RumorSleuth: Joint Detection of Rumor Veracity and User Stance

Mohammad Raihanul Islam
Department of Computer Science
Virginia Tech, VA, USA
raihan8@cs.vt.edu

Sathappan Muthiah
Department of Computer Science
Virginia Tech, VA, USA
sathap1@cs.vt.edu

Naren Ramakrishnan
Department of Computer Science
Virginia Tech, VA, USA
naren@cs.vt.edu

Abstract—The penetration of social media has had deep and far-reaching consequences in information production and consumption. Widespread use of social media platforms has engendered malicious users and attention seekers to spread rumors and fake news. This trend is particularly evident in various microblogging platforms where news becomes viral in a matter of hours and can lead to mass panic and confusion. One intriguing fact regarding rumors and fake news is that very often rumor stories prompt users to adopt different stances about the rumor posts. Understanding user stances in rumor posts is thus very important to identify the veracity of the underlying content. While rumor veracity and stance detection have been viewed as disjoint tasks we demonstrate here how jointly learning both of them can be fruitful. In this paper, we propose *RumorSleuth*, a multi-task deep learning model which can leverage both the textual information and user profile information to jointly identify the veracity of a rumor along with users' stances. Tests on two publicly available rumor datasets demonstrate that *RumorSleuth* outperforms current state-of-the-art models and achieves up to 14% performance gain in rumor veracity classification and around 6% improvement in user stance classification.

Index Terms—Rumor Veracity Classification, Social Network Analysis, Deep Learning

I. INTRODUCTION

One of the major reasons for the widespread popularity of social media is their inherent capability to circulate information worldwide within a matter of minutes. A posted message or tweet can be viewed or shared by thousands of users in a matter of hours. However, such pervasive use of social media without any moderation or oversight is not without its attendant negative impacts as it opens up opportunities for malicious users to spread fake news and rumors alongside true news events.

A rumor generally refers to an interesting piece of information widely circulated through a social network whose authenticity cannot be easily verified. A rumor can later turn out to be true, false or remain unverified. The spread of misinformation and fake news has serious negative impacts as it can lead to deleterious effects to users and society.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ASONAM '19, August 27–30, 2019, Vancouver, Canada

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-6868-1/19/08?/\$15.00

<http://dx.doi.org/10.1145/3341161.3342916>

Fig. 1. Example of true, false and unverified rumor stories surrounding the Ottawa shooting

For example, because of a fake report published by ABC news stating that national security advisor Michael Flynn is going to testify against President Donald Trump in the Russia investigation, the DOW Jones industrial average plummeted around 300 points in just eight minutes¹. As another example, a false rumor was deliberately spread stating a pizzeria in Washington DC is conducting human trafficking in secret, which provoked widespread hate speech and confusion². This constant flow of false information has opened up a new research area viz. rumor detection and veracity classification [11], [21]–[23], [28]. Verifying the authenticity of a piece of information is a challenging task as it not only depends on the content but also on the users who are spreading it. However, majority of the existing work on rumor veracity classification relies primarily on textual and linguistic features [14], [21], [22], [28].

Since a rumor story (whether true or false) can trigger a wide variety of user responses, identifying user stances in such posts is an interesting topic of study. Typically it has been observed that false rumors lead to negative or questioning remarks from users [29]. Moreover, some rumor classification methods exploit user stance as a feature [19], [28]. Based on these observations we believe detecting rumor veracity and user stance can be performed simultaneously and each task can benefit from the other. An overview of recent works on rumor and stance detection is shown in Table I.

¹<https://money.cnn.com/2017/12/01/investing/dow-drops-flynn-trump/index.html>

²<http://www.latimes.com/nation/nationnow/la-na-pizzagate-shooting-20170324-story.html>

Present Work: In this work we present *RumorSleuth*, a machine learning model to detect rumor veracity and classify user stance simultaneously. Our model can handle the content of a rumor post along with additional user information and learn a rich vocabulary of features to support prediction. We take advantage of the recent success of the deep multi-task learning paradigm to provide an efficient model which can handle both tasks. Additionally we effectively integrate variational autoencoders (VAE) into the multi-task framework to model user features. Our contributions in this paper are:

- We propose a multi-task framework to jointly classify rumor veracity and detect user stances in social media threads.
- We successfully integrate two dominant deep learning models (RNN for text and VAE for user profiles) to generate rich features to accurately perform both tasks.
- We also propose an improved version, where we try to explicitly model the correlation between the two tasks by feeding the prediction of one task to other to achieve better performance.
- We evaluate our model over two publicly available datasets and compare it against current state-of-the-art models. Our model is able to outperform other models and obtain up to 14% gain in performance for rumor veracity classification.

