Provenance-based Rumor Detection

Chi Thang Duong [*], Quoc Viet Hung Nguyen [†],
Sen Wang [†], and Bela Stantic [†]

École Polytechnique Fédérale de Lausanne [*], Griffith University [†]

Abstract. With the advance of social media networks, people are sharing contents in an unprecedented scale. This makes social networks such as microblogs an ideal place for spreading rumors. Although different types of information are available in a post on social media, traditional approaches in rumor detection leverage only the text of the post, which limits their accuracy in detection. In this paper, we propose a provenance-aware approach based on recurrent neural network to combine the provenance information and the text of the post itself to improve the accuracy of rumor detection. Experimental results on a real-world dataset show that our technique is able to outperform state-of-the-art approaches in rumor detection.

1 Introduction

With the advance of social media networks, people are sharing user-generated contents in an unprecedented scale. Due to its distributed and decentralized nature, social media provides a platform for information to propagate without any type of moderation. As a result, when an incorrect information propagates on social media networks, it may have a profound impact on real life. For instance, when a fake news claiming two explosions happened in the White House and Barrack Obama got injured was posted by a hacked Twitter account associated with a major newspaper, it caused panic in the society which incurred a loss of $136.5 billion in stock market. This incident shows how a rumor can have a severe impact on our life and it highlights the need for the detection of rumor among different events being discussed on social media networks.

In an attempt to combat fake news, several rumor debunking services such as snopes.com have been created to expose rumors and misinformation. These websites harness collaborative efforts from internet users to identify potential fake news and leverage experts to verify them. As they involve manual labor, the number of events that can be covered are limited and it would take a long time to fact-check an event.

In order to automate this process, several rumor detection models have been proposed. These techniques first design a wide range of features based on the content of the posts[9,6], their characteristics [20,1,16] or the network of users [19,8]. However, feature engineering is a tedious and time-consuming process while the hand-crafted features may not be applicable to a new dataset.

Another important characteristics of rumors on social networks is their temporal nature. When some posts discuss an event, there would be several posts discussing the same event subsequently. These subsequent posts may just be reiteration of the original posts or they could add new information which sheds light on the event. Traditional approaches in rumor detection tend to ignore the temporal nature of the posts. The temporal dependency can also be indicator of rumors. For instance, an event that has many posts providing different views with subsequent posts arguing with previous ones tends to be a rumor as this event is controversial. To the best of our knowledge, there is only one work[10] that models the temporal dependency of the posts explicitly using recurrent neural network(RNN). However, in this work, although various information can be obtained from a post in social media, the users only leverage the textual contents of the posts to classify rumors. We posit that the provenance of the event plays a significant role in identifying rumors. The provenance of an event appears in a post in social media in the form of a link to an article discussing the event. Traditional approaches considered these links as part of the text, hence, provided no special treatment.

Based on this observation, in this work, we propose a provenance-based approach to rumor detection. Our approach considers the provenance of the events appearing as links in the posts as an important source of information. There are several challenges we need to solve in order to leverage the provenance information. First, as the provenance appears as links to some articles in the posts, we need to find a way to model the provenance information. Second, as both the provenance and textual information are present in a post, we also need a way to combine these information in a coherent manner. Third, there are cases that the provenance information is not available. For instance, a tweet may not refer to any article. In these cases, we need to handle the missing of the provenance information while making sure that the classification accuracy does not deteriorate. In order to handle these challenges, we propose a fusion approach that is based on the pooling operation to combine information from the provenance and the text itself. The pooling operation allows our approach to be robust with the missing of the provenance information. In addition, we also leverage RNN to capture the temporal dependency among the posts.

The contributions of this paper are as follows:

- We propose a provenance-based approach to classify events into rumors and non-rumors. Our model also leverages RNN to capture the temporal dependency between the posts.
- We have enriched a social media dataset by adding the provenance information. Our dataset can also be used by subsequent research in this direction.
- Our extensive experiments on a real-world dataset show that our approach is able to outperform state-of-the-art technique significantly.

The rest of the paper is organized as follows. Section 6 introduces related works on the field of rumor detection. Section 3 discusses our general framework to classify rumors. Section 2 explains in detail our provenance-based approach.

Experimental evaluation and analysis are presented in Section 5 while Section 7 concludes the paper.

