Enquiring Minds: Early Detection of Rumors in Social Media from Enquiry Posts

Zhe Zhao
Department of EECS
University of Michigan
zhezhao@umich.edu

Paul Resnick
School of Information
University of Michigan
presnick@umich.edu

Qiaozhu Mei
School of Information
University of Michigan
qmei@umich.edu

ABSTRACT

Many previous techniques identify trending topics in social media, even topics that are not pre-defined. We present a technique to identify trending rumors, which we define as topics that include disputed factual claims. Putting aside any attempt to assess whether the rumors are true or false, it is valuable to identify trending rumors as early as possible.

It is extremely difficult to accurately classify whether every individual post is or is not making a disputed factual claim. We are able to identify trending rumors by recasting the problem as finding entire clusters of posts whose topic is a disputed factual claim.

The key insight is that when there is a rumor, even though *most* posts do not raise questions about it, there may be a *few* that do. If we can find signature text phrases that are used by a few people to express skepticism about factual claims and are rarely used to express anything else, we can use those as detectors for rumor clusters. Indeed, we have found a few phrases that seem to be used exactly that way, including: “Is this true?”, “Really?”, and “What?”. Relatively few posts related to any particular rumor use any of these enquiry phrases, but lots of rumor diffusion processes have **some** posts that do and have them quite **early** in the diffusion.

We have developed a technique based on searching for the enquiry phrases, clustering similar posts together, and then collecting related posts that do not contain these simple phrases. We then rank the clusters by their likelihood of really containing a disputed factual claim. The detector, which searches for the very rare but very informative phrases, combined with clustering and a classifier on the clusters, yields surprisingly good performance. On a typical day of Twitter, about a third of the top 50 clusters were judged to be rumors, a high enough precision that human analysts might be willing to sift through them.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Text Mining

General Terms

Experimentation; Empirical Studies

WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741637.

Keywords

Rumor Detection; Enquiry Tweets; Social Media

1. INTRODUCTION

On April 15th of 2013, two explosions at the Boston Marathon finish line shocked the entire United States. The event dominated news channels for the next several days, and there were millions of tweets about it. Many of the tweets contained rumors and misinformation, including fake stories, hoaxes, and conspiracy theories.

Within a couple of days, multiple pieces of misinformation that went viral on social media were identified by professional analysts and debunked by the mainstream media.[1] These reports typically appeared several hours to a few days after the rumor became popular and only the most widely spread rumors attracted the attention of the mainstream media.

Beyond the mainstream media, rumor debunking Websites such as Snopes.com and PolitiFact.org check the credibility of controversial statements.[2] Such Websites heavily rely on social media observers to nominate potential rumors which are then fact-checked by analysts employed by the sites. They are able to check rumors that are somewhat less popular than those covered by mainstream media, but still have limited coverage and even longer delays.

One week after the Boston bombing, the official Twitter account of the Associated Press (AP) was hacked. The hacked account sent out a tweet about two explosions in the White House and the President being injured. Even though the account was quickly suspended, this rumor spread to millions of users. In such a special context, the rumor raised an immediate panic, which resulted in a dramatic, though brief, crash of the stock market [10].

The broad success of online social media has created fertile soil for the emergence and fast spread of rumors. According to a report of the development of new media in China, rumors were detected in more than 1/3 of the trending events on microblog media in 2012.[3]

Rather than relying solely on human observers to identify trending rumors, it would be helpful to have an automated tool to identify potential rumors. The goal of such a tool would not be to assess

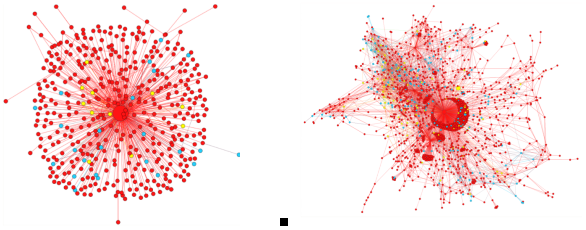
[1] Source: http://www.cnn.com/2013/04/16/tech/social-media/social-media-boston-fakes/, retrieved on March 7, 2015; and http://www.scpr.org/blogs/news/2013/04/16/13322/boston-marathon-bombings-rumor-control-man-on-the/, retrieved on March 7, 2015.

[2] Source: http://www.snopes.com/politics/conspiracy/boston.asp, retrieved on March 7, 2015.

[3] Ironically, this report was misinterpreted by a major news media source, which coined a new rumor that “more than 1/3 of trending topics on Weibo are rumors.” Source: http://truth.cntv.cn/erjiye/20/, retrieved on March 7, 2015.

the veracity of claims made in the rumors, merely to identify when claims were being spread that some people were questioning or disputing. If such a tool can identify rumors **earlier** and with sufficiently high precision, human analysts such as journalists might be willing to sift through all the top candidates to find those that were worth further investigation. They would then assess the veracity of the factual claims. Important rumors might be responded to earlier, limiting their damage. In addition, such a tool could help to develop a large collection of rumors. Previous research on rumor diffusion has included case studies of individual rumors that spread widely (e.g., [16]), but a fuller understanding of the nature of rumor diffusion will require study of much larger collections, including those that reach only modest audiences, so that commonalities and differences between diffusion patterns can be assessed.

We propose a new way to detect rumors **as early as possible** in their life cycle. The new method utilizes the enquiry behavior of social media users as sensors. The key insight is that some people who are exposed to a rumor, before deciding whether to believe it or not, will take a step of information enquiry to seek more information or to express skepticism without asserting specifically that it is false. Some of them will make their enquiries by tweeting. For example, within 60 seconds after the hacked AP account sent out the rumor about explosions in the White House, there were already multiple users enquiring about the truth of the rumor (Figure 1). Table 1 shows some examples of these enquiry tweets.



(a) 60 seconds after the hacked twitter account sent out the White House rumor there were already sufficient enquiry tweets (blue nodes).

(b) Two seconds after the first denial from an AP employee and two minutes before the official denial from AP, the rumor had already gone viral.

Figure 1: Snapshots of the diffusion of the White House rumor. Red, yellow and blue nodes: Twitters spreading, correcting, and questioning the rumor.

Of course, not all tweets about a rumor will be such skeptical enquiries. As features for classifying *individual tweets*, enquiry signals are insufficient. Even if they yielded high precision, the recall would be far too low. As features for classifying *tweet clusters*, however, they provide surprisingly good coverage. Our technique for automatically detecting rumors is built around this signal.

Table 1: Examples of enquiry tweets about the rumor of explosions in the White House

Oh my god is this real? RT @AP: Breaking: Two Explosions in the White House and Barack Obama is injured
Is this true? Or hacked account? RT @AP Breaking: Two Explosions in the White House and Barack Obama is injured
Is this real or hacked? RT @AP: Breaking: Two Explosions in the White House and Barack Obama is injured
How does this happen? #hackers RT @user: RT @AP: Breaking: Two Explosions in the White House and Barack Obama is injured
Is this legit? RT @AP Breaking: Two Explosions in the White House and Barack Obama is injured

We make three contributions in this work. First, we develop an algorithm for identifying *newly emerging, controversial* topics that is scalable to massive stream of tweets. It is scalable because it clusters only *signal tweets* rather than all tweets, and then assigns the rest of the tweets only if they match one of the signal clusters. Second, we identify a set of regular expressions that define the set of signal tweets. This crude classifier of signal tweets based on regular expression matching turns out to be sufficient. Third, we identify features of signal clusters that are independent of any particular topic and that can be used to effectively rank the clusters by their likelihood of containing a disputed factual claim.[4]

The algorithm is evaluated using the Twitter Gardenhose (a 10% sample of the overall tweet stream) to identify rumors on regular uneventful days. We also evaluate it using a large collection of tweets related to the Boston Marathon bombing event. We compare the algorithm to various baselines. It is more scalable and has higher precision and recall than techniques that try to find all trending topics. It detects more rumors, and detects them much earlier than a related technique that treats only debunks or corrections as signals of rumors as well as techniques that rely on tracking all trending topics or popular memes. The performance is also satisfactory in an absolute sense. It successfully detects 110 rumors from the stream of tweets about the Boston Marathon bombing event, with an average precision above 50% among the top-ranked candidates. It also achieves a precision of 33% when outputting 50 candidate rumors per day from analysis of the Gardenhose data.

