On Early-stage Debunking Rumors on Twitter: Leveraging the Wisdom of Weak Learners

Tu Ngoc Nguyen[1], Cheng Li[2], and Claudia Niederée[1]

[1] L3S Research Center / Leibniz Universität Hannover
{tunguyen,niederee}@L3S.de
[2] SAP S/4 Hana Cloud Foundation
cheng.li@SAP.com

Abstract. Recently a lot of progress has been made in rumor modeling and rumor detection for micro-blogging streams. However, existing automated methods do not perform very well for early rumor detection, which is crucial in many settings, e.g., in crisis situations. One reason for this is that aggregated rumor features such as propagation features, which work well on the long run, are - due to their accumulating characteristic - not very helpful in the early phase of a rumor. In this work, we present an approach for early rumor detection, which leverages Convolutional Neural Networks for learning the hidden representations of individual rumor-related tweets to gain insights on the credibility of each tweets. We then aggregate the predictions from the very beginning of a rumor to obtain the overall event credits (so-called *wisdom*), and finally combine it with a time series based rumor classification model. Our extensive experiments show a clearly improved classification performance within the critical very first hours of a rumor. For a better understanding, we also conduct an extensive feature evaluation that emphasized on the early stage and shows that the low-level credibility has best predictability at all phases of the rumor lifetime.

1 Introduction

Widely spreading rumors can be harmful to the government, markets and society and reduce the usefulness of social media channel such as Twitter by affecting the reliability of their content. Therefore, effective method for detecting rumors on Twitter are crucial and rumors should be detected as early as possible before they widely spread. As an example, let us recall of the shooting incident that happened in the vicinity of the Olympia shopping mall, Munich; in a summer day, 2016. Due to the unclear situation at early time, numerous rumors about the event did appear and they started to circulate very fast over social media. The city police had to warn the population to refrain from spreading related news on Twitter as it was getting out of control: *"Rumors are wildfires that are difficult to put out and traditional news sources or official channels, such as police departments, subsequently struggle to communicate verified information to the public, as it gets lost under the flurry of false information."* [3] Figure 1 shows the rumor *sub-events* in the early stage of the event Munich shooting. The first *terror-indicating* "news" –The gunman shouted 'Allahu Akbar'– was widely disseminated on

---

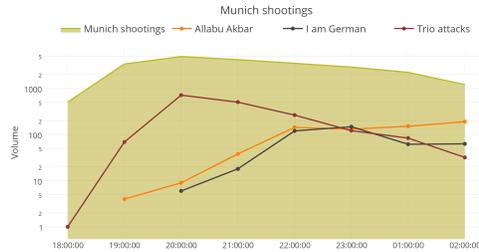[3] Deutsche Welle: http://bit.ly/2qZuxCN

Fig. 1: The *Munich shooting* and its sub-events burst after the first 8 hours, y-axis is English tweet volume.

Twitter right after the incident by an unverified account. Later the claim of three gunmen also spread quickly and caused public tension. In the end, all three information items were falsified.

We follow the rumor definition [24] considering a rumor (or fake news) as a statement whose truth value is unverified or deliberately false. A wide variety of features has been used in existing work in rumor detection such as [6,11,13,18,19,20,23,30,31]. Network-oriented and other aggregating features such as propagation pattern have proven to be effective for this task. Unfortunately, the inherently accumulating characteristic of such features, which require some time (and Twitter traffic) to mature, does not make them very apt for early rumor detection. A first semi-automatic approach focussing on early rumor detection presented by Zhao et al. [32], thus, exploits rumor signals such as enquiries that might already arise at an early stage. Our fully automatic, cascading rumor detection method follows the idea on focusing on early rumor signals on text contents; which is the most reliable source before the rumors widely spread. Specifically, we learn a more complex representation of single tweets using Convolutional Neural Networks, that could capture more hidden meaningful signal than only enquiries to debunk rumors. [7,19] also use RNN for rumor debunking. However, in their work, RNN is used at *event-level*. The classification leverages only the deep data representations of aggregated tweet contents of the whole event, while ignoring exploiting other –in latter stage–effective features such as user-based features and propagation features. Although, tweet contents are merely the only reliable source of clue at early stage, they are also likely to have doubtful perspectives and different stands in this specific moment. In addition, they could relate to rumorous sub-events (see e.g., the Munich shooting). Aggregating all relevant tweets of the event at this point can be of noisy and harm the classification performance. One could think of a sub-event detection mechanism as a solution, however, detecting sub-events at real-time over Twitter stream is a challenging task [22], which increases latency and complexity. In this work, we address this issue by deep neural modeling only at single tweet level. Our intuition is to leverage the "wisdom of the crowd" theory; such that even a certain portion of tweets at a moment (mostly early stage) are weakly predicted (because of these noisy factors), the ensemble of them would attribute to a stronger prediction.

In this paper, we make the following contributions with respect to rumor detection:

– We develop a machine learning approach for modeling tweet-level credibility. Our CNN-based model reaches 81% accuracy for this novel task, that is even hard for human judgment. The results are used to debunk rumors in an ensemble fashion.

– Based on the credibility model we develop a novel and effective cascaded model for rumor classification. The model uses time-series structure of features to capture their temporal dynamics. Our model clearly outperforms strong baselines, especially for the targeted early stage of the diffusion. It already reaches over 80% accuracy in the first hour going up to over 90% accuracy over time.

