quality of the feature extraction also affect the results. When we add the features to the RNN model, we do get better results. On the other hand, we implement PTK and cPTK method, due to hardware limitations, we can not use it on the full dataset, instead we using it on a small dataset, which also confirm the effectiveness of the method.
For the feature engineering, basing on the characteristics of our data, we refined the method to extract both static and variational features. We also explored the relationship between model efficiency and *number of time period*. Based on the modified features, our model performs better than origin model.
For the RNN model, we may adjust the parameters or use different gating function to improve the model performance. Besides, we may use other deep learning motheds on rumor detection.
For the PTK and cPTK

References

M. Mendoza C. Castillo and B. Poblete. 2011. Information credibility on twitter. *In Proceedings of WWW* page 675?684.

Wei Gao Kam-Fai Wong Jing Ma. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. *ACL* pages 708–717.

Wei Gao Prasenjit Mitra Sejeong Kwon Bernard J. Jansen Kam-Fai Wong Meeyoung Cha. Wu S. Yang Jing Ma and K. Q. Zhu. 2016. Detecting rumors from microblogs with recurrent neural networks. *IJ-CAI* pages 3818–3824.

Wei Gao Zhongyu Wei Yueming Lu Jing Ma and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. *CIKM* .

S. Yang K. Wu and K. Q. Zhu. 2015. False rumors detection on sina weibo by propagation structures. *In Proceedings of ICDE* page 675?684.