2018 IEEE 30th International Conference on Tools with Artificial Intelligence

Predicting Stances in Twitter Conversations for Detecting Veracity of Rumors: a Neural Approach

Lahari Poddar, Wynne Hsu, Mong Li Lee
School of Computing
National University of Singapore
{lahari, whsu, leeml}@comp.nus.edu.sg

Shruti Subramaniyam*
Department of Computer Science
Columbia University
s.shruti@columbia.edu

***Abstract*—Detecting rumors is a crucial task requiring significant time and manual effort in forms of investigative journalism. In social media such as Twitter, unverified information can get disseminated rapidly making early detection of potentially false rumors critical. We observe that the early reactions of people towards an emerging claim can be predictive of its veracity. We propose a novel neural network architecture using the stances of people engaging in a conversation on Twitter about a rumor for detecting its veracity. Our proposed solution comprises two key steps. We first detect the stance of each individual tweet, by considering the textual content of the tweet, its timestamp, as well as the sequential conversation structure leading up to the target tweet. Then we use the predicted stances of all tweets in a conversation tree to determine the veracity of the original rumor. We evaluate our model on the SemEval2017 rumor detection dataset and demonstrate that our solution outperforms the state-of-the-art approaches for both stance prediction and rumor veracity prediction tasks.**

I. Introduction

Rumor is a circulating piece of story with questionable veracity or truthfulness. Given the increasing penetration of social media in our daily lives, uncensored news updates by media and individuals have become our main sources of information. It has been reported that more than 63% of social media users use Facebook and Twitter for their primary source of news [1]. Existing rumor debunking websites like `factcheck.org` or `snopes.com` use manual investigative journalism which entails a long delay during which a false story could get widely circulated and become disruptive.

Automatically analyzing and determining the veracity of online content has been of recent interest among the web and natural language processing communities. After the initial work [2] of identifying controversial posts from Twitter, increasingly advanced systems have been developed using a wide range of hand crafted features [3], [4], [5], [6]. These approaches leverage user features derived from their demography, followers, posting and re-tweeting behavior, textual features from the text of the posts, and external knowledge from shared links to external sources. However, designing and maintaining these wide range of features from the rich and evolving information present in social media is non-trivial.

In this work, we address the pressing problem of automatic prediction of veracity of a story around which reactions of the social media users are still unfolding. We argue that when a news item starts spreading over social media, peoples' reactions to it contain clues to its truthfulness. We aim to model such opinions, and arguments put forward by people in order to resolve the veracity of a rumor.

Figure 1 shows a sample conversation on Twitter with a source tweet mentioning an event. The subsequent tweets *reply* either directly to the source tweet, or to other tweets in the conversation. Each tweet in a conversation can be of type *support, deny, query* or *comment* depending on its stance towards the rumor. There is a veracity label for the whole conversation indicating whether the rumor is *true, false* or *unverified*.

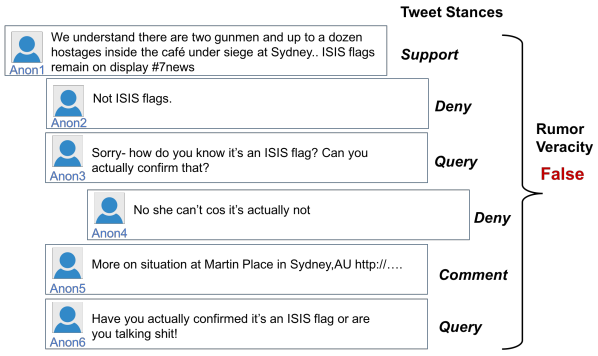


Fig. 1: Sample tweet conversation structure on a rumor claiming ISIS involvement in an attack in Sydney.

We propose a two-step solution to detect the veracity of rumors.

1) Identify the *stances* of all the tweets engaging in a conversation about the rumor.
2) Aggregate the individual tweet stances to predict the rumor's veracity.

The stance prediction component leverages the discourse around a rumor by detecting how users *react* to it in forms of direct/indirect replies. While a person may render direct *support* or outright *deny* a rumor, often people *comment* on a possible rumor tweet with additional information or ask for more information through *queries*. It is important to correctly identify these stances, since their distributions can be distinctive for different classes of rumors e.g. false

*Work done while interning in School of Computing, NUS

2375-0197/18/$31.00 ©2018 IEEE
DOI 10.1109/ICTAI.2018.00021

rumors tend to evoke a lot more *deny* and *query* tweets than a rumor which is true. To this end, we design a novel neural architecture for predicting the stances that considers the conversation tree structure, i.e., the textual content of the target tweet, its timestamp, as well as its context.

To encode the textual content of a tweet, we employ convolutional neural networks inspired by their recent success for natural language processing tasks [7], [8], [9], [10], [11]. We further augment it with attention layer to help the network focus on parts of a tweet that are important for identifying its stance. However, due to the short and conversational nature of tweets, using only the tweet text is often not sufficient to understand its stance, e.g., the tweet "*No she can't cos it's actually not*" in Figure 1. Since this is in response to an ongoing conversation, looking at its preceding tweet "*Sorry- how do you know it's an ISIS flag? Can you actually confirm that?* ", makes its stance clear. We account for the context of a target tweet by taking into consideration all its preceding tweets in the reply chain of a conversation tree. The sequential nature of conversation is captured through a recurrent neural network (RNN) due to its superiority in handling sequential data. Additionally, we observe that not all tweets preceding a target tweet is equally important in understanding its stance. Therefore, we include a tweet-level attention mechanism to help the RNN focus on the relevant parts of the conversation.

