View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Infoscience - École polytechnique fédérale de Lausanne

Fighting Rumours on Social Media

Chi Thang Duong

Distributed Information Systems Laboratory, École Polytechnique Fédérale de Lausanne

***Abstract*—With the advance of social platforms, people are sharing contents in an unprecedented scale. This makes social platforms an ideal place for spreading rumors. As rumors may have negative impacts on the real world, many rumor detection techniques have been proposed. In this proposal, we summarize several works that focus on two important steps of rumor detection. The first step involves detecting controversial events from the data streams which are candidates for rumors. The aim of the second step is to find out the truth values of these events i.e. whether they are rumors or not. Although some techniques are able to achieve state-of-the-art results, they do not cope well with the streaming nature of social platforms. In addition, they usually leverage only one type of information available on social platforms such as only the posts. To overcome these limitations, we propose two research directions that emphasize on 1) detecting rumors in a progressive manner and 2) combining different types of information for better detection.**

***Index Terms*—thesis proposal, candidacy exam write-up, EDIC, EPFL**

I. Introduction

Social media networks are becoming widely popular as they are the means for internet users to share user-generated contents and interact with other people [1], [2], [3]. Due to its distributed and decentralized nature, social media provides a platform for information to propagate without any type of moderation. As a result, when an incorrect information propagates on social media networks, it may have a profound impact on real life. For instance, there was a rumor claiming two explosions happened in the White House and Barrack Obama got injured. It was posted by a hacked Twitter account associated with a major newspaper[4] and it has caused panic in the society which incurred a loss of $136.5 billion in stock market. This incident shows how a rumor can have a severe impact on our life and it highlights the need for the detection of rumor among different events being discussed on social media networks.

In an attempt to combat rumors, several rumor debunking services such as snopes.com have been created to expose rumors and misinformation. These websites harness collaborative efforts from internet users to identify potential rumors and leverage experts to verify them. As they involve manual labor, the number of rumors that can be covered are limited and it would take a long time to debunk a rumor. As a result, there is a need for automatic technique to rumor detection.

Challenges. Given the nature of social platforms, there are several challenges we need to solve to develop a rumor detection technique. First, posts on social platforms are usually short and they may lack of semantics. For instance, tweets are limited to 140 characters and may contain slang, incorrect spellings and teen language. On the other hand, news articles are long and usually well-versed. Due to these differences, traditional credibility assessment techniques that are used on web documents can not be applied to posts on social media. As a result, there is a need for rumor detection techniques that are tailored for social media. Second, as people can share contents without moderation on social platforms, rumors can spread extremely quickly. Rumor detection techniques need to detect rumors as early as possible as late detection may result in severe loss on the real world. Third, social platforms are dynamic in nature. Since contents are posted by users continuously, it is unrealistic for a rumor detection technique to collect all data before giving results. These techniques need to be able to handle the dynamic nature of social contents i.e. they must be able to work on streaming data. Last but not least, as posts on social media such as tweets are short, they may not provide enough information for rumor detection. However, in addition to posts, there are other types of information available on social platforms such as images, user information, network structure... and we need to combine them to obtain a bigger picture, thereby, better accuracy in detection. Leveraging different types of information may also allow us to detect rumors faster as for instance, we do not need to wait to collect a large amount of tweets enough for accurate rumor detection . However, these types of information have different modalities, different structure which makes it a huge challenge in combining them effectively.

In order to propose a rumor detection technique, we first need to understand the nature of rumors e.g. their underlying characteristics. In [5], the authors have collected posts and reshares of posts related to rumors. By studying these posts and their reshares (called *cascades*), the authors are able to infer several important characteristics on how a rumor propagates or the effect of rumor-debunking on rumor diffusion. This study provides insights on rumor propagation and the importance of rumor detection in preventing rumor spreading.

In a rumor detection system, there are two important steps: controversial event detection and rumor detection[6]. Controversial event detection aims to identify events on social media where users have different opinions about their truth values. These controversial events are potential candidates for rumor detection. In the rumor detection step, we focus on identifying the truth values of these events.

There are several works on controversial event detection. There is one particular study[4] that focuses on dealing with the challenge of early detection. The authors of this work aim to detect controversial events by identifying enquiry or verification posts. These posts provide a starting point to identify events of interest. These events are classified to be controversial or not using classifiers constructed on several user-defined features. This work shows that it is possible to detect controversial events early and their detection is an important intermediate step to rumor detection.

Although there are several automatic techniques to rumor detection, they rely on feature engineering which is a tedious and time-consuming process. These features are usually domain and platform-specific, which requires experts to discover new features for each domain/platform. Advances in deep learning have allowed features to be extracted automatically and the feature extraction process is integrated in the classification task. Based on these advances, the authors of [7] have designed a rumor detection technique that incorporates feature extraction and rumor detection in the same process. They are able to achieve higher accuracy in comparison with using hand-crafted features.

Proposal. Although we are able to achieve better performance with deep-learning features, these techniques do not cope well with the streaming nature of social platforms. First, they assume the availability of all data. This is not realistic in a streaming setting where data arrive sequentially. Second, they require several data preprocessing steps to happen before the rumor detection step such as the detection of events. Third, they only leverage one type of information available on social platforms. Given these shortcomings, we aim to develop techniques that are able to handle the dynamic nature of social platforms. Our techniques will combine different types of information which allows for early detection of rumors in the social platforms setting. In addition, we will leverage advances in deep learning such as word embeddings, neural networks to achieve better performance.