2 Recurrent Neural Network for Rumor Detection

2.1 Problem statement

We consider a setting in which a set of users discuss n events which can be rumors or not. We denote a discussion for an event e_i as $d_{it} = \langle e_i, t \rangle$ where t is the time the discussion took place. In our setting, a discussion can be a tweet or a post by a user on a social network. It is worth noting that different events may have different number of discussions. In addition, each event is associated with a label indicating where it is a rumor or not.

The problem we want to solve is given a set of events together with their discussions, classify the events correctly. In particular, given a temporal sequence of discussions $D = \{e_i, t\}$, our goal is to assign a label $l_i \in \{0, 1\}$ for the event e_i where 1 denotes rumor and 0 otherwise. We achieve this by training a feedforward neural network which takes the discussions of an event as input and returns the label for the event. More precisely, given an event e_i and two classes $Y = \{0, 1\}$, we define a neural network that assigns probabilities to all $y \in Y$. The predicted class is then the one with the highest probability:

$$\hat{y} = arg \max_{y} P(Y = y|e) \tag{1}$$

Our network models the temporal characteristics of the dicussions using RNN and leverages the provenance of the tweets to achieve high accuracy.

2.2 Neural Network Model

A feedforward neural network estimates $P(Y = y|e)$ with a parametric function ϕ_θ (Equation 1), where θ refers to all learnable parameters of the network. Given an event e, this function ϕ_θ applies a combination of functions such as

$$\phi_\theta(e) = \phi^L(\phi^{L-1}(\dots \phi^1(e) \dots)), \tag{2}$$

with L the total number of layers in the network.

We denote matrices as bold upper case letters ($\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$), and vectors as bold lower-case letters ($\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$). $\mathbf{A}_i$ represents the i^{th} row of matrix $\mathbf{A}$ and $[\mathbf{a}]_i$ denotes the i^{th} element of vector $\mathbf{a}$. Unless otherwise stated, vectors are assumed to be column vectors. We also denote $|\mathbf{a}|$ to be the dimensionality of the vector $\mathbf{a}$. We now introduce the layers when training linear classifiers with neural networks: the *recurrent neural network* layer, the *linear* layer and the *softmax* layer.

Recurrent Neural Network Among different types of feed-forward neural networks, RNN is the one that can model the sequential characteristics of the input data such as time series or sentences. Given an input sequence $(x_1, ..., x_T)$, RNN processes each input sequentially (from x_1 to x_T), at each step, it updates its hidden state h_i and returns an output o_i. The hidden state vector h_i captures information of the elements that the RNN has seen. More precisely, at the time step i, the network does the following update operations[4]:

$$\begin{aligned}\mathbf{h_i} &= tanh(\mathbf{U}\mathbf{x_i} + \mathbf{W}\mathbf{h_{i-1}} + \mathbf{b}) \\ \mathbf{o_i} &= \mathbf{V}\mathbf{h_i} + \mathbf{c}\end{aligned}$$

where the matrices $\mathbf{U}, \mathbf{V}, \mathbf{W}$ are used, respectively, to convert input vector to hidden vector, hidden vector to output vector and hidden vector to hidden vector. Two vectors $\mathbf{b}, \mathbf{c}$ are the bias vectors and the function $tanh$ is a nonlinearity function. The matrices and the bias vectors are the trainable parameters of the RNN. In order to find these parameters, we compute the gradients of the network using back-propagation thorugh time[18]. However, the RNN as discussed above suffers from the the vanishing or exploding gradients problem which makes it unable to learn from long sequences. A solution to this problem is to implement memory cells in the network to store information over time, which is the idea of Long Short-Term Memory (LSTM)[5,7] and Gated Recurrent Unit (GRU)[2]. In this work, we use LSTM instead of the vanilla RNN to capture long-term dependency in the inputs.

In our setting, we consider an RNN as the first layer in our network. It is modelled by the function $\phi^1_{\theta_1}(e)$ which takes as input the event e which contains $|e|$ discussions and returns the output in the last time step $o_{|e|}$. In particular, $o_{|e|} = \phi^1_{\theta_1}(e)$ and θ_1 is the parameter of the RNN that we need to find. There are two important hyperparameters of the RNN which is the size of the hidden state vector $\mathbf{h_i}$ and the output vector $\mathbf{o_i}$ (denoted as m).

