Final Report: Weibo Rumor detection

Zheng Jiang 515030910561

Abstract

This report describes the Rumor dectction project of class 'Mobile Internet'. In this project,I present a novel method that learns continuous representations of microblog events for identifying rumors. The proposed model is based on recurrent neural networks (RNN) for learning the hidden representations that capture the variation of contextual information of relevant posts over time. In this report, I'll first talk about the background of our project. The following are our design of our different verisons and the conclusion of this project.

1 Introduction

A rumor is an unverified and instrumentally relevant statement of information spread among people. The rapid development of information technology enables more rumors and a faster spread of them. Social psychologists argue that rumors arise in contexts of ambiguity, when the meaning of a situation is not readily apparent, or potential threat, when people feel an acute need for security.Besides, the rapid growth of online social media has made it possible for rumors to spread more quickly. Online social media enable unreliable sources to spread large amounts of unverified information among people. Therefore, it is crucial to design systems that automatically detect misinformation and disinformation.

In our projcet, we mainly focus on twitter when doing rumor detection. We use 4364 events and each event has hundreds to hundreds of thosands of tweets. Since deep neural networks have demonstrated clear advantages for sequences related problems, we use a recurrent neural network to work on the rumor dectection. The advantage maybe that the connections between units in an RNN form a direct cycle and create an internal state of the network that might allow it to capture the dynamic temporal signals characteristic of rumor diffusion.

Utilizing RNN, we model the social context information of an event as a variable-length time series. We assume people, when exposed to a rumor claim, will forward the claim or comment on it, thus creating a continuous stream of posts. This approach learns both the temporal and textual representations from rumor posts under supervision. The experiments on Twitter dataset shows a better performance than the previous work

2 Related work

2.1 Existing methods

The paper I mainly follow is: $DetectingRumorsfromMicroblogswithRecurrentNeuralNetworks^{[1]}$. In this paper, they mianly use the RNN methods to dorumor detecting. I changed the algorithm and get a better result in some of the experiments.

2.2 Recurrent neural network

A recurrent neural network (RNN) is a class of artificial neural network where connections between nodes form a directed graph along a sequence. This allows it to exhibit dynamic temporal behavior for a time sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition[1] or speech recognition.

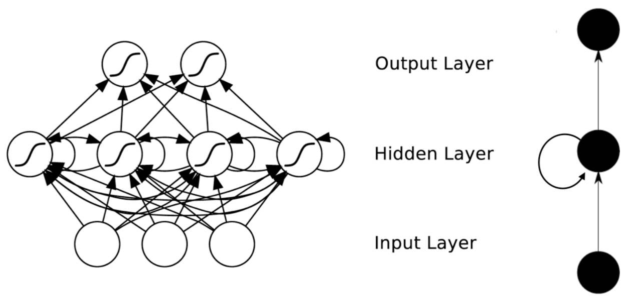

Figure 1: Structure of RNN

2.2.1 Long Short-Term Memory (LSTM)

Unlike the traditional recurrent unit whose state is overwritten at each time step , an LSTM unit maintains a memory cell c_t ct at time t. The output ht of an LSTM unit is computed by the following equations:

$$\begin{aligned}
i_t &= \sigma(x_t W_i + h_{t-1} U_i + c_{t-1} V_i) \\
f_t &= \sigma(x_t W_f + h_{t-1} U_f + c_{t-1} V_f) \\
\widetilde{c_t} &= tanh(x_t W_c + h_{t-1} U_c) \\
c_t &= f_t c_{t-1} + i_t \widetilde{c_t} \\
o_t &= \sigma(x - t W_o + h_{t-1} U_o + c_t V_o) \\
h_t &= o_t tanh(c_t)
\end{aligned}$$

where σ is a logistic sigmoid function. The input gate i_t determines the degree to which the new memory is added to the memory cell. The forget gate f_t decides the extent to which the existing memory is forgotten. The memory c_t is updated by forgetting part of the existing memory and adding new memory $\widetilde{c_t}$. The output gate ot is the amount of output memory.

