Simulating real-time Twitter data from historical datasets

Shane E. Halse
Pennsylvania State University, USA
Seh297@ist.psu.edu

Rob Grace
Pennsylvania State University, USA
Wrg123@ist.psu.edu

Jess Kropczynski
University of Cincinnati
Jess.Kropczynski@uc.edu

Andrea Tapia
Pennsylvania State University, USA
ATapia@ist.psu.edu

ABSTRACT

In this paper, we will discuss a system design for simulating social media data based on historical datasets. While many datasets containing data collected from social media during crisis have become publicly available, there is a lack of tools or systems can present this data on the same timeline as it was originally posted. Through the design and use of the tool discussed in this paper, we show how historical datasets can be used for algorithm testing, such as those used in machine learning, to improve the quality of the data. In addition, the use of simulated data also has its benefits in training scenarios, which would allow participants to see real, non-fabricated social media messages in the same temporal manner as found on a social media platform. Lastly, we will discuss the positive reception and future improvements suggested by 911 Public Service Answering Point (PSAP) professionals.

Keywords

Twitter, Simulation, Crisis Response, Social Media

INTRODUCTION

Does it work? This is a question asked by many software and application developers when designing and developing new tools. This question however becomes much more complex when examining tools and applications designed to work in real time. Furthermore, when the application is designed to be used to support critical decision making that could affect lives of others, it has to work. With this in mind the software development cycle can be broken up into the five stages of; requirements analysis, design, implementation, testing and evolution or iteration.

Unfortunately, the world of software development is one that seems boundless, with each application having different requirements and goals. In order to provide focus to our research, we have chosen the domain of crisis response. In particular, we investigate software development tools for crisis response that look at how to best gather, filter, and curate data produced by citizens such as that found on social media. With people turning to social media to both disseminate and gather information, there is an abundance of information. For example, if we take the social media platform Twitter, infamous Hurricane Sandy during 2012 in which over 20 million tweets were posted over a five-day period (Halse et al. 2016). Next, as crisis responders only require actionable information with a high degree of veracity, much work as looked at how to improve the quality of this data through tools applying advanced filtering and machine-learning algorithms (Halse et al. 2016)(Caragea et al. 2011).

Due the nature of crisis response, the implementation phase of the software development cycle can be a particularly challenging one. Previous work has shown the value of tools for social media analysis for crisis responders along with the desire to use such tools (Alexander 2014). However, it is not without difficulties associate with deployment in the crisis response domain (Jin, Liu, and Austin 2011). These include difficulties in increasing the workload of the responders, especially during a crisis, along with providing adequate training tools

that are based on actual, non-fabricated datasets similar to those that would be seen during and event.

The next two crucial steps in the software development cycle, is that of implementation and testing. It is within these steps that we face two primary challenges in developing tools for crisis response. The first is the unpredictability of certain crisis. Some slow onset crisis, such as hurricanes, floods, and fires, provide enough time for researchers and end users to deploy and monitor software application for data. However fast onset crisis, earthquakes, bombings, and other manmade crisis may require advanced techniques for detection and identification. Furthermore, fast onset crisis also require the users of applications to have active ongoing data collection and analysis. The second issue regarding development of crisis response application is that of the temporal nature of the data collected during a crisis. While many discoveries and tools have been created through the analysis of full datasets, they are difficult to test and improve within a real time environment.

It is these last two steps in the software development cycle, which this paper will address. This is done through presenting a simulation tool designed to allow testing of software used for gathering, filtering and analyzing social media data. We discuss the system architecture, features and the results of a small workshop with crisis response professionals. Additional avenues, such as training, in which a simulator can be implemented, are also discussed.

RELATED WORKS

Social media contains a vast amount of data, be it a community or people sharing information about every day, to large organizations attempting to disseminate time sensitive critical information to the public. This makes it an ideal candidate for data collection and analysis, especially during emergencies when traditional avenues of information gathering and sharing may not be available. It has also become an important tool to reach those impacted by a crisis event (Kavanaugh et al. 2012). While there can be an overabundance of information found within social media sites, many previous research studies have been able to parse out particular information related to a given event. For example, work has shown that through using given keywords and machine learning algorithms, earthquakes events and information can be detected through messages found on Twitter (Crooks et al. 2013)(Sakaki, Okazaki, and Matsuo 2013). Likewise, social media has also been used to capture data for fires, shootings, hurricanes and building collapses (Palen 2008).