To the best of our knowledge, this is the first work that uses two-level attention over textual content as well as at the tweet-level to encode the conversational nature of a tweet in order to understand its stance and in turn predict rumor veracity.

After predicting the stances of all the tweets in a conversation tree, we aggregate the predictions along with the textual contents of the tweets to determine the rumor veracity. We analyze several methods of combining the stance prediction component with the veracity prediction step. We optimize the combined network using a transfer learning approach with full fine tuning of the weights learned in the first step. Experimental results on a real-world dataset from Twitter show that our approach significantly outperforms other competitive methods for both stance prediction and veracity classification.

II. Preliminaries

We use a dataset published as part of the PHEME project [12]. This dataset is subsequently used in a SemEval 2017 task [13]. The data contains online conversations on Twitter, each pertaining to a particular event and the rumors around it.

Each conversation forms a tree T as shown in Figure 2. The root node A is a *source tweet* that initiated the discussion. A directed edge denotes the reply of a tweet to its parent tweet. Each tweet is associated with a timestamp at which it has been posted e.g., tweet C is posted at ts_C. The *conversation sequence* of a tweet is defined by the chain of tweets starting from its parent and going all the way up to the source tweet., e.g., the conversation sequence for tweet I is $\{A, D, F\}$.

To understand the importance of people's stances in determining the veracity of a rumor, we first look at the distribution of stances of tweets concerning rumors of different veracity.

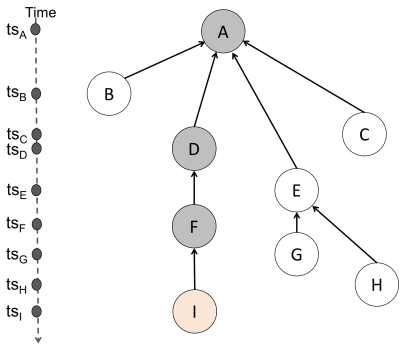


Fig. 2: Sample tweet conversation tree.

	False	True	Unverified
Comment	63.26	63.86	65.32
Support	18.93	22.18	18.46
Deny	11.71	5.99	7.52
Query	6.10	7.96	8.70

TABLE I: Stance distribution of tweets in conversation trees of different types of rumors.

As we can see from Table I, the distribution of stances for different types of rumor are quite discriminating. For example, number of *support* tweets are higher for a true rumor whereas higher number of *deny* tweets are sparked for a rumor which later turned out to be false. Rumors that remained unverified have a greater percentage of query tweets.

III. Proposed Solution

Motivated by our observation of the discriminating stance distribution for different types of rumors, we design a two-step solution that takes into consideration the **C**onversation **T**ree structure. The first step predicts the stances of individual tweets via CT-Stance. The second step aggregates the predicted stances via CT-Veracity.

A. Stance Prediction

We consider three signals for a target tweet: textual content, conversation sequence, and the timestamp that the tweet is posted. Figure 3 shows the overall architecture for our stance predictor model CT-Stance.

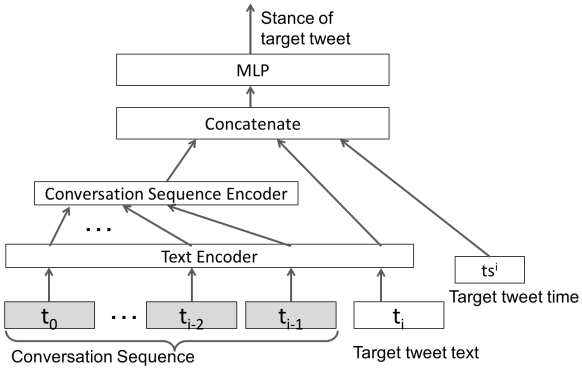


Fig. 3: Architecture of CT-Stance.

