# Early Rumour Detection

**Kaimin Zhou**[2,3] **Chang Shu**[1,2] **Binyang Li**[3*] **Jey Han Lau**[2,4,5]

[1] School of Computer Science, University of Nottingham Ningbo China
[2] DeepBrain
[3] School of Information Science and Technology, University of International Relations
[4] School of Computing and Information Systems, The University of Melbourne
[5] IBM Research Australia

will@deepbrain.ai, scxcs1@nottingham.edu.cn,
byli@uir.edu.cn, jeyhan.lau@gmail.com

## Abstract

Rumours can spread quickly through social media, and malicious ones can bring about significant economical and social impact. Motivated by this, our paper focuses on the task of rumour detection; particularly, we are interested in understanding how *early* we can detect them. Although there are numerous studies on rumour detection, few are concerned with the timing of the detection. A successfully-detected malicious rumour can still cause significant damage if it isn't detected in a timely manner, and so timing is crucial. To address this, we present a novel methodology for early rumour detection. Our model treats social media posts (e.g. tweets) as a data stream and integrates reinforcement learning to learn the number minimum number of posts required before we classify an event as a rumour. Experiments on Twitter and Weibo demonstrate that our model identifies rumours earlier than state-of-the-art systems while maintaining a comparable accuracy.

## 1 Introduction

The concept of rumour has a long history, and there are various definitions from different research communities (Allport and Postman, 1947). In this paper, we follow a commonly accepted definition of rumour, that it is an unverified statement, circulating from person to person and pertaining to an object, event, or issue of public concern and it is circulating without known authority for its truthfulness at the current time, but it may turn out to be true, or partly or entirely false; alternatively, it may also remain unresolved (Peterson and Gist, 1951; Zubiaga et al., 2018).

Rumours have the potential to spread quickly through social media, and bring about significant economical and social impact. Figure 1 illustrates an example of a rumour propagating on TWITTER. The source message started a claim about the cause of Michael Brown's shooting, and it was published shortly after the shooting happened. It claimed that he was shot ten times by the police for stealing candy. The message was retweeted by multiple users on TWITTER, and within 24 hours there were about 900K users involved, either by reposting, commenting, or questioning the original source message. From Figure 1, we see some users (e.g. User 7) question the veracity of the original message. Had the rumour been identified timely and rebutted, its propagation could have been contained.

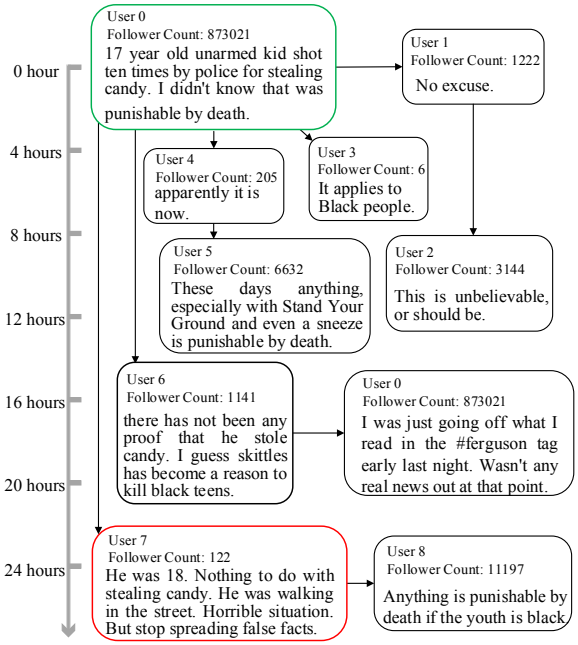


Figure 1: An illustration of a rumour propagating on TWITTER. The green box indicates the source message, and the red box highlights a post that rebuts the rumour.

Most studies (Qazvinian et al., 2011; Zhang et al., 2015) consider rumour detection as a binary classification problem, where they extract various



*Proceedings of NAACL-HLT 2019*, pages 1614–1623
Minneapolis, Minnesota, June 2 - June 7, 2019. 

features to capture rumour indicative signals for detecting a rumour, and a few recent works explore deep learning approaches to enhance detection accuracy (Long et al., 2017; Ruchansky et al., 2017). In all these studies, however, the timeliness of the rumour detection is not evaluated.

There are a few exceptions. In Ma et al. (2015) and Kwon et al. (2017), the authors define a checkpoint (e.g. number of posts or time elapsed after the source message) in the timeline and use all the posts prior to this checkpoint to classify a rumour. The checkpoint is often a pre-determined value for all rumours, and so does not capture the variation of propagation patterns for different rumours.

The focus of our paper is on *early* rumour detection. That is, our aim is to identify rumours as early as possible, while keeping a reasonable detection accuracy. Our early rumour detection system (ERD) features two modules: a rumour detection module that classifies whether an event (which consists of a number of posts) constitutes a rumour, and a checkpoint module that determines when to trigger the rumour detection module.