The rest of the writeup is organized as follows. Section II discusses the work of Friggeri et al.[5] on characteristics of rumors on a social platform. Section III and IV respectively describe the works of Zhao et. al[4] and Ma et al.[7] about techniques to detect controversial events and then, rumors. Section V details the research direction. Finally, Section VI concludes the writeup and discusses plan to realize the proposal.

II. Understanding rumor characteristics

In this section, we discuss the work of Friggeri et al.[5] which provides important insights on the nature of rumors. The authors focus on two important aspects of rumors: i) rumor propagation and ii) rumor evolution. They use facebook to study these aspects. At the time of the study, there are two types of information available on facebook which are photo and text. Correspondingly, given a post, there are two ways that it can be reshared: using the reshare button or through copying and pasting. These two resharing mechanisms are also the ways that rumors can propagate. As users reshare their friends' posts, the original post together with the tree of reshares form an *information cascade*.

A. Understanding rumor propagation

In order to study how rumors propagate, the authors focus on the first resharing mechanism: using reshare button, which is used for photo posts. Photo cascades that are related to a

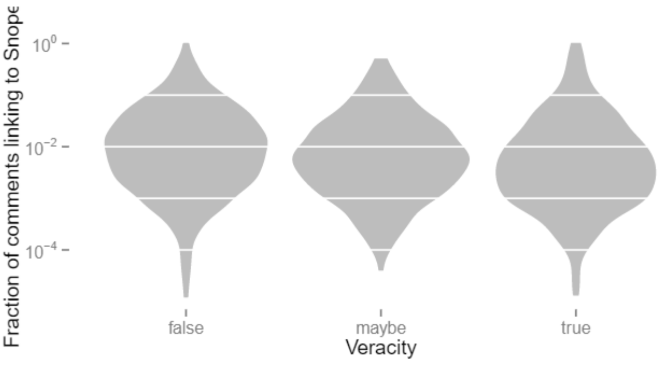


Fig. 1: False rumors are more likely to be snoped[5]

rumor are identified based on the comments on the original post and the reshares. If one or more of these comments contain a Snopes link, the cascade is considered to be about the rumor specified in the Snopes link. It is worth noting that rumors on Snopes are classified to be *true* or *false* or its veracity can not be verified (denoted as *maybe*).

Factors affecting snoping likelihood. Given a rumor cascade, there are various factors affecting its chance to be *snoped*. A cascade is snoped if there is a user commenting a Snopes link on the original post or its reshares. The authors observe that the veracity of the rumors can affect its probability of being snoped. Figure 1 shows that false rumors tend to receive more Snopes links than true or maybe rumors. In addition, false rumors are quick to be snoped as shown in Figure 2. This is expected as false rumors motivate the posters' friends to comment Snopes links. This allows them to warn their friends that the posts are inaccurate. Another factor that can affect the frequency of snoped cascades is the level of controversy of the rumors. Highly controversial stories tend to attract more viewers, hence, they may receive more comments, which increases the chance of containing a Snopes link. The ease at which the article from Snopes can be searched for given the keywords from the posts also affects the chance that a rumor cascade to be snoped. This means rumors that are easy to understand and look up tend to receive more Snopes links.

Effects of snoping on reshare deletion. We have observed that false rumors are more likely to be snoped. We expect that after being snoped, the rumor's poster may want to retract his/her post by deleting it. For instance, by removing a post about a false rumor, the poster indicates that he/she does not want to associate with the rumor. Figure 3 illustrates different probabilities of deletion across different veracity values and whether the reshare is snoped or not. We observe that all rumor reshares have higher probability of being deleted when they receive a comment with a Snopes link. This clearly shows the positive effects of snoping on resharing of rumors. There are various reasons why a snoped cascade tends to be deleted. As discussed above, for false rumors, deletion may come from the resharer's guilt of associating with a false rumor. For true or maybe true rumors, deletion may stem from the inconsistencies between the rumors and the Snopes article e.g. difference in posting age or parts of the rumor is rebutted by the Snopes article. However, the effect of snoping is more dramatic with

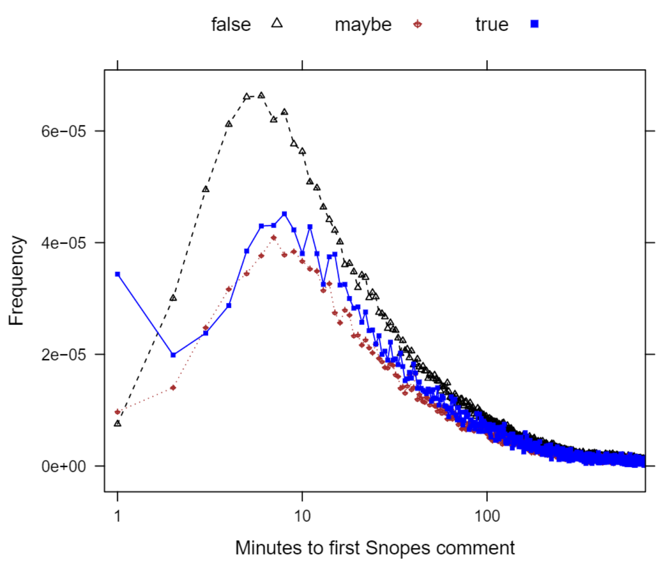


Fig. 2: False rumors are quick to be snoped after being posted[5]