2 Related Work

A variety of issues have been investigated using data, structural information, and the dynamics of the microblogging platform Twitter including event detection [16], spam detection [1,29], or sentiment detection [4]. Work on rumor detection in Twitter is less deeply researched so far, although rumors and their spreading have already been investigated for a long time in psychology [2,5,26]. Castillo et al. researched the information credibility on Twitter[6,11]. The work, however, is based solely on people's attitude (trustful or not) to a tweet not the credibility of the tweet itself. In other words, a false rumor tweet can be trusted by a reader, but it might anyway contain false information. The work still provides a good start of researching rumor detection.

Due to the importance of information propagation for rumors and their detection, there are also different simulation studies [25,27] about rumor propagations on Twitter. Those works provide relevant insights, but such simulations cannot fully reflect the complexity of real networks. Furthermore, there are recent work on propagation modeling based on epidemiological methods [3,13,17], yet over a long studied time, hence how the propagation patterns perform at early stage is unclear. Recently, [30] use unique features of Sina Weibo to study the propagation patterns and achieve good results. Unfortunately Twitter does not give such details of the propagation process as Weibo, so these work cannot be fully applied to Twitter.

Most relevant for our work is the work presented in [20], where a time series model to capture the time-based variation of social-content features is used. We build upon the idea of their *Series-Time Structure*, when building our approach for early rumor detection with our extended dataset, and we provide a deep analysis on the wide range of features change during diffusion time. Ma et al. [19] used Recurrent Neural Networks for rumor detection, they batch tweets into time intervals and model the time series as a RNN sequence. Without any other handcrafted features, they got almost 90% accuracy for events reported in Snope.com. As the same disadvantage of all other deep learning models, the process of learning is a black box, so we cannot envisage the cause of the good performance based only on content features. The model performance is also dependent on the tweet retrieval mechanism, of which quality is uncertain for stream-based trending sub-events.

3 Single Tweet Credibility Model

Before presenting our Single Tweet Credibility Model, we will start with an overview of our overall rumor detection method. The processing pipeline of our classification approach is shown in Figure 2. In the first step, relevant tweets for an event
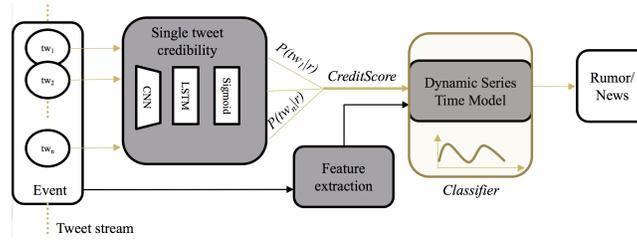
Fig. 2: Pipeline of our rumor detection approach.

are gathered. Subsequently, in the upper part of the pipeline, we predict tweet credibilty with our pre-trained credibility model and aggregate the prediction probabilities on single tweets (CreditScore). In the lower part of the pipeline, we extract features from tweets and combine them with the creditscore to construct the feature vector in a time series structure called Dynamic Series Time Model. These feature vectors are used to train the classifier for rumor vs. (non-rumor) news classification.

Early in an event, the related tweet volume is scanty and there are no clear propagation pattern yet. For the credibility model we, therefore, leverage the signals derived from tweet contents. Related work often uses aggregated content [18,20,32], since individual tweets are often too short and contain slender context to draw a conclusion. However, content aggregation is problematic for hierarchical events and especially at early stage, in which tweets are likely to convey doubtful and contradictory perspectives. Thus, a mechanism for carefully considering the 'vote' for individual tweets is required. In this work, we overcome the restrictions (e.g., semantic sparsity) of traditional text representation methods (e.g., bag of words) in handling short text by learning low-dimensional tweet embeddings. In this way, we achieve a rich hidden semantic representation for a more effective classification.

3.1 Exploiting Convolutional and Recurrent Neural Networks

Given a tweet, our task is to classify whether it is associated with either a news or rumor. Most of the previous work [6,11] on tweet level only aims to measure the *trustfulness* based on human judgment (note that even if a tweet is trusted, it could anyway relate to a rumor). Our task is, to a point, a reverse engineering task; to measure the probability a tweet refers to a *news* or *rumor* event; which is even trickier. We hence, consider this a weak learning process. Inspired by [33], we combine CNN and RNN into a unified model for tweet representation and classification. The model utilizes CNN to extract a sequence of higher-level phrase representations, which are fed into a long short-term memory (LSTM) RNN to obtain the tweet representation. This model, called CNN+RNN henceforth, is able to capture both local features of phrases (by CNN) as well as global and temporal tweet semantics (by LSTM)(see Figure 3).

Representing Tweets: Generic-purpose tweet embedding in [9,28] use character-level RNN to represent tweets that in general, are noisy and of idiosyncratic nature. We discern that tweets for rumors detection are often triggered from professional sources. Hence, they are linguistically clean, making word-level embedding become useful. In this work, we do not use the pre-trained embedding (i.e., *word2vec*), but instead learn
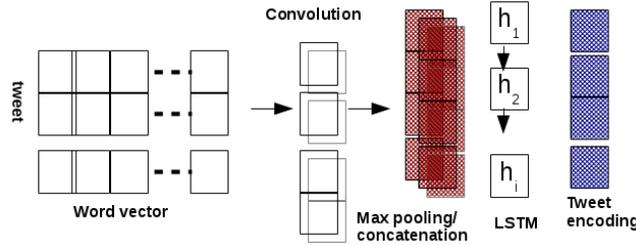
Fig. 3: CNN+LSTM for tweet representation.