Furthering this frameworks, tools have also been developed to allow dynamic detection and capturing of data during a crisis (Bruns and Liang 2012). With many frameworks and analysis methods available, tools have been developed for use by crisis responders. Some of these tools include; Earthquake and reporting system (EARS) (Avvenuti et al. 2014), Tweet4act (Chowdhury, Chowdhury, and Castillo 2013), SensePlace (MacEachren et al. 2011) and TweetTracker (Kumar et al. 2011). Unfortunately, despite the availability of these tools, social media remains a "white whale" for crisis responders. A solution to social media adoption by crisis responders can be addressed through examining two primary concerns; the first is identifying which of the tools are most effective and identifying areas in which they can be improved, with the second being developing simulated training environments for the top performers. This has been approached in previous work which investigated how data can be used to design, develop and improve tools the emergency responders (Hampton et al. 2017). However, this work utilizes data that is fabricated artificially. In this work, we utilize previously collected corpuses from real messages posted on Twitter.

With the first concern in mind, and with the use of simulations in crisis response shown to be an effective method by which to improve the quality of software applications and algorithms used in crisis response (Walker and Giddings 2011), we can begin to standardize the testing environments. Through standardizing the simulation inputs and outputs with the same datasets, we can reduce any additional confounding variable during testing. Previous research has shown that valuable information in system design can be gained, especially through incorporating user centered design (Kluckner and Wendt 2014). This is done through presenting end users with a simulated environment to determine what information they feel is most relevant. Likewise, through incorporating the results of studies into a simulated environment, software developers have an ability to compare one iteration and design to another based on user feedback. Standardizing the simulated environment also opens avenues in which policies and procedures can be created.

In addition to testing, and with respect to the second concern, standardizing simulation environments allows for the development of training protocols. Training for crisis responders with games and simulations has had much success in previous literature (Heldal and Wijkmark 2017) (Parush et al. 2016). These two examples describe how through simulations, training is both reduced in cost and increased in effectiveness. This combats the problem of users being unfamiliar with the system during times of crisis. It is through training that the user can develop a better understanding of the system and learn the limitation or advantages within (Schaefer et al. 2014). Additional works have also stressed the importance of training, especially when the system be utilized is novel and the user is unfamiliar with it (Abbasi and Kumar 2012)(Baumgart et al. 2006). Rather than having to wait for a particular event to occur in real time, simulations within a training environment provide the ability to control which data set and corresponding event is being used. In addition to this, with data collected during previous crisis, crisis

responders can identify particular keywords, search tactics and analysis strategies. This is particularly important when the data used for the simulation has been localized to their area of interest.

SIMULATOR SYSTEM DESIGN ARCHITECTURE

The goal of this simulator is not to create new social media messages, but instead is to take messages that have been collected from previous datasets. These messages would then be sent in the same temporal pattern as they were originally posted. Rather than send a fully collected dataset to a software application, the basic premise of this simulator is to read in the data one message at a time, detect the timestamp for that message and then send the data after a given amount of time has passed. That is, once the simulator has begun running it starts at a predetermined historical timestamp such as 1 hour before the event had occurred. It would then calculate the delay between this time and the timestamp found from the Twitter message. After the simulator had been running for this delay, it would then send that message. For example, if we set our simulators start time to 11:00, and had an event such as a shooting, where the first message about the shooting was posted at 11:57, then it would take 57 minutes for that message to be shown (assuming a timescale of 1:1). This emulates the temporal nature that would be seen from data gathered during a crisis event. Figure 1 below shows how the simulator is connected to existing tools with the minor addition of a data adapter.