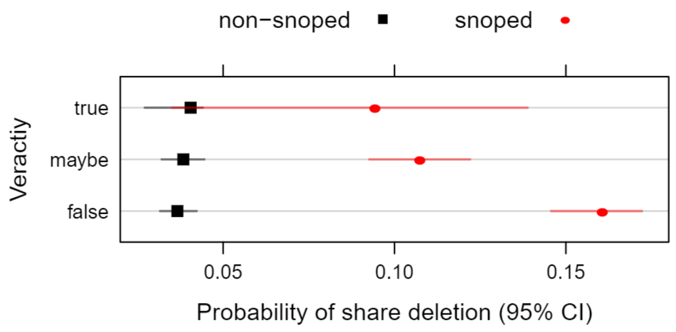


Fig. 3: Probability of deletion among different veracity and snoping values[5]

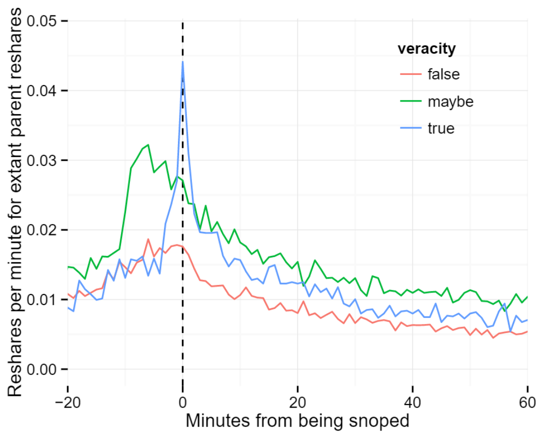


Fig. 4: Snoping decreases the likelihood of sharing in the short term[5]

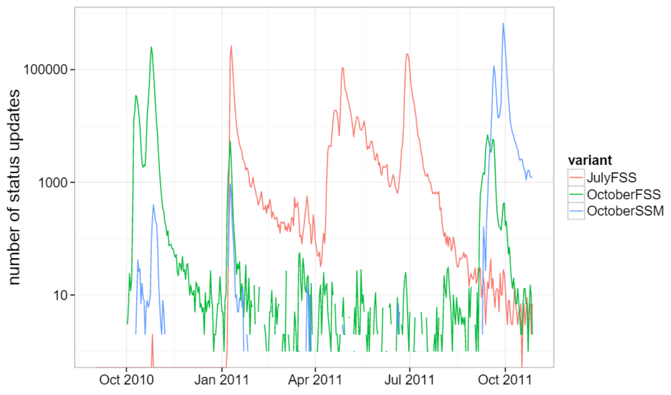


Fig. 5: Burstiness of different variants of a rumor[5]

false rumors. False rumors have the highest probability of deletion when snoped. The difference in probability between false rumors being snoped and non-snoped are 4.4 times, which is the highest among different veracity values.

Effects of snoping on rumor propagation. Snoping does not only stop the spreading of rumors (via deletion of reshares) but also slow down their propagation. We expect that friends of the resharer seeing a snoped reshare are more reluctant to forward it. This hypothesis is confirmed in Figure 4. We observe a sharp decrease in resharing across different types of rumors when a comment with link to a Snopes article is posted. Although this effect is clear with true rumors, we can also observe the effect on false and maybe true rumors. However, the effect is only short-term as the rate of resharing stay nearly the same after 20 minutes of snoping. This is understandable as not all comments of a reshare are visible to the resharer's friends. This is either due to lack of interest or newer comments obscure older ones.

B. *Understanding rumor evolution*

In the previous part, we have discussed how rumors propagate by studying photo cascades. In order to study how rumors evolve, the authors focus on the textual rumors as it is easier for users to create different versions of a textual rumor over time. Textual rumors are usually shared using the copy/paste resharing mechanism as it allows easy modification which facilitates rumor evolution.

More precisely, the authors want to track textual rumors similar to the following one. One rumor says that as the current year's July is special since it contains 5 Fridays, 5 Saturdays and 5 Sundays, the person who shares this information would receive money within 4 days. There are various variants of this rumor with changes in the month e.g. October instead of July and the weekdays e.g. Monday instead of Friday. These rumors can be identified as they usually contain phrases such as "make this your status" or "repost this". By collecting all the posts containing these phrases and clustering them, the authors are able to create clusters of all the posts related to a rumor and its variants. This allows further analysis of the rumor and its variants such as its popularity as shown in Figure 5. We observe that rumors of this type are bursty with peaks during the month mentioned in the rumors. However, during the other months, the rumors do not die out but they still remain in low frequency. These rumors are called *long-standing rumors* which are rumors persisted for a long time.

C. *Discussion*

This paper provides several positive insights regarding rumor detection. First, as cascades from false rumors are quick to be snoped, we expect rumor detection to have a profound impact as it makes the rumors less contagious. More precisely, when a rumor is snoped, it decreases the likelihood that the rumor is shared and increases the chance of being removed. As a result, given an authoritative source which provides the veracity of the rumors, social platforms users can automatically halt the spreading of false rumors. Third, among different types of rumors, false rumors which may have a negative impact on real life are more likely to be snoped. Finally, regarding long-standing rumors, as they are periodic in nature, we can expect when they burst in order to prevent their diffusion. These insights highlight several benefits of having a corpus of snoped rumors, which can be constructed through the rumor detection process.