the word vectors from scratch from our (large) rumor/news-based tweet collection. The effectiveness of fine-tuning by learning task-specific word vectors is backed by [15]. We represent tweets as follows: Let $x_i \in \mathscr{R}$ be the k-dimensional word vector corresponding to the i-th word in the tweet. A tweet of length n (padded where necessary) is represented as: $x_{1:n} = x_1 \oplus x_2 \oplus \cdots \oplus x_n$, where $\oplus$ is the concatenation operator. In general, let $x_{i:i+j}$ refer to the concatenation of words $x_i, x_{i+1}, ..., x_{i+j}$. A convolution operation involves a filter $w \in \mathscr{R}^{hk}$, which is applied to a window of h words to produce a feature. For example, a feature c_i is generated from a window of words $x_{i:i+h-1}$ by: $c_i = f(w \cdot x_{i:i+h-1} + b)$.

Here $b \in \mathscr{R}$ is a bias term and f is a non-linear function such as the hyperbolic tangent. This filter is applied to each possible window of words in the tweet $\{x_{1:h}, x_{2:h+1}, ..., x_{n-h+1:n}\}$ to produce a feature map: $c = [c_1, c_2, ..., c_{n-h+1}]$ with $c \in \mathscr{R}^{n-h+1}$. A max-over-time pooling or dynamic k-max pooling is often applied to feature maps after the convolution to select the most or the k-most important features. We also apply the 1D max pooling operation over the time-step dimension to obtain a fixed-length output.

Using Long Short-Term Memory RNNs: RNN are able to propagate historical information via a chain-like neural network architecture. While processing sequential data, it looks at the current input x_t as well as the previous output of hidden state h_{t-1} at each time step. The simple RNN hence has the ability to capture context information. However, the length of reachable context is often limited. The gradient tends to vanish or blow up during the back propagation. To address this issue, LSTM was introduced in [12]. The LSTM architecture has a range of repeated modules for each time step as in a standard RNN. At each time step, the output of the module is controlled by a set of gates in $\mathscr{R}^d$ as a function of the old hidden state h_{t-1} and the input at the current time step x_t: forget gate f_t, input gate i_t, and output gate o_t.

3.2 CNN+LSTM for tweet-level classification.

We regard the output of the hidden state at the last step of LSTM as the final tweet representation and we add a softmax layer on top. We train the entire model by minimizing the cross-entropy error. Given a training tweet sample $x^{(i)}$, its true label $y_j^{(i)} \in \{y_{rumor}, y_{news}\}$ and the estimated probabilities $\tilde{y}_j^{(i)} \in [0..1]$ for each label $j \in \{rumor, news\}$, the error is defined as:

$$\mathsf{L}(x^{(i)}, y^{(i)}) = 1\{y^{(i)} = y_{rumor}\}log(\tilde{y}_{rumor}^{(i)}) + 1\{y^{(i)} = y_{news}\}log(\tilde{y}_{news}^{(i)}) \qquad (1)$$

where 1 is a function converts boolean values to $\{0,1\}$. We employ stochastic gradient descent (SGD) to learn the model parameters.

4 Time Series Rumor Detection Model

As observed in [19,20], rumor features are very prone to change during an event's development. In order to capture these temporal variabilities, we build upon the Dynamic Series-Time Structure (DSTS) model (time series for short) for feature vector representation proposed in [20]. We base our credibility feature on the time series approach and train the classifier with features from diffent high-level contexts (i.e., users, Twitter and propagation) in a cascaded manner. In this section, we first detail the employed Dynamic Series-Time Structure, then describe the high and low-level ensemble features used for learning in this pipeline step.

4.1 Dynamic Series-Time Structure (DSTS) Model

For an event E_i we define a time frame given by $timeFirst_i$ as the start time of the event and $timeLast_i$ as the time of the last tweet of the event in the observation time. We split this event time frame into N intervals and associate each tweet to one of the intervals according to its creation time. Thus, we can generate a vector $V(E_i)$ of features for each time interval. In order to capture the changes of feature over time, we model their differences between two time intervals. So the model of DSTS is represented as: $V(E_i) = (\mathbf{F}_{i,0}^D, \mathbf{F}_{i,1}^D, ..., \mathbf{F}_{i,N}^D, \mathbf{S}_{i,1}^D, ..., \mathbf{S}_{i,N}^D)$, where $\mathbf{F}_{i,t}^D$ is the feature vector in time interval t of event E_i. $\mathbf{S}_{i,t}^D$ is the difference between $\mathbf{F}_{i,t}^D$ and $\mathbf{F}_{i,t+1}^D$. $V(E_i$) is the time series feature vector of the event E_i. $\mathbf{F}_{i,t}^D = (\tilde{f}_{i,t,1}, \tilde{f}_{i,t,2}, ..., \tilde{f}_{i,t,D})$. And $\mathbf{S}_{i,t}^D = \frac{\mathbf{F}_{i,t+1}^D - \mathbf{F}_{i,t}^D}{Interval(E_i)}$. We use Z-score to normalize feature values; $\tilde{f}_{i,t,k} = \frac{f_{i,t+1,k} - \overline{f}_{i,k}}{\sigma(f_{i,k})}$ where $f_{i,t,k}$ is the k-th feature of the event E_i in time interval t. The mean of the feature k of the event E_i is denoted as $\overline{f}_{i,k}$ and $\sigma(f_{i,k})$ is the standard deviation of the feature k over all time intervals. We can skip this step, when we use Random Forest or Decision Trees, because they do not require feature normalization.

