Course Report for
Detecting Rumors from Microblogs with Recurrent Neural Networks

Tonghui Tang
School of Electronic Information
and Electrical Engineering
Shanghai Jiao Tong University
Student Number: 515030910611
Email:supermantth@163.com

***Abstract*—Microblogging platforms are an ideal place for spreading rumors and automatically debunking rumors is a crucial problem. To detect rumors, existing approaches have relied on hand-crafted features for employing machine learning algorithms that require daunting manual effort. Upon facing a dubious claim, people dispute its truthfulness by posting various cues over time, which generates long-distance dependencies of evidence. Then in the reference paper [1], it presents a method that learns continuous representations of microblog events for identifying rumors which 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 , we implement a RNN model based on the reference paper and try to do some modification to achieve best results.**

1. Introduction[1]

1.1. Rumor

Social psychology literature defines a rumor as a story or a statement whose truth value is unverified or deliberately false [2]. False rumors are damaging as they cause public panic and social unrest. For example, on August 25th of 2015, a rumor about shootouts and kidnappings by drug gangs happening near schools in Veracruz spread through Twitter and Facebook1. This caused severe chaos in the city involving 26 car crashes, because people left their cars in the middle of a street and rushed to pick up their children from school. This incident of a false rumor highlights that automatically predicting the veracity of informa tion on social media is of high practical value.

Debunking rumors at an early stage of diffusion is particularly crucial to minimizing their harmful effects. To distinguish rumors from factual events, individuals and organizations often have relied on common sense and investigative journalism. Rumor reporting websites like snopes.com and factcheck.org are such collaborative efforts. However, because manual verification steps are involved in such efforts, these websites are not comprehensive in their topical cover- age and also can have long debunking delay.

1. This part is mainly from [1]

1.2. Models using Learning Algorithms

Existing rumor detection models use learning algorithms that incorporate a wide variety of features manually crafted from the content, user characteristics, and diffusion patterns of the posts [3] [4] [5] [6] [7] [8], or simply exploited patterns expressed using regular expressions to discover rumors in tweets [9]. Feature engineering is critical, but it is painstakingly detailed, biased, and labor-intensive.

1.3. Deep Neural Networks

In several years, deep neural networks have demonstrated clear advantages for many machine learning problems [10] [11] [3]. given the sequential nature of text streams in social media, recurrent neural networks (RNN) are suitable for rumor detection. For the connections between units in an RNN form a direct cycle and create an internal state of the network [12] that might allow it to capture the dynamic temporal signals characteristic of rumor diffusion.So given the sequential nature of text streams in social media, recurrent neural networks (RNN) are suitable for rumor detection.

1.4. Method in Reference Paper

In the reference paper [1], utilizing RNN, they model the social context information of an event as a variable-length time series. They 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 network model of their model is as shown in Figure1. For each model, there is a embedding layer that encode the origin representation of words into vector to convert the sparse input word vectors into low-dimensional representations.

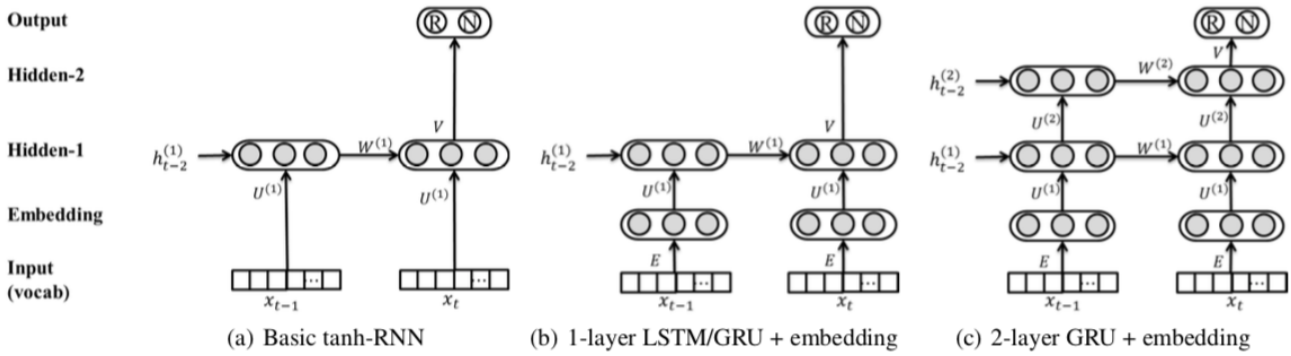

Figure 1. RNN-based rumor detection models in the reference paper

Then there are CNN, LSTM or GRU layers. After go through these layers, there is a full connected layer that output the result.

