A Human-Machine Collaborative System for Identifying Rumors on Twitter

Soroush Vosoughi
The Media Lab
Massachusetts Institute of Technology
Cambridge, MA 02139
Email: soroush@mit.edu

Deb Roy
The Media Lab
Massachusetts Institute of Technology
Cambridge, MA 02139
Email: dkroy@media.mit.edu

***Abstract*—The spread of rumors on social media, especially in time-sensitive situations such as real-world emergencies, can have harmful effects on individuals and society. In this work, we developed a human-machine collaborative system on Twitter for fast identification of rumors about real-world events. The system reduces the amount of information that users have to sift through in order to identify rumors about real-world events by several orders of magnitude.**

I. Introduction

Now more than ever, people turn to social media as their source of news [1], [2], [3]; this is especially true for breaking-news situations, where people crave rapid updates on developing events in real-time. As Kwak et al. have shown, over 85% of all *trending topics*[1] on Twitter are news [3]. Ubiquity and accessibility are among the factors that make social media a great resource for dissemination of breaking-news. These same factors, combined with the relative lack of oversight of such services, make social media fertile ground for the creation and spread of unsubstantiated and unverified information about events happening in the world. Users of Twitter have the incredibly hard task of sifting through a large number of posts, in order to separate substantiated and trustworthy posts from rumors and unjustified assumptions. A case in point of this phenomenon is the social media's response to the Boston Marathon bombings. As much as social media was a great resource for the people living in the greater Boston area, enabling them to stay up-to-date on the situation as it enfolded, it led to several unfortunate instances of false rumors being spread, and innocent people being implicated in witch-hunts [4], [5], [6].

In this paper, we present a preliminary overview of a novel human-machine collaborative system for fast identification of potential emerging rumors about real-world events. We define a rumor to be an unverified assertion about an event. On Twitter, that translates to a collection of tweets, all asserting the same unverified statement spreading through Twitter. Our system can be used by journalists, emergency services and consumers of news to greatly reduce the amount of data they have to sift through to separate unsubstantiated claims from facts. Note that this system is designed for identification of rumors and not verification. The output of our system, however, can be used as the input to verification systems, several of which have been created for Twitter [7], [8], [9], [10].

[1]Trending topics are those topics being discussed more than others on Twitter.

II. System Overview

An overview of the system can be seen in Figure 1. The input to our system is a collection of tweets about an event, specified by the user through a boolean query (e.g., *Boston AND Bombing* in this illustration). Our system consists of two major parts, an assertion detector and a hierarchical clustering module. Raw tweets about an event feed directly into the assertion detector, which automatically filters the tweets for only those containing assertions. The output of the assertion detector feeds directly into the hierarchical clustering module, the output of which is a collection of clusters. These clusters contain messages that have propagated through Twitter; these are potential rumors about the event of interest. The user can then sort through these clusters to sift interesting clusters from the uninteresting and noisy ones. In the next sections, we will explain the two modules in greater detail.

III. Assertion Detection

An assertion is an utterance that commits the speaker to the truth of the expressed proposition. Figure 2 shows two tweets about the Boston Marathon bombings. The tweet shown in Figure 2a contains an assertion while the tweet shown in Figure 2b does not. More generally, assertions are a class of speech-acts. Speech-acts have performative function in language and communication. For instance, we perform speech acts when we offer an apology, greeting, request, complaint, invitation, compliment, or refusal (to name a few). We used a state-of-the-art supervised Twitter speech-act classifier, which utilizes both syntactic and semantic features of tweets for classification, to detect assertions in tweets [9], [11]. This classifier can identify several different speech-acts in tweets: assertions, expressions, questions, recommendations and requests.

For the purposes of this work only we were only interested in classifying assertions, thus we modified the speech-act classifier into a binary assertion classifier. We trained the classifier on $7,000$ manually labelled tweets about several different real-world events, with 3290 (47%) of those tweets containing assertions and the rest containing one of the other speech-acts (i.e., expressions questions, etc).

