Implement Rumor Detection on Twitter with Top-down Tree-structured Recursive Neural Network

Anonymous Authors[1]

Abstract

In our present day, social media applications are not only for social networking but are also used as a platform for people to propagate and consume news. Therefore, automatic rumor detection has become an important issue. In this paper, I have implemented a top-down tree-structured recursive neural network for rumor detection on the two public Twitter datasets. Results demonstrate that the model can detect rumor with an accuracy of 77.7% and 79.1% on Twitter15 and Twitter16, respectively.

1. Motivation and Problem Setting

Rumor or fake news can be defined as information that emerges and spreads among people whose truth value is unverified or intentionally false.(DiFonzo & Bordia, 2007). As technology advances, rumors can be spread quickly through the Internet, especially through social media outlets such as Twitter and Facebook. According to the research, fake news diffuses more quickly and more broadly than truth since people are more likely to share fake news which is usually more novel than true news.(Vosoughi et al., 2018) Moreover, social bots are created to magnify the spread of fake news, being responsible for many postungs of political fake news during the 2016 U.S. election and 2017 French election.(Vosoughi et al., 2018) Rumor may cause people to make ill-informed decisions or may even have a significant and harmful impact on public safety. For example, just after the 2017 Manchester terrorist attack, rumors about missing children, terror attacks, etc. near the site of the Manchester terrorist attack are spread and caused public panic.(Vasu et al., 2018) Thus, rumor detection emerged as an important social issue solution. However, it is labor-intensive and time-consuming to identify each rumor amongst a tremendous amount of information on the social media. Therefore, automatic rumor detection is necessary to develop in order to assist people in assessing the truthfulness of the information they consume on social media.

Recent analysis shows that Twitter can "self-correct" some falsehood propagated by users through sharing evidence and opinions with one another.(Zubiaga et al., 2018) Based on this fact, Jing et al. proposed a tree-structured recursive model to capture the signal from the propagation tree of each source tweet on the social media.(Ma et al., 2018) In this paper, I implement this tree-structured recursive neural networks to classify whether a piece of news posted on Twitter is non-rumor, false rumor, true rumor, or unverified rumor. Non-rumor means that the source post is a piece of news and unverified rumor means that the post is a rumor and still unverified. False rumor represents that the post is verified to be false and true rumor is post that is verified to be true. Since the codes implemented with Theano are released by the author, the main goal that I want to achieve is speeding up the training process of the model with the same or higher accuracy rate.

2. Related Work

Many automatic rumor detection approaches were proposed with varying performance and accuracy. In this paper, I select three papers that use different models to detect rumor or fake news. The first method is the tree-structured recursive neural networks proposed by Jing et al., which has 73.7% accuracy of detecting rumors on public dataset Twitter16(Ma et al., 2018). The second model is recurrent and convolutional networks proposed by Yang et al., which shows 86.3% on the Twitter16(Yang et al., 2019). The last approach is a unsupervised framework presented by Yang Liu and Yi-Fang Wu, which has 75.9% on LIAR dataset(Liu & Wu, 2018).

2.1. Rumor Detection on Twitter with Tree-structured Recursive Neural Networks

This method integrates semantic content and propagation structure of each piece of news (source tweet and its corresponding responsive tweets) to capture features and generate a representation of it. Each source tweet is the root node of a

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

tree and its relevant responsive tweets are the other nodes in the tree placed in chronological order. The directions of the edges in the tree decide whether the model is a bottom-up tree structure or a top-down tree structure.

In the top-down tree structure, the directions of edges are from parent nodes to their children nodes and the Gated Recurrent Unit (GRU) is used to recursively estimate the states of each node in the tree from top to bottom. Eventually, all learned representations are projected to the states of leaf nodes which are fed into a max-polling layer to output a single state and predict whether the news is non-rumor, false rumor, true rumor, or unverified rumor.