ERD treats incoming posts as a data stream and monitors the posts in real time. When ERD receives a new post, this post — along with all prior posts of the same event — will be used to decide if it constitutes an appropriate checkpoint to trigger the rumour detection module. ERD integrates reinforcement learning for the checkpoint module to guide the rumour detection module, using its classification accuracy as a reward. Through reinforcement learning ERD is able to learn the minimum number of posts required to identify a rumour. In other words, ERD can dynamically determine the appropriate checkpoint for different rumours, and this feature is the core novelty of our methodology.

To evaluate our approach, we use standard microblog data sets from WEIBO and TWITTER. We compare our method with benchmark rumour detection systems (Ma et al., 2016; Ruchansky et al., 2017; Dungs et al., 2018) and found that ERD could on average identify rumours within 7.5 and 3.4 hours with an accuracy of 93.3% and 85.8% on WEIBO and TWITTER respectively. Our detection accuracy performance is better than a state-of-the-art system that that detects rumours within 12 hours.

To summarise, we present a novel methodology for rumour detection. Unlike most rumour detection systems, our approach determines the checkpoint for each event dynamically, by learning when it should classify it as a rumour. Our experimental results showed that ERD outperforms state-of-the-art methods over two benchmark data sets in detection accuracy and timeliness. Our proposed framework is flexible and the individual modules (i.e. the rumour detection and checkpoint module) can be extended to incorporate more complex networks for further improvements. An open source implementation of our model is available at: `https://github.com/DeepBrainAI/ERD`.

## 2 Related Work

Traditionally, research on rumour detection has mainly focused on developing handcrafted features for machine learning algorithms (Qazvinian et al., 2011). Takahashi and Igata (2012) propose a method for rumour detection on Twitter using cue words and tweets statistics. Yang et al. (2012) apply two new types of features — client-based and location-based features — to rumour detection on Sina Weibo. Beyond this, user-based (Liang et al., 2015) and topic-based (Yang et al., 2015) features have also been explored. Friggeri et al. (2014) demonstrate that there are structural differences in the propagation of rumours and non-rumours, and Wu et al. (2015) and Ma et al. (2017) experiment with using these propagation patterns extensively to improve detection.

More recently, deep learning models are explored for the task. Compared to traditional machine learning approaches, these deep learning models tend to rely less on sophisticated handcrafted features. Ma et al. (2016) introduce a rumour detection model for microblogs based on recurrent networks. The input to their model is simple tf-idf features but it outperforms models leveraging handcrafted features. Sampson et al. (2016) show that implicit linkages between conversation fragments improve detection accuracy. Long et al. (2017) present a deep attention model that learns a hidden temporal representation for each sequential posts to represent the hypothesis. Ruchansky et al. (2017) integrate textual, user response, and source information into their neural models and achieve better performance.

Most of these works focus on detection accuracy, and so largely ignore the timing of the detection. Ma et al. (2015) develop a dynamic time

series structure to incorporate temporal information to the features to understand the whole life cycle of rumours. Zhao et al. (2015) propose a detection model using a set of regular expressions to find posts that question or rebut the rumour to detect it earlier. Dungs et al. (2018) present an approach that checks for a rumour after 5 or 10 retweets. These models are interested in *early* rumour detection, although the checkpoint for triggering a detection is pre-determined, and succeeding posts after the checkpoint are usually ignored. On a similar note but a different task, Farajtabar et al. (2017) experiment with reinforcement learning by combining it with a point process network activity model to detect fake news and found some success.

## 3 Model Architecture

Let $E$ denote an event, and it consists of a series of relevant posts $x_i$, where $x_0$ denotes the source message and $x_T$ the last relevant message.[1] The objective of early rumor detection is to make a classification decision whether $E$ is a rumour as early as possible while keeping an acceptable detection accuracy.[2]

As shown in Figure 2, ERD has two modules: a rumour detection module (RDM) that classifies whether an event is a rumour, and a checkpoint module (CM) that decides when the rumour detection module should be triggered. The checkpoint module plays an important role here, as it is responsible for the timeliness of a detection.

### 3.1 Rumor Detection Module (RDM)

RDM contains three layers: a word embedding layer that maps input words into vectors, a max-pooling layer that extracts important features of a post, and a GRU (Cho et al., 2014) that processes the sequential posts of an event.

In the word embedding layer, we map words in post $x_i$ into vectors, yielding vectors $e_i^j$ for each word. To capture the most salient features of a post, we apply a max pooling operation (Collobert et al., 2011; Kim, 2014; Lau et al., 2017), producing a fixed size vector $m_i$:

$$m_i = \text{maxpool}([\mathbf{W}_m e_i^0; \mathbf{W}_m e_i^1; ...; \mathbf{W}_m e_i^K])$$

where $K$ is the number of words in the post. Henceforth $\mathbf{W}$ in all equations are model parameters.