We evaluated the classifier using 5-fold cross validation. The *F1* score, which is the harmonic mean of recall and

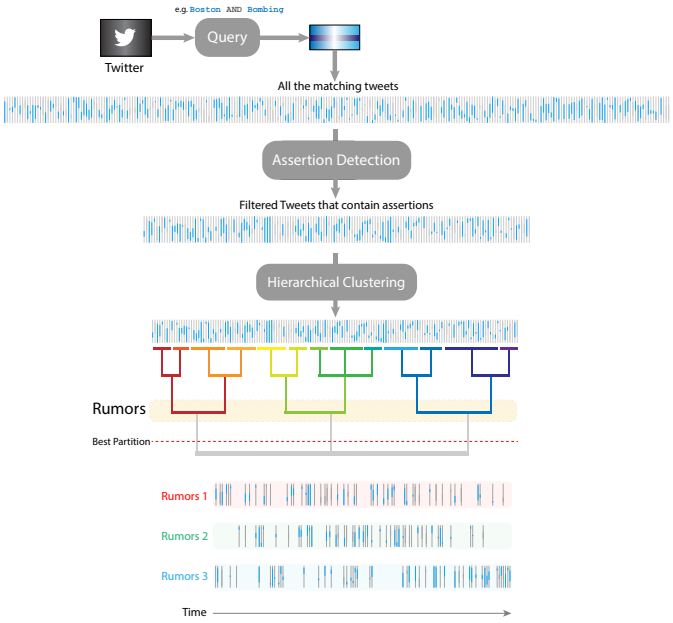

Fig. 1: The pipeline of our system. The input to the system is a collection of tweets about an event specified by the user through a boolean query (e.g., *Boston AND Bombing* in this illustration). The output is a collection of potential rumors.

(a) An assertion. (b) Not an assertion.

Fig. 2: Two example tweets. Tweet (a) contains an assertion, while tweet (b) does not.

precision (see Equation (1)), for classifying assertions was .86.

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad (1)$$

In order to better understand the performance of the assertion classifier, we look at the *receiver operating characteristic (ROC)*, or the ROC curve of the classifier. An ROC curve is a plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the *true positive rate (TPR)* against the *false positive rate (FPR)* at various threshold settings. The definitions for TPR and FPR are shown in Equation (2). In the equation, *TP* stands for true positive, *TN* for true negative, *FP* for false positive, and *FN* for false negative.

$$TPR = \frac{TP}{TP + FN} \quad FPR = \frac{FP}{FP + TN} \quad (2)$$

Using the TPR and FPR we can plot the ROC curve of the assertion classifier, shown in Figure 3. According to the ROC curve, at a false positive rate of .28, the true positive rate would be 1.0. This means that in order to get all the tweets containing assertions correctly classified, we would have to tolerate 28% of non-assertion containing tweets mistakenly classified as containing assertions. Depending on the application that the system is being used for, one might be able to tolerate this level of noise in order to capture all assertion containing tweets. However, if the application requires much more precision, then a different point on the operating curve can be picked. For example, at only 15% false positive rate, the tool will correctly identify 90% of all tweets containing assertions. This parameter is under the control of the user who can tune it to different values depending on the application.

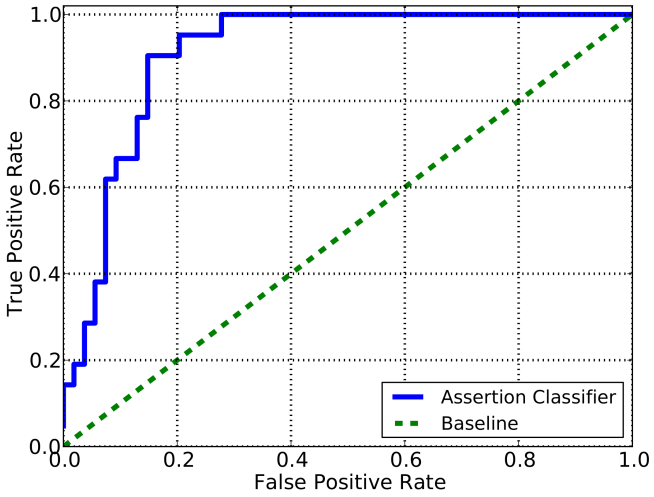

Fig. 3: The receiver operating characteristic (ROC) curve of the assertion classifier.