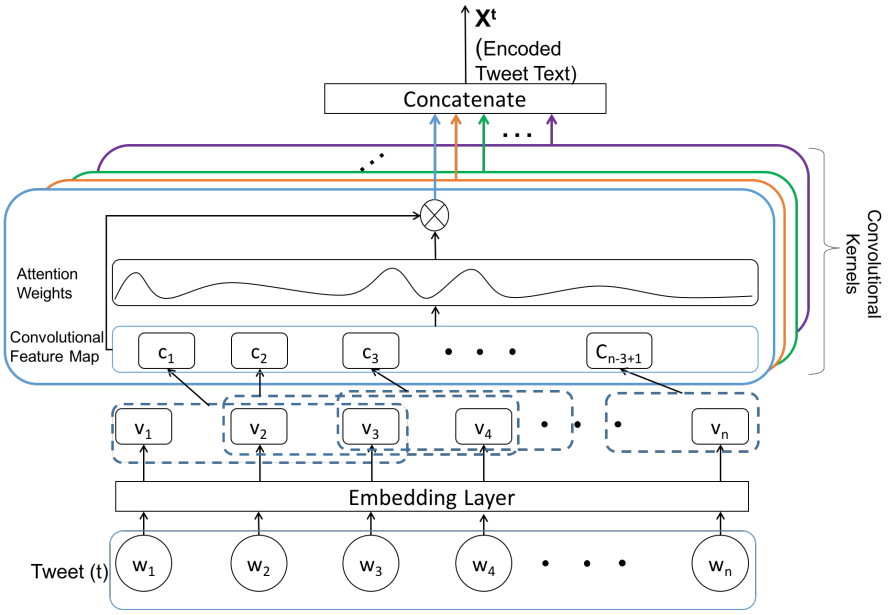


Fig. 4: Text Encoder.

Each tweet is first encoded by a CNN-based text encoder, and an RNN-based conversation sequence encoder is used to represent the context of the target tweet. The encoded representations of the signals are thereafter used for prediction. Details of the network components are given below.

Tweet Text Encoder We first encode the text of an individual tweet t, denoted as a collection of words $t = \{w_1, w_2, \cdots, w_n\}$. Figure 4 shows the text encoder.

Each word is embedded in a lower dimensional space so that a tweet is now represented as a sequence of word vectors $\{\mathbf{v_1}, \mathbf{v_2}, \cdots, \mathbf{v_n}\}$ where $\mathbf{v_i} \in \mathbb{R}^d$. We initialize the word vectors using pre-trained Glove embeddings [14] but tune it during training to capture the intrinsic features of the specific task at hand.

We apply a one dimensional convolution followed by a $tanh$ non-linearity on the sequence of word vectors. The convolutional kernel is parameterized by $\mathbf{W} \in \mathbb{R}^{d \times l}$, $b \in \mathbb{R}$ where d is the dimension of a word and l is the filter length. It processes l consecutive word vectors and maps them to a single output which can be used as a feature. For example, a feature c_i is generated from a window of words $\mathbf{v_{i:i+l-1}}$ by

$$c_i = tanh(\mathbf{W}.\mathbf{v_{i:i+l-1}} + b) \tag{1}$$

The kernel slides over the embedded vectors of each *l-gram* and produces a map of features $\mathbf{c} = [c_1, c_2, ..., c_{n-l+1}]$ as the output. The output is padded to make its length the same as the input length i.e. n.

Traditionally, a standard max-over-time pooling operation [15] is performed over the feature map to produce the single most important feature. However, multiple non-consecutive sections of a tweet could be important in understanding its stance, making max pooling insufficient.

In order to identify the parts of a tweet that are important in determining its stance, we use a self-attention [16] mechanism over the output of the convolutional layers. For each *l-gram* $\mathbf{v_{i:i+l-1}}$, we compute a weight a_i to determine the contribution of its corresponding feature vector c_i to the stance of the whole tweet and get an attention vector $\mathbf{a} = \{a_1, a_2, \cdots, a_n\}$ as

$$\mathbf{a} = softmax(tanh(\mathbf{W}.\mathbf{c})) \tag{2}$$

The tweet representation for a kernel j is computed as:

$$x_j = \sum_i^n a_i c_i \tag{3}$$

We use three different filter lengths ($l \in \{2, 3, 4\}$) and 128 such kernels for each filter length to detect multiple features and concatenate all extracted features to get the final tweet text representation denoted as $\mathbf{x^t}$.

Conversation Sequence Encoder Next, we encode the preceding tweets in the conversation sequence of a target tweet t by using a bi-directional RNN. The input to the bi-directional RNN is the encoded tweet text representations $\{\mathbf{x^1}, \mathbf{x^2}, \cdots, \mathbf{x^{t-1}}\}$. Figure 5 shows the conversation sequence encoder.

The RNN reads the sequence in left to right direction in the forward pass and creates a sequence of hidden states $\{\mathbf{h_f^1}, \mathbf{h_f^2}, \cdots, \mathbf{h_f^{t-1}}\}$, where $\mathbf{h_f^i} = RNN(\mathbf{x^i}, \mathbf{h_f^{i-1}})$ is a function for which we use a GRU [17]. In the backward pass, it reads the input sequence in reverse order and returns a sequence of hidden states $\{\mathbf{h_b^{t-1}}, \mathbf{h_b^{t-2}}, \cdots, \mathbf{h_b^1}\}$. The forward and backward hidden states are then concatenated to create the encoded hidden state of a context tweet $\mathbf{h^i} = [\mathbf{h_f^i}, \mathbf{h_b^i}]$ considering all its surrounding tweets.