4.2 Features for the Rumor Detection Model

In selecting features for the rumor detection model, we have followed two rationales: a) we have selected features that we expect to be useful in early rumor detection and b) we have collected a broad range of features from related work as a basis for investigating the time-dependent impact of a wide variety of features in our time-dependence study. In total, we have constructed over 50 features[4] in the three main categories i.e., *Ensemble*, *Twitter* and *Epidemiological* features. We refrained from using network features, since they are expected to be of little use in early rumor detection [8], since user networks around events need time to form. Following our general idea, none of our features are extracted from the content aggregations. Due to space limitation, we describe only our main features as follows.

---

[4] details are listed in the Appendix.

Ensemble Features. We consider two types of Ensemble Features: features accumulating crowd wisdom and averaging feature for the Tweet credit Scores. The former are extracted from the surface level while the latter comes from the low dimensional level of tweet embeddings; that in a way augments the sparse crowd at early stage.

CrowdWisdom: Similar to [18], the core idea is to leverage the public's common sense for rumor detection: If there are more people denying or doubting the truth of an event, this event is more likely to be a rumor. For this purpose, [18] use an extensive list of bipolar sentiments with a set of combinational rules. In contrast to mere *sentiment* features, this approach is more tailored *rumor* context (difference not evaluated in [18]). We simplified and generalized the "dictionary" by keeping only a set of carefully curated *negative words*. We call them "debunking words" e.g., *hoax*, *rumor* or *not true*. Our intuition is, that the attitude of doubting or denying events is in essence sufficient to distinguish rumors from news. What is more, this generalization augments the size of the crowd (covers more 'voting' tweets), which is crucial, and thus contributes to the quality of the crowd wisdom. In our experiments, "debunking words" is an high-impact feature, but it needs substantial time to "warm up"; that is explainable as the crowd is typically sparse at early stage.

CreditScore: The sets of single-tweet models' predicted probabilities are combined using an *ensemble averaging*-like technique. In specific, our pre-trained $CNN + LSTM$ model predicts the credibility of each tweet tw_{ij} of event E_i. The *softmax* activation function outputs probabilities from 0 (rumor-related) to 1 (news). Based on this, we calculate the average prediction probabilities of all tweets $tw_{ij} \in E_i$ in a time interval t_{ij}. In theory there are different sophisticated ensembling approaches for averaging on both training and test samples; but in a real-time system, it is often convenient (while effectiveness is only affected marginally) to cut corners. In this work, we use a sole training model to average over the predictions. We call the outcome CreditScore.

5 Experimental Evaluation

5.1 Data Collection

To construct the training dataset, we collected rumor stories from online rumor tracking websites such as **snopes.com** and **urbanlegends.about.com**. In more detail, we crawled 4300 stories from these websites. From the story descriptions we manually constructed queries to retrieve the relevant tweets for 270 rumors with high impact. Our approach to query construction mainly follows [11]. For the news event instances (non-rumor examples), we make use of the manually constructed corpus from Mcminn et al. [21], which covers 500 real-world events. In [21], tweets are retrieved via Twitter firehose API from 10th of October 2012 to 7th of November 2012. The involved events are manually verified and relate to tweets with relevance judgments, which results in a high quality corpus. From the 500 events, we select top 230 events with the highest tweet volumes (as a criteria for event impact). Furthermore, we have added 40 other news events, which happened around the time periods of our rumors. This results in a dataset of 270 rumors and 270 events. The dataset details are shown in Table 1. To serve our learning task. we then constructs two distinct datasets for (1) single tweet credibility and (2) rumor classification.

Training data for single tweet classification. Here we follow our assumption that an event might include sub-events for which relevant tweets are rumorous. To deal with

Type	Min Volume	Max Volume	Total	Average
News	98	17414	345235	1327.82
Rumors	44	26010	182563	702.06

Table 1: Tweet Volume of News and Rumors

this complexity, we train our single-tweet learning model only with manually selected *breaking and subless* [5] events from the above dataset. In the end, we used 90 rumors and 90 news associated with 72452 tweets, in total. This results in a highly-reliable large-scale ground-truth of tweets labelled as *news*-related and *rumor*-related, respectively. Note that the labeling of a tweet is inherited from the event label, thus can be considered as an semi-automatic process.

5.2 Single Tweet Classification Experiments

For the evaluation, we developed two kinds of classification models: traditional classifier with handcrafted features and neural networks without tweet embeddings. For the former, we used 27 distinct surface-level features extracted from single tweets (analogously to the Twitter-based features presented in Section 4.2). For the latter, we select the baselines from NN-based variations, inspired by state-of-the-art short-text classification models, i.e., Basic tanh-RNN , 1-layer GRU-RNN, 1-layer LSTM, 2-layer GRU-RNN, FastText [14] and CNN+LSTM [33] model. The hybrid model CNN+LSTM is adapted in our work for tweet classification.

Single Tweet Model Settings. For the evaluation, we shuffle the 180 selected events and split them into 10 subsets which are used for 10-fold cross-validation (we make sure to include near-balanced folds in our shuffle). We implement the 3 non-neural network models with Scikit-learn[6]. Furthermore, neural networks-based models are implemented with TensorFlow [7] and Keras[8]. The first hidden layer is an embedding layer, which is set up for all tested models with the embedding size of 50. The output of the embedding layer are low-dimensional vectors representing the words. To avoid overfitting, we use the 10-fold cross validation and dropout for regularization with dropout rate of 0.25.