Linear layer This layer applies a linear transformation to its inputs $\mathbf{x}$:

$$\boldsymbol{\phi}^l(\mathbf{x}) = \mathbf{W}^l\mathbf{x} + \mathbf{b}^l \tag{3}$$

where $\mathbf{W}^l$ and $\mathbf{b}^l$ are the trainable parameters with $\mathbf{W}^l$ being the weight matrix, and $\mathbf{b}^l$ is the bias term.

In our model, we use two linear layers after the RNN layer to first convert the output of the RNN to a hidden vector space and then, convert from this hidden vector space to a score vector for the classes. In particular, the second layer of our network $\phi^2_{\theta_2}(\mathbf{o})$ takes as input the output of the first layer (the output vector $\mathbf{o}$) and returns a vector from a hidden space $\mathbf{p}$. More precisely, the layer ϕ^2 takes the vector $\mathbf{o} \in \mathbb{R}^m$ as input and uses the matrix $\mathbf{W}^2 \in \mathbb{R}^{p \times m}$ and $\mathbf{b}^2 \in \mathbb{R}^p$ to convert $\mathbf{o}$ to the hidden space vector $\mathbf{k} \in \mathbf{R}^p$. Similarly, the third layer of our network $\phi^3_{\theta_3}(\mathbf{k})$ takes as input the output of the second layer (the hidden space vector $\mathbf{k}$) and returns a score vector $\mathbf{s} \in \mathbb{R}^2$.

Softmax layer Given an input x, the penultimate layer outputs a score for each class $\mathbf{s} \in \mathbb{R}^2$. The probability distribution is obtained by applying the softmax activation function:

$$P(Y = y|e) \propto \phi_\theta(e, y) = \frac{\exp([\mathbf{s}]_y)}{\sum_{k=1}^{2} \exp([\mathbf{s}]_{y_k})} \tag{4}$$

2.3 Training

In summary, our network is modelled as a function ϕ_θ which is a combination of functions where each function represents a layer. The parameter θ, which combines all the trainable parameters in the network, is obtained by minimizing the negative log-likelihood using stochastic gradient descent(SGD):

$$L(\theta) = \sum_{(e,y)} -\log P(Y = y|e) \propto \sum_{(e,y)} -\log \big(\phi_\theta(e, y)\big). \tag{5}$$

3 Provenance-based Approach

In this section, we discuss how to obtain the provenance of the events that we use in our models, and we present our technique for leveraging provenance to classify the events.

3.1 Provenance of an event

When an event is discussed on social media, it usually has a source or several sources backing it up. When a post discusses an event, it tends to cite one of those sources. For instance, when a tweet mentions an event, it may include the link to the article. We consider these articles as the provenance of the event. As these articles contain detail information about the event, they provide several indicators of rumor. Based on this observation, we also include the provenance of the event into our model.

We consider a *discussion* (e.g. a tweet or a post) to be composed of a *text*, or both an *article* and a text. The article appears in a discussion in form a hyperlink. When both article and text are present, we assume that they are semantically related, e.g. the text is a summary of the article. Traditional techniques in rumor detection only leverage the textual information. As a result, they represent each discussion using only the text. In our setting, as we also consider the provenance of the event, we propose a technique to model a discussion using the text and article information. The text and article information are represented as feature vectors.

3.2 Feature vectors

Before diving into the detail how to represent a discussion its text and article, we discuss the process to represent the article and text in a discussion as feature vectors $(\gamma(i), \psi(s))$ as they are the foundation to construct a post vector

representing a discussion. A good article and text representation can affect the performance of our approach heavily. We model the article i in a discussion with an article feature vector $\gamma(i) \in \mathbb{R}^n$. Similarly, we represent the piece of text s with a textual feature vector $\psi(s) \in \mathbb{R}^m$.

Text Representation In order to represent a text, we aim to convert it from its original format (i.e. words) to an n-dimensional vector. We first calculate the tf-idf values of all words in all the posts. The tf-idf value of a word reflects its importance based on its presence frequency in all the posts relative to the number of posts it appears in. We keep the top-K words with the highest tf-idf values as the vocabulary. Each post is then represented using the words in the vocabulary as a vector of length $|K|$. The value of the i-th element in the vector is 0 if the i-th word in the vocabulary does not appear in the text of the post.