2.2.2 Gated Recurrent Unit (GRU)

Gated recurrent unit (GRU) is a gating mechanism in recurrent neural networks. There are several variations on the full gated unit, with gating done using the previous hidden state and the bias in various combinations, and a simplified form called minimal gated unit. GRU merges the input gate and output gate in LSTM to update gate and also merges the cell unit and hidden state. The following is the equations of a layer of GRU:

$$\begin{aligned}
Z_t &= \sigma(x_t U_z + h_{t-1} W_z) \\
r_t &= \sigma(x_t U_r + h_{t-1} W_r) \\
\widetilde{h_t} &= tanh(x_t h_t + (h_{t-1} \cdot r_t) W_h) \\
h_t &= (1 - z_t) \cdot h_{t-1} + z_t \cdot \bar{h_t}
\end{aligned}$$

where a reset gate r_t determines how to combine the new input with the previous memory, and an update gate z_t defines how much of the previous memory is cascaded into the current time step, and h_t denotes the candidate activation of the hidden state h_t.

3 Our Design

Since individual microblog posts are short in nature, containing very limited context, we mainly focus on the event instead of the posts in each event. By analysising the key posts in each event, we can finally drive the conclusion of whether the event is a rumor of not. My design is to seperate the problem into two parts: one is to process the posts to get a suitable structure for neural networks which could stand for the charateristic of the event, the other is the RNN model. We define a set of given events as E $= E_i$, where each event $E_i = (m_{i.j}, t_{i,j})$ consists of ideally all relevant posts $m_{i,j}$ at timestamp $t_{i,j}$, and the task is to classify each event as a rumor or not.

3.1 Data processing

In the data processing part, I tried some methods besides the paper I follow, but the results are not so well. The accuracy drops if I use my definition of data process. I guess it maybe because my definition widens the amount of posts which increases the noise since there are many irrelevant posts in the dataset. So I finally use the algorithm in the paper[1] that has a good combination of posts and time line. The following shows the partition of time intervals which all contain a series of posts that stand for the characteristics of the events.

```
Input  : Relevant posts of $E_i = \{(m_{i,j}, t_{i,j})\}_{j=1}^{n_i}$,
         Reference length of RNN $N$
Output: Time intervals $I = \{I_1, I_2, \ldots\}$
/* Initialization                                  */
$L(i) = t_{i,n_i} - t_{i,1}$;  $\ell = \frac{L(i)}{N}$;  $k = 0$;
while true do
    $k$ ++;
    $U_k \leftarrow Equipartition(L(i), \ell)$;
    $U_0 \leftarrow$ {empty intervals} $\subseteq U_k$;
    $U'_k \leftarrow U_k - U_0$;
    Find $\bar{U}_k \subseteq U'_k$ such that $\bar{U}_k$ contains continuous
      intervals that cover the longest time span;
    if $|\bar{U}_k| < N$ && $|\bar{U}_k| > |\bar{U}_{k-1}|$ then
        /* Shorten the intervals                   */
        $\ell = 0.5 \cdot \ell$;
    else
        /* Generate output                         */
        $I = \{I_o \in \bar{U}_k | I_1, \ldots, I_{|\bar{U}_k|}\}$;
        return $I$;
    end
end
return $I$;
```

Figure 2: Algorithm for constructing variablelength time series given the set of relevant posts of an event and the reference length of RNN

For each event, this algorithm sperate the posts to many intervals by time line. Then it find the longest continous non-empty intervals. The set of intervals should also meet with the demand that the number of intervals should larger than N(set by ourselves as the reference length of RNN). If the intervals do nott meet the demand, it will halve the length of interval until finally meets. As a result, the output should be a list of intervals contains major useful posts of that event. Then by analysising the frequency of words in the posts, we get the top-K words as the input of the RNN structure.

3.2 Model

The structure of the models are quite simple and are showed in the following picture. E is the word embedding weight matrix, U, W, V correspond to the parameters of hidden layers and output layers. R means rumor and N means non-rumor.