III. Controversial event detection

The above work has shown several benefits of rumor detection. An important insight is that even after the rumor has spread, rumor detection can still have an impact as it can halt or slow down the propagation. However, it is important that some rumors are detected and prevented from spreading as early as possible e.g. the rumor about explosions in the White House[4]. One of such work that aims to detect rumors early is [4]. The premise of the work is to detect controversial events early as they are potential candidates for rumors. This step lays the foundation for the rumor detection step in which the truth values of the controversial events are identified. In [4], the rumor detection step is delegated to experts. Instead of manual labor, after controversial events are detected, we can employ automatic approach such as the one discussed in the next section. In what follows, we summarize the work of [4] on controversial event detection.

A. *Setting*

In this work, the authors consider a stream of social media posts. Each post discusses an *event* and many posts can discuss the same event. There are events that posters express conflicting opinions about their truth values and the truth values of the events are not known at the time of posting. For example, at the time when the rumor about the explosions at the White House was posted, there were several tweets asking whether the event took place while some users did not believe that the event happened. These events are called *controversial events*. Upon encountering these events, people may express skepticism by raising questions, corrections or verifications about their truth values. When the truth values of these controversial events are false, they are considered as *rumors*. The authors want to detect these controversial events as they are the input for the rumor detection step.

More precisely, we consider a stream of posts $S = \langle (p_1, t_1), (p_2, t_2), ...\rangle$ in social media as input where p_i is a post and t_i is the time of posting. An event is defined in this setting as a set of posts $E_i = \{(p_j, t_j)\}$ that discuss a story in real life. The set of all events is denoted as $\mathbb{E} = \{E_i\}$.

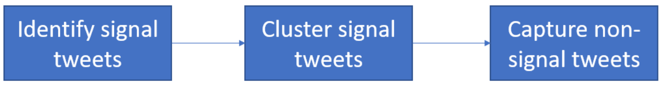


Fig. 6: Controversial event detection process

Regular Expressions
is (that|this|it) true
wh[a]*t[?!][?1]*
(real? | really? | unconfirmed)
(rumor|debunk)
(that|this|it) is not true

TABLE I: Patterns to extract signal tweets

A controversial event E_i is an event such that there is a post $p_j \in E_i$ that questions or denies the story discussed by the event E_i.

Goal of the paper. Given a stream of posts $S = \langle (p_1, t_1), (p_2, t_2), ...\rangle$, the paper aims to detect all controversial events $\mathbb{E}_c \subseteq \mathbb{E}$.

B. *Approach*

The authors propose two steps to detect controversial events. In the first step, they aim to identify potential controversial events from the data stream. Depending on their nature, these candidates may have different level of controversy. As a result, in the second step, the authors want to rank these candidates. This has many benefits such as highly controversial events can be investigated before less controversial ones.

1) Generating candidates: The authors notice that controversial events contain verification, correction or enquiry posts. Their intuition is to detect tweets of these types (called *signal tweets*) and then extend to non-signal tweets. The detection process is illustrated in Figure 6. First, the system will detect enquiry, correction and verification tweets as they contain special indicators. As these signal tweets may belong to different events, they will be clustered according to their similarity. Each cluster is considered to be a controversial event. In addition to signal tweets, controversial events also include other types of tweets such as statements. These non-signal tweets are collected by matching them with a representative statement extracted from the cluster. In the following, we discuss these steps in detail.

Identify signal tweets. As signal tweets are enquiry, correction or verification tweets, they contain special patterns that allow the system to detect them. In order to identify these patterns, the authors perform a statistical analysis on a sample set of tweets which corresponds to 5 controversial events. From the analysis, they are able to extract a set of regular expression to detect signal tweets. These regular expressions are shown in Table I.

Cluster signal tweets. The set of detected signal tweets may cover different controversial events. In order to separate these events, the authors propose to cluster the tweets as similar tweets tend to discuss the same event. This requires a measure of similarity between the tweets and a clustering technique. For the similarity, they leverage Jaccard coefficient which is defined between two tweets p_i, p_j as follows:

$$\mathbb{J}(p_i, p_j) = \frac{Ngram(p_i) \cap Ngram(p_j)}{Ngram(p_i) \cup Ngram(p_j)}$$

In other words, each tweet is represented as a set of N-grams and the similarity is measured based on these sets. One observation is that tweets related to an event are nearly duplicate as they can be retweets or tweets with minor modifications. As a result, by setting a high similarity threshold, the system only needs to handle a small subset of highly-similar tweets instead of maintaining a full similarity matrix. This observation motivates the authors to use connected component detection to leverage the sparsity of the similarity matrix. A graph of tweets is constructed where there is an edge between nearly-duplicate tweets. A connected component detection algorithm is run on this graph and each connected component is considered as a cluster.

Capture non-signal tweets. As tweets belonging to an event are similar since they use similar vocabulary, non-signal tweets are also similar to signal tweets. The only difference is that signal tweets contain enquiry terms as shown in Table I. Based on this observation, the authors propose to extract a statement that represents each cluster. This statement is used to match with non-signal tweets from the data steam. This reduces computation time a lot as it prevents redundant comparisons between all signal tweets in a cluster and non-signal tweets from the stream. For each cluster, the statement is extracted by concatenating the most frequent 3-grams that appears in 80% of signal tweets in the cluster. Non-signal tweets that have Jaccard similarity score with the statement higher than a threshold are included into the corresponding cluster.