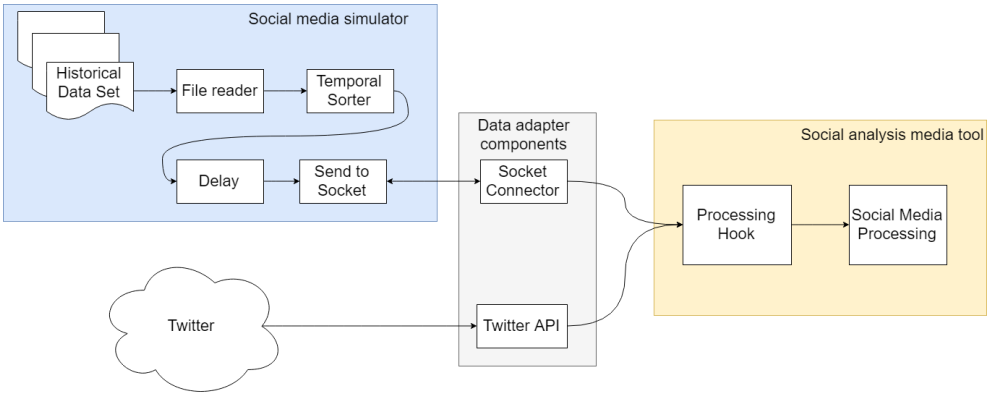

Figure 1 - Overview of system usage

From Figure 1 we can see that the social media simulator is implemented in place of collecting data directly from Twitter. The concept here is to allow developers to leverage previously collected datasets, that are crisis related, to test and improve their designs. The goal of the simulator is not to replace data collected from Twitter, however there is not always a crisis occurring that can be used for development purposes. It is through the adapter component that the source of the data would be selected. In the diagram above we have shown the adapter as a separate component however it can be easily incorporated into the existing social media analysis tools. This way the developers can choose if they would prefer live data, (assuming a crisis is ongoing) or simulated data based on a selected dataset.

Data

In order to create a tool that leverages historical datasets to produce a simulated social media stream, historical datasets are needed. The datasets used for testing in our case was those publicly available from crisislex.com and described in the works of (Olteanu et al. 2014). This collection contains data collected on over 30 different crisis events. As each data set has the potential to contain many different data points, we began by defining some of the fields that would be required to create a simulation stream. As the simulator's main purpose was to deliver tweets from these datasets in the same temporal nature to how they were originally found, the first required data point is that of a timestamp. Next, the simulator detects one of two possible conditions: data containing a messageID, and data with no messageID but does have message context. In some cases, the creators of a dataset did not collect additional information concerning a message, but instead only a timestamp and a messageID were collected. If no messageID is available then only the message content data field is used. While it is possible for the datasets to contain additional informational fields, our simulator is primarily concerned with displaying each of the tweet's message in the same temporal order originally seen. Each of the data points are explained in more detail in the following sections.

Timestamp (Required)

The timestamp for the data is required as this serves as the basis for when to send or show the data. This tool looks for the timestamp with the earliest value and uses this as the simulators start time (startTimestamp). It then

calculates the value (in milliseconds) between the earliest timestamp and the next timestamp (nextTimestamp). This is done using the following formula deltaTime = nextTimestamp – startTimestamp. One the value for how much time exists between the start time and corresponding tweet's timestamp (deltaTime) has been calculated, a delay of deltaTime is implemented before displaying the message data. For example to increase the speed at which messages are displayed we can divide deltaTime by the scale factor. The pseudo code snippet looks like this:

```
Loop for each message in dataset
deltaTime =   nextTimestamp – startTimestamp;
        timePassed = nowTime – codeExecutionStartTime;
        if(timePassed > deltaTime/timeScaleFactor)
                Show/Send message
endLoop
```

It is important to note that while the simulator is designed to present messages found in a dataset at the same speed they were originally displayed, a timeScaleFactor was added to allow for accelerated testing. For example, if we wanted to test the performance of an analysis algorithm in pseudo realtime, a time scale factor could be used as some of the datasets were collected over a period of many days. However, if the timescale is changed it is important to adjust the parameters of any tools that expect data in real time. Furthermore, by using a timeScaleFactor variable, we provide two features of the simulator. The first is to allow computational tools to be tested at an accelerated rate, whereas the second feature offered is the ability to provide end user the ability to slow or speed up the data. In training scenarios it may be necessary to pause or slow the data stream down to allow for explanation and topics to be discussed.

MessageID (Required if no message content)

Some datasets do not contain information beyond a timestamp and a messageID. This is particularly true with Twitter datasets that require a process knowns rehydration. Rehydration occurs when only as some situations Twitter do not allow the message contents of a tweet to be shared but instead only allows the sharing of the tweet's timestamp and its tweet ID. In these cases, the simulator connects to the Twitter API and requests information for the given messageID. If the contents of a message are not available, the simulation tool connects to the Twitter API to retrieve the relevant message data. As such, the message ID for each message within a data set is required if there is no corresponding content. In addition, with messageIDs other Meta data can be collected for further analysis such as number of retweets, likes, media links and poster's information.

Message content (Optional)

Message content is a field found in some of the datasets and is essentially the body or message of a tweet that contains only the text portion of a Twitter message. The simulator is designed to use this message field if it is found in the corresponding dataset. As the primary interest of the researchers in this paper was to present message content to a separate analysis tool, this field was used when found. If however the dataset did not contain the contents of messages, the rehydrating process described above was used.