2. RELATED WORK

2.1 Detection Problems in Social Media

Although rumors have long been a hot subject in multiple disciplines (e.g., [9, 31, 22]), research on identifying rumors from online social media through computational methods has only begun in recent years. Our previous work has shown that particular known rumors can be retrieved with a high accuracy by training a machine learning classifier for each rumor [28]. Here we seek to identify new rumors, not necessarily retrieve all the tweets related to them.

Much previous research has tried to develop classifiers for a more challenging problem than ours, automatically determining whether a meme that is spreading is true or false ([35, 4, 14, 17]). Application domains have included "event rumors" in Sun et al. [33], and fake images on Twitter during Hurricane Sandy [16]. The "Truthy" system attempts a related classification problem, whether a spreading meme is spreading "organically" or whether it is being spread by an "astroturf" campaign controlled by a single person or organization [29, 30].

Identifying the truth value of an arbitrary statement is very difficult, probably as difficult as any natural language processing problems. Even if one knows what the truth is, the problem is related to textual entailment (recognizing whether the meaning of one given statement can be inferred from another given statement), the accuracy of the art of which is lower than 70% on balanced lab data sets [8]. This is even harder for short posts in social media.

Thus, most existing approaches that attempt to classify the truthfulness of spreading memes utilize information beyond the content of the posts, usually by analyzing the collective behavior of how users respond to the target post. For example, many studies identified the popularity of a post (e.g., number of posts that retweeted

[4] At the risk of redundancy, we emphasize that our technique does not make any attempt to assess whether rumors are true or not, or classify or rank them based on the probability that they are true. We rank the clusters based on the probability that they contain a *disputed* claim, not that they contain a *false* claim.

or replied to the post) as a significant signal. This information is used either directly as features of the "rumor" classifier (e.g., [4, 14, 35, 17, 33, 34]), or as filters to prescreen candidate topics (e.g., to only consider the most popular posts [15] or "trending topics" [4, 14]), or both [4, 14]. Other work identified burstiness [34], temporal patterns [17, 15], or the network structure of the diffusion of a post/topic [30, 4, 32, 17] as important signals.

Most of these features of the tweet collection can only be collected after the rumor has circulated for a while. In other words, these features only become meaningful when the rumor has already reached and been responded to by many users. Once these features become available, we also make use of them in our classifier that ranks candidate rumor clusters. However, since we have set ourselves the easier task of detecting controversial fact-checkable claims, rather than detecting false claims, we are able to rely for initial detection on content features that are available much earlier in a meme's diffusion.

Some existing work uses corrections made by authoritative sources or social media users as a signal. For example, Takahashi and Igata tracked the clue keyword "false rumor" [34]. Both Kwon et al. [17] and Friggeri et al. [13] tracked the judgments made by rumor debunking websites such as Snopes.com. Studies of rumors on Weibo.com also tracked official corrections made by the site [35, 33]. These correction signals are closer in spirit to those we employ. They suffer, however, from limited coverage and delays, only working after a rumor has attracted the attention of authoritative sources. In our experiments we will compare the recall and earliness of rumor detection through our system using both correction and enquiry signals to a more limited version of our system that uses only correction signals.

Another related problem is detecting and tracking trending topics [20] or popular memes [18]. Even if they are effective at picking up newly popular topics, they are not sufficiently precise to serve as trending rumor detectors, as most topics and memes in social media are not rumors. As an example, Sun et al. collected 104 rumors and over 26,000 non-rumor posts in their experiment [33]. Later in this paper, we will compare the precision of the candidate rumors filtered using our method and those filtered through trending topics and meme tracking.

2.2 Question Asking in Social Media

Another detection feature used in related work is question asking. Mendoza et al. found on a small set of cases that false tweets were questioned much more than confirmed truths [21]. Castillo et al. therefore used the number (and ratio) of question marks as a feature to classify the credibility of a group of tweets. The same feature is adopted by a few follow-up studies [14, 16].

In fact, the behavior of information seeking by asking questions on online social media has drawn interest from researchers in both social sciences and computer science (e.g., [6, 24, 25, 37]). Paul et al. analyzed a random sample of 4,140 tweets with question marks [25]. Among the set of tweets, 1,351 were labeled as questions by Amazon Mechanical Turkers. Morris et al. conducted surveys on if and how people ask questions through social networks. They analyzed the survey responses and presented findings such as how differently users ask questions via social media and via search engines, and how different cultures influence the behaviors [24, 23, 36]. These studies have proved that question asking is a common behavior in social media and provided general understanding of the types of questions people ask.

To study question asking behavior at scale, our previous work detected and analyzed questions from billions of tweets [37]. The analysis pointed out that the questions asked by Twitter users are tied to real world events including rumors. These findings inspired us to make use of question asking behavior as the signal for detecting rumors once they emerge.

Though inspired by the value of question marks as features for classifying the truth value of a post, for our purposes we need a more specific signal. Previous work has shown that only one third of tweets with question marks are real questions, and not all questions are related to rumors [25, 37]. In this paper, we carefully select a set of regular expressions to identify enquiry tweets that are indicative of rumors.

3. PROBLEM DEFINITION

3.1 Defining a Rumor

Many variations of the definition of rumors have been proposed in the literature of sociology and communication studies [26]. These different definitions generally share a few insights about the nature of rumors. First, rumors usually arise in the context of ambiguity, and therefore the truth value of a rumor appears to be uncertain to its audience. Second, although the truth value is uncertain, a rumor does not necessarily imply *false* information. Instead, the term "false rumor" is usually used in these definitions to refer to rumors that are eventually found to be false. Indeed, many pieces of *truthful* information spread as rumors because most people don't have first-hand knowledge to assess them and no trusted authorities have fact-checked them yet. Having such intuitions and following the famous work of DiFonzo and Bordia in social psychology [9], we propose a practical definition:

"A rumor is a controversial and fact-checkable statement."

We make the following remarks to further clarify this definition:

- "Fact-checkable": In principle, the statement has a truth value that could be determined right now by an observer who had access to all relevant evidence. This excludes statements that cannot be fact-checked or those whose truth value will only be determined by future events (e.g., "Chelsea Clinton will run for president in 2040.").
- "Controversial (or Disputed)": At some point in the life cycle of the statement, some people express skepticism (e.g., verifications, corrections, statements of disbelief or questions). This excludes statements that are fact-checkable but not disputed (e.g., "Bill Clinton tried marijuana," as Clinton himself has admitted it.).
- Any statement referring to a statement meeting the criteria above is also classified as a rumor. This includes statements that point to several other rumors (e.g., "Click the link `http://...` to see the latest rumors about Boston Bombing.").