Article Representation An article contains different types of information such as its text, images. However, the images may not contain indicators of rumor. As a result, among different types of information in an article, we only consider its main text. We also follow the same approach from text representation by modeling each article by its tf-idf vector.

3.3 Joint Fusion

From the article and text vector of a discussion, there are many ways to construct a post vector representing the discussion. However, the technique to represent the discussion needs to take into account the missing of the article information. For instance, there are cases that a post may not always explicitly refer to an article and the technique must be robust to these absences. We propose joint fusion which is a technique to combine article and text vector which is able to handle the absence of the article.

Joint fusion takes the article vector and text vector $\gamma(i), \psi(s)$ as input and applies the pooling operation to obtain the post vector $\mathbf{x}$:

$$\mathbf{x} = \text{pooling}(\gamma(i), \psi(s)) \tag{6}$$

The pooling function can be either a *component-wise max* pooling, or an *average* pooling. In this work, we leverage the max pooling and we will extend this work with the average pooling in the future works.

It is worth noting that the pooling operation requires the vectors $\gamma(i) \in \mathbb{R}^n$ and $\psi(s) \in \mathbb{R}^m$ to have the same size. This can be done by adding an extra linear layer to γ (i.e. the network that extracts article feature vector). Assuming $n > m$, the linear layer is as follows:

$$\tilde{\gamma}(i) = \tilde{\mathbf{W}}\gamma(i) + \tilde{\mathbf{b}} \tag{7}$$

where $\tilde{\gamma}(i) \in \mathbb{R}^m$, $\tilde{\mathbf{W}} \in \mathbb{R}^{n \times m}$ and $\tilde{\mathbf{b}} \in \mathbb{R}^m$. The input to joint fusion is then two vectors $\tilde{\gamma}(i)$ and $\psi(s)$. The trainable parameters of joint fusion are $\theta = \{\tilde{\mathbf{W}}, \tilde{\mathbf{b}}\}$.

4 Putting it all together

Recall that our model takes an event as input and returns a prediction for the label of the event. As the event is composed of several discussions, and the discussion contains different types of information, we first model the discussion by combining information from the text and its provenance as discussed in Section 3. Then, we use the events with the post vectors as input to the network. The complete model of our approach is shown in Figure 1. We train the model to find all the parameters in an end-to-end manner which means the parameters of the network and of the joint fusion are trained together. The training is similar to the one discussed in Section 2.3.

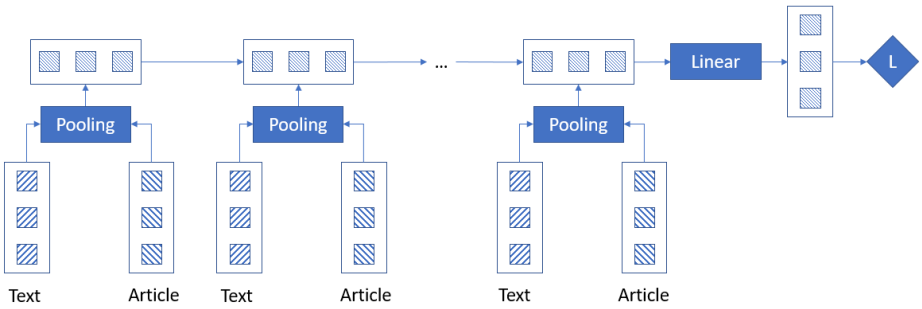

Fig. 1: Model of the network

5 Experiments

In this section, we evaluate our proposed approach on a real-world dataset.

5.1 Dataset

To the best of our knowledge, there is no large-scale dataset for rumor detection that contains both the texts and their provenance. For textual content only, there are the *twitter* and *weibo* dataset which were produced by Ma et. al.[10]. This motivates us to construct a new dataset by adding the article information.

After inspecting the *weibo* dataset, we observe that there is no article information to be added. The reason is that each post from weibo contains only text without any link to an outside article. As a result, in this work, we only focus on the *twitter* dataset[10]. For this dataset, rumor and non-rumor events were identified using a real-time rumor debunking service[1]. The authors of the dataset then extracted keywords from Snopes and used the keywords to query tweets in real-time from Twitter. Due to legal restrictions, only the tweet IDs from the dataset are published instead of its content. Based on these tweet IDs, we crawled their contents including its texts. For the tweets that contain links to a article, we also followed the link and crawled the main text of the article.