Single Tweet Classification Results. The experimental results of are shown in Table 2. The best performance is achieved by the CNN+LSTM model with a good accuracy of 81.19%. The non-neural network model with the highest accuracy is RF. However, it reaches only 64.87% accuracy and the other two non-neural models are even worse. So the classifiers with hand-crafted features are less adequate to accurately distinguish between rumors and news.

---

[5] the terminology *subless* indicates an event with no sub-events for short.

[6] scikit-learn.org/

[7] https://www.tensorflow.org/

[8] https://keras.io/

Model	Accuracy
CNN+LSTM	**0.8119**
2-layer GRU	0.7891
1-layer GRU	0.7644
1-layer LSTM	0.7493
Basic RNN with tanh	0.7291
FastText	0.6602
Random Forest	**0.6487**
SVM	0.5802
Decision Trees	0.5774

Table 2: Single Tweet Classification Performance

Feature	Importance
PolarityScores	0.146
Capital	0.096
LengthOfTweet	0.092
UserTweets	0.087
UserFriends	0.080
UserReputationScore	0.080
UserFollowers	0.079
NumOfChar	0.076
Stock	0.049
NumNegativeWords	0.030
Exclamation	0.023

Table 3: Top Features Importance

Discussion of Feature Importance For analyzing the employed features, we rank them by importances using RF (see 3). The best feature is related to sentiment polarity scores. There is a big difference between the sentiment associated to rumors and the sentiment associated to real events in relevant tweets. In specific, the average polarity score of news event is -0.066 and the average of rumors is -0.1393, showing that rumor-related messages tend to contain more negative sentiments. Furthermore, we would expect that verified users are less involved in the rumor spreading. However, the feature appears near-bottom in the ranked list, indicating that it is not as reliable as expected. Also interestingly, "IsRetweeted" feature is pretty much useless, which means the probability of people retweeting rumors or true news are similar (both appear near-bottom in the ranked feature list).

It has to be noted here that even though we obtain reasonable results on the classification task in general, the prediction performance varies considerably along the time dimension. This is understandable, since tweets become more distinguishable, only when the user gains more knowledge about the event.

5.3 Rumor Datasets and Model Settings

We use the same dataset described in Section 5.1. In total –after cutting off 180 events for pre-training single tweet model – our dataset contains 360 events and 180 of them are labeled as rumors. Those rumors and news fall comparatively evenly in 8 different categories, namely *Politics*, *Science*, *Attacks*, *Disaster*, *Art*, *Business*, *Health* and *Other*. Note, that the events in our training data are not necessarily subless, because it is natural for high-impact events (e.g., *Missing MH370* or *Munich shooting*) to contain sub-events. Actually, we empirically found that roughly 20% of our events (mostly news) contain sub-events. As a rumor is often of a long circulating story [10], this results in a rather long time span. In this work, we develop an event identification strategy that focuses on the first 48 hours after the rumor is peaked. We also extract 11,038 domains, which are contained in tweets in this 48 hours time range.

Rumor Detection Model Settings. For the time series classification model, we only report the best performing classifiers, SVM and Random Forest, here. The parameters of SVM with RBF kernel are tuned via grid search to $C = 3.0$, $\gamma = 0.2$. For Random

Forest, the number of trees is tuned to be 350. All models are trained using 10-fold cross validation.

5.4 Rumor Classification Results

We tested all models by using 10-fold cross validation with the same shuffled sequence. The results of these experiments are shown in Table 4. Our proposed model (***Ours***) is the time series model learned with Random Forest including all ensemble features; $TS-SVM$ is the baseline from [20], and $TS-SVM_{all}$ is the $TS-SVM$ approach improved by using our feature set. In the lower part of the table, RNN_{el} is the RNN model at event-level [19]. As shown in the Table 4 and as targeted by our early detection approach, our model has the best performance in all case over the first 24 hours, remarkably outperforming the baselines in the first 12 hours of spreading. The performance of RNN_{el} is relatively low, as it is based on aggregated *contents*. This is expected as the news (non-rumor) dataset used in [19] are crawled also from snopes.com, in which events are often of small granularity (aka. subless). As expected, exploiting contents solely at event-level is problematic for high-impact, evolving events on social media. We leave a deeper investigation on the sub-event issue to future work.

Model	Accuracy in hours								
	1	6	12	18	24	30	36	42	48
Ours	**0.82**	**0.84**	**0.84**	<u>0.84</u>	**0.87**	<u>0.87</u>	**0.88**	<u>0.89</u>	**0.91**
$TS-SVM_{all}$	<u>0.76</u>	0.79	<u>0.83</u>	0.83	**0.87**	**0.88**	0.86	0.89	<u>0.90</u>
$TS-SVM_{Credit}$	0.73	<u>0.80</u>	0.83	**0.85**	0.85	0.86	<u>0.88</u>	**0.90**	<u>0.90</u>
$TS-SVM$ [20]	0.69	0.76	0.81	0.81	0.84	0.86	0.87	0.88	0.88
RNN_{el} [19]	0.68	0.77	0.81	0.81	0.84	0.83	0.81	0.85	0.86
$SVM_{static}+Epi$ [13]	0.60	0.69	0.71	0.72	0.75	0.78	0.75	0.78	0.81
$SVM_{static}+SpikeM$ [17]	0.58	0.68	0.72	0.73	0.77	0.78	0.78	0.79	0.77
SVM_{static} [31]	0.62	0.70	0.70	0.72	0.75	0.80	0.79	0.78	0.77

Table 4: Performance of different models over time (bold for best accuracy, underlined for second-to-best). TS indicates time-series structure; we separate the TS models (upper) with the static ones (lower).