We experimented with replacing GRU by LSTM [18] which resulted in similar performance at the cost of longer training time due to larger number of parameters. We use stacked bi-directional GRUs where the output hidden states of a layer are

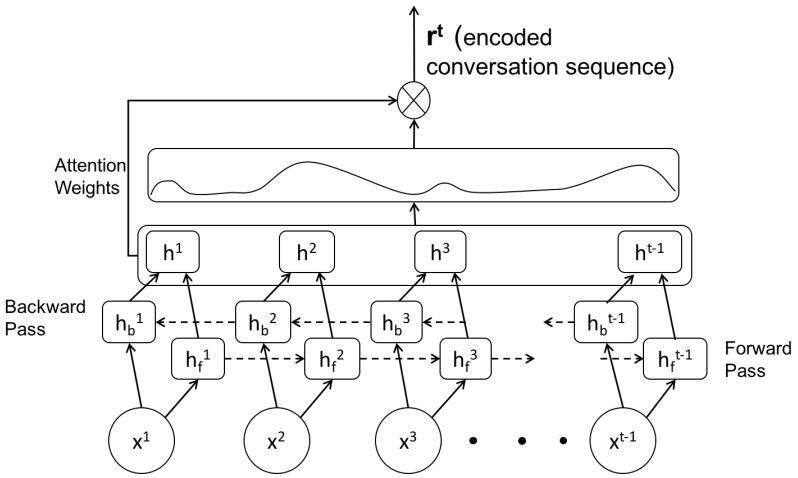


Fig. 5: Conversation Sequence Encoder

fed as input sequence to the next layer. The output of the last such layer is considered as the feature vector for the context of the target tweet.

We apply a tweet-level attention over the conversation sequence to focus on the important tweets in the conversation. We compute the context attention weights a_i^c for the feature vector $\mathbf{h_i}$ corresponding to each tweet in the conversation sequence. The attention vector $\mathbf{a^c} = \{a_1^c, a_2^c, \cdots, a_{t-1}^c\}$ is then multiplied with the corresponding features $\mathbf{h_i}$, and a weighted sum is calculated (similar to Equation 3) to get the context representation $\mathbf{r^t}$.

Temporal Signal Encoder. The time elapsed since the source tweet could influence the type of response tweets it generates. For example, when an unverified news emerges, people typically voice their opinions from pre-conceived notions and the limited evidences available at that time to *support* or *deny* the claim. However, as time progresses and more evidences come in, we observe that people try to reason and evaluate the repercussions of the event by *commenting* on earlier tweets with posts like 'why this outrage let's calm down' , 'no one would care if a white kid was shot but now people care because he is black', 'maybe he left his Taser in the car and so he used his gun' and so on.

To study this observation further, we plot the percentage of tweets belonging to the majority two stance classes (*comment* and *support*) arriving within varying time windows. Figure 6 shows that as more time elapses since the source tweet, the percentage of reply tweets commenting on the rumor increases while the percentage of support decreases. This motivates us to use the temporal information as a signal in our network. For a target tweet t, we encode its temporal feature ts^t as the difference (in seconds) between the posting time of the source tweet and that of the target tweet.

CT-Stance Predictor Given a target tweet t, we concatenate its text representation $\mathbf{x^t}$, its context representation $\mathbf{r^t}$, and temporal feature ts^t to form the final tweet representation $\mathbf{z^t} = [\mathbf{x^t}; \mathbf{r^t}; ts^t]$. The vector $\mathbf{z^t}$ is fed through stacked fully

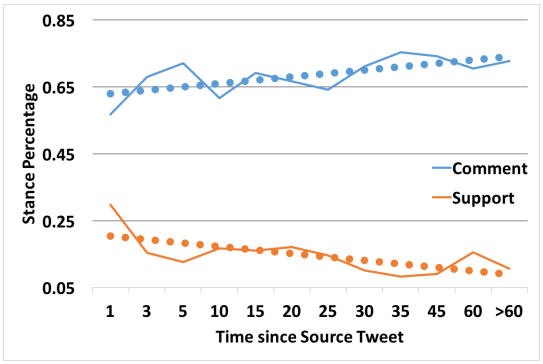


Fig. 6: Distributions of tweets belonging to *comment* and *support* class over time. Dotted lines show the trends that with time *comments* increase while percentage of *support* decreases.

connected layers and the output of the last layer is passed through a *softmax* layer to output a probability distribution over the four stance classes.

$$p(\mathbf{y^t_{stance}}|\mathbf{z^t}) = softmax(\mathbf{W}.\mathbf{z^t} + \mathbf{b}) \quad (4)$$

where $\mathbf{y^t_{stance}}$ is the probability distribution over the four stance classes for the tweet t. The model is trained using cross-entropy loss function and optimized with Adam optimizer [19].

B. Veracity Prediction

In order to classify the veracity of a rumor, we take as input a complete conversation tree T (recall Figure 2). Based on the conversation tree, each individual tweet t (its textual content and timestamp) and its conversation sequence is first fed through CT-Stance to obtain the probability distribution over stances for each tweet, denoted as $\mathbf{y^t_{stance}}$. Figure 7 shows the architecture of the veracity classification model CT-Veracity.