To capture the temporal relationship between multiple posts, we use a GRU (Cho et al., 2014):

$$h_i = \text{GRU}(m_i, h_{i-1}) \quad (1)$$

We take the final state $h_N$ ($N$ = number of posts received to date) and use it to perform rumour classification:

$$p = \text{softmax}(\mathbf{W}_p h_N + b_p) \quad (2)$$

where $p \in \mathbb{R}^2$, i.e. $p^0$ ($p^1$) gives the probability of the positive (negative) class.[3]

### 3.2 Checkpoint Module (CM)

Rather than setting a static checkpoint when to classify an event as a rumour, CM learns the number of posts needed to trigger RDM. To this end, we leverage deep reinforcement learning to identify the optimal checkpoint. We reward CM based on RDM's accuracy and also penalise CM slightly every time it decides to not trigger RDM (and continue to monitor the event). This way CM learns the trade-off between detection accuracy and timeliness. The reward function is detailed in Section 3.3.

We use the deep Q-learning model (Mnih et al., 2013) for CM. The optimal action-value function $Q^*(s, a)$ is defined as the maximum expected return achievable under state $s$, which can be formulated as follows:

$$Q^*(s, a) = E_{s' \varepsilon}[r + \gamma \max_{a'} Q_i(s', a')|s, a]$$

where $r$ is the reward value, $\gamma$ the discount rate, and the optimal action in all action sequence $a'$ is selected to maximise the expected value of $r + \gamma Q^*(s', a')$.

The optimal action-value function obeys the Bellman equation and is used for iterative value update:

$$Q_{i+1}(s, a) = E[r + \gamma \max_{a'} Q_i(s', a')|s, a]$$

The above iterative algorithm will converge and reach the optimal action value function, i.e. $Q_i \rightarrow Q^*$ when $q \rightarrow \infty$ (Sutton et al., 1998).

[1] Relevant posts are defined as retweets or responses to a source message.

[2] The earliest possible time to classify $E$ is when we receive the first post $x_0$.

[3] Although sigmoid activation is more appropriate here as it is a binary classification task, we used the softmax function because in preliminary experiments we considered a third neural class.

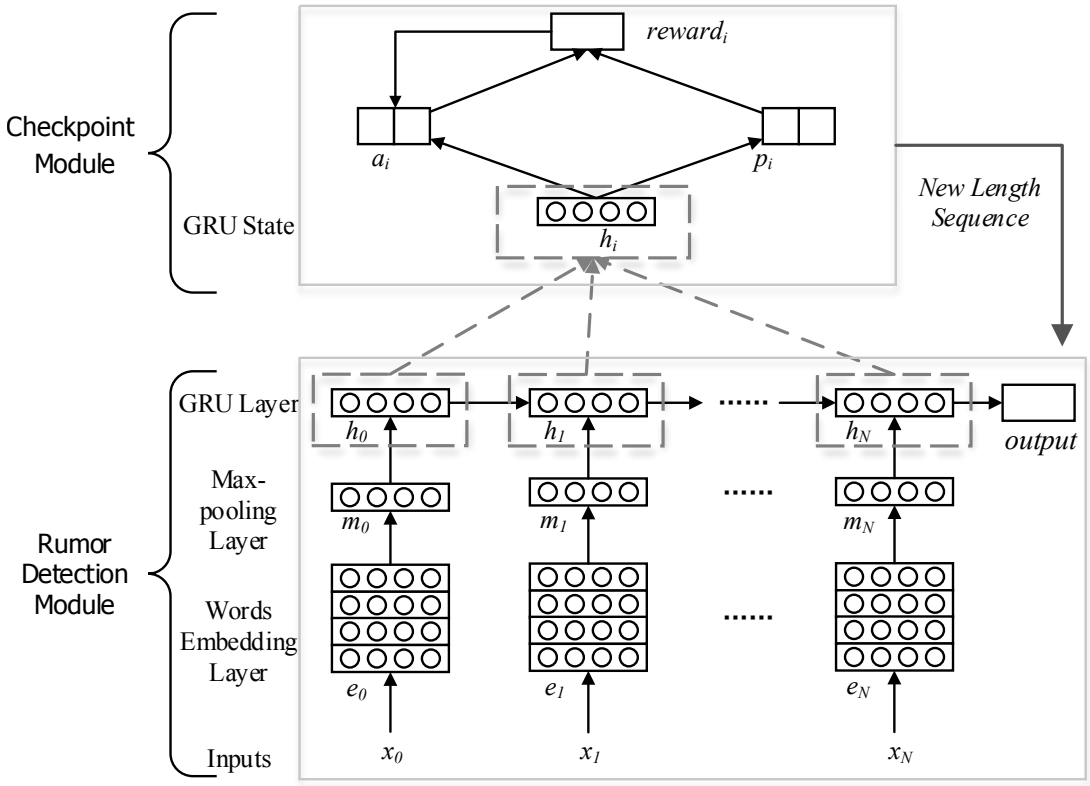


Figure 2: Architecture of ERD.