II. PROBLEM FORMULATION

Rumors constitute interesting news content or stories with questionable credibility. Due to their inherently appealing nature, rumors are widely circulated among users of social media platforms. A rumor can turn out to be *True*, *False* or *Unverified*. Examples of three types of rumors are shown in Fig. 1. Besides while discussing about such rumor stories users can opt to either *support* the rumor, *question* its veracity, *deny* the information or simply *comment* on it. Such actions taken by an user comprises the stance he/she takes w.r.t to the rumor. In the following section, we formally introduce how we tackle the problem of identifying rumor and the stances of the participating users.

Problem Statement: We are given a collection of conversational threads $\mathcal{T} = \{T_1, T_2, \dots, T_{|\mathcal{T}|}\}$. Each thread T_i begins with a post sharing a news story or an incident (i.e. a rumor). Each entry $t_{ij} \in T_i$ can be represented as a tuple $t_{ij} = \langle c_{ij}, u_{ij}, s_{ij} \rangle$. Here $u_{ij} \in \mathcal{U}$ and c_{ij} denote the user and the textual content of the post respectively. $\mathcal{U}$ denotes the set of all users. s_{ij} represents the type of response or the stance the user takes (e.g. *supporting*, *denying*, *commenting* or *questioning*). Suppose the set of stances is $\mathcal{S}$. Now each conversational thread is associated with a label R_i , which represents whether the incident being discussed is *true*, *false* or *unverified*. We are also given a set of user features $\mathcal{F} \in \mathbb{R}^{|\mathcal{U}| \times m}$ where m denotes the feature size.

Given the above information, our goal is to train a model which can predict the veracity of a set of conversational threads (true, false, or unverified) and classify the stances users take on those threads.

TABLE I
COMPARISON OF *RumorSleuth* AGAINST SEVERAL RECENTLY PROPOSED RUMOR VERACITY CLASSIFICATION METHODS. AS WE CAN SEE, *RumorSleuth* CAN EXPLOIT BOTH CATEGORIES OF FEATURES (TEXTUAL AND USER) AND CAN ACCOMPLISH BOTH THE UNDERLYING TASKS: RUMOR VERACITY AND STANCE CLASSIFICATION.

Methods	textual feature	user feature	rumor veracity detection	stance classification
Castillo et al. [4]	✓	✓	✓	✗
Ma et al. [21]	✓	✓	✓	✗
Lukasik et al. [20]	✓	✗	✗	✓
Ma et al. [22]	✓	✗	✓	✓
Kochkina et al. [14]	✓	✗	✓	✓
<i>RumorSleuth</i>	✓	✓	✓	✓

III. OUR APPROACH

In this paper we adopt a multi-task learning approach to jointly learn the rumor class and stances. Multi-task learning refers to learning two or more tasks together leveraging shared structures and correspondences. One of the reasons for adopting a multi-task approach is to effectively exploit correlation inherent in the tasks. For example if an incident is highly controversial or suspected then the responses around it are likely to be of a disbelieving, denying or questioning nature, and vice versa. Similarly a believable story is likely to generate supportive reactions.

Inspired by the recent success of deep neural networks in multi-task learning [18], we present a recurrent neural network (RNN) based multi-task model to jointly classify rumor and stance. The model has one shared layer and additional task-specific layers. The architecture of model is shown in Fig. 2.