The probability distribution over four stance classes of individual tweets are then averaged to obtain a probability

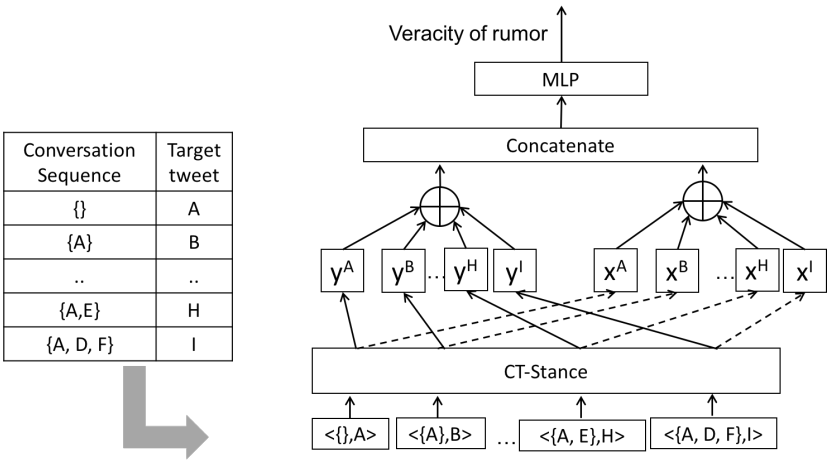


Fig. 7: Architecture of CT-Veracity. Each row in the table shows the conversation sequence for a target tweet from the conversation tree.

distribution over stances for the complete tree.

$$\mathbf{y^T_{stance}} = \frac{1}{|T|} * \sum_{t \in T} \mathbf{y^t_{stance}} \tag{5}$$

where $|T|$ denotes the number of tweets in T.

Apart from the output stance probability distribution, the stance predictor component learns a tweet text representation $\mathbf{x^t}$ for each tweet t in T. We combine these individual tweet representations to form a textual representation of T by taking an average,

$$\mathbf{x^T} = \frac{1}{|T|} * \sum_{t \in T} \mathbf{x^t} \tag{6}$$

Thereafter, the stance distribution and textual representation of the tree are concatenated and fed through a fully connected layer with softmax to predict the veracity of the rumor discussed in the conversation tree.

$$\mathbf{y^T_{veracity}} = softmax(\mathbf{W}.[\mathbf{y^T_{stance}}; \mathbf{x^T}] + \mathbf{b}) \tag{7}$$

where ; denotes concatenation operation and $\mathbf{y^T_{veracity}}$ is the probability distribution over three veracity classes.

Now we move on to address the coupling of CT-Stance into the architecture of CT-Veracity model. We first note that the data for the veracity prediction task is considerably smaller than the stance prediction task, since there is a single veracity label for a whole conversation tree in contrast to a label for each tweet stance. To overcome this challenge we adopt a transfer learning approach for training CT-Veracity.

In transfer learning, a base network is trained first, and then the learned features are reused or *transferred* to a second network to be trained on a target task. It has proven to be a powerful learning tool when the target dataset is much smaller compared to the base dataset. For neural networks, the weights of the first n layers from a pre-trained base network are copied to the first n layers of the target network and the remaining layers of the target network are initialized randomly.

Following this principle we pre-train our base network (CT-Stance) and copy the corresponding layer weights to our target network (CT-Veracity). While training CT-Veracity, we backpropagate the error into the transferred features from CT-Stance as well, essentially *fine-tuning* them.

IV. Experiments

We carry out a comprehensive set of experiments to evaluate our proposed solution. We use the online Twitter conversation dataset of the SemEval 2017 Challenge [13]. The training dataset consists of tweets spanning eight events such as the 'Charlie Hebdo shooting in Paris', 'The Ferguson unrest in the US', and 'The GermanWings plane crash' etc. The test data consists of conversation trees related to some events from the training set as well as two unseen events. We report the results after averaging five runs on the test set.

A. Evaluation of CT-Stance

We first compare CT-Stance with the following state-of-the-art neural stance prediction models that consider different input signals:

- CNN [20]. This method uses a convolutional neural network on the target tweet text to predict its stance.
- Branch-LSTM [21]. This method uses the entire conversation tree for predicting stances of each of its nodes.
- CT-Stance$^-$. This is the same as CT-Stance except that the temporal signal is not used. In other words, it only considers the target tweet text and the conversation sequence.

Table II shows the results. We observe that considering the target tweet as well as the conversation sequence is important in understanding the discourse properly and predicting its stance. Incorporating the temporal information helps in boosting the performance further.

We note that, although the branch-LSTM [21] obtains a competitive score, it uses some input signals which might not be available in a real-time system. In order to predict the stance of a tweet, it looks up *all* the tweets in the tree, including the ones posted in the *future* with respect to the target tweet. On the other hand, our model only uses the

Model	Input Signals	Accuracy
CNN [20]	Target tweet text	70.06%
Branch-LSTM [21]	Entire conversation tree (includes *future tweets*)	78.4%
CT-Stance$^-$	Target tweet text, conversation sequence	78.02%
CT-Stance	Target tweet text, conversation sequence, time	**79.86%**

TABLE II: Comparison of Stance Prediction Models that consider different subsets of input signals. Our model CT-Stance achieves the best performance when considering all three realistically available signals.