The above definition of rumor is effective in practice. As we describe below, human raters were able to achieve high inter-rater reliability labeling statements as rumors or not.

3.2 The Computational Problem

Based on the conceptual definition, we can formally define the computational problem of real-time detection of rumors.

DEFINITION 1. *(Rumor Cluster). We define a rumor cluster R as a group of social media posts that are either declaring, questioning, or denying the same fact claim, s, which may be true or false. Let S be the set of posts declaring s, E be the set of posts questioning s, and C be the set of tweets denying s, then $R = S \cup E \cup C$. We say s is a candidate rumor if $S \neq \emptyset$ and $E \cup C \neq \emptyset$.*

Naturally, posts belonging to the same rumor cluster can either be identical to each other (e.g., retweets) or paraphrase the same fact claim. Posts that are enquiring about the truth value of the fact claim are referred to as *enquiry posts* (E) and those that deny the fact claim are referred to as *correction posts* (C).

DEFINITION 2. *(Real-time Rumor Detection). Consider the input of a stream of posts in social media,* $\mathcal{D} = \langle (d_1, t_1), (d_2, t_2) \ldots \rangle$*, where* $d_i, i \in [1, 2, \cdots]$ *is a document posted at time* t_i*. The task of real-time rumor detection is to output a set of clusters* $\mathcal{R}_t = \langle R_{t,1}, R_{t,2}, \ldots, R_{t,l} \rangle$ *at time t after every time interval* Δt*, where the fact claim* $s_{t,j}$ *of each cluster* $R_{t,j} \in \mathcal{R}_t$ *is a candidate rumor.*

Given any time point t where a new set of clusters are output, the clusters must satisfy that

$$\forall R_{t,j} \in \mathcal{R}_t, \exists (d', t') \in R_{t,j} \; s.t. \; t - \Delta t < t' \leq t$$

This means that the output rumor clusters at time t must contain at least one tweet posted in the past time interval Δt. Clearly, a cluster about a fact claim s can accumulate more documents over time, such that $R_{t_1,j} \subseteq R_{t_2,j}$ if $t_1 < t_2$ and $s_{t_1,j} = s_{t_2,j} = s$. Therefore, we can naturally define the first time (t_1 in the previous example) where a rumor cluster about a fact claim s is output as the *detection time* of the candidate rumor s. Our aim is to minimize the delay from the time when the first tweet about the rumor is posted to the detection time.

4. EARLY DETECTION OF RUMORS

We propose a real-time rumor detection procedure that has the following five steps.

1. **Identify Signal Tweets.** Using a set of regular expressions, the system selects only those tweets that contain skeptical enquiries: verification questions and corrections. These are the signal tweets.
2. **Identify Signal Clusters.** The system clusters the signal tweets based on overlapping content in the tweets.
3. **Detect Statements.** The system analyzes the content of each signal cluster to determine a single statement that defines the common text of the cluster.
4. **Capture Non-signal Tweets.** The system captures all non-signal tweets that match any cluster's summary statement, turning a signal cluster into a full candidate rumor cluster.
5. **Rank Candidate Rumor Clusters.** Using statistical features of the clusters that are independent of the statements' content, rank the candidate clusters in order of likelihood that their statements are rumors (i.e., controversial and fact-checkable).

The algorithm operates on a real-time tweet stream, where tweets arrive continuously. It outputs a ranked list of candidate rumor clusters at every time interval Δt, where Δt could be as small as the interval when the next tweet arrives. In practice, it will be easier to think of the time interval as, for example, an hour or a day, with many tweets arriving during that interval.

The system first matches every new tweet posted in that interval to rumor clusters detected in the past, using the same method of capturing non-signal tweets (component 4). Tweets that do not match to any existing rumors will go through the complete set of five components listed above, with a procedure described in Figure 2. If very short time intervals are used, signal tweets from recent past intervals that were not matched to any rumor clusters may also be included in this procedure. Below, each of the five steps are described in more detail.

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

Table 2: Patterns used to filter Enquiries and Corrections

4.1 Identify Signal Tweets

The first module of our algorithm extracts enquiry tweets. Not all enquiries are related to rumors [37]. A tweet conveying an information need can be either of the following cases:

- It requests a piece of factual knowledge, or a verification of a piece of factual knowledge. Factual knowledge is objective and fact-checkable. For example: "According to the Mayan Calendar, does the world end on Dec 16th, 2013?"
- It requests an opinion, idea, preference, recommendation, or personal plan of the recipient(s), as well as a confirmation of such information. This type of information is subjective and not fact-checkable.

We hypothesize that only verification/confirmation questions are good signals for rumors. In addition, we expect that corrections (or debunks) are also good signals. To extract patterns to identify these good signals, we conducted an analysis on a labeled rumor dataset.

Discover patterns of signal tweets.

We analyzed 10,417 tweets related to five rumors published in [28], with 3,423 tweets labeled as either verifications or corrections. All tweets are lowercased and processed with the Porter Stemmer [27]. We extracted lexical features from the tweets: unigrams, bigrams and trigrams. Then we calculated the Chi-Square score for each feature in the data set. Chi-Squared test is a classical statistical test of independence and the score measures the divergence from the expected distribution if one assumes a feature is independent of the class label [12]. Features with high Chi-Square scores are more likely to appear only in tweets of a particular class. Patterns which appear excessively in verification and correction tweets but are underrepresented in other tweets were selected to detect signal tweets. From the patterns with high Chi-Square scores, human experts further selected those which are independent of any particular rumor. The patterns we selected are listed in Table 2.

As a way to identify all the tweets containing rumors, this set of regular expressions has relatively low recall. Even on the 3,423 tweets labeled as either verifications or corrections in our training data, only 572 match these regular expressions. In the signal tweet identification stage, however, it is far more important to have a high precision. Low recall of *signal tweets* may still be sufficient to get high recall of *signal clusters*. By identifying patterns that are more likely to appear only in signal clusters, even though these patterns only appear a few times inside each rumor cluster, our framework can make use of them to detect many rumors. Note that although current patterns are discovered from a data set containing only five rumors, we could in principle rerun this process after we have more rumors labeled by our rumor detection framework, shown as the dotted line in Figure 2.

4.2 Identify Signal Clusters

When a tweet containing a rumor emerges, many people either explicitly retweet it, or create a new tweet containing much of the

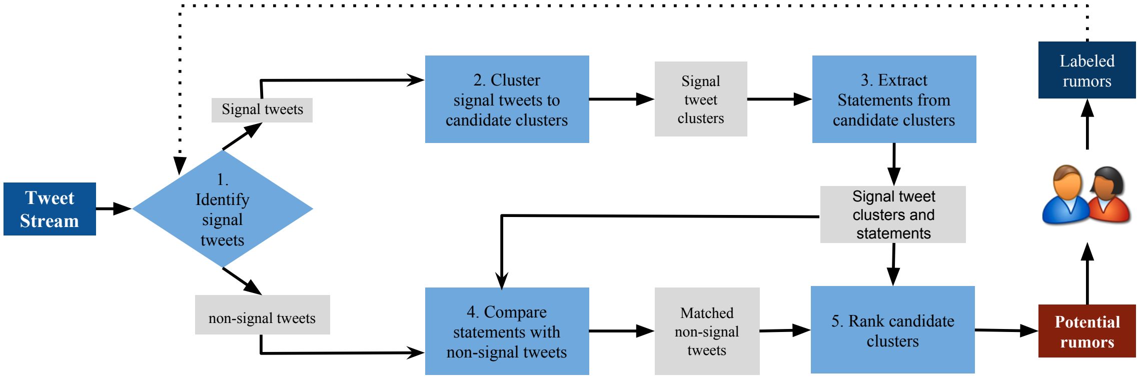

Tweet Stream
1. Identify signal tweets
Signal tweets
non-signal tweets
2. Cluster signal tweets to candidate clusters
Signal tweet clusters
3. Extract Statements from candidate clusters
Signal tweet clusters and statements
4. Compare statements with non-signal tweets
Matched non-signal tweets
5. Rank candidate clusters
Potential rumors
Labeled rumors

Figure 2: The procedure of real-time rumor detection.

original text. Therefore, tweets spreading a rumor are mostly near duplicates, as illustrated in Table 1. By clustering, we aim to group all the near duplicates, which are either retweets or tweets containing the original rumor content.