In the bottom-up tree structure, the direction of edges are from responsive nodes to their responded nodes and GRU is used to recursively estimate the states of each node in the tree from bottom to top. Finally, all learned representations are embedded into the state of the root node which will be used to predict whether the news is non-rumor, false rumor, true rumor, or unverified rumor. However, experiment results indicated that the top-down model has better performance than the bottom-up model and the author believes that it is due to the larger information loss during the recursive estimation process in the bottom-up model. The final output of the bottom-up model only relies on the representation of a single node but the output of the top-down model depends on several leaf nodes.

There are three important parts that the author did not explain clearly in the paper. First, the author did not define each class clearly, especially the difference between the non-rumor and unverified rumor, which makes the understanding of the result slightly vague. Second, the rule for terminating the training process of the model is not defined clearly, that is the number of the maximum epochs and the definition of the convergence of the loss value are not specified which may lead to the performance comparison between the models to be unfair and unclear. Finally, the author did not specify the starting learning rate they used for their experiment results which makes it difficult to repeat and obtain the same result for their experiment.

2.2. Detection of Fake News Through Propagation Path Classification with Recurrent and Convolutional Networks

In this method, the propagation path of every piece of news is first modeled as a multivariate time series which includes tuples that represent characteristics of users who participated in propagating the news. After transforming the multivariate time series into fixed-length multivariate sequences, GRU is used to encode each sequence as a vector of global variation of user characteristics and the convolutional network is used to encode every sequence as a vector of local variation of user characteristics. Finally, every two vectors are concatenated into a vector and fed into the neural network to predict whether it is true news or fake news. Figure 1 is provided by Yang et al. which shows the structure of this model(Yang et al., 2019). There is only one part that the author did not state clearly; whether the user characteristics collected are the data up to date or at the moment that the user posted their posts.

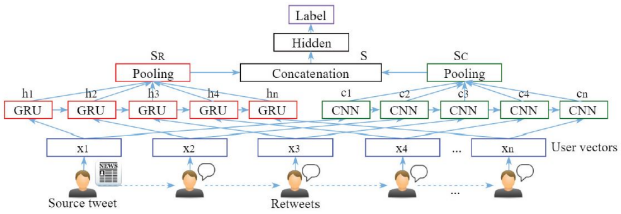

Figure 1. Structure of the propagation path classification with recurrent and convolutional networks.

The results of this model show much higher accuracy than the tree-structured recursive neural network model proposed by Jing et al.(Ma et al., 2018) However, this paper only defines two cases: true or false. For the output label, such a direct comparison of the accuracy between these two models may be unfair. Nonetheless, this paper shows the fact that the user profile of the posts on social media is also a key factor to detect whether a piece of news posted is true or false. I believe it is reasonable since there are some companies and individuals setting up social bots on the social media networks to manipulate public opinion. Therefore, it is evident that a combination of the features of the content of the post, propagation structure, and user characteristics may achieve an even higher accuracy for rumor detection.

2.3. Unsupervised Fake News Detection with A Generative Approach

In this model, all tweets posted by verified users on every news article are collected and unverified users' social engagements of these tweets are also collected which includes likes, retweets, and replies.. Figure 2 shows the hierarchical user engagement model that the author uses for collecting data. The author assumes that verified users may have higher credibility to differentiate between true news and fake news and uses the Bayesian probability graphical model to generate truth latent variables of news and user credibilities. Figure 2 is provided liu et al. which shows the probability graphical model(Liu & Wu, 2018). In the figure, darker nodes are observed variables and white nodes are latent variables. Each news i has a x_i to represent whether it is true ($x_i = 1$) or fake ($x_i = 0$). Every verified user j has a $y_{i,j}$ to represent whether he or she thinks the news i is true ($y_{i,j} = 1$) or fake ($y_{i,j} = 0$). Each unverified user k has a $z_{i,j,k}$ to represent that whether he or she agree ($z_{i,j,k} = 1$) or disagree ($z_{i,j,k} = 0$) with user j's opinion on news i.