CM takes as input the hidden states produced by the GRU in RDM to compute the action-value function using a two-layer feedforward network:

$$a_i = \mathbf{W}_a(\text{ReLu}(\mathbf{W}_h h_i + b_h)) + b_a \quad (3)$$

where $a_i \in \mathbb{R}^2$ is the action value for *terminate* ($a_i^0$) or *continue* ($a_i^1$) at post $x_i$. Note that a random action will be taken with the probability of $\theta$ irrespective to the action value $a_i$.

### 3.3 Joint Training

We train both RDM and CM jointly, and the training process is similar to that of generative adversarial networks (Goodfellow et al., 2014). The checkpoint module serves as the generator for action sequences, while the detection module is the discriminator. A key contrast, however, is that the two modules are working cooperatively rather than adversarially.

CM is trained using RDM's accuracy as reward. To compute the reward, we first pre-train RDM based on cross entropy:

$$-\sum_j [L_j \log(p_j^0) + (1 - L_j)(\log(p_j^1))] + \alpha l_2$$

where $L_j$ is a binary label indicating the true class for event $E_j$, $p$ is computed based on Equation (2), $l_2$ is the L2 loss for RDM parameters, and $\alpha$ is a hyper-parameter for scaling $l_2$.

We then train CM while keeping RDM's parameters fixed. In each step of the training, new posts that have arrived and previous GRU states are first fed to the RDM to produce the new states (Equation (1)), which will in turn be used by CM to calculate the action values (Equation (3)). This decides whether the system takes the *continue* or *terminate* action. If *terminate* is chosen, the reward is given in accordance to RDM's prediction; otherwise, a small penalty is incurred:

$$r_i = \begin{cases} \log M, & \textit{terminate} \text{ with correct prediction} \\ -P, & \textit{terminate} \text{ with incorrect prediction} \\ -\varepsilon, & \textit{continue} \end{cases}$$

where $M$ is the number of correct predictions accumulated thus far, $P$ is a large value to penalise an incorrect prediction, and $\varepsilon$ is a small penalty value for delaying the detection.

To optimise our action value function, we apply the deep Q-learning approach with the experience replay algorithm (Mnih et al., 2013). Based on the optimal action-value function $Q^*(s, a)$, the objective of the action value function $y_i$ is given as

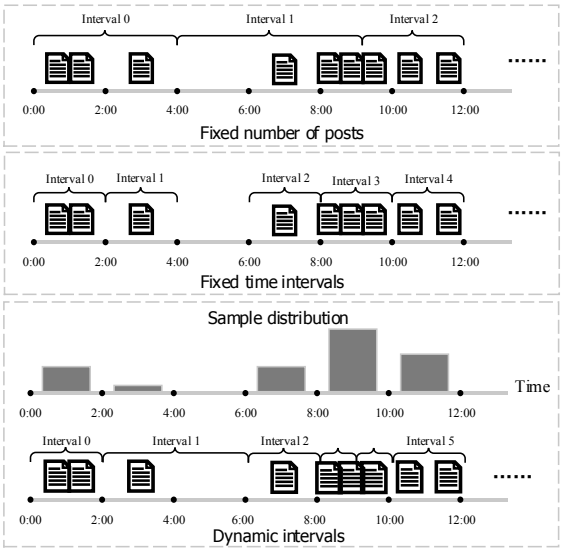


Figure 3: Three bucketing strategies to process streaming posts in batches.

follows:

$$y_i = \begin{cases} r_i, & \textit{terminate} \\ r_i + \gamma \max_{a'} Q(h_{i+1}, a'; \theta), & \textit{continue} \end{cases}$$

where $\gamma$ is the discount rate that decides how much experience is taken into consideration. And lastly, CM is optimised by minimising the cost:

$$(y_i - a_i)^2$$

We train CM and RDM in an alternating fashion, i.e. we train CM for several iterations while keeping RDM's parameters fixed, and then we move to train RDM for several iterations while keeping CM's parameters fixed. Training converges when CM's reward value stabilises between consecutive epochs.