Variants of CT-Stance	Accuracy
Text Encoder + Concatenation	72.21%
Text Encoder with Attention + Concatenation	74.35%
Text Encoder + Conversation Encoder	77.50%
Text Encoder with Attention + Conversation Encoder	79.17%
CT-Stance (Text Encoder with Attention + Conversation Encoder with Attention)	**79.86%**

TABLE III: Performance of architecture variants of CT-Stance. Using a sequence encoder for the conversation greatly improves the accuracy compared to simple concatenation. The model achieves the best scores with the use of attention at both text and tweet levels.

Model	Input Signals	Accuracy
GRU-2 [22]	Tweet texts	45.85%
CAMI [23]	Tweet texts	50.0%
Bi-GRU-2	Tweet stances (ground truth)	50.57%
CT-Veracity	Tweet texts, tweet stances (predicted by CT-Stance)	**57.14%**

TABLE IV: Comparison of Rumor Veracity Prediction Models. This demonstrates the effectiveness of tweet stances in determining a rumor's veracity. Our CT-Veracity model achieves the best performance compared to the state-of-the-art rumor detection methods.

preceding tweets in the conversation sequence for predicting stance of a target tweet, which is more realistic. From the results, we can observe that in comparison to branch-LSTM, our model achieves comparable scores using only the realistically available conversation sequence (CT-Stance$^-$) and outperforms using temporal information (CT-Stance).

Next, we investigate the effectiveness of the various components in CT-Stance by implementing the following variants:

- Text Encoder + Concatenation. We use the convolution layers as the text encoder and concatenate the hidden text representations to form the conversation sequence.
- Text Encoder with Attention + Concatenation. We use the convolution layers with attention as text encoder and concatenate the hidden text representations to form the conversation sequence.
- Text Encoder + Conversation Encoder. We use convolution layers as text encoder and use 2 layers of stacked Bidirectional GRU as conversation sequence encoder.
- Text Encoder with Attention + Conversation Encoder with attention. We use the convolution layers with attention as text encoder and use 2 layers of stacked Bidirectional GRU as conversation sequence encoder.

For fair comparison, the final prediction layers and the input signals for all the variants are kept identical.

Table III shows the results. As we can see from the results, encoding the conversation sequence properly using bidirectional GRUs produces a huge improvement over a simple concatenation. This is in line with most of the recent works that have found the efficacy of RNNs in sequence representation across domains. We also note that using attention mechanism further boosts the performance by enabling the model to concentrate on important parts for stance prediction.

B. Evaluation of CT-Veracity

Veracity is a three class (*true, false, unverified*) classification task and we use accuracy as its performance metric.

We compare CT-Veracity with the following state-of-the-art rumor detection approaches:

- GRU-2 [22]. This uses two stacked GRU layers to encode the sequence of textual contents of tweets being posted about the rumor.
- CAMI [23]. This uses convolutional neural network to encode consecutive tweets of an event.

We note that these previous works addressing similar tasks have modeled the tweet texts directly for veracity prediction, without considering a tweet's stance, unlike our approach. Therefore, we also design the following baseline to investigate if knowing the ground truth stances of tweets helps improve the accuracy of veracity prediction.

- Bi-GRU-2. This is a baseline that considers only the sequence of ground truth stances for the tweets and use two layers of stacked Bidirectional GRU to encode it. This baseline demonstrates the rumor detection accuracy achievable by only considering stances of tweets.

We make two key observations from the results shown in Table IV. Firstly, we observe that the ability to detect rumors is greatly benefited by directly considering the stances of tweets compared to only its textual contents as demonstrated by baseline Bi-GRU-2. Secondly, CT-Veracity model outperforms the competitive methods comfortably by considering both the stances as well as the tweet contents.

As the CT-Veracity model considers the predicted stances of tweets, the accuracy of the CT-Stance model and their coupling plays an important role in determining the overall accuracy.

In the next set of experiments, we investigate how the coupling strategy can influence the CT-Veracity model performance by evaluating multiple alternatives.

We consider the following,

- Pipeline model. We train the CT-Stance model first. Thereafter, we use the predicted stances from it, and the encoded text representations of the tweets from the text encoder component as input to CT-Veracity.
- Joint model. We train a single model using a multi-objective loss function that optimizes both the stance prediction and veracity prediction tasks together.
- Transfer learning with frozen weights. We train CT-Stance first and copy the weights of the corresponding layers in the complete model. The weights of the text encoder component are kept frozen during the training of CT-Veracity. This is a prevalent practice for training on smaller datasets, to avoid learning those parameters which are already learnt well in another task in order to avoid overfitting [24].

Method	**Accuracy**
Pipeline model	41.23%
Joint model	44.45%
Transfer learning with frozen weights	50.15%
CT-Veracity (Transfer learning with fine tuning)	**57.14%**

TABLE V: Performance of Variants of CT-Veracity.

Table V shows the results. We firstly observe that the transfer learning based approaches outperform both the joint and the pipeline model. This is due to (i) the dependency between stance prediction and veracity prediction tasks, and (ii) imbalance between dataset sizes. For the joint model, the network tries to optimize both objectives together, and learns a sub-optimal stance prediction model possibly due to overfitting on the veracity prediction task. In the pipeline model, since CT-Stance is trained independently, the overall stance prediction accuracy is the best. However, the recall for the under-represented stance classes (*query, deny*) are lower than the majority classes (*comment, support*). This affects the veracity prediction accuracy since they are the most discriminative classes for determining rumors (as shown in Table I).