Other Information (Optional)

As mentioned previously the datasets that we used for testing the Twitter simulator contained many different data fields. With this in mind, we limited the required fields to only that of a timestamp and a MessageID. However, additional fields such as coding fields for each tweet can easily be added. This in turn would then be added to the output to be used later by other applications.

Input data

Due to the similarity of the datasets used during testing, a comma separated (.csv) file structure was used for the input file. This is not however a limitation but rather a design choice as most of our test datasets were contained within the csv files. In a similar manner, any delimited file could be used as the input to the simulator with minor modification. Furthermore, while not implemented in the current vision of the simulator, additional data input formats could be added in the future such as database connections or JSON and XML formats.

Data handling

The simulator is responsible for parsing data from a given input file, selecting the appropriate fields, analyzing the time sequence of tweets, creating a connection endpoint and outputting the data to that connection endpoint. The code for the simulator was written in Java. As we are using .csv dataset file the first step of the simulator is to read each file in. it then parses out fields such as timestamps, messageID, message content and any additional fields contained. Furthermore, if additional information is required, the simulator would query the Twitter API using the messageID as a reference to request information about the respective message. After this, a time sequence is calculated using the earliest tweet as a reference, delays are calculated using the formula described in

the Timpstamp section. After the delay has occurred (amount of time between first tweet's timestamp and current timestamp) the data from a particular tweet is converted into a JSON format and sent to the endpoint. Any tools connected to that endpoint would then be responsible for processing the data.

Output data

The output data is sent via a connected socket. The simulator begins by opening a listening port and then sends the processed data to any applications connected to that socket for the given tweet's timeline. The format of the output of the simulator uses Javascript Object Notation (JSON). This entails that any application wishing to leverage the simulator should be capable of handling this format. However, this format could be replaced or additional formats added in future releases.

TWITTER SIMULATOR USE CASE

In order to test the effectiveness of our Twitter simulator we deployed a simple prototype in an active 911 public service answering point (PSAP). This center was located in a city with a population over 100,000 people. The first visit took place in May of 2018, with the primary goal on initial discovery. That is, during this visit we were interested in gaining a better understanding of how 911 centers operate and if social media could provide value to the existing processes. The primary goals of the first visit was to conduct observational research, interviews and workshops in order to discover what tasks and procedures were used. In addition, we also investigated how social media is currently used, and the possible future avenues of its use within the 911 environment. Lastly, challenges involving communicating data obtain from social media and potential other sources were discussed. These interviews shed light on how the data from social media should be displayed and presented for use by the center. It also helped us understand the importance of specific pieces of information, in particular the following categories emerged: what was the location or "where" was the person making the tweet, what was the timeline of the incident or "when" did it occur, "who" was involved in the incident, "what" was the nature of the incident, where there "weapons" or use of weapons mentioned and "why" had the incident occurred. These information points were referred to as the 6w's.

We then conducted an in person, informal workshop. This involved simulating data using the 2013 airport shootings in Los Angeles USA dataset found in the CrisisLexT26 dataset from crisislex.org (Olteanu, Vieweg, and Castillo n.d.). As the airport shooting occurred at 9:24 am only the hours of 8am – 12pm were used. A timescale of 15 was used meaning that the total simulation took 16 minutes. The first reference to LAX and shooting did not occur until the fifth minute of the simulation. This was used across three different participant groups. In addition to monitoring and examining how the prototype was used, we also observed the interaction between the participants to determine how the data from social media may be integrated into the existing process and if any disruptions had occurred.

In order to mimic the PSAP's existing procedures, sets of three participants were used, one as a traditional call taker, one as a social media analysist and one as a dispatcher. Each of these participants were 911 professionals working in the call center with multiple years' experience. All participants had access to the social media prototype interface user interface.

The social media prototype GUI was built on the ELK stack. The ELK stack is an open source software platform consisting of three primary components (https://www.elastic.co/). The first is a mechanism to import data from one format into another. In our case this was the role of the simulator tool, which imported data from the T26 LAX shooting dataset into the stack. The second mechanism is the elastic search analytics engine, which performs similar to a database. The third component was Kibana which is a user interface designed to display information from the elastic search environment. This allows you to create custom dashboards and interfaces. An example of this interface can be seen in Figure 2.