However, in this paper, author didt point out clearly the which is the input. the input is one word or a sentence, If it is one word, then the time step will be the longest length of top k. If it is a sentence, then the time step will be the interval.

so while firstly implementing the paper, I use my own approaches to handle the data.

2. Related Work[2]

Automatic rumor detection from social media is based on traditional classifiers that detect misinformation stemming from the pioneering study of information credibility on Twitter [3]. In following works [4] [6] [7] [8], different sets of hand-crafted features were proposed and incorporated to determine whether a claim about an event is credible. Most of these prior works attempted to classify the veracity of spreading memes using information other than the text content, for instance, the popularity of a post (e.g., the number of retweets or replies of the post), the features relevant to determine a users credibility, etc. However, feature engineering is painstakingly labor intensive.

Some prior studies focused on capturing the temporal traits of rumors during their propagation. [5] introduced a time-series-fitting model based on the temporal properties of a single feature tweet volume. [7] extended the model using dynamic time series to capture the variation of a set of social context features over time. [13] characterized the structure of misinformation cascades on Facebook by analyzing comments with links to rumor debunking websites. [9] worked on early rumor detection using cue terms such as not true, unconfirmed or debunk to find questioning and denying tweets.

[1] use the temporal properties of the representations, but the features are learned automatically via an RNN given the fundamental term representation in each time segment. Also, it learns representations that are significantly more complex than these explicit signals; these representations can capture the hidden implications and dependencies over time. And it is related to studies detecting spammers [14] [15] and fake images on Twitter [16], and the Truthy system [17] [18] that differentiates whether a meme is spreading organically or is being spread by an astroturf campaign.

3. RNN: Recurrent Neural Network[3]

An RNN is a type of feed-forward neural network that can be used to model variable-length sequential information such as sentences or time series. A basic RNN is formalized as follows: given an input sequence $(x_1, ..., x_T)$, for each time step, the model updates the hidden states$(h_1, ..., h_T)$ and generates the output vector $(o_1, ..., o_T)$, where T depends on the length of the input. From t = 1 to T , the algorithm iterates over the following equations:

$$h_t = tanh(Ux_t + Wh_{t-1} + b)$$

$$o_t = Vh_t + c$$

where U, W and V are the input-to-hidden, hidden-to-hidden and hidden-to-output weight matrices, respectively b and c are the bias vectors, and $tanh(.)$ is a hyperbolic tangent nonlinearity function.

Typically, the gradients of RNNs are computed via back-propagation through time [19]. In practice, because of the vanishing or exploding gradients [20], the basic RNN cannot learn long-distance temporal dependencies with gradient-based optimization. One way to deal with this is to make an extension that includes memory units to store information over long time periods, commonly known as Long Short-Term Memory (LSTM) unit [21] [22] and Gated Recurrent unit (GRU) [3] . Here, we briefly introduce the two structures.

2. This part is mainly from [1]

3. This part is mainly from [1]

3.1. Long Short-Term Memory (LSTM)

Unlike the traditional recurrent unit whose state is overwritten at each time step (equations 1), an LSTM unit maintains a memory cell ct at time t. The output ht of an LSTM unit is computed by the following equations [22] [21]

$$i_t = \sigma(x_t W_i + h_{t-1} U_i + c_{t-1} V_i)$$

$$f_t = \sigma(x_t W_f + h_{t-1} U_f + c_{t-1} V_f)$$

$$\tilde{c}_t = tanh(x_t W_c + h_{t-1} U_c)$$

$$c_t = f_t + c_{t-1} + i_t \tilde{c}_t$$

$$o_t = \sigma(x_t W_o + h_{t-1} U_o + c_t V_o)$$

$$h_t = o_t tanh(c_t)$$

where σ is a logistic sigmoid function. The input gate it determines the degree to which the new memory is added to the memory cell. The forget gate ft 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 $\tilde{c}_t$.The output gate o_t is the amount of output memory.

3.2. Gated Recurrent Unit (GRU)

Similar to an LSTM unit, a GRU has gating units that modulate the flow of the content inside the unit, but a GRU is simpler with fewer parameters. The following equations are used for a GRU layer [3]:

$$z_t = \sigma(x_t U_z + h_{t-1} W_z)$$

$$r_t = \sigma(x_t U_r + h_{t-1} W_r)$$

$$\tilde{h}_t = tanh(x_t H_h + h_{t-1} r_t W_h)$$

$$h_t = (1 - z_t) h_{t-1} + z_t \tilde{h_t}$$

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.

4. RNN-based Rumor Detection

Compared to the reference paper, we use the basic split method corresponding to each event.In problem state and the algorithm spliting the Event, it is almost the same between my method and reference paper. Next , I will introduce my implementation work following the approach in the reference paper.