There are many different clustering algorithms such as the K-Means [19]. Many have a high computational cost and/or need to keep an $N \times N$ similarity matrix in memory. Given that we expect tweets about the same rumor to share a lot of text, we can trade off some accuracy for efficiency. In contrast to exploratory clustering tasks where documents may be merely similar, we want to cluster tweets that are near duplicates. Therefore, using an algorithm such as connected component clustering can be efficient and effective enough for our purposes. A connected component in an undirected graph is a group of vertices, every pair of which are reachable from each other through paths. An undirected graph of tweets is built by including an edge joining any tweet pair with a high similarity.

We use the Jaccard coefficient to measure similarity between tweets. Given two tweets d_a and d_b, the similarity between d_a and d_b can be calculated as:

$$J(d_a, d_b) = \frac{|Ngram(d_a) \cap Ngram(d_b)|}{|Ngram(d_a) \cup Ngram(d_b)|}$$

Here $Ngram(d_a)$ and $Ngram(d_b)$ are the 3-grams of tweets d_a and d_b. Jaccard distance is a commonly used indicator of the similarity between two sets. The similarity values from 0 to 1 and a higher value means a higher similarity.

To further improve efficiency, we use the Minhash algorithm [2] to reduce the dimensionality of the Ngram vector space, which makes calculating Jaccard similarity much faster. The Minhash algorithm is used for dimensionality reduction and fast estimation of Jaccard similarities. In our approach, we randomly generate 50 hash functions based on the md5 hash function. Then we use the 50 corresponding Minhash values to represent each tweet. In our implementation of the connected component clustering algorithm, we set the threshold for adding an edge at 0.6 (60% of the hashed dimensions for the two tweets are equal).

The connected components in this graph are the clusters. We create a cluster for each group of three or more tweets connected together. Connected components can be found by either breadth-first search or depth-first search, which has a linear time complexity $O(E)$, where E is the number of edges. Since we want to cluster tweets that are near duplicates, setting a high similarity threshold (0.6) yields a relatively small number of edges.

At this point, the procedure will have obtained a set of candidate rumor clusters $\mathcal{R}$. The next stage extracts, for each cluster R_i, the statement s_i that the tweets in the cluster promote, question, or attempt to correct. In our approach, for each rumor cluster, we extract the most frequent and continuous substrings (3-grams that appear in more than 80% of the tweets) and output them in order as the summarized statement. We keep the summarization component simple and efficient in this study, though algorithms such as LexRank [11] may improve the performance of text summarization.

4.3 Capture Non-signal Tweets

After the statement that summarizes the tweets in a signal cluster is extracted, we use that statement as a query to match similar non-signal tweets from the tweet stream, tweets that are related to the cluster but do not contain enquiry patterns. To be consistent, we still use the Jaccard similarity and select tweets whose similarity score with the statement is higher than a threshold (0.6 in our implementation). This step partially recovers from the low recall of signal tweet detection using limited signal patterns.

Note the efficiency gain that comes from matching the non-signal tweets only with the statements summarizing signal tweets. In particular, it is not necessary to compare each of the non-signal tweets with each other non-signal tweet. Non-signal tweets may form other clusters but we do not need to detect those clusters: they do not contain nuclei of three connected signal tweets and thus are unlikely to be rumor clusters.

4.4 Score Candidate Rumor Clusters

After we have complete candidate rumor clusters, including both signal and non-signal tweets, we score them. A simple way to output clusters is to rank them by popularity. The number of tweets in a cluster measures the statement's popularity. The most popular candidates, however, may not be the most likely to be rumors. There may be statistical properties of the candidate rumor clusters that are better correlated with whether they really contain disputed factual claims.

We extracted 13 statistical features of candidate clusters that are independent of any particular substantive content. We then trained classifiers using these features, to obtain a better ranking function. The features are listed as follows.

- Percentage of signal tweets (1 feature): the ratio of signal tweets to all tweets in the cluster.

- Entropy ratio (1 feature): the ratio of the entropy of the word frequency distribution in the set of signal tweets to that in the set of all tweets in the cluster.

- Tweet lengths (3 features): (1) the average number of words per signal tweet; (2) the average number of words per any tweet in the cluster; and (3) the ratio of (1) to (2).
- Retweets (2 features): the percentage of retweets among the signal tweets and the percentage of retweets among all tweets in the cluster.
- URLs (2 features): the average number of URLs per signal tweet and the average number per any tweet in the cluster.
- Hashtags (2 features): the average number of hashtags per signal tweet and the number per any tweet in the cluster.
- @Mentions (2 features): the average number of usernames mentioned per signal tweet and the number per any tweet in the cluster.

We use rumor clusters labeled by human annotators to train a classifier that ranks the candidate clusters by their likelihood of being rumors (detail described in section 5.3). We select two commonly used classifiers, the Support Vector Machine (SVM [7]) with the LIBSVM implementation [5], and the Decision Trees [1] with the Matlab implementation [5]. The detailed results are shown in Section 5.

5. EXPERIMENT SETUP

In this section, we present empirical experiments to evaluate the proposed method of early detection of rumors.

5.1 Data Sets

We first selected two different collections of tweets. One focuses on a specific high-profile event, i.e. the Boston Marathon bombing in April 2013. The other consists of a random sample of tweets from a month that was not unusually eventful.

BOSTON MARATHON BOMBING (BOSTON). Two bombs exploded at the finish line of the annual Boston Marathon competition on April 15th, 2013.[6] We chose this context as a typical example of unpredictable real-world events.

To obtain a complete set of tweets related to this event, we collected tweets containing keywords such as "Boston," "marathon," and "explosion" and their combinations, starting several hours after the explosion, using the official tracking API provided by Twitter. The tracking API returned all tweets containing those keywords after 23:29 GMT. To collect tweets posted before this time point, we used the Twitter search API to get tweets containing the same set of keywords. In summary, we collected 10,240,066 tweets through the search API (13:30 GMT, April 14 to 23:29 GMT, April 15) and 23,001,329 tweets through the tracking API (23:29 GMT, April 15 to May 10, 2013), adding up to 30,340,218 unique tweets in the entire data set.

GARDENHOSE. Besides the stream related to a major event, we are also interested in the performance of the proposed method in detecting rumors from everyday tweets. We thus collected a tweet stream in a random month of the year 2013 (November 1 to November 30, 2013), through the official stream API with Gardenhose access (10% sample of the real-time stream of all tweets). This data set contains 1,242,186,946 tweets. Although the size is forty times larger than the BOSTON set, we anticipated that the density of rumors in this everyday tweet stream may be lower.