### 3.4 Bucketing Strategy

For processing efficiency purposes, instead of processing each incoming post individually, we experiment with several bucketing strategies that group posts together and process them in batches. As Figure 3 illustrates, we group posts based on: (1) a fixed number of posts (FN), e.g. every 3 posts (i.e. 3 posts are combined together forming 1 single post); (2) a fixed time interval (FT), e.g. every 2 hours; and (3) a dynamic interval (DI) that ensures the number of posts collected in an interval is close to the mean number of posts collected in an hour in the full data set.

| Statistics | WEIBO | TWITTER |
|---|---|---|
| User# | 2,746,818 | 49,345 |
| Posts# | 3,805,656 | 103,212 |
| Events# | 4,664 | 5,802 |
| Rumors# | 2,313 | 1,972 |
| Non-rumours | 2,351 | 3,830 |
| Avg. hours per event | 2,460.7 | 33.4 |
| Avg. # of posts per event | 816 | 17 |
| Max # of posts per event | 59,318 | 346 |
| Min # of posts per event | 10 | 1 |

Table 1: Statistics of WEIBO and TWITTER.

## 4 Experiment

### 4.1 Data Set

We experiment with two data sets: WEIBO and TWITTER, developed by Ma et al. (2016) and Zubiaga et al. (2016) respectively.[4]

Statistics of the data sets is presented in Table 1. Even though both data sets have a comparable number of events, WEIBO is an order of magnitude larger than TWITTER as there are more posts per event. We reserve 10% of the events as the validation set for hyper-parameter tuning and early stopping, and split the rest in a ratio of 3:1 for training and test partitions.

### 4.2 Model Comparison

As a baseline, we use an SVM with tf-idf features. We also include several state-of-the-art rumour detection systems for comparisons: CSI (Ruchansky et al., 2017) on WEIBO; CRF (Zubiaga et al., 2016) and HMM (Dungs et al., 2018) on TWITTER; and GRU-2 (Ma et al., 2016) on both data sets. For GRU-2 (Ma et al., 2016) we also report performance on several variants that use a different recurrent network: simple RNN with tanh activation (RNN); single-layer LSTM (LSTM); and single-layer GRU (GRU-1).

CSI is a neural model that integrates text and users representations to classify rumours. CRF and HMM are classical models that use crowd opinions (a.k.a. stance) of the event for classification. GRU-2 is based on a two-layer GRU that captures contextual information of posts with tf-idf features as inputs.

[4]There is a small difference in the definition of a "rumour" in these two data sets. For WEIBO, all labelled rumours are false rumours (i.e. the source message contains verified untruthful statements), where else for TWITTER, rumours maybe truthful, untruthful, or unverified.

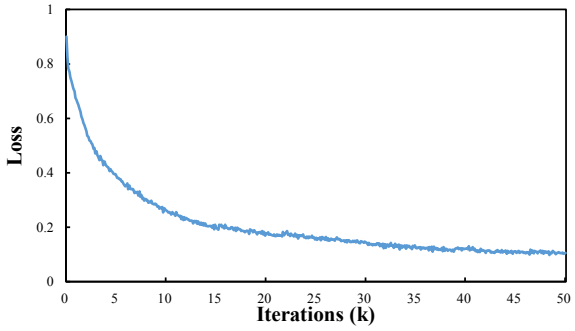


Figure 4: Loss over time during joint training.

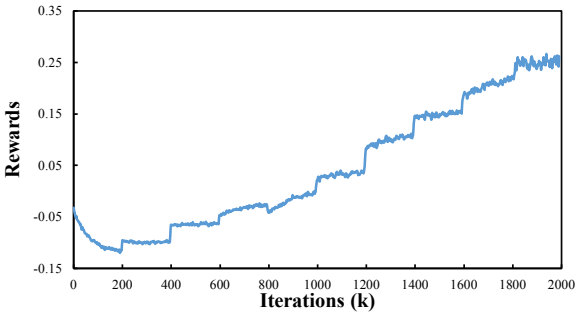


Figure 5: Reward over time during joint training.

### 4.3 Preprocessing and Hyper-parameters

We preprocess each post by segmenting them into words, and remove all stop words.[5] We pre-train word embeddings and kept them fixed during training.[6] $\theta$ is set to 0.01 and $\gamma$ to 0.95; both values are determined empirically based on validation data. We use the Adam optimiser (Kingma and Ba, 2014) with a learning rate of 0.001 during joint training, which we found to produce stable training.

### 4.4 Training Loss and Reward

We present the training loss and reward values over time during joint training in Figure 4 and Figure 5. We pre-train RDM for 2 epochs before joint training, and then we train RDM and CM in an alternating fashion for 1 epoch and 200K iterations respectively. We can see that loss declines steadily after 20K iterations and converges at around 50K iterations. The reward curve, on the other hand, fluctuates more as the reward was calculated based on the accuracy of RDM. When switching between training RDM and CM, the reward value tends to change abruptly, although over time we see a consistent improvement.

[5]For TWITTER, words are tokenised using white spaces, and stopword list is based on NLTK (Bird et al., 2009). For WEIBO, Jieba is used for tokenisation: https://pypi.org/project/jieba/; and stopword list is a customised list based on: http://blog.sina.com.cn/s/blog_a19ab3770102wjav.html.