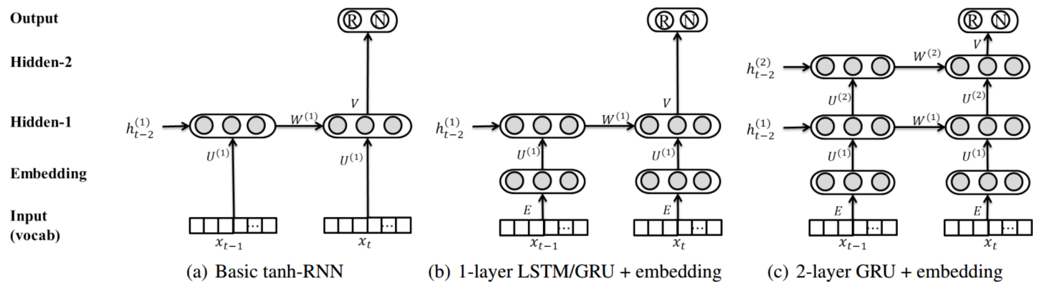

Figure 3: Structure of the models

Model Training. I train all the RNN models by employing the derivative of the loss through back-propagation with respect to all of the parameters. We use the AdaGrad algorithm for parameter update. We empirically set the vocabulary size K as 5,000, the embedding size as 100, the size of the hidden units as 128 and the learning rate as 0.5. We iterate over all the training events in each epoch and continue until the loss value converges or the maximum epoch number is met.

4 Experiments and results

4.1 Data collection

We use the same dataset as the paper we follow form the following url: `http://alt.qcri.org/wgao/data/rumdect.zip`, and the statics of the dataset is listed as follow:

Table 1: The dataset of Weibo

Statistic	Weibo
Users #	2,746,818
Posts #	3,805,656
Events #	4664
Rumors #	2313
Non-Rumors #	2351
Avg. time length/event	2,460.7 hours
Avg. # of posts/event	816
Max # of posts/event	59,318
Min # of posts/event	0

4.2 Experiment result

Table 2: Rumor detection results

Method	My Accuracy	Baseline
LSTM-1	0.936	0.896
LSTM-2	0.926	0.910

The table above shows the results of model (b),(c) which increases 4% accuracy and 1.6% accuracy separatedly. The Table 2 shows the performance of all the systems. Our models outperform all the baselines on both models (b) and (c). The one layer LSTM RNN model achieves 93.6% accuracy on Weibo datasets which is 4% higher than the baseline. And the two layer LSTM RNN achieves 92.6% accuracy on Weibo datasets which is 1.6% higher than then baseline. Our method is similar to that in the paper. As we use the same RNN model with similar hyper-parameters, the difference may at the data process part. The difference in words dividing and feature extraction may lead to different result. In my work, I use jieba and TfidfVectorizer to work on the two parts.

In my experiment, the accuracy of multi-layer RNN is lower than taht o fsingle-layer. I think this may beacause of overfitting. In LSTM-2 the accuracy of training set reaches 98.1% with loss 0.072 while the accuracy of testing set is 92.6% with loss 0.259. Though we got 3,805,656 posts from Weibo, these posts only consist of 4664 events, which is a quite small dataset. So in this situation, comlpex deep neural networks may not act good enough since it is easier to cause overfitiing.

5 Conclusion

In this project, I followed a deep learning framework for rumor debunking in IJCAI 2016[1] and achieved a better result.My method learns RNN models by utilizing the variation of aggregated information across different time intervals related to each event. We empirically evaluate our RNN-based methods which perform significantly better than the stateof-the-arts. The following are the future improvements:

1. Since our multi-layer RNN acts worse than single-layer RNN, it might be a good idea to widden the dataset in the future.
2. Due to the time limit, I accomplished few RNN models, more models could be tried in the future work.
3. Besides RNN, maybe other algorithms like unsupervised learning methods could be taken into account.

Finally, I'd like to thank professor Xingbing Wang and Luoyi Fu for the class Mobile Network. I really gained a lot through this class and the project.

References

[1] Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, Meeyoung Cha.*Detecting Rumors from Microblogs with Recurrent Neural Networks* IJCAI 2016