To process such data sets of over a billion records, we implemented our methods in MapReduce and conducted the experiments on a 72 core Hadoop cluster (version 0.20.2). The main components of our framework, including filtering, clustering and retrieval algorithm are implemented using Apache Pig (version 0.11.1).

5.2 Baselines and Variants of Methods

To obtain a comprehensive understanding of the effectiveness of the overall method, the identifiers of signal tweets, and the algorithms used to rank statements, we tested six variants of the method. The first four variants all rank candidate rumors purely by popularity, the number of tweets in the cluster. They vary in the algorithm used to identify signal tweets. The last three variants all use both enquiry and correction tweets as signals. They vary in the algorithm used to rank the candidate rumor clusters.

Variant 1 (baseline 1): Trending Topics. This straightforward baseline method directly clusters all the input tweets. It treats all the tweets as signal tweets rather than selecting only a subset. This method echoes the common approaches to detecting trending topics and then identifying rumors among them [4, 14]. After tweets are clustered and statements are extracted, this baseline method simply outputs the top candidate clusters with the largest number of tweets.

Variant 2 (baseline 2): Hashtag Tracking. Hashtags are well recognized signals for detecting trending and popular topics and have been used previously in rumor detection [30, 34]. As a second baseline method, popular hashtags (i.e., those that appear more than 10 times) are used to filter the signal tweets. Tweets containing these hashtags are clustered and statements are extracted from these clusters. Again, clusters with the largest number of tweets are presented to the user.

Variant 3 (baseline 3): Corrections Only. One novel contribution of our approach is the utilization of enquiry tweets as early signals of rumors. To test the performance of these signals, we downgraded our identifier of signal tweets by only using correction tweets as filtering signals, i.e., tweets containing the correction patterns in Table 2, including "rumor," "debunk," or "(this|that|it) is not true." Certain correction patterns such as the phrase "false rumor" have been used previously in the literature to identify rumors [34]. The same clustering and statement detection procedures were applied to tweets filtered with the correction signals. The largest candidate rumor clusters are output by the system.

Variant 4: Enquiries and Corrections. Besides the baseline methods, we included three variants that treat both enquiries and corrections as signal tweets, using the complete set of patterns from Table 2. To enable comparison with the baselines, variant 4 still ranks the candidate rumor clusters purely by popularity.

Variant 5: SVM ranking. This version ranks the candidate rumor clusters based on their scores using the trained *SVM* classifier. Like Variant 4, it treats both enquiries and corrections as signal tweets.

Variant 6: Decision tree ranking. This version ranks the candidate rumor clusters based on their scores using the trained *Decision Tree* classifier. Like Variants 4 and 5, it treats both enquiries and corrections as signal tweets.

[5] `http://www.mathworks.com/help/stats/classificationtree-class.html`, retrieved on March 7, 2015.

[6] `http://en.wikipedia.org/wiki/Boston_Marathon_bombings`, retrieved on March 7, 2015.

5.3 Ground Truth

We recruited two human annotators to manually label candidate rumors (i.e. rumor clusters as defined in Section 3) as either a real rumor or not. The annotators made decisions based on the statement extracted from the cluster, actual tweets in the cluster, and other useful information about the statement through Web search. To train the annotators, we developed a codebook according to the definition of rumors discussed in Section 3 which includes both the definition and examples of rumor and non-rumor statements. After being trained, both annotators labeled all the top-ranked candidate rumor clusters extracted by either the *Popularity* method, the *Decision Tree* method, or the *Correction Signal* method, from the first week of the GARDENHOSE data set and the first two days and the eighth day of the BOSTON data set. At most 10 clusters per hour per method were annotated for the BOSTON data set and at most 50 clusters per day per method were annotated for the GARDENHOSE data set. These added up to 639 candidate rumor clusters. The inter-rater reliability was satisfactory, achieving a Cohen's Kappa score of 0.76. Such a high agreement also demonstrates the coherence of our definition of rumors. For the statements the two annotators did not agree on, an expert labeled them and broke the tie. Another 1,440 clusters generated by the first two baseline methods (*Trending Topics* and *Hashtags*) on the same 72 hours of the BOSTON data set were then labeled by one of the two annotators after they were well trained. It took an average about 80 hours for each annotator to finish the labeling task including training.

5.4 Evaluation Metrics

We selected several quantitative metrics to evaluate the effectiveness and efficiency of our proposed method. We calculated precision@N, which is the percentage of real rumors among the top N candidate rumor clusters output by the a method. Since it is not practical in general to manually label a complete data set with tens of millions of tweets and hundreds of thousands of clusters, we cannot directly evaluate the actual recall of a rumor detection method. However, the number of rumors each method returns can be an indirect way to understand whether one method can detect more rumors than another. Another important dimension of the effectiveness of a rumor detection system is how early a rumor can be detected. We calculated the detection time, the time when the algorithm was first able to identify it as a candidate rumor. Finally, we also evaluated the scalability of the proposed method by plotting the scale of the data against the running time of the algorithm.

6. EXPERIMENT RESULTS

6.1 Effectiveness of Enquiry Signals

We evaluated the effectiveness of rumor detection algorithms using different signals. We first compared the precision of the top-ranked candidate rumor clusters output by different methods. For a fair comparison, all methods ranked the candidate rumor clusters simply using the *popularity* (i.e. number of tweets in each cluster).

6.1.1 Precision of Candidate Rumor Clusters

We compared the precision of our proposed methods using both enquiry and correction signals with all three baseline methods on the BOSTON data set. Note that both the baselines 1 and 2 (*Trending Topics* and *Meme Tracking*) have to cluster a huge number of tweets, nearly all the incoming tweets of every time period, hence they cannot handle the scale of all tweets in the GARDENHOSE data set. Therefore, we compare the proposed methods with only Baseline 3 (*Correction Signals*) on the GARDENHOSE data set. These results are summarized in Table 3. Clearly, the use of both enquiry and correction signals typically detects more rumors than using no signal (trending topics) or using memes (meme tracking), and the top-ranked rumor clusters are much more precise. Using both enquiry and correction signals, our method detected 110 rumors from the stream of tweets related to the Boston Marathon bombing, with an average precision@10 above 50% (half of the top 10 candidate clusters output by the system are real rumors). On the stream of everyday tweets, this method detected 92 rumors from a random month of 2013, with the average precision@50 above 26% (one fourth of the top 50 clusters output by the system are real rumors).

Table 3: Precision of rumor detection using different signals. Candidate rumors ranked by popularity only. Maximum number of output rumor clusters: 10 per hour for BOSTON and 50 per day for GARDENHOSE.

Method	Data Set	Candidates Detected	Real Rumors	Precision
Trending Topics	BOSTON	720	71	0.099
Hashtag Tracking	BOSTON	720	35	0.049
Corrections only	BOSTON	109	52	0.466
Enquiries+ Corrections	BOSTON	194	**110**	**0.521**
Corrections only	GARDENHOSE	312	87	**0.279**
Enquiries+ Corrections	GARDENHOSE	350	**92**	0.263

Some interesting observations can be made from these results. Detecting trending topics or tracking popular memes can reveal some rumors, but they both suffer from a low precision among the candidate clusters (lower than 10%), and thus miss many rumors if the user can only check a certain number of candidates (i.e., 10 per hour or 50 per day). This is because both methods inevitably introduce many false positives, popular statements that are not disputed. Detection using correction signals only also misses half of the rumors in the Boston event, probably because the behavior of debunking rumors is less common than enquiries in social media, as it certainly requires more effort of the users.