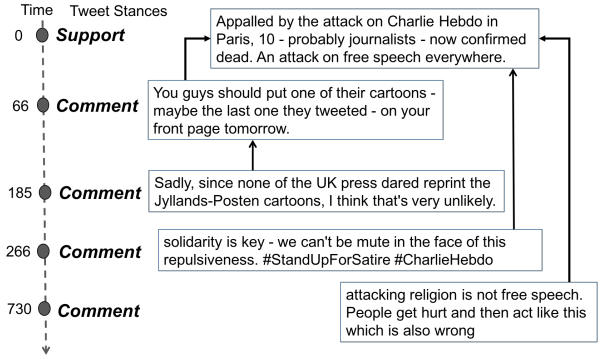
On the contrary, as the transfer learning with fine tuning approach is able to change the weights in stance prediction component, the overall accuracy of the stance prediction component decreases slightly but recall for the other three classes increase significantly. This helps in achieving high accuracy for veracity prediction. We note that by freezing the transferred weights, it becomes non-trivial for gradient-descent to optimize a network that has been split in-between. This can be attributed to task-specific co-adaptation of neighboring layers [24].

C. Case Study

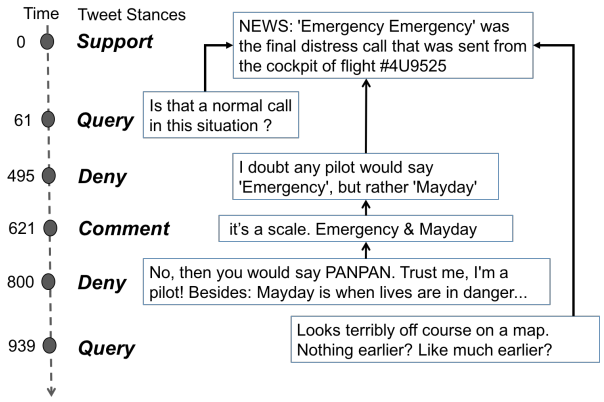
Finally, we present a study for different cases of rumor detection successfully handled by our model. Figure 8 shows the conversation trees within the first few minutes for two different types of rumors.

In Figure 8(a), a rumor regarding 'Charlie Hebdo shooting in Paris' is presented. We observe that the responding tweets mostly are expressing solidarity or voicing personal opinions, but are not raising questions regarding the event. Hence our model predicted its veracity to be *true*.

In Figure 8(b), a *false* rumor about the 'final distress call from Flight 4U9525' is depicted. We observe that some people respond by expressing doubts regarding the nature of the distress call mentioned in the source tweet. These initiate a conversation where people start pointing out the inconsistencies in the reported information invoking further queries and denials. Considering this conversation structure and the presence of many *deny* tweets our model successfully predicts it to be a false rumor.



(a) True Rumor



(b) False Rumor

Fig. 8: Illustration for conversation trees for two rumors within the first few minutes. The unit of time is in seconds on the time-line.

V. Related Work

Research on rumor veracity have utilized hand-crafted features such as posting and re-tweeting behavior, textual content and links to external sources [3], [4], [5], [6]. In the recent

SemEval 2017 Challenge [13], many have used hand crafted feature-based approaches to tackle the task of rumor detection in conjunction with stance prediction [25], [26], [27], [28].

Several works have examined using propagation patterns to detect rumors [29], [30], [31]. The cascading spread of misinformation in Facebook through photos and their captions, have been studied by analyzing comments linking to rumor debunking websites [32]. In [29], [30], a time-series model captures the periodic bursts in volume particular to false rumors whereas [31] use tree kernels to capture the propagation pattern.

The work in [33] considers the enquiring reactions of people to detect rumours. However, they use a handful of cue terms such as 'not true', 'unconfirmed' or 'debunk' to find questioning and denying tweets. [34] employ Hawkes process to use both stance and temporal information of tweets but disregard their conversation structure.

Advances in deep learning have motivated researchers to explore solutions for the rumor debunking problem using recurrent neural networks [22] and convolutional neural networks [23]. [22] use the temporal sequence of tweets as a variable length time series and represent them using stacked Gated Recurrent Units (GRU) [17]. [23] use CNN instead of GRU for the task. These deep learning based methods outperform hand-crafted feature based methods due to their ability to model higher dimensional complex interactions between the underlying features. However, none of them utilizes the conversational context of tweets to analyze their stances towards a rumor and determine its veracity.

VI. Conclusion

In this work, we have examined the problem of rumor detection from analyzing the conversations sparked around an event on social media. To this end, we have designed a neural network architecture that captures the stances of peoples' posts towards the rumor and accumulates them in order to predict its veracity. We employ convolution with attention mechanism to encode a tweet's textual content and use RNN with tweet-level attention mechanism to capture the conversation sequence. Experimental results on a real-world Twitter dataset demonstrate that our stance prediction model outperforms state-of the art models. Additionally, coupling the stance prediction model with the veracity classification model using transfer learning with full fine tuning achieves significant improvement over state-of-the-art rumor detection methods.