4.1. Problem Statement

Individual microblog posts are short in nature, containing very limited context. A claim is generally associated with a number of posts that are relevant to the claim. We are not interested at the individual level, but at the aggregate level. Therefore, predicting the veracity of each post is not our focus here. Instead, we concentrate on detecting rumors at the event-level, comprised of a set of relevant posts. 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.

4.2. Variable-length Time Series

We could model each post as an input instance and construct an RNN modeling the time series with a sequence length equal to the number of posts. However, there could be tens of thousands of posts in a popular event. We have only a single output unit indicating the class at the last time step of each event. Back propagation through a large number of time steps with only a final-stage loss will be computationally expensive as well as ineffective. Hence, we batch posts into time intervals and treat them as a single unit in a time series that is then modeled using an RNN sequence. A reference length of RNN sequence is adopted for constructing the time series.

Time spans representing densely populated with posts in the diffusion should be captured properly; the number of time intervals adopted approximates the reference length of RNN. Algorithm 1 describes the procedure. Initially, we divide the entire timeline equally into N intervals (i.e., N is the reference length). Then, our system tries to discover the set of non-empty intervals U_0 (i.e., each interval in U 0 has at least one tweet) by removing the empty ones in the set U0, from which those continuous intervals whose overall time span is the longest are chosen into the set $\hat{U}$. If the number of intervals in $\hat{U}$ is lower than N and the number of intervals is more than that of the previous round, we halve the intervals and continue partitioning; otherwise, it returns the discovered continuous intervals given by $\hat{U}$. Note that the length of entire time series, though is close to N, varies among different events, whereas the length of individual intervals in an event is equal.

4.3. Word to Vector

After we divide the posts in each event into several intervals. We should think about how to represent the data in each interval.

Firstly, we use $tfidf$ algorithm to select top-K words in each interval. Then we use a vector to represent each word using the vector set download from the website[4] in which we choose the Weibo vector set corresponding to our dataset.

4. https://github.com/Embedding/Chinese-Word-Vectors

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 ++;
  $U_k \leftarrow Equipartition(L(i), \ell)$;
  $U_0 \leftarrow \{\text{empty intervals}\} \subseteq U_k$;
  $U_k' \leftarrow U_k - U_0$;
  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;

Algorithm 1: Algorithm for constructing variable-length time series given the set of relevant posts of an event and the reference length of RNN

In this vector set, each word is represented by a vector **x** which has the length of 300, thus for each interval, we concat all the represent of words together.

Therefore, for each interval, we could get the representation $\mathbf{i} = [\mathbf{x}_1, ..., \mathbf{x}_K]$ which has the length of 300K.

4.4. Structures of Models

4.4.1. Basic model. Based on the model illustrated in Figure1, we implement these models almost the same with the reference. However, there are still a little modification.

Before entering into the RNN layer, we use a mask layer to set the length to be the same value.

Instead of using squared error, we use logloss error to evaluate the model. Let g_c, where c denotes the class label, be the ground-truth class an event. For each training instance (i.e., each event), our goal is to minimize the logloss error between the probability of each event of the prediction and ground truth:

$$min \sum_c g_c \log pc + (1 - g_c) \log(1 - p_c) + \sum_i ||\theta_i||^2$$

where g_c and p_c are the gold and predicted distributions, respectively, θ_i represents the model parameters to be estimated, and the L2-regularization penalty is used for trading off the error and the scale of the problem

4.4.2. Model with CNN. Long-distance dependencies are important for capturing the patterns in rumors and hidden signals over the life cycle of the event. Actually, we also try to use CNN to find some local feature during each inteveral.the model is shown in Figure3

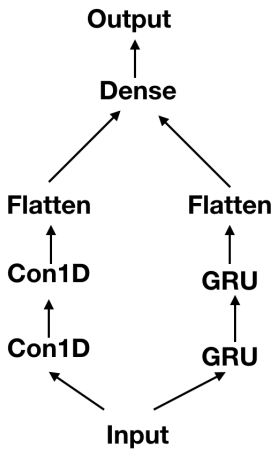

Figure 2. RNN with CNN

4.5. Model Training

We train all the models by employing the derivative of the loss through backpropagation [23] with respect to all of the parameters. We use the AdaGrad algorithm [24] for parameter update. We empirically set the vocabulary size K as 10, the size of the hidden units as 125 and the learning rate as 0.01. We iterate over all the training events in each epoch and continue until the loss value con- verges or the maximum epoch number is met.

5. Experiments and Results

5.1. Data Collection

We use the datasets released by the reference paper about sina Weibo. In this dataset,There are about 4500 event. And the positive instance and negative instance are almost the same.