[1] snopes.com

Table 1: Statistics of the dataset

Statistics	Twitter
Involved Users	231535
Total Posts	586162
Total Events	992
Total Rumors	498
Total Non-Rumors	494

Table 2: Model hyperparameters

Parameter	Value
Text vector size	$wvdim = 5000$
Article vector size	$avdim = 5000$
Hidden vector size of RNN	$nhid = 800$
Low rank output size	$lowrank = 400$
Classification main task weight	$\lambda = 3$
Fusion layer function	max

However, some of the tweets from the original dataset are missing as they were removed by the users or Twitter. As a result, we can only collected 586162 tweets. Over 60% of them contain a link to an article. The statistics of the dataset is shown in Table1.

5.2 Experimental settings

We compare our proposed approach (joint fusion) with a baseline that does not leverage the provenance information. This baseline is similar to the original approach to rumor detection by Ma et. al.[10].

As some events has thousands of posts, it is inefficient to back propagate through time with such amount of posts. As a result, instead of considering each post separately, we group the posts into partitions and make these partitions as the input to the RNN. For each event, we split the posts in to N partition where each partition has nearly the same amount of posts. It is worth noting that the partitions retain the temporal information among the posts i.e. first partition contains posts that occur first. For all the tweets inside a partition, we concatenate them and generate a longer tweet. Similarly, we also concatenate the main text of the articles appearing in the tweets in a partition.

For regularization, we used a dropout layer with a dropout probability of 0.5 right after the RNN layer to reduce overfitting. We also use a dropout probability of 0.5 for the RNN layer following the suggestion from [17]. In addition, it is reported that factorizing the linear classifier into low rank matrices may improve the classification accuracy[13]. We also followed this approach by adding a linear layer right before the last layer to map the output vector from the RNN layer to a hidden vector space with a size of *nhid*. Regarding the hyperparameters, we tested different values of them on the validation set and select the ones that gave the best results. Table 2 describes other hyperparameters. Our models were trained with a learning rate set to 0.01.

The models were trained on a server equipped with a Tesla GPU. We use 10% events for testing and 10% for validation, the rest is used for training. We use the same splits for all the models in our experiments. All the source codes and the datasets will be released upon the publication of this work.

5.3 Effectiveness of the provenance-based approach

Table 3 shows the experimental results of our approach in comparison with the baseline. The results show that our provenance-based approach is able to outperform the baseline significantly on three metrics: accuracy, recall and F-measure. For instance, our technique has an accuracy of 0.85, which is a 9% relative improvement in comparison with the baseline. The superior performance of our technique is also demonstrated by the recall metric. We are able to recall 92% of events while the baseline approach can only achieve 70%, which is 22% difference. These results show the effectiveness of our provenance-based approach as adding the provenance information allows us to improve the performance significantly. Although our approach has lower precision than the baseline, the difference is extremely small (2%).

Table 3: Performance of the provenance-based approach

Method	Accuracy	Precision	Recall	F-measure
LSTM [1]	0.78	0.83	0.70	0.76
ProvBased [2]	0.85	0.81	0.92	0.86

[1] $wvdim = 5000$, $nhid = 800$, $lowrank = 400$
[2] $wvdim = 5000$, $avdim = 5000$, $nhid = 800$, $lowrank = 400$

5.4 Effects of RNN hidden vector size

In this experiment, we want to analyze the effects of the hidden vector size ($nhid$) to the performance of our approach and the baseline. In order to analyze the effect of $nhid$, we fix other parameters ($wvdim = 5000$ and $lowrank$=400). The experimental results are shown in Figure 2. It is clear that our approach has better accuracy, recall and F-measure over different values of $nhid$. For instance, the recall of our approach is always higher than 0.8 while the recall of the baseline is always lower.