IV. Clustering of Assertions

The second part of the system, shown in Figure 1, is a clustering module. Generally speaking, hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. This module takes as input the output of the assertion detector, which is a set of tweets containing assertions. The output of the clustering module is a collection of clusters, each containing tweets with similar assertions. There are two strategies for hierarchical clustering [12]:

- *Agglomerative*: This is a "bottom up" approach; each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- *Divisive*: This is a "top down" approach; all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

The complexity of agglomerative clustering is polynomial at $O(n^3)$, while the complexity of divisive clustering is exponential at $O(2^n)$. Given the potentially large number tweets about an event, we decided to use Hierarchical Agglomerative Clustering (HAC), given its lower complexity.

A. Data Preparation

Before the tweets are passed to the clustering module, they are cleaned up. All mentions of usernames (which are denoted by the @ symbol) were removed. All retweet symbols (which

are denoted by the *rt* symbol), were also removed. Tweets also contain very informal language and as such, characters in words are often repeated for emphasis (e.g., the word *good* is used with an arbitrary number of *o*'s in many tweets). Any character that was repeated more than two times was removed (e.g., *goooood* was replaced with *good*). Finally, all words in the tweets were stemmed using *Porter Stemming* [13].

A large portion of tweets contain links to other websites, though these links are mostly not meaningful semantically, we did not remove them. This is because many rumors originate on sites other than Twitter and in some cases these links are useful in identifying rumors seeping into Twitter from other sources.

B. Similarity Function

For agglomerative clustering, there needs to be a way to decide which clusters should be combined. This is achieved through the use of a metric that measures the distance between pairs of observations, or tweets in our case. The similarity function that we used for HAC of tweets is *TF-IDF* combined with *cosine similarity*. TF-IDF, or Term FrequencyInverse Document Frequency, is a method of converting text into numbers so that it can be represented meaningfully by a vector [14]. TF-IDF is the product of two statistics, *TF* or Term Frequency and *IDF* or Inverse Document Frequency.

Term Frequency measures the number of times a term (word) occurs in a document. Since each document will be of different size, we need to normalize the document based on its size. We do this by dividing the Term Frequency by the total number of terms. TF considers all terms as equally important, however, certain terms that occur too frequently should have little effect (for example, the term *"the"*). And conversely, terms that occur less in a document can be more relevant. Therefore, in order to weigh down the effects of the terms that occur too frequently and weigh up the effects of less frequently occurring terms, an Inverse Document Frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely. Generally speaking, the Inverse Document Frequency is a measure of how much information a word provides, that is, whether the term is common or rare across all documents.

The exact formula for calculating TF is shown in Equation (3). Here, t is the term being processed, d is the document, and the function *f(t,d)* measure the raw frequency of t in d.

$$TF(t,d) = 0.5 + \frac{0.5 \times f(t,d)}{max\{f(w,d) : w \in d\}} \tag{3}$$

The formula for calculating IDF is shown in Equation (4). Here, N is the total number of documents in the corpus, and $|\{d \in D : t \in d\}|$ is the number of documents where the term t appears.

$$IDF(t,D) = \log \frac{N}{1 + |\{d \in D : t \in d\}|} \tag{4}$$

Using the definitions of TF and IDF, the TF-IDF is then calculated as shown in Equation (5).

$$TFIDF(t,d,D) = TF(t,d) \times IDF(t,D) \tag{5}$$

Using TF-IDF, we derive a vector for each tweet. The set of tweets in our collection is then viewed as a set of vectors in a vector space with each term having its own axis. We measure the similarity between two tweets using the formula shown in Equation (6). Here, $d1 \cdot d2$ is the dot product of two documents, and $\|d1\| \times \|d2\|$ is the product of the magnitude of the two documents.