5.2. Experimental Results

For the three RNN-based models, for GRU and LSTM remember more long-term information. GRU and LSTM perform well; GRU is slightly better. Compared to RNN-based model, the CNN-combined model has a slightly better performance.However, the overall performance is still lower than the performance in the reference paper.

6. Conclusion

Compared to the reference paper, in which the worst accuracy is about 0.87, the best accuracy is about 0.83. Thus

Weibo dataset				
Class	Accuracy	Precision	Recall	F_1
tanh-RNN	0.873	0.816	0.964	0.884
LSTM	0.896	0.846	0.968	0.913
GRU	0.908	0.871	0.958	0.913

TABLE 1. REFERENCE RESULT IN WEIBO DATASET

Weibo dataset				
Class	Accuracy	Precision	Recall	F_1
tanh-RNN	0.745	0.740	0.732	0.736
LSTM	0.781	0.769	0.802	0.776
GRU	0.811	0.802	0.816	0.802
CNN-RNN	0.824	0.817	0.830	0.814

TABLE 2. OUR RESULT IN WEIBO DATASET

the problem may be the process of handling data, author did not state the process clearly, so I deal with the data in my own way. Due to lacking much understanding of how to deal with text and not have some tricks to capture the import feature of sequence of text, the result is not as good the origin result even when I modify the network and actually this modification improve the performance. It is regretful that I fail to achieve a better result compared to the reference paper. At the same time, by comparing the result with those of the traditional method, actually we can conclude that with regard to the text, RNN is an effective way to capture the import feature at present.

References

[1] 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." in *IJCAI*, 2016, pp. 3818–3824.

[2] G. W. Allport and L. Postman, "The psychology of rumor." *American Journal of Sociology*, 1947.

[3] K. Cho, B. V. Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *Computer Science*, 2014.

[4] Y. Ding, J. Han, J. Tang, and P. S. Yu, "Proceedings of the acm sigkdd workshop on mining data semantics," in *ACM SIGKDD Workshop on Mining Data Semantics*, 2012.

[5] S. Kwon, M. Cha, K. Jung, W. Chen, and Y. Wang, "Prominent features of rumor propagation in online social media," in *IEEE International Conference on Data Mining*, 2014, pp. 1103–1108.

[6] X. Liu, A. Nourbakhsh, Q. Li, R. Fang, and S. Shah, "Real-time rumor debunking on twitter," in *ACM International on Conference on Information and Knowledge Management*, 2015, pp. 1867–1870.

[7] 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 *ACM International on Conference on Information and Knowledge Management*, 2015, pp. 1751–1754.

[8] K. Wu, S. Yang, and K. Q. Zhu, "False rumors detection on sina weibo by propagation structures," in *IEEE International Conference on Data Engineering*, 2015, pp. 651–662.

[9] Z. Zhao, P. Resnick, and Q. Mei, "Enquiring minds: Early detection of rumors in social media from enquiry posts," in *International Conference on World Wide Web*, 2015, pp. 1395–1405.

[10] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *International Conference on Machine Learning, ICML 2011, Bellevue, Washington, Usa, June 28 - July*, 2016, pp. 1017–1024.

[11] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," vol. 4, pp. 3104–3112, 2014.

[12] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[13] A. Friggeri, L. A. Adamic, D. Eckles, and J. Cheng, "Rumor cascades," *Dalton Transactions*, vol. 43, no. 16, pp. 6108–19, 2014.

[14] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: social honeypots + machine learning," in *International Acm Sigir Conference on Research Development in Information Retrieval*, 2010, pp. 435–442.

[15] A. H. Wang, "Don't follow me: Spam detection in twitter," in *International Conference on Security and Cryptography*, 2011, pp. 142–151.

[16] A. Gupta, H. Lamba, P. Kumaraguru, and A. Joshi, "Faking sandy:characterizing and identifying fake images on twitter during hurricane sandy," in *International Conference on World Wide Web*, 2013, pp. 729–736.

[17] J. Ratkiewicz, M. Conover, M. Meiss, B. Gonalves, A. Flammini, and F. Menczer, "Detecting and tracking political abuse in social media," in *International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July*, 2011.

[18] J. Ratkiewicz, M. Conover, M. Meiss, S. Patil, A. Flammini, and F. Menczer, "Truthy:mapping the spread of astroturf in microblog streams," in *International Conference Companion on World Wide Web*, 2011, pp. 249–252.

[19] Rumelhart, E. David, Hinton, E. Geoffrey, Williams, and J. Ronald, "Learning representations by back-propagating errors," vol. 323, no. 6088, pp. 399–421, 1986.

[20] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 2002.

[21] A. Graves, "Generating sequences with recurrent neural networks," *Computer Science*, 2013.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] R. Collobert, "Natural language processing from scratch."

[24] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 7, pp. 257–269, 2011.