[6]For WEIBO, the embeddings are pre-trained using word2vec (Mikolov et al., 2013) on a separate Weibo data set we collected. For TWITTER, the embeddings are pre-trained GloVe embeddings (Pennington et al., 2014). Unknown words are initialised as zero vectors.

| Method | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| FN | **0.874** | **0.808** | 0.835 | **0.821** |
| FT | 0.861 | 0.771 | **0.850** | 0.808 |
| DI | 0.814 | 0.771 | 0.767 | 0.769 |

Table 2: Classification performance for 3 bucketing strategies on TWITTER.

### 4.5 Results

#### 4.5.1 Bucketing Strategy

Recall that we explore 3 different methods to group posts in order to process them in batches (Section 3.4). Here we evaluate them on rumour classification accuracy over the validation set of TWITTER. Note that we do not use CM here (and hence no reinforcement learning is involved) — we simply use all posts of an event to perform rumour classification with RDM. In terms of metrics we use standard accuracy, precision, recall and F1 scores. Results are presented in Table 2.

We see FN produces the best performance, and so FN is used for all following experiments as the default bucketing strategy.[7] As certain events have a long delay between posts, we also incorporate a maximum delay of one hour before processing the posts in a batch.

#### 4.5.2 Detection Accuracy

In this section, we assess how accurately the models classify rumours. All baselines and benchmark systems uses all posts of an event to perform classification, with the exception of HMM which uses only the first 5 posts. For our models, we present: (1) the full model ERD, which uses a subset of posts for classification (checkpoint decided by CM); and (2) RDM, which uses the full set of posts. Results are detailed in Table 3 and 4.

We can see that RDM outperforms all models across most metrics, including state-of-the-art rumour detection systems CSI (marginally) and CRF (substantially). ERD, on the other hand, performs very competitively, outperforming most

[7]FN value: 5 posts for WEIBO and 2 posts for TWITTER.

| Method | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| Baseline | 0.724 | 0.673 | 0.746 | 0.707 |
| RNN | 0.873 | 0.816 | 0.964 | 0.884 |
| LSTM | 0.896 | 0.846 | **0.968** | 0.913 |
| GRU-1 | 0.908 | 0.871 | 0.958 | 0.913 |
| GRU-2 | 0.910 | 0.876 | 0.956 | 0.914 |
| CSI* | 0.953 | — | — | 0.954 |
| RDM | **0.957** | **0.950** | 0.963 | **0.957** |
| ERD | 0.933 | 0.929 | 0.936 | 0.932 |

Table 3: Detection accuracy on WEIBO. '*' denotes values taken from the original publications.

| Method | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| Baseline | 0.612 | 0.355 | 0.465 | 0.398 |
| RNN | 0.785 | 0.707 | 0.659 | 0.682 |
| LSTM | 0.796 | 0.719 | 0.683 | 0.701 |
| GRU-1 | 0.800 | 0.735 | 0.685 | 0.709 |
| GRU-2 | 0.808 | 0.741 | 0.694 | 0.717 |
| CRF* | — | 0.667 | 0.566 | 0.607 |
| HMM* | — | — | — | 0.524 |
| RDM | **0.873** | 0.817 | **0.823** | **0.820** |
| ERD | 0.858 | **0.843** | 0.735 | 0.785 |

Table 4: Detection accuracy on TWITTER. '*' denotes values taken from the original publications.

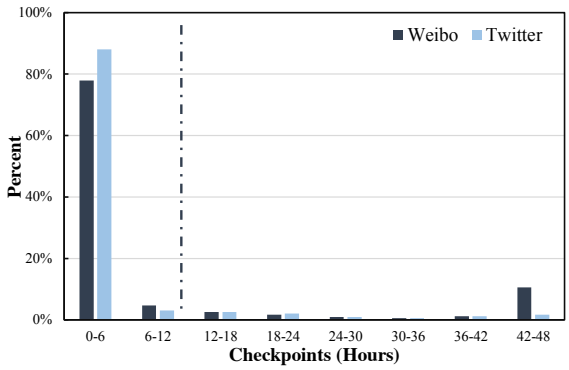


Figure 6: Proportion of events classified by ERD over time. Dashed line indicates the optimal checkpoint (12 hours) for GRU-2.