$$Similarity(d1,d2) = \frac{d1 \cdot d2}{\|d1\| \times \|d2\|} \tag{6}$$

C. Hierarchical Agglomerative Clustering

Using the similarity function that was described, we can use hierarchical agglomerative clustering (HAC) to cluster similar assertions together. The arrangement of the clusters produced by HAC can be illustrated using a dendrogram. Figure 4 shows a sample dendrogram depicting HAC.

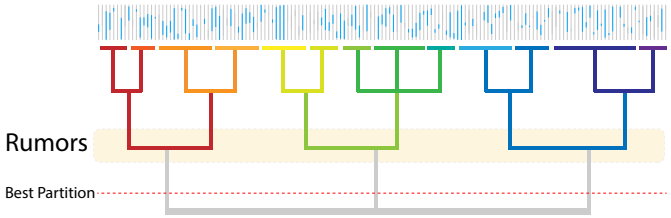

Fig. 4: A sample dendrogram depicting hierarchical agglomerative clustering.

As can be seen in the figure, at the very first level all tweets belong to their own clusters, so there are as many clusters as there are tweets. At the very root of the tree, is a single cluster, containing all the tweets. It is up to the user to pick the level at which the clusters are to be used (this is called the *best partition* in Figure 4). These clusters are potential rumors. A partition higher in the tree (further from the root) would yield more rumor clusters, with each rumor containing fewer number of tweets. Conversely, a partition closer to the root would yield fewer clusters, with each containing greater number of tweets. Depending on the application and the particular event that is being looked at, different *best partitions* can be picked by the user. For example, if the event in question is a very local event, meaning that there are not many tweets about the event, then perhaps a partition further from the root would be more useful and vice-versa.

V. Analysis and Evaluation

An insightful way of measuring the performance of our system is to measure the *bandwidth reduction* of information afforded by our system. Bandwidth reduction is an important measurement because it can help demonstrate the usefulness and utility of our system in real-world situations. For example, a journalist trying to sort out false and true information from millions of tweets about a real-world event (as was the case with the Boston Marathon bombings), has a Sisyphean task. However, our system can make the task much less daunting and more manageable for our hypothetical journalist by greatly

reducing the amount of information he or she has to sort through

We would like to estimate the bandwidth reduction afforded by our system. Of course, the bandwidth reduction would depend on the level of the partition used in HAC, but generally speaking, the number of clusters or rumors is somewhere between hundreds to thousands of rumors, depending on the size of the event. For example, in the case of the Boston Marathon bombings, there were about 20 million English-language tweets about the event (as picked up by a simple user-defined query). Through the assertion filter, our system reduced this number by 50% to around 10 million tweets containing assertions. Finally, using HAC, the system further reduced the bandwidth by clustering these assertions into somewhere between 100 and 1000 clusters (depending on the partition set by the user). This is a reduction by four orders of magnitude, making it much more manageable for the user to search for potential rumors.

By utilizing word clouds (which are used to show the most prominent words in documents), one can quickly and reliably examine the salient content of each cluster to identify rumors. Using this method, we manually examined the rumor clusters generated from the Boston Marathon bombings dataset. Though many of the clusters were junk, we were able to identify many of the major rumors that propagated on Twitter during the event [2]. Figure 5 shows the word cloud for one of the rumor clusters generated by our system. It is clear that this cluster has captured the "man on the roof" rumor about the Boston Marathon bombings. This was the rumor that a man was seen on the roof of a building just as the explosion occurred on the street below [15].