2) Ranking candidate controversial events: After the previous step, we have obtained a set of controversial events where each event contains all of its related tweets including signal and non-signal tweets. It is worth noting that each event has different level of controversy. For instance, an event that contains a larger amount of signal tweets tends to be more controversial. A highly-controversial event has a better chance to be a rumor. As a result, there is a need to rank controversial events by their level of controversy. In addition, as the authors want to leverage experts in assessing the veracity of the events, the ranking also helps the experts in selecting ones that worth investigation. There are several features that are helpful in ranking controversial events. These features are shown in Table II. From these features, the authors train a decision trees classifier to give a score to each controversial event. The labels for these events used for training are obtained from human labellers.

C. Validation

In order to verify the performance of their techniques, the authors construct two controversial event datasets. The first dataset contains the Tweets related to the Boston marathon bombing in a specific period, the second dataset contains 10% of the Tweets in the month 11/2013. The authors compare their techniques with several baselines. These baselines are different clustering techniques where they cluster the tweets using different criteria and consider each cluster as a controversial event. As the number of tweets is extremely large, the authors measure the performance based on precision@N. This means they only need to label a small number of candidate controversial events returned by the techniques instead of identifying all the controversial events in the dataset. In both datasets, the proposed technique is able to outperform the baselines. It returns the highest number of real controversial events with a precision@10 of 0.521 for the Boston dataset.

D. Discussion

This paper presents a framework to detect controversial events from streams of social platforms. It allows to detect controversial events early when they are posted on social platforms. This work lays a foundation for the detection of rumors as these controversial events are potential candidates for rumors. The paper makes several optimizations such as using connected component clustering instead of traditional clustering techniques or reusing 3-grams representation in different steps of the process. These optimizations has helped in handling the online nature of social platforms. However, the paper has some shortcomings that we discuss below.

First, the paper only tackles the controversial event detection problem. It needs to leverage manual experts to detect rumors from these events. This slows down the whole rumor detection process significantly. Second, the paper uses feature engineering to design several hand-crafted features. Feature engineering is tedious and not robust to domain and platform changes. Given a new platform or a new domain, experts need to be used to design new features. Third, the paper limits to using only tweets to detect controversial events. There are several other information such as user and network statistics that may help in the detection process.

IV. Rumor detection

Although the previous work[4] allows to extract controversial events early from streaming data of social platforms, they do not consider the veracity of these events. The task of assessing the truth values of these events are assigned to human experts, which would defeat the purpose of detecting controversial events early as human verification may take a long time. As a result, there is a need to assess the veracity of controversial events (i.e. rumor detection) automatically. One approach which is able to achieve state-of-the-art results in rumor detection is [7]. Their technique leverages several advances in deep learning such as recurrent neural network. In the following, we summarize the work of Ma et. al.[7] in detail.

A. Setting and Approach

In this work, the authors consider a setting in which they are given a set of controversial events $\mathbb{E} = \{E_i\}$ which are the output of the controversial event detection step. Similarly, each event is a set of all related posts $E_i = \{(p_j, t_j)\}$. The aim of this work is to classify each event whether it is a rumor or not. In particular, given an event $E_i = \{(p_j, t_j)\}$, the rumor detection problem is to assign a label $y_i \in \{0, 1\}$ for the event E_i where 1 denotes rumor and 0 otherwise.

In the following, we denote matrices as bold upper case letters ($\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$), and vectors as bold lower-case letters ($\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$). $\mathbf{A}_i$ represents the i^{th} row of matrix $\mathbf{A}$ and $[\mathbf{a}]_i$ denotes the i^{th} element of vector $\mathbf{a}$. Unless otherwise stated, vectors are assumed to be column vectors. We also denote $|\mathbf{a}|$ to be the dimensionality of the vector $\mathbf{a}$.

B. Approach

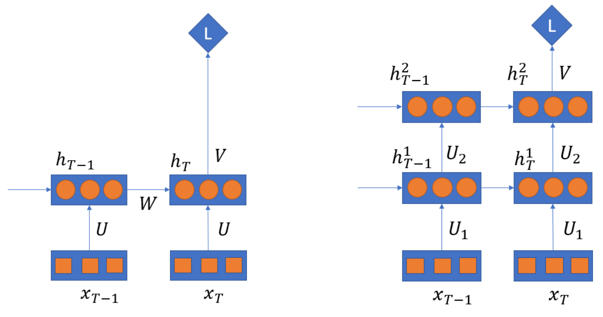
In order to solve this problem, the authors make an observation that when some posts discuss an event, there would be several posts discussing the same event subsequently (similar to cascades discussing in Section II). These subsequent posts may just be reiteration of the original posts or they could add new information which sheds light on the veracity of the event. There is a temporal dependency between the posts in an event and the temporal dependency may indicate whether the event is a rumor or not. Based on this observation, the authors aim to capture this temporal dependency explicitly using recurrent neural network (RNN), which allows them to achieve better detection accuracy.