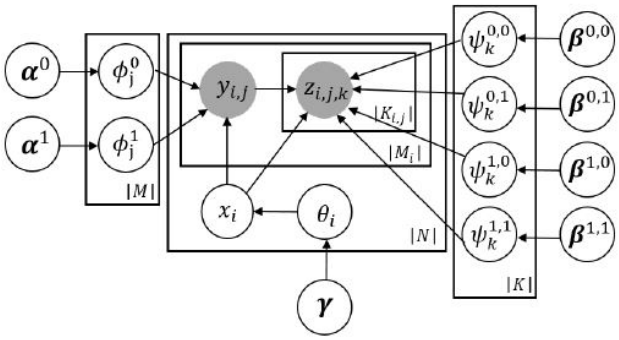

Figure 2. The probabilistic graphic model.

Bernoulli distribution is used to generate x, y, and z with parameters θ, ϕ, and φ, respectively. Beta distribution is used to generate parameters θ, ϕ, and φ with hyperparameter γ, α, and β, respectively. User credibility of verified user j is modeled by true positive rate ϕ_j^1 and false positive rate ϕ_j^0 and user credibility is modeled by $\varphi_k^{0,0}$, $\varphi_k^{0,1}$, $\varphi_k^{1,0}$, and $\varphi_k^{1,1}$. Next, maximum of joint probability of x, y, z, θ, ϕ, and φ are estimated by collapsed Gibbs sampling algorithm to obtain a collapsed Gibbs sampler. The algorithm is shown by the figure 3 provided by liu et al.(Liu & Wu, 2018). Finally, calculate the average values of the samples generated by Gibbs sampler and round them to be 0 or 1 as the output which represents whether the news is estimated as fake news or true news.

Algorithm 1: Collapsed Gibbs Sampling

Randomly initialize $x_i^{(0)}$ with 0 or 1, $\forall i \in \mathcal{N}$;
Initialize counts m for $\forall j \in \mathcal{M}$ and n for $\forall k \in \mathcal{K}$;
Sample record $R \leftarrow \emptyset$;
for $t = 1 \rightarrow$ *iter_num* **do**
  foreach *news* $i \in \mathcal{N}$ **do**
    Sample $x_i^{(t)}$ using Equation (8);
    Update counts;
  if $t >$ *burn-in* & t % *thinning* $= 0$ **then**
    $R \leftarrow R \cup \{\boldsymbol{x}^{(t)}\}$;
return $\frac{1}{|R|} \sum_{\boldsymbol{x}^{(t)} \in R} \boldsymbol{x}^{(t)}$;

Figure 3. The collapsed Gibbs sampling algorithm.

There is one set-up of the model that I think may not always be true. The author treats retweets as positive opinions without checking the content of the retweet. Sometimes, people may retweet news and state their disagreement, thus I think it may not be a cautious assumption for the model.

This model is implemented with unsupervised learning which is a big difference between this model and the previous two models mentioned above. The accuracy of the model seems to be between the previous two models, but it uses different public datasets in which most of the news is related to politics. Therefore, it is hard to tell whether the performance of this model is better or not. However, this model also generates user credibility estimation which may help estimate the user profiles and generate parameter input for other rumor detection models.

3. Solution Approach

In this section, I first introduce Top-down Tree-structured, then Gated Recurrent unit, followed by the recursive neural network, and finally explain why the model works.

3.1. Top-down Tree-structured

In the Twitter dataset, each source tweet (news) is the root node of a propagation tree and its responsive tweets are the children nodes in the tree of which orders are based on their responsive relationships. Figure 4 demonstrates an example of a propagation tree of a true rumor. In the top-down tree