CreditScore and CrowdWisdom. As shown in Table 5, *CreditScore* is the best feature in overall. In Figure 4 we show the result of models learned with the full feature set with and without *CreditScore*. Overall, adding *CreditScore* improves the performance, especially for the first 8-10 hours. The performance of *all-but-CreditScore* jiggles a bit after 16-20 hours, but it is not significant. *CrowdWisdom* is also a good feature which can get 75.8% accuracy as a single feature. But its performance is poor (less than 70%) in the first 32 hours getting better over time (see Table 5). Table 5 also shows the performance of *sentiment* feature (*PolarityScores*), which is generally low. This demonstrates the effectiveness of our *curated* approach over the *sentiments*, yet the crowd needs time to unify their views toward the event while absorbing different kinds of information.
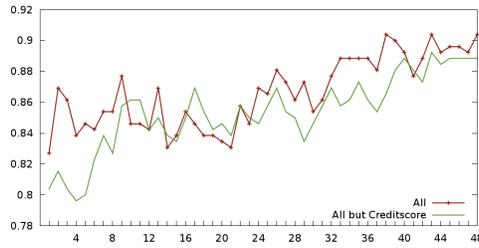
Fig. 4: Accuracy: All features with and without CreditScore.

Features				Ranks						
Hours	1	6	12	18	24	30	36	42	48	AVG
CreditScore	1	0	0	0	0	0	0	0	0	0.08
CrowdWisdom	34	38	21	14	8	5	5	2	2	13.18
PolarityScores	12	15	23	28	33	33	34	31	32	28

Table 5: Importance ranking of CreditScore, CrowdWisdom and PolarityScores over time; 0 indicates the best rank.

Case Study: Munich Shooting. We showcase here a study of the Munich shooting. We first show the event timeline at an early stage. Next we discuss some examples of misclassifications by our "weak" classifier and show some analysis on the strength of some highlighted features. The rough event timeline looks as follows.

– At 17:52 CEST, a shooter opened fire in the vicinity of the Olympia shopping mall in Munich. 10 people, including the shooter, were killed and 36 others were injured.
– At 18:22 CEST, the first tweet was posted. There might be some certain delay, as we retrieve only tweets in English and the very first tweets were probably in German. The tweet is *"Sadly, i think there's something terrible happening in #Munich #Munchen. Another Active Shooter in a mall. #SMH"*.
– At 18:25 CEST, the second tweet was posted: *"Terrorist attack in Munich????"*.
– At 18:27 CEST, traditional media (BBC) posted their first tweet. *"'Shots fired' in Munich shopping centre - http://www.bbc.co.uk/news/world-europe-36870800a02026 @TraceyRemix gun crime in Germany just doubled"*.
– At 18:31 CEST, the first misclassified tweet is posted. It was a tweet with shock sentiment and swear words: *"there's now a shooter in a Munich shopping centre.. What the f*** is going on in the world. Gone mad"*. It is classified as *rumor-related*.

We observe that at certain points in time, the volume of rumor-related tweets (for sub-events) in the event stream surges. This can lead to *false positives* for techniques that model events as the aggregation of all tweet contents; that is undesired at critical moments. We trade-off this by debunking at single tweet level and let each tweet vote for the credibility of its event. We show the *CreditScore* measured over time in Figure 5(a). It can be seen that although the credibility of some tweets are low (rumor-related), averaging still makes the *CreditScore* of Munich shooting higher than the average of news

events (hence, close to a *news*). In addition, we show the feature analysis for Contain-News (percentage of URLs containing news websites) for the event *Munich shooting* in Figure 5(b). We can see the curve of *Munich shooting* event is also close to the curve of average news, indicating the event is more news-related.
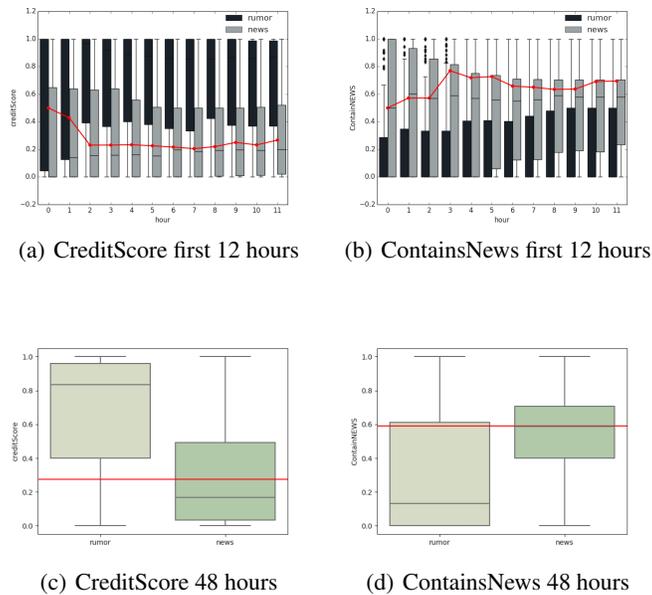

(a) CreditScore first 12 hours   (b) ContainsNews first 12 hours

(c) CreditScore 48 hours   (d) ContainsNews 48 hours

Fig. 5: Creditscore and ContainsNews for *Munich shooting* in red lines, compared with the corresponding average scores for *rumor* and *news*.