A. Shared Layer Description

Recall that a conversational thread T is a sequence of textual responses along with their corresponding user and stance information. Suppose n refers to the number of posts in the thread. First we convert the responses c_j into a fixed size vector $\mathbf{d}_j$ using a TF-IDF score. The TF-IDF vectors $\mathbf{d}_j$ are then passed through two task specific embedding layers. Suppose the outputs of the embedding layer are $\mathbf{d}_j^R$ and $\mathbf{d}_j^S$. Now both embeddings are fed to the shared RNN layer.

$$h_1^R, \dots, h_n^R = LSTM_{shared}(\mathbf{d}_1^R, \dots, \mathbf{d}_n^R) \quad (1a)$$

$$h_1^S, \dots, h_n^S = LSTM_{shared}(\mathbf{d}_1^S, \dots, \mathbf{d}_n^S) \quad (1b)$$

After passing through the shared layer we obtain two fixed-size representations of the input sequence one for each task, where the size of the hidden state h_j^R or h_j^S is K_1 . Now this representation can be input to the task-specific layers for subsequent prediction.

B. User Representation

In rumor classification research, usually only linguistic features of a post are used [14], [21], [22] with the notable exception of [15]. However, it has been shown that the characteristics of a user also has an impact on rumor propagation [4].

Fig. 2. Block architecture of *RumorSleuth* and *RumorSleuth-CO*. Additional components of *RumorSleuth-CO* are shown in dashed line. Here $\odot$ and $\oplus$ denote concatenation and addition respectively.

To demonstrate this we plot the CCDF (complementary cumulative distribution function) of friends and status count of users who first shared the rumor posts in Fig. 3(a) and (b) for the PHEME dataset, one of the datasets used in our evaluation. It can be seen that false rumor discussions are mostly initiated by users who have comparatively lower friends and status counts. We also show the (%) of verified and non-verified accounts for both true and false rumors in Fig. 3(c) and observe that most of the time the account of false rumor starters are not verified. This analysis supports our hypothesis that rumor veracity classification can benefit from leveraging user profile information.

One of the possible ways to incorporate user feature into our model is to apply unsupervised learning methods such as autoencoders to obtain a compact feature representation. Then the latent embedding of the autoencoder can be used in the task specific layers in conjunction with the shared embedding (i.e. h_j^R and h_j^S) for supervised classification. For the choice of autoencoder we use a variational autoencoder (VAE) [13]. A VAE is a non-linear latent variable model based on variational principles. Given a set of user features $\mathcal{F}$, any latent variable model can be expressed as $p(f, z) = p(f|z)p(z)$, where $f \in \mathcal{F}$ and $z \in \mathbb{R}^{K_2}$. The prior of the latent variable is chosen to be a standard Gaussian distribution i.e. $\mathcal{N}(0, 1)$. The conditional distribution $p(f|z)$ is implemented by a neural network which is also the decoder of the model. However, mapping from f to z is bit tricky as it requires us to compute $p(f) = \int p(f|z)p(z)dz$, which is intractable in general. A VAE uses a variational approximation $q(z|f)$ for this purpose where $q(z|f)$ is derived from a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$; here, μ and σ is estimated through another neural network, i.e., the encoder. Now according to the variational

Fig. 3. (a) and (b) shows the CCDF of friends and status count of users who initiate true and false rumors. (c) shows (%) of verified and non-verified accounts that start a rumor post.

Bayes formulation, the KL divergence between $p(z|f)$ and $q(z|f)$ should be minimized. It can be shown that minimizing this KL divergence is equivalent to maximizing the Evidence Lower Bound (ELBO) $= E[\log p(f|z)] - \mathcal{D}_{KL}[q(z|f)||p(z)]$. Therefore the loss function of VAE can be cast as,

$$\mathcal{O}_v = -E[\log(p(f|z))] + \mathcal{D}_{KL}[q(z|f)||p(z)] \quad (2)$$

Here $-E[\log(p(f|z))]$ is the reconstruction loss or the negative log-likelihood. The second term represents the KL divergence between the encoders distribution $q(z|f)$ and the distribution of the latent variable z . It is necessary for the model to allow for deterministic calculation of gradients w.r.t. to the parameters of $q(z|f)$ i.e., μ and σ , for backpropagation. Thus it is common to re-parameterize the model to avoid stochastic sampling of z by moving to a deterministic form $z = \mu + \sigma \times \epsilon$ where $\epsilon \in \mathcal{N}(0, 1)$.

We use the following user features in our model: number of friends, followers, statuses, public lists the user belong to and boolean indicators that represents whether the profile is verified, whether it has a URL and if the profile has a textual description.