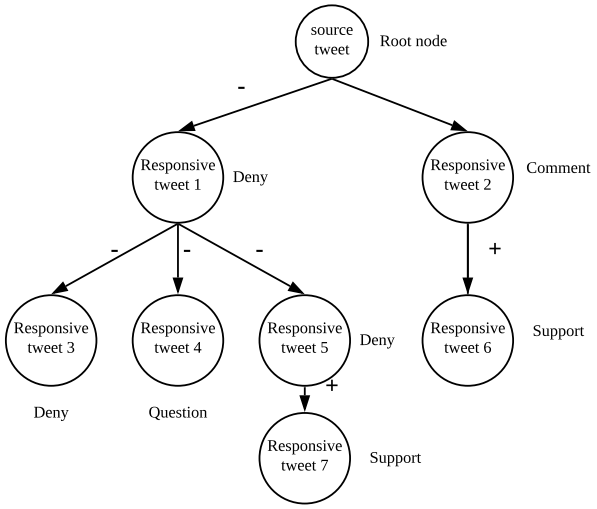

Figure 4. Propagation tree of a true rumor. The possible relationship between a parent node and a child node may be neutral(commenting), negative(denying or questioning), or positive(supporting). Tweet1 and tweet2 are the child nodes of the source tweet. Tweet3, tweet4, tweet7, tweet6 are the leaf nodes of the tree.

structure, edges point from a parent node to child nodes meaning that information flows from the root of the tree down to the leaves of the tree.

3.2. Gated Recurrent Unit

Gated Recurrent Unit (GRU) was proposed by Cho et al.(Cho et al., 2014) which performs well on tasks that are related to capturing long-term dependencies. In the top-down tree-structured model, the representation of each

node is calculated by combining input of itself and its parent node. GRU is used as the hidden unit to learn these textual representations and can be formulated as:

$$\hat{x}_j = x_j E$$
$$r_j = \sigma(W_r \hat{x}_j + U_r h_{p(j)})$$
$$z_j = \sigma(W_z \hat{x}_j + U_z h_{p(j)})$$
$$\hat{h}_j = tanh(W_h \hat{x}_j + U_h(h_{p(j)} \odot r_j)))$$
$$h_j = (1 - z_j) \odot h_{p(j)} + z_j \odot \hat{h}_j$$

where x_j is the original input vector of node j, E denotes the parameter matrix for transforming input as transformed representation $\hat{x}_j$. W_* and U_* are the weight connections inside GRU. x_j and $h_{p(j)}$ are the hidden states of node j and parent of node j, respectively. r_j is the reset gate vector of node j which decides how to combine the current input $\hat{x}_j$ with the memory of parent node and z_j is the update gate vector which determines how much information from parent node needs to be embedded into the current node j.

3.3. Recursive Neural Network

Recursively traverse the tree from top to down, the learned representations are eventually embedded into the leaf nodes in the tree. Subsequently, to align with the size of the output layer, the vectors of all leaf nodes in a tree are fed into a max-pooling layer to obtain a vector in which every dimension is the maximum value selected for all nodes. Finally, the softmax function is used in the output layer to predict the label of the tree (source tweet). Through supervised learning, a classifier can be trained with labeled source tweets. Figure 5 shows the steps of the model.

3.4. Insight of Model

On the social media, a post denying a false rumor often attracts repliers to support or affirm it. On the contrary, a post denying a true rumor often triggers replier to question or disagree with it.(Ma et al., 2018) Therefore, using GRU to transform the signals provided by the propagation tree into representation and enhance it by recursive neural networks can predict whether a rumor is true, false, or unverified.

4. Implementation

In this paper, I modified the source codes released by Jing et al.(Ma et al., 2018)[1] and implement the model with PyTorch.

Input data of the model is preprocessed by origin author with 5000-vocabulary text classifier and 5-fold cross-validation. In the implementation, data are loaded to construct as propagation trees and rearranged as three lists in recursive order

[1]Released source codes from Jing et al. can be accessed from $https://github.com/majingCUHK/Rumor_RvNN$.

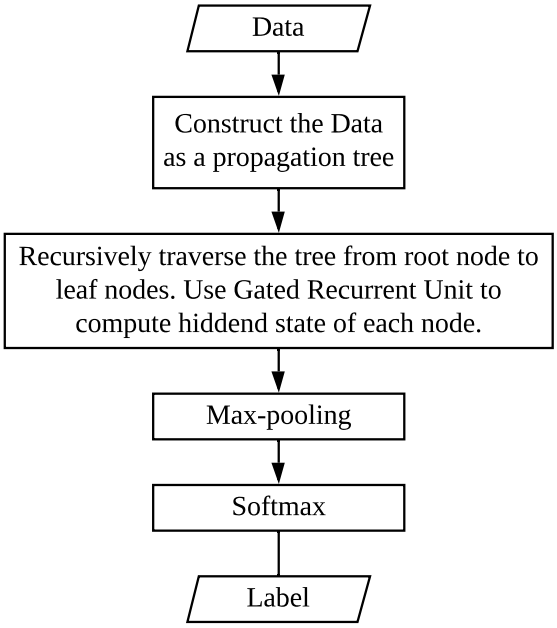

Figure 5. Steps of the top-down tree-structured recursive neural network.