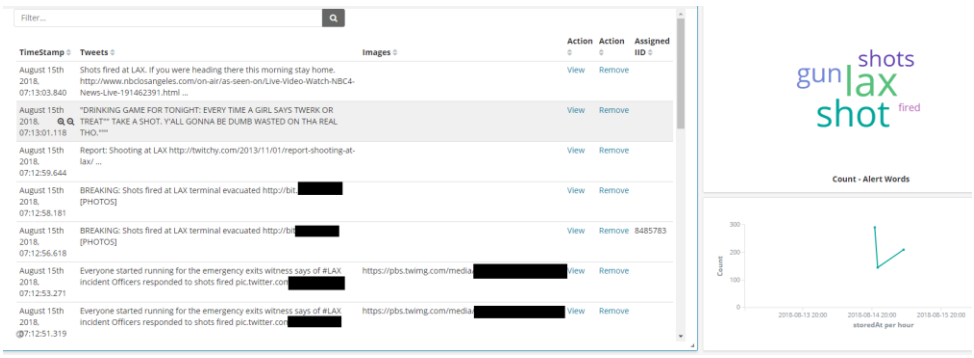

Figure 2 - Example Interface

Figure 2 illustrates a simple user interface developed using Kibana. While there is much room for improvement the goal of this work was to develop a tool capable of simulating Twitter data collection in real time. From the figure we have three primary data points displayed. The first is a table containing a list of tweets (which updates in real time) along with the associate timestamps, and image links. The other columns, action (view), action (remove) and assign incident id (iid) were used in conjunction with our prototype. The first action (view) allows the user to view the details of the tweet with second action (remove) allowing the user to remove a particular tweet from the database (Elasticsearch). Lastly, assigned IID allowed users to connect a tweet to a particular incident within their system. The assig IID was for display purposes only.

Overall, the participants were impressed with the ability to present previous data in a real-time format. Unfortunately, they did say that there was too much data being presented to fast. It was then explained that the data they were seeing was at an accelerated rate. Two recommendations to combat this issue were discussed and presented. The first was to narrow down the data further to show 10-15 minutes before the incident and 30-40 minutes after that. This was suggested, as they (crisis responders) would only really be interested in the event onset and possibly any victims. Once the suspect had been apprehended, further data would not necessarily be required. While we only used the airport shooting data during our workshop it is highly suggested that further work is done to test slow onset crisis such as floods, and hurricanes.

The second recommendation was to filter out more noise and allow for custom searches. As the tweets shown to the users were based on the dataset, no further filtering was implemented. Note that the datasets were collected using various keywords and collection methods originally. While this could be a side effect of the rate at which the tweets were been shown, allowing custom filtering to be setup within the simulator would allow users to fine tune the data that is shown during training exercises.

The last recommendation was to use a localized dataset. While the LAX dataset did have some clear useful information in it, those who were unfamiliar with the lingo of area may not be able to identify points of interest. This was found to be especially true when images were presented, as the PSAP was not located in the LAX area, they were unable to identify where the picture may have been taken. Additional examples of images were presented in another session to which the crisis responders were quickly able to identify where the picture was taken.

CONCLUSION

Previous research done on datasets has mostly looked at datasets as a whole in identifying patterns within the messages. These patterns are then used in the design of new methods for data collection and analysis. While this technique has been somewhat successful, two primary issues still exist. The first is that when messages are collected in real-time access to the overall picture, or the full dataset, is not available until after the event has occurred. The developers of these tools must then examined how each of them performed iterate the design and then wait for another event to occur to test the changes. Secondly, with respect to training, Twitter contains a vast amount of information, with many unique ways of sharing information. Unfortunately, developing a tool that attempts to create these messages for training purposes becomes less than ideal as different areas or languages may report things in different ways. Lastly, by involving the 911 operators in the design the prototype we can discover key data points and information that they would use within the existing process.

In order to address these issues, this paper describes the design and creation of a tool to present and simulate Twitter message from previously collected datasets. By doing this, the first issues is combatted the tweets are presented based on their timestamp meaning that tools connected to the simulator would not be able to analyze the full dataset at once and instead only receive data up to a given time point with new data flowing in as time progresses. This temporal delivery can also serve as a method for testing a systems robustness and assist in discovering the limits or requirements of an analysis application. A Twitter simulator can also assist as an input to existing training tools. Because we are leveraging existing datasets, actual tweet messages are used. This means that the issues of tweet fabrication can be overcome as the tweets are written by humans and not auto generated.