1) Recurrent Neural Network: Among different types of neural networks, RNN is the one that can model the sequential characteristics of the input data such as time series or sentences. Given an input sequence $(\mathbf{x}_1, ..., \mathbf{x}_T)$, RNN processes each input sequentially (from $\mathbf{x}_1$ to $\mathbf{x}_T$), at each step, it updates its hidden state $\mathbf{h}_i$ and returns an output $\mathbf{o}_i$. The hidden state vector $\mathbf{h}_i$ captures information of the elements that the RNN has seen. More precisely, at the time step i, the network does the following update operations[8]:

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

where the matrices $\mathbf{U}, \mathbf{V}, \mathbf{W}$ are used, respectively, to convert input vector to hidden vector, hidden vector to output vector and hidden vector to hidden vector. Two vectors $\mathbf{b}, \mathbf{c}$ are the bias vectors and the function $tanh$ is a nonlinearity function. The matrices and the bias vectors are the trainable parameters of the RNN. In order to find these parameters, we compute the gradients of the network using back-propagation through time[9].

Variants. In addition to the vanilla tanh-RNN, there are several variants of RNN. The RNN as discussed above suffers from the the vanishing or exploding gradients problem which makes it unable to learn from long sequences[10]. A solution to this problem is to implement memory cells in the network to



(a) tanh-RNN **(b) 2-layer RNN**

Fig. 7: Different variants of RNN

store information over time, which is the idea of Long Short-Term Memory (LSTM)[11]. Another improvement is to stack several layers of RNNs. This increases the capacity of the network, which may improve the classification accuracy. The architectures of different RNNs are illustrated in Figure 7.

2) Neural Network Model: In the following, we discuss how the authors of [7] leverage RNN to detect rumors. They achieve this by training a feedforward neural network which takes the posts of an event as input and returns the label for the event. More precisely, given an event E and two classes $Y = \{0, 1\}$, they define a neural network that assigns probabilities to all labels $y \in Y$. The predicted class is then the one with the highest probability:

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

The feedforward neural network estimates $P(Y = y|E)$ with a parametric function ϕ_θ. More precisely, given an event E, the function ϕ_θ applies a combination of functions such as

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

where θ refers to all learnable parameters of the network and L the total number of layers.

RNN layer. In our setting, we consider an RNN as the first layer in our network. It is modelled by the function $\phi^1_{\theta_1}(E)$ which takes as input the event E which contains $|E|$ posts and returns the output in the last time step $\mathbf{o}_{|E|}$. In particular, $\mathbf{o}_{|E|} = \phi^1_{\theta_1}(E)$ and θ_1 is the parameter of the RNN that we need to find.

It is worth noting that the input to the first layer of the network (the RNN layer) must be a sequence of vectors $(\mathbf{x}_1, ..., \mathbf{x}_T)$ (as shown in Figure 7). This means there is a need to convert the texts from the posts of an event $E = \{(p_i, t_i)\}$ to a sequence of vectors. The authors achieve this by constructing

Features	Description
Percentage of signal tweets	Ratio of signal and non-signal tweets in a cluster
Entropy ratio	Ratio of entropy of vocabulary distributions of signal and non-signal tweets
Tweet lengths	Avg. # of words of signal tweets; avg. # of tweets in a cluster; their ratio
Retweets	% of retweets among signal tweets in a cluster, % of retweets in a cluster
URLs	Avg. # of URLs in signal tweets; avg. # of URLs in a cluster
Hashtags	Avg. # of hashtags in signal tweets; avg. # of hashtags in a cluster
Mentions	Avg. # of mentions in signal tweets; avg. # of mentions in a cluster

TABLE II: Features to rank controversial events

a vocabulary containing the top-K terms with the highest tf-idf values. Each post is then modelled as a tf-idf vector of K elements.

Softmax layer. Given an input E, the function $\phi^1_{\theta_1}(E)$ outputs a vector score $\mathbf{o} \in \mathbb{R}^2$ which contains the score for each class $y \in Y$. In order to convert the score to probabilities, a softmax activation function is applied on the vector $\mathbf{o}$. The probability distribution over the labels is computed as follows:

$$P(Y = y|E) = \frac{\exp([\mathbf{o}]_{y+1})}{\sum_{k=1}^{2} \exp([\mathbf{o}]_{y_k+1})} \tag{3}$$

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

$$L(\theta) = \sum_{(E,y)} -\log P(Y = y|E) \tag{4}$$

C. Validation

In order to verify the effectiveness of their approach, the authors construct two rumor datasets from Twitter and Weibo. For Twitter, they pick 498 rumors from Snopes and extract the keywords from the Snopes article about the rumors. The keywords are used to search for tweets related to the rumors on Twitter. For Weibo, they collect a set of known rumors and their related posts using the Sina community management center. To balance both datasets, they also add some non-rumor events. Each rumor and non-rumor event is just a collection of posts. The authors compare their approach with several baselines which use hand-crafted features and traditional classifiers such as decision tree to detect rumors. Their approach is able to outperform these baselines significantly, achieving state-of-the-art result with an accuracy of 0.881 for the Twitter dataset and 0.91 for the Weibo dataset.

D. Discussion

This paper can be considered as a continuation from the work discussed in Section III as the input of this work is the output of the controversial event detection step from [4]. As a result, one may leverage the work in [4] to detect all controversial events and then, use this work to assess their veracity. This work is the first to leverage deep learning technique in rumor detection and it is able to achieve state-of-the-art results. However, there are some weak points that we analyze below.