by the functions provided by the original author. The three lists are tree list (list of IDs of tweets in tree order), word list (list of word frequency list in tree order), and index list (list of word index list in tree order).

Subsequently, the three lists are fed into the top-down tree-structured recursive neural networks and transformed as tree tensor, word tensor, and index tensor, respectively. In the order of the items in tree tensor, word tensor and index tensor of each node are fed into GRU to compute its hidden state. Eventually, the hidden states of the leaf nodes of the tree are obtained and fed into the max-pooling layer to form a combined single state for the whole tree. Finally, this single state is passed to the softmax function in the output layer and the probabilities for the four classes are generated as the result. Figure 6 demonstrates the structure of the model implemented with a three-node propagation tree example.

The sum of the squared difference between the result and the actual value is calculated as loss which can be formulated as:

$$L(y, \hat{y}) = \sum_{n=1}^{N} \sum_{c=1}^{C} (y_c - \hat{y}_c)^2$$

where y_c and $\hat{y}_c$ are the actual value and the predicted probability of a class, respectively. C is the number of classes and N is the number of the training cases. The model is trained to minimize this loss by updating the model parameters after each iteration of training.

There are 12 parameters in the model which includes a pa-

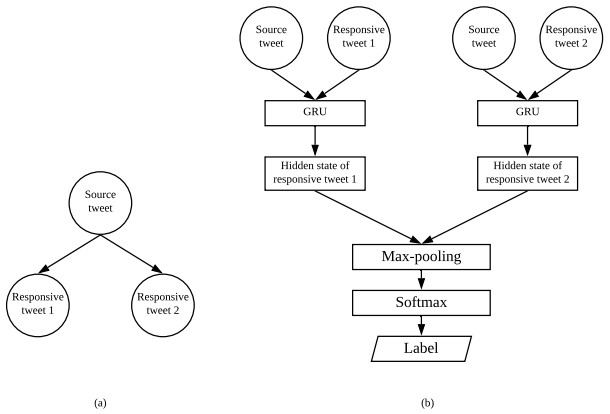

Figure 6. Structure of top-down tree-structured recursive neural network examplified by three-node propagation tree. Figure (a) is the three-node propagation tree and figure (b) is the model implementation for the tree.

rameter matrix for transforming the input of a current node, 2 weight matrices and a bias vector for update gate of GRU, 2 weight matrices and a bias vector for reset gate of GRU, 2 weight matrices and a bias vector for output of GRU, and a weight matrix and a bias vector for output of the model. All the bias vectors are initialized with zeros and other matrices are initialized with random normal distribution. After each training case, these parameters are updated using back-propagation through Adaptive Moment Estimation (ADAM) optimizer. The number of the epoch, vocabulary size, and the hidden dimension are set as 300, 5000, and 100.

For each epoch, all training cases are iterated over to train the model and then the trained model is tested by the testing cases. The accuracy rate is computed by averaging the true accept rates of all the four classes. During the training, I continued the training process until the accuracy rate is converged (continuously shows the same accuracy for eight times) or the number of the epoch reaches 300.

There are two major differences between my implementation and the source code of the original author. Firstly, instead of using Theano for modeling, I used PyTorch to build the model. Second, for the optimizer, I chose to use ADAM which is different from the Stochastic gradient descent (SGD) used by the original author.

5. Experiments

5.1. Dataests

For experiment, I use two public datasets, Twitter15 and Twitter16, which are released by Ma et al. Table 1 shows the detailed statistics of the datasets(Ma et al., 2017). In the datasets, data are provided in tree structure and every tree is labeled with one of the four classes which includes non-rumor, false rumor, true rumor, or unverified rumor.