6   Conclusion

In this work, we propose an effective cascaded rumor detection approach using deep neural networks at tweet level in the first stage and wisdom of the "machines", together with a variety of other features in the second stage, in order to enhance rumor detection performance in the early phase of an event. The proposed approach outperforms state of the art methods for early rumor detection. There is, however, still considerable room to improve the effectiveness of the rumor detection method. The support for events with rumor sub-events is still limited. The current model only aims not to misclassify long-running, multi-aspect events where rumors and news are mixed and evolve over time as false positive.

Appendix A   Time Period of an Event

The time period of a rumor event is hard to define. One reason is a rumor may be created for a long time and kept existing on Twitter, but it did not attract the crowd's attention. However it can be triggered by other events after a uncertain time and suddenly spreads as a bursty event. E.g., a rumor[9] claimed that Robert Byrd was member of KKK. This rumor has been circulating in Twitter for a while. As shown in Figure 6(a) that almost every day there were several tweets talking about this rumor. But this rumor was triggered by a picture about Robert Byrd kissing Hillary Clinton in 2016 [10] and Twitter users suddenly noticed this rumor and it was bursted. And what we are really interested in is the tweets which are posted in hours around the bursty peak. We defined the hour with the most tweets' volume as t_{max} and we want to detect the rumor event as soon as possible before its burst, so we define the time of the first tweet before t_{max} within 48 hours as the beginning of this rumor event, marked as t_0. And the end time of the event is defined as $t_{end} = t_0 + 48$. We show the tweet volumes in Figure 6 of the above rumor example.
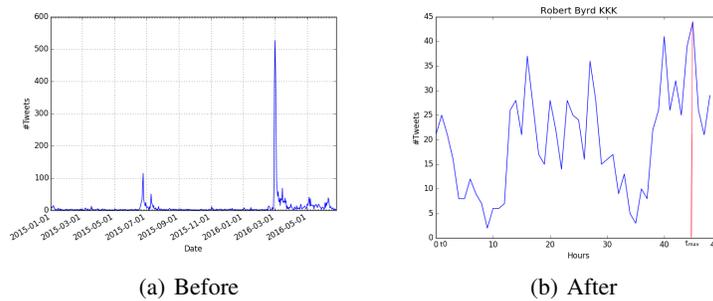

(a) Before   (b) After

Fig. 6: tweet volume of the rumor event of Robert Byrd at full scale and after selected time period

---

[9] http://www.snopes.com/robert-byrd-kkk-photo/

[10] http://www.snopes.com/clinton-byrd-photo-klan/

Appendix B   Full Features

Category	Feature	Description
Twitter Features	Hashtag	% tweets contain #hashtag [6][18][24][11][18]
	Mention	% tweets mention others @user [6][18][24][11][18]
	NumUrls	# URLs in the tweet [6][24][11][31][18]
	Retweets	average # retweets [18]
	IsRetweet	% tweets are retweeted from others [6][11]
	ContainNEWS	% tweets contain URL and its domain's catalogue is News [18]
	WotScore	average WOT score of domain in URL [11]
	URLRank5000	% tweets contain URL whose domain's rank less than 5000 [6]
	ContainNewsURL	% tweets contain URL whose domain is News Website
Text Features	LengthofTweet	average tweet lengths [6][11]
	NumOfChar	average # tweet characters [6][11]
	Capital	average fraction of characters in Uppercase [6]
	Smile	% tweets contain $: ->, : -), ; ->, ; -)$ [6][11]
	Sad	% tweets contain $: -<, : -(, ; ->, ; -($ [6][11]
	NumPositiveWords	average # positive words [6][11][31][18]
	NumNegativeWords	average # negative words [6][11][31][18]
	PolarityScores	average polarity scores of the Tweets [6][31][18]
	Via	% of tweets contain via [11]
	Stock	% of tweets contain $ [6][11]
	Question	% of tweets contain ? [6][18]
	Exclamation	% of tweets contain ! [6][18]
	QuestionExclamation	% of tweets contain multi Question or Exclamation mark [6][18]
	I	% of tweets contain first pronoun like I, my, mine, we, our [6][11][18]
	You	% of tweets contain second pronoun like U, you, your, yours [6]
	HeShe	% of tweets contain third pronoun like he, she, they, his, etc. [6]
User Features	UserNumFollowers	average number of followers [6][11][18]
	UserNumFriends	average number of friends [6][11][18]
	UserNumTweets	average number of users posted tweets [6][11][31][18]
	UserNumPhotos	average number of users posted photos [31]
	UserIsInLargeCity	% of users living in large city [31][18]
	UserJoinDate	average days since users joining Twitter [6][31][18]
	UserDescription	% of user having description [6][31][18]
	UserVerified	% of user being a verified user[31][18]
	UserReputationScore	average ratio of #Friends over (#Followers + #Friends) [18]
Epidemiological Features	β_{SIS}	Parameter β of Model SIS [13]
	α_{SIS}	Parameter α of Model SIS [13]
	β_{SEIZ}	Parameter β of Model SEIZ [13]
	b_{SEIZ}	Parameter b of Model SEIZ[13]
	l_{SEIZ}	Parameter l of Model SEIZ [13]
	p_{SEIZ}	Parameter p of Model SEIZ [13]
	ε_{SEIZ}	Parameter ε of Model SEIZ [13]
	ρ_{SEIZ}	Parameter ρ of Model SEIZ [13]
	R_{SI}	Parameter R_{SI} of Model SEIZ [13]
SpikeM Model Features	P_s	Parameter P_s of Model Spike [17]
	P_a	Parameter P_a of Model SpikeM [17]
	P_p	Parameter P_p of Model SpikeM [17]
	Q_s	Parameter Q_s of Model SpikeM [17]
	Q_a	Parameter Q_a of Model SpikeM [17]
	Q_p	Parameter Q_p of Model SpikeM [17]
Crowd Wisdom	CrowdWisdom	% of tweets containing "Debunking Words" [18] [32]
CreditScore	CreditScore	average CreditScore