Using correction signals achieves a high precision among detected candidates. This is not surprising as statements already explicitly corrected or referred in tweets as "rumors" are likely to in fact be disputed factual claims. Interestingly, using enquiry as well as correction signals yields a similar precision.

6.1.2 Earliness of Detection

One of the most important objectives of our study is to detect emerging rumors as early as possible so that interventions can be made in time. Correction signals may appear only in a later stage of a rumor's diffusion. If this is the case, detecting rumors using such signals may have less practical value, as the rumors may have already spread widely. To verify this and further understand the usefulness of enquiry signals, we measured the earliness of detection. We computed the difference between the time points at which the same rumor was first detected by different methods, assuming that the algorithms are run in batch mode to output results only once per hour. The results are summarized in Table 4.

We first compare the method which uses both enquiry and correction signals with Baseline 3 (correction signals only). Since different methods may yield different clustering results and/or statements for the same rumor, we manually matched the 52 rumors detected by *correction only* from the BOSTON data set with the 110

Table 4: Earliness of detection comparing to Enquiries+ Corrections: enquiry signals help to detect rumors hours earlier.

Method	Data Set	Rumors detected	Rumors matched	Average delay
Corrections only	BOSTON	52	46	+4.3h
Trending Topics	BOSTON	71	53	+3.6h
Hashtag Tracking	BOSTON	35	31	+2.8h

rumors detected by both *enquiries and corrections*. We obtained 46 rumors detected in the top 10 results per hour by both methods.

There are 27 rumors that *enquiries and corrections* detected at least one hour earlier than *correction only*. The two detected the rest of the 19 rumors in the same hour. The detection of a rumor using *enquiries and corrections* is on average 4.3 hours earlier.

For example, at 20:00 (GMT) April 15th the *enquiries and corrections* algorithm would have output the popular rumor that the police identified a Saudi national as the suspect. This was one hour earlier than people started to realize it was false and tweet corrections. For another widespread rumor about an 8-year-old girl who died in the explosion, *enquiries and corrections* identified it almost one day earlier than tracking correction signals only.

In theory, how early can rumors be detected through enquiry signals, if candidate rumors were output continuously rather than hourly? We marked the time points when the system captures **at least three** signal tweets. On average, a real-time system that tracks enquiry signals can *hypothetically* detect a rumor after its first appearance in 9.6 minutes. To collect at least three correction tweets, a method has to wait for 236.7 more minutes on average. Not all candidate rumor clusters are actually output by our algorithms, so precision would have to be sacrificed to detect all rumors that quickly.

We also compare *enquiries and corrections* with Baseline 1 (trending topics), and Baseline 2 (meme tracking), on the earliness of detection. For Baseline 1, we matched the 71 rumors detected by *trending topics* with 110 rumors detected by our method. We obtained 53 common rumors detected by both methods. On average these rumors were detected as trending topics 3.6 hours later than using enquiry+correction signals. For Baseline 2, we matched the 35 rumors detected by *meme tracking* with rumors detected by our method. 31 of them are matched. On average these rumors were detected as trending memes 2.8 hours later than using enquiry+correction signals. The earliness of our detection method compared to other methods passed paired-sample t-test at significance level of 0.01.

In brief, we see that the use of enquiry tweets as signals not only detects more rumors, but also detects them hours faster than tracking trending topics or popular memes. Tracking correction signals, although it yields high precision, is the latest among all methods.

6.2 Ranking Candidate Rumor Clusters

We assessed the benefits produced by ranking the candidate clusters, using the 13 statistical features described in the previous section. We tested the performance of ranking functions based on Support Vector Machines and Decision Trees compared with two other baseline methods. The first baseline ranks the clusters by the number of tweets inside each cluster, referred to as *Popularity*. The second baseline ranks the clusters based on the retweet ratio in the cluster of tweets, which was reported as an indicative feature of rumors [34]. The second baseline is referred to as *Retweet Ratio*.

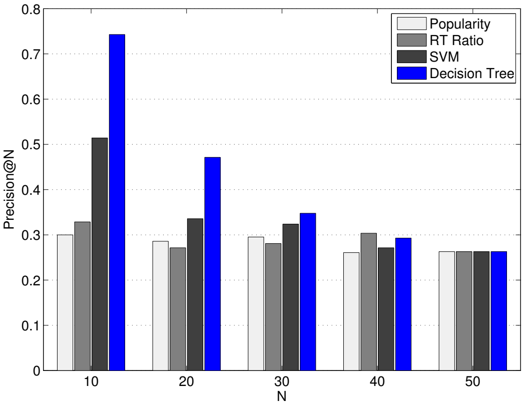


Figure 3: Precision@N of different ranking methods

We applied the ranking algorithms to the 350 candidate rumor clusters labeled by human annotators (the 50 most popular clusters for each of the seven days from 2013-11-1 to 2013-11-7 in GARDENHOSE data set). We used 6 days of labeled results to train the classifiers and the remaining day's results to test the algorithms. We did a 7-fold cross validation, holding out each of the seven days and computing the average performance. Figure 3 shows the results. We used Precision@N to evaluate different ranking algorithms. In this figure, we can see that *retweet ratio* has comparable performance to *popularity*, which remains about 0.3 no matter what N is. Our ranking algorithm significantly improves the Precision@N when N is small. The precision of the top 10 statements per day is above 0.7 using a Decision Tree, which outperforms SVM and the baseline methods. Of course, as N approaches 50, the precisions equalize since all the algorithms are essentially re-ranking the 50 most popular items. Note we are dealing with the classification task on only hundreds of examples and the 13 features we extracted are in different scales. In such case a decision tree is easier to tune than the more sophisticated SVM [3]; this may explain why the Decision Tree algorithm achieved a better performance than SVM. The improvement of Precision@10 and Precision@20 made by the decision tree compared to other methods passed the paired-sample t-test at significance level of 0.01.

Next, we tested whether the decision tree algorithm would find more rumors if it was able to suggest its own 50 top-ranked candidate clusters among all the candidates instead of reranking the most popular 50. In the previous figure, it was restricted to re-ranking the 50 most popular ones. We evaluated the performance using Precision@N. Figure 4 shows the results for the GARDENHOSE data set. We used tweets from 2013-11-1 to 2013-11-3 in the GARDENHOSE data set to train the decision tree and then used tweets from 2013-11-4 to 2013-11-7 to test, with popularity ranking as the baseline. Results show that we can not only improve Precision@N when N is small, but also find more rumors in 50 output statements. 33% of our output statements are rumors. The improvement of Precision@N when $N \leq 40$ made by our ranking algorithm passed the paired-sample t-test at significance level of 0.01.

In order to verify that the ranking algorithm is not overfitting only one data set, We also applied the decision tree trained using 7 days of labeled results in GARDENHOSE data set to rank rumor clusters detected hourly from BOSTON data set. We got similar results as in Figure 4. The average precision at 2, 4, 6, 8 an 10 in an hour is improved compared to *popularity* based ranking. The features used at the top levels of the decision tree include percentage of signal tweets and the average numbers of words, URLs and @mentions per any tweet in the cluster.