C. Rumor Classification Layer

Since the rumor veracity is detected for each conversational thread, we can treat it as a sequence classification problem. For the classification of rumor veracity we take the concatenation of the last hidden state of the shared layer $h_n^R \in \mathbb{R}^{K_1}$ and the latent VAE representation for the initiator of the thread $z_1 \in \mathbb{R}^{K_2}$ as the feature. Then we apply a fully connected softmax layer to obtain the rumor class probabilities. The reason behind using z_1 is from our observation in Fig. 3.

$$\tilde{h}_n^R = h_n^R \odot z_1 \quad (3a)$$

$$w_{rl} = V_r \tilde{h}_n^R + b_r \quad (3b)$$

$$p_{l|\tilde{h}_n^R} = \frac{\exp(w_{rl})}{\sum_{k \in L} \exp(w_{rk})} \quad (3c)$$

Given a set of training instances the objective for rumor veracity is

$$\mathcal{O}_r = \min_{\theta_a, \theta_r} \sum_{i=1}^{|\mathcal{T}|} \left[-\log(p_{l=R_i} | \tilde{h}_{in}^R) \right] \quad (4)$$

Here θ_a and θ_r are the parameters from the shared layer and rumor specific layer receptively.

D. Stance Specific Layer

Stance classification is different from rumor veracity as we have to classify the stance of each post in a conversational thread. This problem can be treated as a sequence labelling problem. For each individual post t_j we compute the feature representation by concatenating the output from the shared LSTM layer h_j^S and the corresponding latent user vector. This feature representation can then be used in a dense softmax layer to perform stance classification. However, empirically we found that directly using the output from the shared layer is not an adequate feature representation for stances. Thus we pass the output of the shared layer to another LSTM network specific to the stance classification task.

$$\tilde{h}_j^S = h_j^S \odot z_j \quad (5a)$$

$$\tilde{p}_1, \dots, \tilde{p}_n = LSTM_P(\tilde{h}_1^S, \dots, \tilde{h}_n^S) \quad (5b)$$

$$w_{sc}^j = V_s \tilde{p}_j + b_s \quad (5c)$$

$$p_{s|\tilde{p}_j} = \frac{\exp(w_{sc}^j)}{\sum_{k \in S} \exp(w_{sk}^j)} \quad (5d)$$

The objective of stance classification is then

$$\mathcal{O}_s = \min_{\theta_a, \theta_s} \sum_{i=1}^n \sum_{j=1}^n [-\log(p_{s=s_{ij}} | \tilde{p}_{ij})] \quad (6)$$

Here θ_a and θ_s are the parameters from the shared layer and stance-specific layer respectively.

E. Introducing RumorSleuth-CO

Previous works like [19] has shown that information regarding stance of users can help predict rumor veracity. We use this idea as a motivation to create an enhanced version of our model. In this enhanced version of *RumorSleuth* named *RumorSleuth-CO* we use the predicted probability distribution of one task to improve the other task's predictive capability. Specifically we compute the probability distribution of each task using logits from Eqn. 3b and 5c. Now for rumor veracity classification, the probability distribution of stance classification w_{sc}^j is fed to a fully connected neural network, then it's output is added with w_{rl} from Eqn. 3b to obtain a joint rumor score x_{rl} . Now this joint rumor score is passed through a softmax layer to obtain the final probability distribution for each rumor class. For *RumorSleuth-CO*, $\mathcal{O}_r$ is to minimize the negative log likelihood of the joint rumor score.

The stance classification also undergoes a similar process. We use another fully connected neural network to transform w_{rl} for a thread and add it with each w_{sc}^j of that thread to get the joint score for stance classes x_{sc}^j . In this way each task contributes to other task's prediction and improve each others prediction. For *RumorSleuth-CO*, $\mathcal{O}_s$ is to minimize the negative log likelihood of the joint stance score.

Now the final objective function is to minimize all three objective functions i.e. $\mathcal{O}_r, \mathcal{O}_s, \mathcal{O}_v$. Let us assume set of all the paramter is Θ . Since the VAE model uses only the user profile information not the textual part, we use a different optimizer to compute the variational loss (Eqn. 2) at the beginning and then use the pre-trained z to jointly minimize $\mathcal{O}(\Theta_{rs}) = \mathcal{O}_r + \mathcal{O}_s$. We use adam optimizer to minimize the objective function.