Table 1. Detailed statistics of datasets Twitter16.

STATISTIC	TWITTER15	TWITTER16
# OF USERS	276,663	173,487
# OF SOURCE TWEETS	1,490	818
# OF SOURCE THREADS	331,612	204,820
# OF NON-RUMORS	374	205
# OF FALSE RUMORS	370	205
# OF TRUE RUMORS	372	205
# OF UNVERIFIED RUMORS	374	203
AVG. # OF POSTS/TREE	223	251
MAX # OF POSTS/TREE	1,768	2,765
MIN # OF POSTS/TREE	55	81
AVG. TIME LENGTH/TREE	1,337 HOURS	848 HOURS

5.2. Rumor Classification Performance

For experimental evaluation, I set the learning rate as 0.001 and ran the model with the two datasets. Table 2 displays the results and Figure 7 shows the changing of the accuracy rate versus the time in minutes. As the table and figure show, the accuracy rate of the model on Twitter16 is slightly higher than that of Twitter15. It may be caused by the difference between the number of posts per tree of the two datasets. From table 1, we can see that the number of posts per tree on Twitter16 is larger than that of Twitter15 which means that the tree structures in Twitter16 are longer and can provide more information for the representation of a tree.

Table 2. Ststidtics of the result of the model trained and tested by Twitter15 and Twitter16.

STATISTICS	TWITTER15	TWITTER16
ACCURACY RATE	77.7%	79.14%
CONVERGED TIME	623 MINUTES	506 MINUTES
CONVERGED EPOCH	60	105
LOSS AT CONVERGED	0.014	0.009

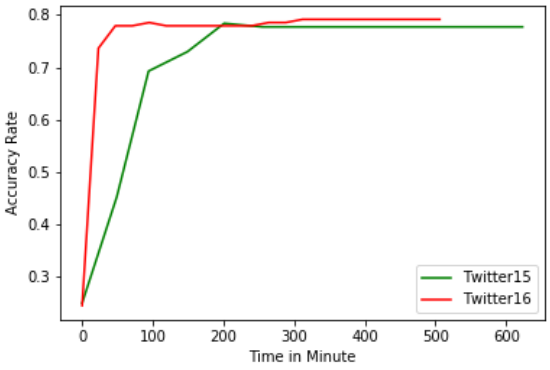

Figure 7. Results of the model tested by datasets Twitter15 and Twitter16.

Table 3. Ststidtics of the results of my implementation tested with Twitter16.

STATISTICS	ACC	PREC	RECLL	F1
NON-RUMOR	82.8%	66.7%	63.4%	65%
FAKE NEWS	90.2%	82.1%	78.1%	80%
TRUE NEWS	92.6%	85.4%	85.4%	85.4%
UNVERIFIED NEWS	92.6%	81.8%	90%	85.7%

According to the paper of the original author, the accuracy rate of the model built with Theano is 72.3% and 73.7% for Twitter15 and Twitter16(Ma et al., 2018). The main reason that my results are better than the original may be the different choices for the optimizer. Although the original author used AdaGrad to improve the performance of SGD, SGD cannot handle saddle point well and tends to become stuck at a local minimum. On the contrary, ADAM can handle these two issues better and allows the model to converge at a higher accuracy rate more rapidly.

To evaluate the performance of the model more clearly, Table 3 provides the accuracy, precision, recall, and F1 of the model I implemented tested with Twitter16. On the precision and recall part, it seems that non-rumor class does not perform as well as other classes and this issue is also shown in the results of the original paper. The reason is that the propagation structure of the non-rumor tree is more diverse and flat, therefore, people tend to respond to non-rumor with little informative content.

5.3. Comparison between models

For the performance comparison, I tested three versions of models with Twitter16 which are the model built with Theano and SGD optimizer (ThSGD model), the model built with Pytorch and SGD optimizer (PyTSGD model), and the model built with Pytorch and ADAM optimizer (PyTADAM model). Table 3 and figure 4 demonstrates the results of each model.