References

[1] M. Barthell, E. Shearer, J. Gottfried, and A. Mitchell, "The evolving role of news on twitter and facebook," http://www.journalism.org/2015/07/14/the-evolving-role-of-news-on-twitter-and-facebook/, 2015.

[2] V. Qazvinian, E. Rosengren, D. R. Radev, and Q. Mei, "Rumor has it: Identifying misinformation in microblogs," in *EMNLP*, 2011.

[3] C. Castillo, M. Mendoza, and B. Poblete, "Information credibility on twitter," in *WWW*, 2011.

[4] F. Yang, Y. Liu, X. Yu, and M. Yang, "Automatic detection of rumor on sina weibo," in *SIGKDD Workshop on Mining Data Semantics*, 2012.

[5] S. Sun, H. Liu, J. He, and X. Du, "Detecting event rumors on sina weibo automatically," in *Asia-Pacific Web Conference*. Springer, 2013.

[6] X. Liu, A. Nourbakhsh, Q. Li, R. Fang, and S. Shah, "Real-time rumor debunking on twitter," in *CIKM*, 2015.

[7] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[8] S. W.-t. Yih, X. He, and C. Meek, "Semantic parsing for single-relation question answering," in *ACL*, 2014.

[9] F. Meng, Z. Lu, M. Wang, H. Li, W. Jiang, and Q. Liu, "Encoding source language with convolutional neural network for machine translation," in *ACL-IJCNLP*, 2015.

[10] Y. Gong and Q. Zhang, "Hashtag recommendation using attention-based convolutional neural network." in *IJCAI*, 2016.

[11] L. Zhining, G. Xiaozhuo, Z. Quan, and X. Taizhong, "Combining statistics-based and cnn-based information for sentence classification," in *ICTAI*. IEEE, 2016.

[12] L. Derczynski and K. Bontcheva, "Pheme: Veracity in digital social networks." in *UMAP Workshops*, 2014.

[13] L. Derczynski, K. Bontcheva, M. Liakata, R. Procter, G. W. S. Hoi, and A. Zubiaga, "Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours," *arXiv preprint arXiv:1704.05972*, 2017.

[14] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation." in *EMNLP*, 2014.

[15] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, 2011.

[16] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *ICLR*, 2017.

[17] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *Proceedings of NIPS Deep Learning and Representation Learning Workshop*, 2014.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, 1997.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[20] Y.-C. Chen, Z.-Y. Liu, and H.-Y. Kao, "Ikm at semeval-2017 task 8: Convolutional neural networks for stance detection and rumor verification," in *Workshop on Semantic Evaluation (SemEval-2017)*, 2017.

[21] E. Kochkina, M. Liakata, and I. Augenstein, "Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm," *arXiv preprint arXiv:1704.07221*, 2017.

[22] J. Ma, W. Gao, P. Mitra, S. Kwon, B. J. Jansen, K.-F. Wong, and M. Cha, "Detecting rumors from microblogs with recurrent neural networks." in *IJCAI*, 2016.

[23] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A convolutional approach for misinformation identification," in *IJCAI*, 2017.

[24] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *NIPS*, 2014.

[25] A. Srivastava, G. Rehm, and J. M. Schneider, "Dfki-dkt at semeval-2017 task 8: rumour detection and classification using cascading heuristics," in *Workshop on Semantic Evaluation (SemEval-2017)*, 2017.

[26] V. Singh, S. Narayan, M. S. Akhtar, A. Ekbal, and P. Bhattacharyya, "Iitp at semeval-2017 task 8: A supervised approach for rumour evaluation," in *Workshop on Semantic Evaluation (SemEval-2017)*, 2017.

[27] F. Wang, M. Lan, and Y. Wu, "Ecnu at semeval-2017 task 8: Rumour evaluation using effective features and supervised ensemble models," in *Workshop on Semantic Evaluation (SemEval-2017)*, 2017.

[28] O. Enayet and S. R. El-Beltagy, "Niletmrg at semeval-2017 task 8: Determining rumour and veracity support for rumours on twitter." in *Workshop on Semantic Evaluation (SemEval-2017)*, 2017.

[29] S. Kwon, M. Cha, K. Jung, W. Chen, and Y. Wang, "Prominent features of rumor propagation in online social media," in *ICDM*, 2013.

[30] S. Kwon, M. Cha, and K. Jung, "Rumor detection over varying time windows," *PloS one*, 2017.

[31] J. Ma, W. Gao, and K.-F. Wong, "Detect rumors in microblog posts using propagation structure via kernel learning," in *ACL*, 2017.

[32] A. Friggeri, L. A. Adamic, D. Eckles, and J. Cheng, "Rumor cascades." in *ICWSM*, 2014.

[33] Z. Zhao, P. Resnick, and Q. Mei, "Enquiring minds: Early detection of rumors in social media from enquiry posts," in *WWW*, 2015.

[34] M. Lukasik, P. Srijith, D. Vu, K. Bontcheva, A. Zubiaga, and T. Cohn, "Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter," in *ACL*, 2016.