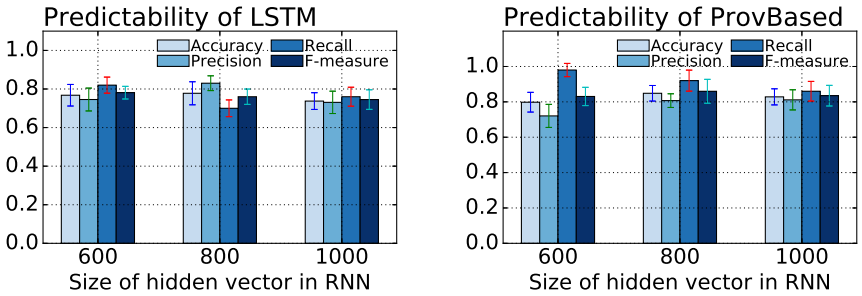

Fig. 2: Effects of hidden vector space on predictive power

5.5 Effects of lowrank vector space

In this experiment, we want to analyze the effect of the lowrank vector space to the performance of two approaches. Similarly, in order to analyze this parameter, we fix $wvdim = 5000$ and $nhid = 800$. It is clear in this experiment that our approach outperforms the baseline significantly. For instance, when the size of the lowrank vector space is 200, our accuracy is 0.81 while the baseline's is only 0.72. We also observe the same pattern with the F-measure metric. When the lowrank vector space is 400, our approach achieves an F-measure of 0.83 while the baseline's F-measure is only 0.78. We are able to achieve the highest performance when the lowrank vector space is 400, which is the value we chose for our model.

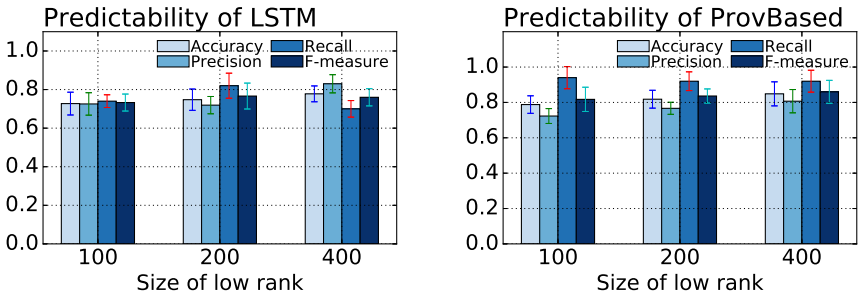

Fig. 3: Effects of low rank layer on predictive power

5.6 Effects of word vector space

In this experiment, we want to analyze the effects of the input vector size to the two techniques. For the sake of simplicity, we set the article and text vector size to the same value. Similarly, we fix other parameters to $nhid = 800$ and $lowrank = 400$. Once again, our approach is able to outperform the baseline across different values of the word vector space. For instance, when the word vector space size is 5000, the baseline has only the precision value higher than 0.8. On the other hand, our approach has 3 metrics which have a higher value than 0.8. We also observe this phenomenon when the word vector space is 10000. Our approach consistently outperform the baseline across all metrics. These experiments confirm our observation that our approach is able to have higher performance due to the addition of the provenance information.

6 Related Work

Rumor detection can be considered a binary classification task. Traditional works on automatic rumor detection aim to construct some classifiers based on hand-crafted features. Several works [3,12,6,15] leveraged linguistic features such as word usage or presence of conjunctions or pronouns. For instance, the authors of [6] found out that fake news usually contain swear words. Many works followed

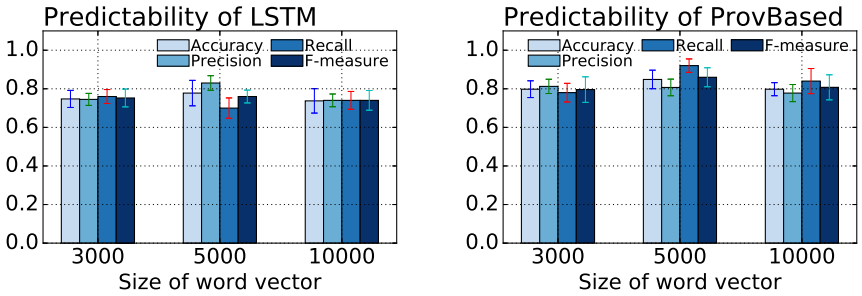

Fig. 4: Effects of word vector size on predictive power

a different direction in which they do not take into account the content of the posts. For instance, some statistical features such as the number of retweets or replies are considered[1,20,11]. Similarly, some user-level features are also used such as the credibility or readability of the users [1,9,14]. A different approach is to examine network level features to detect rumors. For instance, by constructing a tree representing how the messages in an event are related, the authors of [19] is able to classify whether the root is a rumor or not. However, the problem with these hand-crafted features is that the feature engineering process is tedious and time-consuming. In addition, the selected features are usually data-specific and/or domain-specific, which hinders their generality.