From the results of ThSGD model and PyTSGD model, we can see that the accuracy rates of these two models are about the same, but the time needed for PyTSGD model to be trained and tested is much less than that of ThSGD model. This speed improvement is because the implementation of the recursive neural network is PyTorch instead of Theano. Furthermore, from the results of PyTSGD model and PyTADAM model, we can see that the model with ADAM optimizer can converge faster and get a higher accuracy rate than the model with SGD optimizer. This is consistent with the conclusion I made above that ADAM algorithm can escape from a local minimum more easily than SGD algorithm.

Table 4. Ststidtics of the results of the three models trained and tested with Twitter16.ACC. RATE means accuracy rate, CONV. TIME means time consumed to let the model converge, CONV. EPOCH means the number of epochs trained to let the model converge, and TIME/EPOCH means time spent to run an epoch.

STATISTICS	THSGD	PYTSGD	PYTADAM
ACC. RATE	71.78%	70.55%	79.14%
CONV. TIME	33HOURS	15HOURS	8HOURS
CONV. EPOCH	300	190	105
TIME/EPOCH	0.11HOURS	0.079HOURS	0.076HOURS

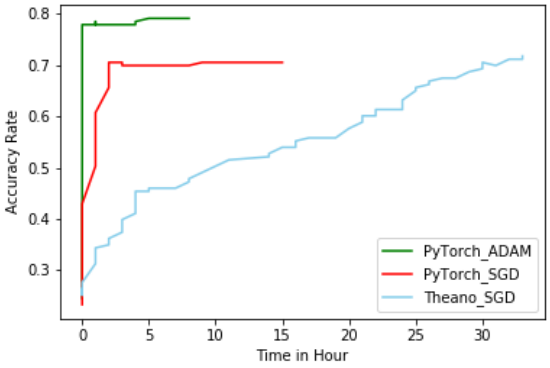

Figure 8. results of the three models trained and tested with Twitter16.

6. Conclusion and Discussion

According to the results of my experiments, there are two main findings. First, using Pytorch as the framework to build the tree- structured recursive neural network model can speed up the training and testing process. Second, ADAM optimizer performed better than SGD optimizer (with Adagrad rule) on detecting rumor with the tree- structured recursive neural network model and can let the model converge at higher accuracy more quickly.

From the second paper that I selected, which detects fake news through propagation path classification with recurrent and convolutional Networks, it is shown that user characteristics of the posts are important factors for detecting rumors. Moreover, the third paper I selected provides the method to estimate user credibility from user profiles. Therefore, collecting user profiles of the posts and utilizing the user credibility estimation method to generate as a parameter of the model may improve the accuracy of the model by a nontrivial amount.

References

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-

decoder approaches, 2014.

DiFonzo, N. and Bordia, P. Rumor, gossip and urban legends. *Diogenes*, 54(1):19–35, 2007.

Liu, Y. and Wu, Y.-F. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *AAAI Conference on Artificial Intelligence*, 2018.

Ma, J., Gao, W., and Wong, K.-F. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 708–717, Vancouver, Canada, 2017. Association for Computational Linguistics.

Ma, J., Gao, W., and Wong, K.-F. Rumor detection on twitter with tree-structured recursive neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1980–1989, Melbourne, Australia, 2018. Association for Computational Linguistics.

Vasu, N., Ang, B., Teo, T.-A., Jayakumar, S., Raizal, M., and Ahuja, J. *Fake news: National security in the post-truth era*. RSIS, 2018.

Vosoughi, S., Roy, D., and Aral, S. The spread of true and false news online. *Science*, 359(6380):1146–1151, 2018.

Yang, S., Shu, K., Wang, S., Gu, R., Wu, F., and Liu, H. Unsupervised fake news detection on social media: A generative approach. In *AAAI Conference on Artificial Intelligence*, pp. 5644–5651, 2019.

Zubiaga, A., Aker, A., Bontcheva, K., Liakata, M., and Procter, R. Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)*, 51 (2):32, 2018.