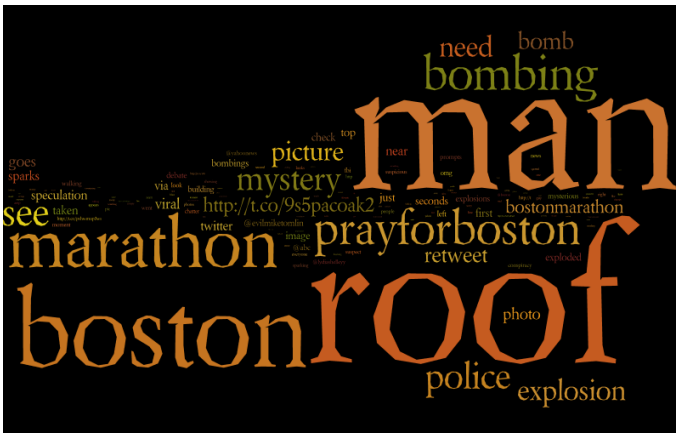

Fig. 5: Word cloud of one of the rumor clusters, capturing the "man on the roof" rumor from the Boston Marathon bombings dataset.

VI. Conclusions and Future Work

In this paper, we presented a preliminary view of a human-machine collaborative system for fast identification of rumors on Twitter. The system drastically reduces the "bandwidth" of the information that users have to look through in order to identify rumors about real-world events.

As this work is in its early stages, there are several directions that are open to exploration in the future. The most immediate direction is the utilization of more sophisticated hierarchical clustering methods. For example, ones that are able to capture topic level similarity, such as the method developed by Kuang, et al. [16]. Moreover, we would like expand our evaluation from our current qualitative approach to a more objective, quantitative one.

Acknowledgment

The authors would like to thank Mostafa "Neo" Mohsenvand and Helen Zhou for their help with the preparation of this document.

[2]See http://www.snopes.com/politics/conspiracy/boston.asp for a list of rumors.

References

[1] S. Laird, "How social media is taking over the news industry," April 2012, http://mashable.com/2012/04/18/social-media-and-the-news/[mashable.com; posted 18-April-2012].

[2] W. Stassen, "Your news in 140 characters: exploring the role of social media in journalism," *Global Media Journal-African Edition*, vol. 4, no. 1, pp. 116–131, 2010.

[3] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 591–600.

[4] L. Kundani, "When the tail wags the dog: Dangers of crowdsourcing justice," July 2013, http://newamericamedia.org/2013/07/when-the-tail-wags-the-dog-dangers-of-crowdsourcing-justice.php/[newamericamedia.org; posted 27-July-2013].

[5] D. Lee, "Boston bombing: How internet detectives got it very wrong," April 2013, http://www.bbc.com/news/technology-22214511/[bbc.com; posted 19-April-2013].

[6] M. Valdes, "Innocents accused in online manhunt," April 2013, http://www.3news.co.nz/Innocents-accused-in-online-manhunt/tabid/412/articleID/295143/Default.aspx/[3news.co.nz; posted 22-April-2013].

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

[8] 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*. IEEE, 2013, pp. 1103–1108.

[9] S. Vosoughi, "Automatic detection and verification of rumors on twitter," Ph.D. dissertation, Massachusetts Institute of Technology, 2015.

[10] S. Vosoughi and D. Roy, "Predicting the veracity of rumors on twitter," *Submitted to The Transactions on Knowledge Discovery from Data*, 2015.

[11] S. Vosoughi and D. Roy, "Tweet acts: A speech act classifier for twitter," *Submitted to PLoS ONE*, 2015.

[12] O. Maimon and L. Rokach, *Data mining and knowledge discovery handbook*. Springer, 2005, vol. 2.

[13] M. F. Porter, "An algorithm for suffix stripping," *Program: electronic library and information systems*, vol. 14, no. 3, pp. 130–137, 1980.

[14] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2011.

[15] L. Effron, "Mystery "man on the roof" sparks boston marathon chatter." 2013.

[16] D. Kuang, J. Choo, and H. Park, "Nonnegative matrix factorization for interactive topic modeling and document clustering," in *Partitional Clustering Algorithms*. Springer, 2015, pp. 215–243.