Another problem with these approaches is that they do not consider the temporal information of the posts. As the posts are usually temporally related, ignoring this information has a negative effect on the accuracy of rumor detection. Recently, the authors of [10] has leveraged RNN to capture the temporal information of the posts while used deep learning to construct the features automatically. Although this work is the most similar to our work, there are some differences. The approach in [10] does not take into account the provenance of the events, which is an important information to detect rumors. As the tweets are short in nature and they may contain similar phrases, leveraging the provenance information allows us to improve the accuracy significantly.

7 Conclusions

In this paper, we propose a provenance-based approach to detect rumor. Our model is able to combine the provenance information with the textual content to improve the classification accuracy significantly. In addition, our model is robust as it is able to handle the missing of the provenance information. In order to showcase our model, we also enriched a real-world dataset with provenance information which allow us to test our approach in a real-world scenario. Future research directions will go towards adding different types of information such as network or user-level features. In addition, it is worth investigating whether the provenance information is reliable or not.

References

1. Castillo, C., Mendoza, M., Poblete, B.: Information credibility on twitter. In: WWW. pp. 675–684 (2011)
2. Cho, K., Van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259 (2014)
3. Feng, V.W., Hirst, G.: Detecting deceptive opinions with profile compatibility. In: IJCNLP. pp. 338–346 (2013)
4. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), `http://www.deeplearningbook.org`
5. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
6. Gupta, A., Kumaraguru, P., Castillo, C., Meier, P.: Tweetcred: Real-time credibility assessment of content on twitter. In: International Conference on Social Informatics. pp. 228–243. Springer (2014)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation 9(8), 1735–1780 (1997)
8. Hung, N.Q.V., Thang, D.C., Weidlich, M., Aberer, K.: Minimizing efforts in validating crowd answers. In: SIGMOD. pp. 999–1014 (2015)
9. Liu, X., Nourbakhsh, A., Li, Q., Fang, R., Shah, S.: Real-time rumor debunking on twitter. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. pp. 1867–1870. ACM (2015)
10. Ma, J., Gao, W., Mitra, P., Kwon, S., Jansen, B.J., Wong, K.F., Cha, M.: Detecting rumors from microblogs with recurrent neural networks. In: IJCAI (2016)
11. Ma, J., Gao, W., Wei, Z., Lu, Y., Wong, K.F.: 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. pp. 1751–1754. ACM (2015)
12. Markowitz, D.M., Hancock, J.T.: Linguistic traces of a scientific fraud: The case of diederik stapel. PloS one 9(8), e105937 (2014)
13. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
14. Nguyen, Q.V.H., Duong, C.T., Nguyen, T.T., Weidlich, M., Aberer, K., Yin, H., Zhou, X.: Argument discovery via crowdsourcing. VLDBJ pp. 1–25 (2017)
15. Nguyen, T.T., Duong, C.T., Weidlich, M., Yin, H., Nguyen, Q.V.H.: Retaining data from streams of social platforms with minimal regret. In: IJCAI (2017)
16. Nguyen, T.T., Nguyen, Q.V.H., Weidlich, M., Aberer, K.: Result selection and summarization for web table search. In: ICDE. pp. 231–242 (2015)
17. Pham, V., Bluche, T., Kermorvant, C., Louradour, J.: Dropout improves recurrent neural networks for handwriting recognition. In: Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on. pp. 285–290. IEEE (2014)
18. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Cognitive modeling 5(3), 1 (1988)
19. Wu, K., Yang, S., Zhu, K.Q.: False rumors detection on sina weibo by propagation structures. In: Data Engineering (ICDE), 2015 IEEE 31st International Conference on. pp. 651–662. IEEE (2015)
20. Yang, F., Liu, Y., Yu, X., Yang, M.: Automatic detection of rumor on sina weibo. In: Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics. p. 13. ACM (2012)