First, this work assumes that all the posts that are related to an event are available beforehand. This requirement may not be practical in a streaming setting where data arrive in a consecutive manner. There is a need for a progressive algorithm that can handle the streaming nature of social platforms. Second, although this work does not require the manual task of feature engineering, it restraints itself by using td-idf vectors. There are several approaches in deep learning that allow to convert a piece of text to a vector such as paragraph vector or word embeddings. Third, similar to the work in [4], this paper only leverages the tweets for rumor detection. There are other types of information such as user or network that can help in increasing the accuracy of detection.

V. Research proposal

We propose two research directions towards detecting rumors from streams of social platforms including combination of information for better detection and progressive techniques to handle the online nature of streaming data.

A. Information Combination for Rumor Detection

Existing works on rumor detection tend to use only one type of information available on social platforms which are the posts (e.g. tweets, statuses). This is the most straightforward approach as the posts are easily-accessible. However, there are other types of information such as user statistics, images, information propagation that would help in detecting rumors. In the following, we discuss several types of information on social platforms and how they can be used and combined to detect rumors [12], [13], [14], [15].

Textual information. Posts on social media such as tweets, statuses belong to this type of information. In addition to posts, when an event is discussed on social media, it usually has an article or several articles backing it up. These articles can be referenced in the posts and they also belong to the textual information type. Articles and posts provide several linguistic and syntactical features that can be used to detect rumors. For instance, the writing style such as word choice, sentence structure, paragraph structure can indicate whether the posts or articles are about a rumor or not. A rumor tweet may contain keywords that can facilitate its diffusion while the writing style of a rumor article may try to stimulate strong emotions from readers. As posts on social media are abundant and easy to collect, there are a lot of works[7], [4] on rumor detection using only the posts.

Provenance. Provenance is the information related to the authors of the posts or the articles. The provenance information could be several statistics about the author such as the number of friends, number of posts, author description, network of friends ... These information has been used for credibility assessment of documents on the Web[16] and they can also be applied in rumor detection. For instance, an author that has many friends who post rumor tweets tend to follow suit. As a result, posts from this author may be rumor-related. Provenance and textual information are two typical types of information that have been used extensively in traditional credibility assessment techniques for Web documents [17], [18], [19], [20], [21].

Inconsistencies. As rumors are events where truth values are false, they tend to be made up. As a result, there are inconsistencies in the information related to the events. These inconsistencies are clear indicators of rumors. For instance, the image and the text from a rumor article may not be about the same event. As the rumor is fabricated, the authors of the article may reuse the image from another event in the past.

TABLE III: Performance of combining articles and tweets

Method	Accuracy	Precision	Recall	F-measure
Tweet-only[7]	0.78	**0.83**	0.70	0.76
Article+Tweet	**0.85**	0.81	**0.92**	**0.86**

There are other types of inconsistencies such as time, location, user statistics... For example, posts from a normal user usually have few comments as comments are usually from friends. Posts with many comments that are unrelated to the user are more likely to be about controversies, which could be a rumor.

Information propagation. There are several information regarding how an event propagates that can be used to detect rumors. First, rumors are usually contagious. This means they tend to diffuse extremely fast and they are more likely to be shared by a large amount of users. Second, there are differences in how a rumor propagates. For instance, an original rumor tweet may come from a normal unpopular user while retweets from this original rumor tweet have high popularity. On the other hand, a normal tweet may come from a popular user and retweets from this tweet are from normal users[22]. Detecting these phenomena on how a piece of information propagates can help us to identify rumors.

In this write-up, we have discussed two papers[7], [4] that leverage posts on social media to detect rumors. In order to verify our hypothesis that the combination of different types of information can help to detect rumors, we have conducted a preliminary study. In this study, we combine the information from the tweets and the articles referenced by the tweets. The results in Table III have confirmed our hypothesis. In order to combine tweets and articles, we have to deal with many challenges. First, as each post may also contain information from the article, we need a way to combine information from the tweets and the articles in a coherent manner. Second, there are cases that the article is not available in the tweet. In these cases, we need to handle the missing of the article information while making sure that the classification accuracy does not deteriorate [23], [24], [25], [26], [27], [28].

B. Progressive Rumor Detection

Traditional works on rumor detection tend to follow a post-processing approach, where the detection is done statically after all the data is acquired. For social platforms, this requirement is unrealistic as waiting for all the data to be available, the rumors may already have done their damage. In addition, even if we can wait to collect all data, the amount of data generated from social platforms may exceed any reasonable limit for permanent storage. To handle this problem, we go beyond state-of-the-art approaches by devising an online rumor detection technique that can handle incoming data in a progressive manner.

First, it is worth noting that by combining different types of information, we are able to detect rumors faster. For instance, if we use only the tweets, we need to wait to collect a large amount of tweets that is enough to achieve a reasonable accuracy in detection. However, by combining different types of information, the detection can be done earlier as we can obtain different aspects of the event. For example, given some suspect tweets, we can collect information from the articles referenced in the tweets, their authors and the authors' friends.

Second, we will leverage advances in deep learning to improve the detection accuracy while decreasing detection delay. Better information representation may allow us to detect rumors with less amount of data. For instance, the work in [7] converts the tweets into vector using tf-idf. However, this type of conversion lose a lot of semantics as it does not consider the relation between the words. By using word embeddings or paragraph vectors, we can capture the semantics of the tweets, thereby, decreasing the amount of tweets we need to collect.