Table 6: Features of Time Series Rumor Detection Model

References

1. F. Ahmed and M. Abulaish. An mcl-based approach for spam profile detection in online social networks. In *Proceedings of TrustCom*, pages 602–608. IEEE, 2012.
2. G. W. Allport and L. Postman. The psychology of rumor. 1947.
3. Y. Bao, C. Yi, Y. Xue, and Y. Dong. A new rumor propagation model and control strategy on social networks. In *Proceedings of ICWSM*, pages 1472–1473. ACM, 2013.
4. L. Barbosa and J. Feng. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of ACL*, pages 36–44, 2010.
5. J. Borge-Holthoefer and Y. Moreno. Absence of influential spreaders in rumor dynamics. *Physical Review E*, 85(2):026116, 2012.
6. C. Castillo, M. Mendoza, and B. Poblete. Information credibility on twitter. In *Proceedings of WWW*, pages 675–684. ACM, 2011.
7. T. Chen, L. Wu, X. Li, J. Zhang, H. Yin, and Y. Wang. Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection. *arXiv preprint arXiv:1704.05973*, 2017.
8. M. Conti, D. Lain, R. Lazzeretti, G. Lovisotto, and W. Quattrociocchi. It's always april fools' day! on the difficulty of social network misinformation classification via propagation features. *CoRR*, abs/1701.04221, 2017.
9. B. Dhingra, Z. Zhou, D. Fitzpatrick, M. Muehl, and W. W. Cohen. Tweet2vec: Character-based distributed representations for social media. *arXiv preprint arXiv:1605.03481*, 2016.
10. A. Friggeri, L. A. Adamic, D. Eckles, and J. Cheng. Rumor cascades. 2014.
11. A. Gupta, P. Kumaraguru, C. Castillo, and P. Meier. Tweetcred: Real-time credibility assessment of content on twitter. In *SocInfo*. Springer, 2014.
12. S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
13. F. Jin, E. Dougherty, P. Saraf, Y. Cao, and N. Ramakrishnan. Epidemiological modeling of news and rumors on twitter. In *Proceedings of SNA-KDD*, 2013.
14. A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.
15. Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.
16. D. Kimmey. Twitter event detection. 2015.
17. S. Kwon, M. Cha, K. Jung, W. Chen, and Y. Wang. Prominent features of rumor propagation in online social media. In *Proceedings of ICDM*, 2013.
18. X. Liu, A. Nourbakhsh, Q. Li, R. Fang, and S. Shah. Real-time rumor debunking on twitter. In *Proceedings of CIKM*, pages 1867–1870. ACM, 2015.
19. 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.
20. J. Ma, W. Gao, Z. Wei, Y. Lu, and K.-F. Wong. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of CIKM*, 2015.
21. A. J. McMinn, Y. Moshfeghi, and J. M. Jose. Building a large-scale corpus for evaluating event detection on twitter. In *Proceedings of CIKM*, 2013.
22. P. Meladianos, G. Nikolentzos, F. Rousseau, Y. Stavrakas, and M. Vazirgiannis. Degeneracy-based real-time sub-event detection in twitter stream. In *Proceedings of ICWSM*, 2015.
23. M. Mendoza, B. Poblete, and C. Castillo. Twitter under crisis: can we trust what we rt? In *Proceedings of the first workshop on social media analytics*, pages 71–79. ACM, 2010.
24. V. Qazvinian, E. Rosengren, D. R. Radev, and Q. Mei. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of EMNLP*, 2011.
25. E. Seo, P. Mohapatra, and T. Abdelzaher. Identifying rumors and their sources in social networks. In *SPIE*, 2012.

26. C. R. Sunstein. *On rumors: How falsehoods spread, why we believe them, and what can be done*. Princeton University Press, 2014.

27. R. M. Tripathy, A. Bagchi, and S. Mehta. A study of rumor control strategies on social networks. In *Proceedings of CIKM*, pages 1817–1820. ACM, 2010.

28. S. Vosoughi, P. Vijayaraghavan, and D. Roy. Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 1041–1044. ACM, 2016.

29. A. H. Wang. Don't follow me: Spam detection in twitter. In *Proceedings of SECRYPT*, pages 1–10. IEEE, 2010.

30. K. Wu, S. Yang, and K. Q. Zhu. False rumors detection on sina weibo by propagation structures. In *Proceedings of ICDE*, pages 651–662. IEEE, 2015.

31. F. Yang, Y. Liu, X. Yu, and M. Yang. Automatic detection of rumor on sina weibo. In *Proceedings of MDS*. ACM, 2012.

32. Z. Zhao, P. Resnick, and Q. Mei. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of WWW*, 2015.

33. C. Zhou, C. Sun, Z. Liu, and F. Lau. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*, 2015.