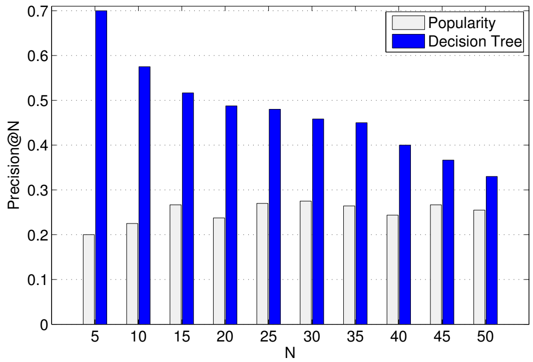


Figure 4: Precision@N if rumor clusters are ranked by the Decision Tree. One third of top 50 clusters are real rumors.

6.3 Efficiency of Our Framework

We have shown that our rumor detection algorithm is effective in detecting rumors in their early stage with reasonable precision. We now show that our framework is computationally efficient. Our framework first filters tweets with specific signals, then uses clustering to detect statements in this smaller group of tweets and at last outputs potential rumor statements. Compared to approaches that first generate trending topics and then identify rumors, we reduce the cost significantly in the detection process.

We first tested the time cost of our algorithm (*decision tree ranking* in Section 5.2 which uses both enquiry and correction signals) compared to baseline methods of running the algorithm on one batch of tweets from one time interval. We started from 1,000 tweet batches randomly sampled from tweets in our data set, then increased the number of tweets to 100,000,000 exponentially. Figure 5 shows the results. The x-axis shows the number of tweets to be processed in log scale. The baseline methods here are the Baseline 1 and 2 from Section 5.2, which try to detect trending topics or popular memes (hashtags) first. For baseline methods, we used the same clustering and ranking implementations as our method except they don't filter tweets with enquiry or correction signals and they don't have to retrieve tweets back after clustering. When the scale reaches one million tweets, Baseline 1 cannot finish in hours. Our method performs consistently and does not take much longer even at the 10 million scale: it can process 100 million tweets in about 80 minutes. It is intuitive that *Meme Tracking* achieves an intermediate performance. It clusters only those tweets that contain popular and trending hashtags, and thus scales somewhat better than clustering all tweets to find trending topics, but is still not as efficient as our method, which clusters a much smaller number of signal tweets.

We also tested our algorithm on one month's tweets from the GARDENHOSE data set, collected at November 2013. We set the time interval to be a day. The average number of tweets every day in the GARDENHOSE data set was about 40 million. As we would expect, experiment results indicate that the time cost does not increase significantly after processing several days, even with the accumulation of older rumor clusters. On average it took 28.77 minutes for our algorithm to finish detecting rumors each day and took 14.38 hours in total to process the 1.2 billion tweets in the entire month.

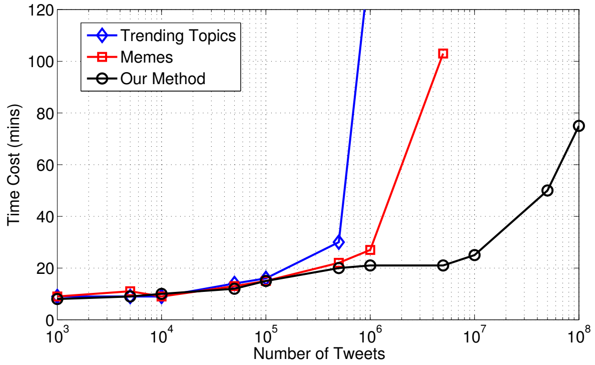


Figure 5: Running time vs. batch size.

6.4 Discussion

We have shown in the experiments that tweets asking verification questions or making corrections to controversial statements are very important signals of rumors early in their life cycle. Some users who have an information need of evaluating a rumor will post tweets either asking verification questions to express suspicions, or will correct the rumor after their investigation. Verification questions are particularly useful because they appear much sooner and thus facilitate earlier detection of rumors.

Not all clusters that include tweets that ask verification questions or use correction phrases are actually rumors. We identified 13 different statistical features of clusters of tweets, such as average tweet length and percentage of signal tweets, etc. By training a decision tree model, we built a powerful ranking algorithm that ranks tweet clusters by how likely they are to be rumors (i.e., controversial, fact-checkable claims). The precision we can achieve is much higher than without the ranking algorithm.

We demonstrated that our proposed framework can scale up well. By clustering only the small set of signal tweets, we avoided the computational cost of detecting popular statements or trending topics from the entire corpus. Our proposed framework is robust even if the number of tweets to be processed exceeds 100 million.

To give the readers a sense of the end-to-end operation of our system, we present the rumors detected in the two data sets. For the BOSTON data set, we identified the top 5 candidate rumor clusters each hour. Figure 6 plots the number of tweets hourly of each identified rumor statement. The dotted blue line in the background shows the number of tweets arriving in that hour.

Although the small set of regular expressions may yield a low recall of enquiry signals, when the candidate rumor clusters detected by our system are labeled by human experts, they can be used to enrich the set of signals. Indeed, using statistical feature selection techniques [12], one can extract features that are highly representative in the rumor clusters and underrepresented in the non-rumor clusters. These patterns can be approved by human experts and added into the pipeline to identify more signal tweets, thus improving precision and recall of rumor detection. By using rumors of our two data sets labeled by annotators, we have already discovered a few additional promising patterns such as "scandal?", or "fact check." We leave the iterative improvement of the signal patterns to future work.

One may be curious about what types of rumors are being circulated in a random week of tweets. From the GARDENHOSE data set, we output the 10 top-ranked rumors for each day and tracked them (Figure 7). We also show example statements extracted from

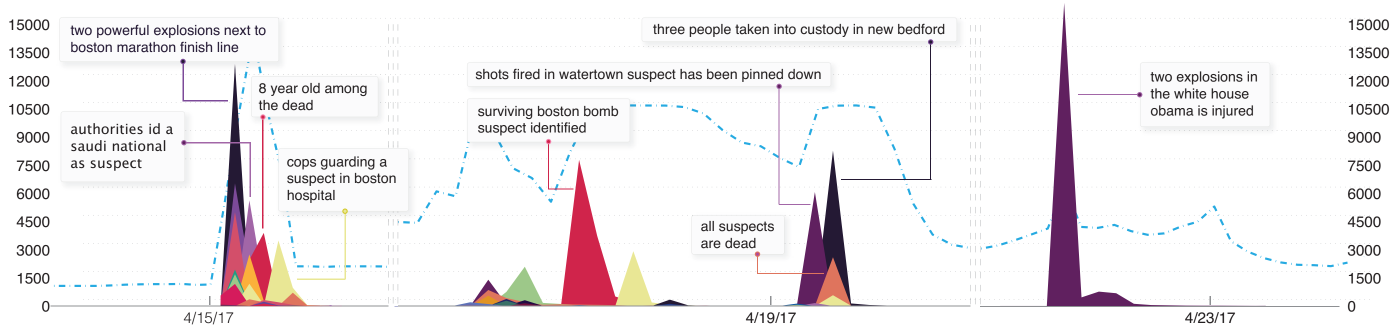


Figure 6: Tracking detected rumors about Boston Marathon bombing

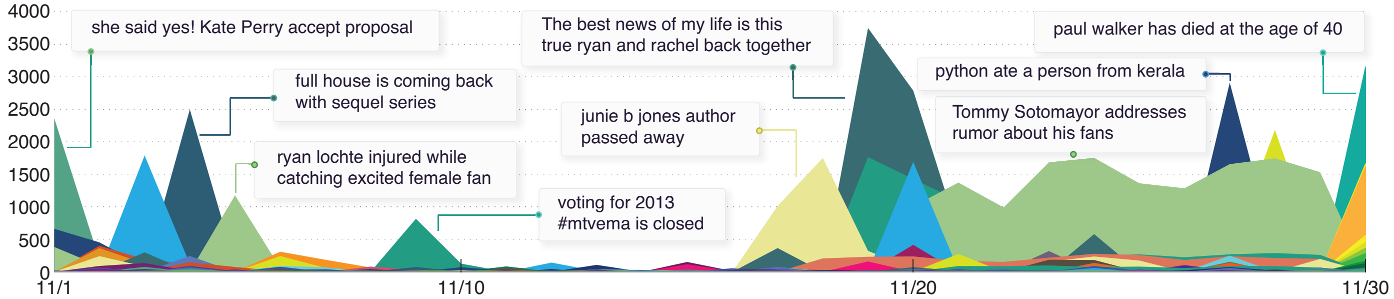


Figure 7: Tracking detected rumors in November 2013

the most popular rumor clusters. Most everyday rumors turn out to be gossip about celebrities, with occasionally emerging anecdotes like “python ate person.”