TABLE II

GENERAL INFORMATION REGARDING THE TWO DATASETS USED FOR EVALUATION. HERE F, T AND U REFERS TO FALSE, TRUE AND UNVERIFIED RUMOR AND S, D, C, Q MEANS SUPPORT, DENY, COMMENT AND QUESTION.

Dataset	Stats.	F	T	U	Overall
PHEME	# of users	613	1563	1260	3359
	max len	46	111	104	111
	avg. len	14.05	15.99	20.73	17.15
	Stance Info	S	D	C	Q
Twitter15-16	# of stance	929	347	3456	361
	# of users	762	307	2377	326
	# of users	16446	13354	8603	37062
	max len	313	334	312	334
	avg. len	31.48	25.35	24.84	27.22

IV. EXPERIMENTAL FINDINGS

A. Experimental Setup

a) *Datasets*: We utilize two datasets. The first is the PHEME rumor scheme dataset [29] which contains 297 conversational threads discussing several breaking news events. Each conversational thread is associated with one of the three labels: *true*, *false* or *unverified*. Moreover each comment in a thread is also annotated with one of the following user stances: *support*, *deny*, *comment* or *question*.

The second dataset we use is a combination of two datasets known as Twitter15 [19] and Twitter16 [21]. We refer to this dataset in this paper as Twitter15-16. The combined dataset contains 1729 conversational threads. However this dataset does not provide stance information of its users (thus, we show the stance performance on the PHEME dataset only). General information regarding the two datasets are shown in Table II.

b) *Baseline Methods*: We compare *RumorSleuth* against several methods including state-of-the-art methods capable of performing either of the two tasks. Description of these methods are given below:

Castillo et al. [4]: This model manually extracts several linguistic, temporal and user features to detect the veracity of rumors. Although originally the work was intended to only detect rumors, here we use the same set of features to classify stance as well.

Doc2Vec [16]: We treat each tweet as a document and apply a doc2vec approach to generate an embedding for each tweet. Then we train a logistic regression classifier to identify rumor veracity and stance.

Shared-MLP: This is a generic deep multi-task architecture using an RNN shared layer and a MLP layer for each task.

BOW: This model uses a shared bag-of-words model and a separate MLP layer for each task.

MT-US [22]: This is a state-of-the-art model which uses a deep multi-task model to detect rumor veracity and stance.

MT-ES [22]: This is the other model proposed in [22], which is an enhancement of the previous model. Here additional task-specific layers are used to recognize signals specific to each task.

Fig. 4. Comparison of performance between all the other methods and the variants of *RumorSleuth* for rumor veracity classification.

RumorSleuth variants: We have compared the baselines with two variants of *RumorSleuth*.

RumorSleuth: This is our proposed model for detecting the rumor veracity and user stance which uses a uniform shared architecture augmented by a VAE for generating features from user profile information.

RumorSleuth-CO: This is an extension of basic *RumorSleuth* which also takes the probability distribution of one task into account while making the prediction for other task.

c) Parameter Settings: We set the input embedding size of the LSTM to 256, batch size to 15, vocabulary size $\mathcal{V}$ to 5000 for all the methods. For *RumorSleuth* variants we set the latent size of the VAE to 8 and the encoder and decoder size to 16. We made a 80-10-10 split of the dataset for training, testing, and validation. We also set the number of epochs to 50.

B. How does RumorSleuth perform against other models in detecting rumor veracity?

To answer this question we demonstrate the performance of each model in terms of micro and macro F1 scores in Fig. 4. We make the following observations from this figure.

- We can see that for both datasets *RumorSleuth* outperforms all other methods. For the PHEME dataset *RumorSleuth* obtains a 6–8% gain against the second best model (MT-US and MT-ES). For Twitter15-16 dataset the gain is modest (~3%). Because it uses user-profile information *RumorSleuth* can generate a richer vocabulary of features than other models to classify rumor veracity.
- *RumorSleuth-CO* provides even more performance boost in case of rumor veracity classification. For PHEME the gain is 13–14%, while for Twitter15-16 improvement is around 5% against the best baseline model. This demonstrates the effectiveness of explicitly modeling the correlation between the two tasks.
- The hand-tuned feature based model, i.e. Castillo et al. [4] demonstrates decent performance compared to the doc2vec model. As this model performs extensive feature extraction from multiple perspectives, e.g. linguistic, users, temporal it performs quite well.
- We also notice that Shared-MLP does not perform very well compared to other multi-task model as it does not use task-specific layers or user information like ours.