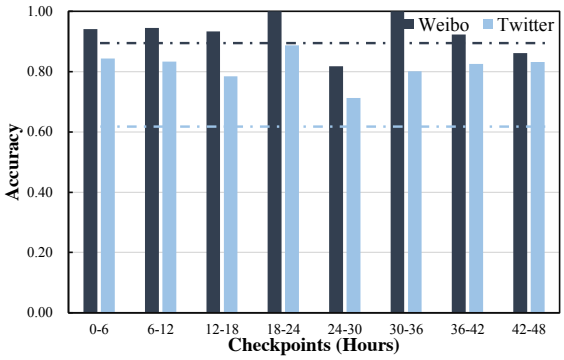


Figure 7: Detection accuracy of ERD over time. Dashed lines indicates GRU-2's accuracies.

benchmark systems and baselines, with the exception of CSI on WEIBO. Note, however, that unlike most other systems, ERD leverages only a subset of posts for rumour classification. HMM is the only benchmark system on TWITTER that uses a subset (first 5), and its performance is markedly worse than that of ERD (which uses 4.03 posts on average).

#### 4.5.3 Detection Timeliness

Next we evaluate the timeliness of the detection, and we focus on comparing our system with GRU-2 (Ma et al., 2016), as it performed competitively in Section 4.5.2. Note that GRU-2 uses a manually set checkpoint (12 hours) that were found to be optimal, while ERD determines the checkpoint dynamically.

We first present the proportion of events that are classified by ERD over time (6-hour interval) in Figure 6.[8] We see that for both WEIBO and TWITTER, the majority of the events (approximately 80%) are classified within the first 6 hours. GRU-2's optimal checkpoint is 12 hours (dashed line), and so ERD is detecting rumours much earlier than GRU-2.

We next present the classification accuracy of these events over time (again, in 6-hour interval) in Figure 7. ERD generally outperforms GRU-2 (dashed lines) over all checkpoints. To be fair, checkpoints that are longer than 12 hours are not exactly comparable, as ERD uses more posts than GRU-2 in these instances. But even if we consider only the first 2 intervals (0-6 and 6-12 hours), ERD still outperforms GRU-2 across both data sets, demonstrating that ERD detects rumours earlier and more accurately.

For the two checkpoints on WEIBO where GRU-2 outperforms ERD, in the first checkpoint (24-30) we find that there are only 5 events and so the difference is unlikely to be statistically robust. For the second checkpoint (42-48), we hypothesise that these events are possibly the diffi-

[8] We include all events (whether it is a true of false positive) that CM decides to checkpoint.

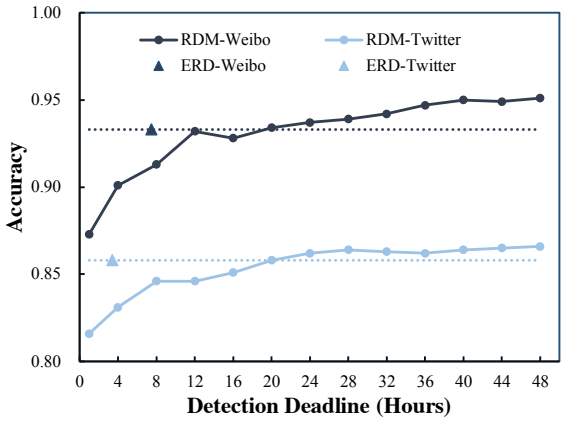


Figure 8: Detection accuracies of ERD and RDM over time.

| Interval | Salient Words | Translation |
|---|---|---|
| 18:41 – 18:44 | 大闸蟹，毒性，激素，有害，吃惊 | hairy crabs, toxicity, hormone, harmful, amazed |
| 18:48 – 18:51 | 大闸蟹，爆出，消息，吃惊，上市 | hairy crabs, bursts, message, amazed, on the market |
| 18:51 – 18:59 | 美食，为何，这样，晕，同城会 | delicious food, why, so, dizzy, one city club |
| 18:59 – 19:09 | 敢吃吗，吃得起，喜欢，惨，偷笑 | dare to eat, afford to eat, like, miserable, laughing |
| 19:11 – 19:15 | 食品安全，真的吗，失望，神马，不能 | food safety, really, disappointment, what, cannot |
| **Rumour Detected** | | |
| 19:34 – 19:49 | 是不是，大闸蟹，吃不成，疑问，围观 | is it, hairy crabs, cannot eat, doubt, look around |

Table 5: Case study of a rumour on WEIBO.

cult cases, and as such the classification decision is deferred until much later (and classification performance is ultimately still low due to its difficulty).

To understand the advantage of incorporating reinforcement learning (CM) for rumour detection, we compute the detection accuracy over time for ERD and RDM in Figure 8. The dashed lines indicate the average accuracy performance of ERD, which detects rumours on average in 7.5 and 3.4 hours on WEIBO and TWITTER respectively. The solid lines show the accuracy performance of RDM, which increases over time as it has more evidence. For RDM to achieve the performance of ERD, we see that it requires approximately at least 20 hours of posts on both data sets. This highlights the importance of the checkpoint module, which allows ERD to detect rumours much earlier. In certain events, they are detected within 3 minutes.