7. CONCLUSION

One post of a rumor in social media can sometimes spread beyond anyone’s control. A rumor about two explosions in the White House is a perfect example of how a single tweet out of more than 9,000 tweeted in the same second spreads and causes real damage.

In social media, users share information based on different types of needs, including the need to verify controversial information. We point out that such information needs can not only help spread rumors, but also provide the first clue for detecting them.

Based on this important observation, we designed a rumor detection approach. We cluster only those tweets that contain enquiry patterns (the signal tweets), extract the statement that each cluster is making, and use that statement to pull back in the rest of the non-signal tweets that discuss that same statement. We then rank the clusters based on statistical features that compare properties of the signal tweets within the cluster to properties of the whole cluster.

Extensive experiments show that our proposed method can detect rumors effectively and efficiently in their early stage. With a small Hadoop cluster, in about half an hour we process 10% of all the tweets posted in one day on Twitter. If we output 50 candidate statements, about one third of them are real rumors, and about 70% of the top ranked 10 clusters are rumors.

There is still considerable room to improve the effectiveness of the rumor detection method. We can improve the filtering of enquiry and correction signals by training a classifier rather than relying on manually selected regular expressions. We can further develop a method to automatically update the filtering patterns in real time to prevent potential spamming of the detection system. We can also explore more features for each statement and train a better ranking algorithm for candidate rumor clusters. Another direction is to adopt this method to detect rumors automatically and generate a large data set of rumors, which can benefit many potential analyses such as finding features that are potentially correlated to the truth value of a rumor, or analyzing general diffusion patterns or the life cycle of rumors.

8. ACKNOWLEDGMENT

We thank Cheng Li and Yuncheng Shen for generating the visualizations. This work is partially supported by the National Science Foundation under grant numbers IIS-0968489 and IIS-1054199. It is also partially supported by the DARPA under award number W911NF-12-1-0037.

9. REFERENCES

[1] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[2] A. Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE, 1997.

[3] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

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

[5] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[6] E. H. Chi. Information seeking can be social. *IEEE Computer*, 42(3):42–46, 2009.

[7] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[8] I. Dagan, O. Glickman, and B. Magnini. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer, 2006.

[9] N. DiFonzo and P. Bordia. *Rumor psychology: Social and organizational approaches.* American Psychological Association, 2007.

[10] P. Domm. False rumor of explosion at white house causes stocks to briefly plunge; ap confirms its twitter feed was hacked., April 2013.

[11] G. Erkan and D. R. Radev. Lexrank: graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, pages 457–479, 2004.

[12] G. Forman. An extensive empirical study of feature selection metrics for text classification. *The Journal of machine learning research*, 3:1289–1305, 2003.

[13] A. Friggeri, L. A. Adamic, D. Eckles, and J. Cheng. Rumor cascades. In *Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media*, 2014.

[14] A. Gupta and P. Kumaraguru. Credibility ranking of tweets during high impact events. In *Proceedings of the 1st Workshop on Privacy and Security in Online Social Media*, page 2. ACM, 2012.

[15] A. Gupta, H. Lamba, and P. Kumaraguru. $1.00 per rt# bostonmarathon# prayforboston: Analyzing fake content on twitter. In *eCrime Researchers Summit (eCRS), 2013*, pages 1–12. IEEE, 2013.

[16] A. Gupta, H. Lamba, P. Kumaraguru, and A. Joshi. Faking sandy: characterizing and identifying fake images on twitter during hurricane sandy. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 729–736. International World Wide Web Conferences Steering Committee, 2013.

[17] S. Kwon, M. Cha, K. Jung, W. Chen, and Y. Wang. Prominent features of rumor propagation in online social media. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1103–1108. IEEE, 2013.

[18] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 497–506. ACM, 2009.

[19] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[20] M. Mathioudakis and N. Koudas. Twittermonitor: trend detection over the twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 1155–1158. ACM, 2010.

[21] 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.

[22] M. R. Morris, S. Counts, A. Roseway, A. Hoff, and J. Schwarz. Tweeting is believing?: understanding microblog credibility perceptions. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 441–450. ACM, 2012.

[23] M. R. Morris, J. Teevan, and K. Panovich. A comparison of information seeking using search engines and social networks. *ICWSM*, 10:23–26, 2010.

[24] M. R. Morris, J. Teevan, and K. Panovich. What do people ask their social networks, and why?: a survey study of status message q&a behavior. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1739–1748. ACM, 2010.

[25] S. A. Paul, L. Hong, and E. H. Chi. Is twitter a good place for asking questions? a characterization study. In *ICWSM*, 2011.

[26] S. C. Pendleton. Rumor research revisited and expanded. *Language & Communication*, 18(1):69–86, 1998.

[27] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[28] V. Qazvinian, E. Rosengren, D. R. Radev, and Q. Mei. 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, 2011.

[29] J. Ratkiewicz, M. Conover, M. Meiss, B. Gonçalves, A. Flammini, and F. Menczer. Detecting and tracking political abuse in social media. In *ICWSM*, 2011.

[30] J. Ratkiewicz, M. Conover, M. Meiss, B. Gonçalves, S. Patil, A. Flammini, and F. Menczer. Truthy: mapping the spread of astroturf in microblog streams. In *Proceedings of the 20th international conference companion on World wide web*, pages 249–252. ACM, 2011.

[31] R. L. Rosnow. Inside rumor: A personal journey. *American Psychologist*, 46(5):484, 1991.

[32] E. Seo, P. Mohapatra, and T. Abdelzaher. Identifying rumors and their sources in social networks. In *SPIE Defense, Security, and Sensing*, pages 83891I–83891I. International Society for Optics and Photonics, 2012.

[33] S. Sun, H. Liu, J. He, and X. Du. Detecting event rumors on sina weibo automatically. In *Web Technologies and Applications*, pages 120–131. Springer, 2013.

[34] T. Takahashi and N. Igata. 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, 2012.

[35] F. Yang, Y. Liu, X. Yu, and M. Yang. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 13. ACM, 2012.

[36] J. Yang, M. R. Morris, J. Teevan, L. A. Adamic, and M. S. Ackerman. Culture matters: A survey study of social q&a behavior. *ICWSM*, 11:409–416, 2011.

[37] Z. Zhao and Q. Mei. Questions about questions: An empirical analysis of information needs on twitter. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1545–1556. International World Wide Web Conferences Steering Committee, 2013.