Fig. 5. Comparison of performance between all the other methods and variants of *RumorSleuth* for user stance classification.

Fig. 6. (a) Kernel density estimation (KDE) of true class distribution for PHEME dataset. (b) 2D projection of latent vector of the users who initiates a rumor in Twitter15-16 dataset.

C. How does RumorSleuth perform against other methods in terms of stance classification?

We show the performance of all models for stance classification in Fig. 5. We can see that *RumorSleuth* variants outperform other methods in both micro and macro F1 score. Methods based on handcrafted features like Castillo et al. do not perform well on stance classification as the collected features do not capture the stance label well. Although the doc2vec and Shared-MLP formulations perform better than Castillo et al. w.r.t. the micro F1 score, they all show low macro F1 scores depicting their bias against certain classes. The MT-ES model performs very well compared to MT-US because of its enhanced layer representation; however it falls short of *RumorSleuth*. *RumorSleuth-CO* outperforms all other methods in terms of macro-F1 score gaining around 6% improvements.

D. How useful is it to model the correlation between rumor and stance explicitly in case of RumorSleuth-CO?

To answer this question, we plot the kernel density estimation (KDE) of true class distribution in Fig. 6(a). We can see that the joint probability distribution from x_{rl} shown in red (addition of rumor score w_{rl} and rumor prob. dist. from stance) is more towards the right (i.e. it makes confident predictions) than the other two densities. This demonstrates that co-joining the probability distribution of both tasks in *RumorSleuth-CO* improves the performance. We observe similar trends for stance classification (not shown for brevity).

E. How distinct are the latent user vectors?

To evaluate the significance of VAE, we train *RumorSleuth* using only the latent user vectors. We show the 2D projection of the vector with the predicted rumor class in Fig. 6(b). We observe that even in a 2D projection the false and true classes are quite distinguishable. However, unverified rumors are spread around both the classes. Our hypothesis is that unverified rumors share characteristics of both false and true rumors w.r.t. the latent user vectors.

V. RELATED WORK

Rumor Detection: Identifying rumors or gossip content has been studied in several domains [12], [24] because of its impact in social life [6]. Studies on rumor detection using supervised learning with hand-crafted features was conducted in [4], [15]. Propagation structures of information have also been used to identify rumor and news credibility [26]. The dynamics of rumors in social media has also been analyzed in [8], [25]. Finally, there has been studies related to mitigating the spread of rumors [1], [3].

Recently several deep learning methods have been proposed to detect rumors [11], [14], [21]. These models primarily utilizes recurrent neural network model like LSTM for text modeling. However, these models lack any mechanism to integrate the user profile information unlike ours.

Stance Classification: Stance classification is considered an important task in natural language processing, e.g. sentiment analysis, ideology detection [7], [10]. Lukasik et al. [20] propose a Hawkes process model to detect stance in rumor. Recently deep neural models have also been proposed for stance classification [5], [7]. Chen et al. propose UTCNN in [5] which can ingest three types of information (i.e. user preference, topic and comments) to classify stances. Du et al. [7] proposes a target specific attention model for stance classification. However, these models are only stance specific models and cannot perform rumor detection.

Variational Autoencoders: Autoencoders learn a compact representation of the data in an unsupervised manner. The learned representation can be used for any downstream prediction tasks like text classification [27]. The variational autoencoder [13] is a recent addition to autoencoder models and has seen a flurry of interest in many applications, e.g., vision [9], NLP [2], and collaborative filtering [17]. However, none of the models apply VAE to generate user features to detect rumor veracity or stance classification as done here.

VI. CONCLUSION

We have proposed a novel model to jointly detect rumor veracity and user stance using both textual and user features. Due to the increasing importance of social media in every sector, it is crucial to detect the truthfulness of rumors accurately. Additionally detection of user stance can also be very useful in characterizing rumors. Our proposed variants of *RumorSleuth* is very effective in both tasks. *RumorSleuth* leverages the commonalities of both tasks to generate a richer vocabulary of features to effectively classify rumors and stances. In future we

wish to incorporate additional tasks into the model to support its specialization to specific domains such as cyberbullying, identifying conflict in discourse etc.