#### 4.5.4 Case Study

To provide a qualitative analysis for our approach, we showcase an example of a rumour event from WEIBO in Table 5. We present a set of salient words (second column) and their translations (third column) extracted from posts published during a particular period (first column) using simple tf-idf features.

The rumour was started by a message claiming that hairy crabs contain harmful hormones and toxins on August 18th, 2012. After the message was posted, within 12 hours 2.3M users participated in its propagation, either by re-posting, commenting, or questioning the original source message. The rumour spread quickly and led to significant economic damage to the aquaculture industry in China. Officially the rumour was rebutted after 24 hours, but in Table 5 we see that ERD detects the rumour in 34 minutes.

## 5 Conclusions

We present ERD, an early rumour detection system. Rather than setting a static checkpoint that determines when an event should be classified as rumour, ERD learns dynamically the minimum number of posts required to identify a rumour. To this end, we integrate reinforcement learning with recurrent neural networks to monitor social media posts in real time to decide when to classify rumours. We evaluate our model on two standard data sets, and demonstrate that ERD identifies rumours within 7.5 hours and 3.4 hours on WEIBO and TWITTER on average, compared to 12 hours of a competitive system. In terms of detection accuracy, ERD achieves a performance of 93.3% and 85.8%, which is comparable to state-of-the-art rumour detection systems.

## Acknowledgements

This work is partially funded by the National Natural Science Foundation of China (61502115, U1636103, U1536207). We would also like to thank Wei Gao and Jing Li for their valuable suggestions.

## References

Gordon W Allport and Leo Postman. 1947. The psychology of rumor.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python — An-*

*alyzing Text with the Natural Language Toolkit*. O'Reilly Media, Sebastopol, USA.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. 12:2493–2537.

Sebastian Dungs, Ahmet Aker, Norbert Fuhr, and Kalina Bontcheva. 2018. Can rumour stance alone predict veracity? In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3360–3370.

Mehrdad Farajtabar, Jiachen Yang, Xiaojing Ye, Huan Xu, Rakshit Trivedi, Elias Khalil, Shuang Li, Le Song, and Hongyuan Zha. 2017. Fake news mitigation via point process based intervention. In *International Conference on Machine Learning*, pages 1097–1106.

Adrien Friggeri, Lada A Adamic, Dean Eckles, and Justin Cheng. 2014. Rumor cascades. In *ICWSM*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Sejeong Kwon, Meeyoung Cha, and Kyomin Jung. 2017. Rumor detection over varying time windows. *PloS one*, 12(1):e0168344.

Jey Han Lau, Timothy Baldwin, and Trevor Cohn. 2017. Topically driven neural language model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 355–365.

Gang Liang, Wenbo He, Chun Xu, Liangyin Chen, and Jinquan Zeng. 2015. Rumor identification in microblogging systems based on users' behavior. *IEEE Transactions on Computational Social Systems*, 2(3):99–108.

Yunfei Long, Qin Lu, Rong Xiang, Minglei Li, and Chu-Ren Huang. 2017. Fake news detection through multi-perspective speaker profiles. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 252–256.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *IJCAI*, pages 3818–3824.

Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1751–1754. ACM.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 708–717.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Warren A Peterson and Noel P Gist. 1951. Rumor and public opinion. *American Journal of Sociology*, 57:159–167.

Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1589–1599. Association for Computational Linguistics.

Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 797–806. ACM.

Justin Sampson, Fred Morstatter, Liang Wu, and Huan Liu. 2016. Leveraging the implicit structure within social media for emergent rumor detection. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2377–2382. ACM.

Richard S Sutton, Andrew G Barto, Francis Bach, et al. 1998. *Reinforcement learning: An introduction*. MIT press.

Tetsuro Takahashi and Nobuyuki Igata. 2012. Rumor detection on twitter. In *Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS), 2012 Joint 6th International Conference on*, pages 452–457. IEEE.

Ke Wu, Song Yang, and Kenny Q Zhu. 2015. False rumors detection on sina weibo by propagation structures. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 651–662. IEEE.

Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. 2012. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 13. ACM.

Zhifan Yang, Chao Wang, Fan Zhang, Ying Zhang, and Haiwei Zhang. 2015. Emerging rumor identification for social media with hot topic detection. In *Web Information System and Application Conference (WISA), 2015 12th*, pages 53–58. IEEE.

Qiao Zhang, Shuiyuan Zhang, Jian Dong, Jinhua Xiong, and Xueqi Cheng. 2015. Automatic detection of rumor on social network. In *Natural Language Processing and Chinese Computing*, pages 113–122. Springer.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1395–1405. International World Wide Web Conferences Steering Committee.

Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. 2018. Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)*, 51(2):32.

Arkaitz Zubiaga, Maria Liakata, and Rob Procter. 2016. Learning reporting dynamics during breaking news for rumour detection in social media. *arXiv preprint arXiv:1610.07363*.