Third, we will analyze the data to find out which information on social media is important for rumor detection. This allows us to filter out irrelevant information, thereby speeding up the detection process. The analysis also helps us in designing rules that can be used to detect rumors. Rule-based detection is suitable for streaming setting as it is able to process data extremely fast.

VI. Conclusions

This write-up discusses several techniques in rumor detection. It first demonstrates several characteristics of rumors that provides insights for the rumor detection task. It then explains two important steps in rumor detection which are the controversial event detection and rumor detection step. These two steps are demonstrated by two state-of-the-art techniques. Although these techniques have achieve high-quality results, they still have several weak points regarding handling the online nature of data streams and the usage of information. In this proposal, we go beyond state-of-the-art by proposing a progressive rumor detection technique that also leverages different types of information. We will realize our approach in the subsequent works.

References

[1] H. Yin, L. Chen, W. Wang, X. Du, N. Q. V. Hung, and X. Zhou, "Mobi-sage: A sparse additive generative model for mobile app recommendation," in *ICDE*, 2017, pp. 75–78.

[2] N. Q. V. Hung, C. T. Duong, N. T. Tam, M. Weidlich, K. Aberer, H. Yin, and X. Zhou, "Argument discovery via crowdsourcing," *VLDB J.*, pp. 511–535, 2017.

[3] N. T. Tam, M. Weidlich, D. C. Thang, H. Yin, and N. Q. V. Hung, "Retaining data from streams of social platforms with minimal regret," in *IJCAI*, 2017, pp. 2850–2856.

[4] Z. Zhao, P. Resnick, and Q. Mei, "Enquiring minds: Early detection of rumors in social media from enquiry posts," in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1395–1405.

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

[6] A. Zubiaga, A. Aker, K. Bontcheva, M. Liakata, and R. Procter, "Detection and resolution of rumours in social media: A survey," *arXiv preprint arXiv:1704.00656*, 2017.

[7] 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 *Proceedings of IJCAI*, 2016.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[10] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[11] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[12] N. Q. V. Hung, D. C. Thang, M. Weidlich, and K. Aberer, "Minimizing efforts in validating crowd answers," in *SIGMOD*, 2015, pp. 999–1014.

[13] N. T. Tam, D. C. Thang, N. Q. V. Hung, and K. Aberer, "An evaluation of diversification techniques," in *DASFAA*, 2015, pp. 215–231.

[14] N. Q. V. Hung, N. T. Tam, C. V. Tuan, T. K. Wijaya, Z. Miklos, K. Aberer, A. Gal, and M. Weidlich, "Smart: A tool for analyzing and reconciling schema matching networks," in *ICDE*, 2015, pp. 1488–1491.

[15] N. Q. V. Hung, D. C. Thang, M. Weidlich, and K. Aberer, "Erica: Expert guidance in validating crowd answers," in *SIGIR*, 2015, pp. 1037–1038.

[16] C. Castillo, M. Mendoza, and B. Poblete, "Information credibility on twitter," in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 675–684.

[17] T. T. Nguyen, Q. V. H. Nguyen, M. Weidlich, and K. Aberer, "Result selection and summarization for web table search," in *ICDE*, 2015, pp. 231–242.

[18] N. Q. V. Hung, S. Sathe, D. C. Thang, and K. Aberer, "Towards enabling probabilistic databases for participatory sensing," in *CollaborateCom*, 2014, pp. 114–123.

[19] Q. V. H. Nguyen, S. T. Do, T. Nguyen Thanh, and K. Aberer, "Privacy-preserving schema reuse," in *DASFAA*, 2014, pp. 234–250.

[20] Q. V. H. Nguyen, T. T. Nguyen, N. T. Lam, and K. Aberer, "Batc: a benchmark for aggregation techniques in crowdsourcing," in *SIGIR*, 2013, pp. 1079–1080.

[21] Q. V. H. Nguyen, T. Nguyen Thanh, T. Lam Ngoc, and K. Aberer, "An evaluation of aggregation techniques in crowdsourcing," in *WISE*, 2013, pp. 1–15.

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

[23] N. Q. V. Hung, H. Jeung, and K. Aberer, "An evaluation of model-based approaches to sensor data compression," *TKDE*, pp. 2434–2447, 2013.

[24] Q. V. H. Nguyen, X. Luong, Z. Miklos, T. Quan, and K. Aberer, "Collaborative schema matching reconciliation," in *CoopIS*, 2013, pp. 222–240.

[25] A. Gal, M. Katz, T. Sagi, M. Weidlich, K. Aberer, H. Q. V. Nguyen, Z. Miklós, E. Levy, and V. Shafran, "Completeness and ambiguity of schema cover," in *CoopIS*, 2013, pp. 241–258.

[26] H. Q. V. Nguyen, T. K. Wijaya, Z. Miklós, K. Aberer, E. Levy, V. Shafran, A. Gal, and M. Weidlich, "Minimizing human effort in reconciling match networks," in *ER*, 2013, pp. 212–226.

[27] A. Gal, T. Sagi, M. Weidlich, E. Levy, V. Shafran, Z. Miklós, and N. Q. V. Hung, "Making sense of top-k matchings: A unified match graph for schema matching," 2012, p. 6.

[28] D. T. Anh, V. H. Tam, and N. Q. V. Hung, "Generating complete university course timetables by using local search methods." in *RIVF*, 2006, pp. 67–74.