Acknowledgements: This work is supported in part by the National Science Foundation via grants DGE-1545362, IIS-1633363.

REFERENCES

- [1] M. Amoroso et al. Contrasting the spread of misinformation in online social networks. In *AAMAS*, pages 1323–1331, 2017.
- [2] S. R. Bowman et al. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.
- [3] C. Budak, D. Agrawal, and A. El Abbadi. Limiting the spread of misinformation in social networks. In *WWW*, pages 665–674, 2011.
- [4] C. Castillo, M. Mendoza, and B. Poblete. Information credibility on twitter. In *WWW*, pages 675–684, 2011.
- [5] W.-F. Chen and L.-W. Ku. UTCNN: a deep learning model of stance classification on social media text. In *COLING*, pages 1635–1645, 2016.
- [6] N. Difonzo and P. Bordia. Rumors influence: Toward a dynamic social impact theory of rumor. In *Science of social influence: Advances and future progress*, pages 271–296, 2007.
- [7] J. Du, R. Xu, Y. He, and L. Gui. Stance classification with target-specific neural attention networks. In *IJCAI*, 2017.
- [8] A. Friggeri, L. Adamic, D. Eckles, and J. Cheng. Rumor cascades. In *ICWSM*, pages 101–110, 2014.
- [9] K. Gregor et al. DRAW: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.
- [10] K. Hasan and V. Ng. Stance classification of ideological debates: Data, models, features, and constraints. In *IJCNLP*, pages 1348–1356, 2013.
- [11] Z. Jin et al. Multimodal fusion with recurrent neural networks for rumor detection on microblogs. In *ACM MM Conf.*, pages 795–816, 2017.
- [12] M. A. Kamins, V. S. Folkes, and L. Perner. Consumer responses to rumors: Good news, bad news. *J Consum Psychol*, 6(2):165–187, 1997.
- [13] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [14] E. Kochkina, M. Liakata, and A. Zubiaga. All-in-one: Multi-task learning for rumour verification. *arXiv preprint arXiv:1806.03713*, 2018.
- [15] S. Kwon, M. Cha, and K. Jung. Rumor detection over varying time windows. *PLoS one*, 12(1):e0168344, 2017.
- [16] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196, 2014.
- [17] X. Li and J. She. Collaborative variational autoencoder for recommender systems. In *KDD*, pages 305–314, 2017.
- [18] P. Liu, X. Qiu, and X. Huang. Recurrent neural network for text classification with multi-task learning. In *IJCAI*, pages 2873–2879, 2016.
- [19] X. Liu et al. Real-time rumor debunking on twitter. In *CIKM*, pages 1867–1870, 2015.
- [20] M. Lukasik et al. Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter. In *ACL*, volume 2, pages 393–398, 2016.
- [21] J. Ma et al. Detecting rumors from microblogs with recurrent neural networks. In *IJCAI*, pages 3818–3824, 2016.
- [22] J. Ma, W. Gao, and K.-F. Wong. Detect rumor and stance jointly by neural multi-task learning. In *WWW*, pages 585–593, 2018.
- [23] V. Qazvinian, E. Rosengren, D. R. Radev, and Q. Mei. Rumor has it: Identifying misinformation in microblogs. In *EMNLP*, pages 1589–1599, 2011.
- [24] R. L. Rosnow. Inside rumor: A personal journey. *American Psychologist*, 46(5):484, 1991.
- [25] S. Vosoughi, D. Roy, and S. Aral. The spread of true and false news online. *Science*, 359(6380):1146–1151, 2018.
- [26] K. Wu, S. Yang, and K. Q. Zhu. False rumors detection on sina weibo by propagation structures. In *ICDE*, pages 651–662, 2015.
- [27] W. Xu, H. Sun, C. Deng, and Y. Tan. Variational autoencoder for semi-supervised text classification. In *AAAI*, pages 3358–3364, 2017.
- [28] Z. Zhao, P. Resnick, and Q. Mei. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *WWW*, pages 1395–1405, 2015.
- [29] A. Zubiaga et al. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PLoS one*, 11(3), 2016.