In being able to test this simulator with actual crisis response professionals, we gained valuable insight into the design. The following design recommendation were discovered

- If training or humans are being used to analyze data, a timescale for accelerated tweet delivery should be used very cautiously
- The timescale feature should be used only for computational analysis and tool testing
- Crisis response professionals would only need to see a very time limited margin of tweets to assess the value of the data depending on the dataset
- Consider incorporating custom filtering into the simulator for training purposes
- When at all possible use localized datasets

The design and development of this Twitter simulator is still ongoing, with many features to be added in the future. First with respect to the data input, this system would benefit from the ability to connect to databases as

well as accept input files in different formats besides the .csv datasets used. Likewise, datasets from social media platforms other than Twitter would need to be incorporated into the design depending on the availability of their APIs. Next, while intentionally left out of our design, incorporating filter methods could also be added for those wishing to use the simulator without needing to connect it to an existing application. Lastly, while we selected the JSON format as our output due to familiarity, additional output formats would be needed including; database connection, REST calls and raw file outputs.

REFERENCES

Abbasi, Ma, and Shamanth Kumar. 2012. "Lessons Learned in Using Social Media for Disaster Relief-ASU Crisis Response Game." *Social Computing, ...*: 282–89. http://link.springer.com/chapter/10.1007/978-3-642-29047-3_34.

Alexander, David E. 2014. "Social Media in Disaster Risk Reduction and Crisis Management." *Science and Engineering Ethics* 20(3): 717–33.

Avvenuti, Marco et al. 2014. "EARS (Earthquake Alert and Report System): A Real Time Decision Support System for Earthquake Crisis Management." *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*: 1749–58. http://doi.acm.org/10.1145/2623330.2623358.

Baumgart, Leigh A, Ellen J Bass, Brenda Philips, and Kevin Kloesel. 2006. "Emergency Management Decision-Making during Severe Weather." *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50(3): 381–85. http://pro.sagepub.com/cgi/content/abstract/50/3/381.

Bruns, Axel, and Yuxian Eugene Liang. 2012. "Tools and Methods for Capturing Twitter Data during Natural Disasters." *First Monday* 17(4): 1–17.

Caragea, Cornelia et al. 2011. "Classifying Text Messages for the Haiti Earthquake." *Proceedings of the 8th International ISCRAM Conference* (May): 1–10.

Chowdhury, Roy, Soudip Roy Chowdhury, and Carlos Castillo. 2013. "Tweet4act : Using Incident-Specific Profiles for Classifying Crisis-Related Messages." *Proceedings of the 10th International ISCRAM Conference* (May): 834–39.

Crooks, Andrew, Arie Croitoru, Anthony Stefanidis, and Jacek Radzikowski. 2013. "#Earthquake: Twitter as a Distributed Sensor System." *Transactions in GIS* 17(1): 124–47.

Halse, Shane E., Andrea H. Tapia, Anna C. Squicciarini, and C Caragea. 2016. "An Emotional Step Towards Automated Trust Detection in Crisis Social Media." *ISCRAM 2016 proceedings*.

Hampton, Andrew J et al. 2017. "Constructing Synthetic Social Media Stimuli for an Emergency Preparedness Functional Exercise."

Heldal, Ilona, and Cecilia Hammar Wijkmark. 2017. "Simulations and Serious Games for Firefighter Training : Users ' Perspective." (May): 868–78.

Jin, Y., B. F. Liu, and L. L. Austin. 2011. "Examining the Role of Social Media in Effective Crisis Management: The Effects of Crisis Origin, Information Form, and Source on Publics' Crisis Responses." *Communication Research* 41(1): 74–94. http://crx.sagepub.com/content/41/1/74.abstract.

Kavanaugh, Andrea L et al. 2012. "Social Media Use by Government : From the Routine to the Critical." *Government Information Quarterly* 29(4): 480–91. http://dx.doi.org/10.1016/j.giq.2012.06.002.

Kluckner, Sigmund, and Willi Wendt. 2014. "Designing for the User : Tailoring a Simulation Software Interface to the Needs of Crisis Managers." (May): 528–32.

Kumar, Shamanth, Geoffrey Barbier, Mohammad Ali Ali Abbasi, and Huan Liu. 2011. "TweetTracker: An Analysis Tool for Humanitarian and Disaster Relief." *Fifth International AAAI Conference on Weblogs and Social Media*: 661–62.

MacEachren, Alan M. et al. 2011. "SensePlace2: GeoTwitter Analytics Support for Situational Awareness." *VAST 2011 - IEEE Conference on Visual Analytics Science and Technology 2011, Proceedings*: 181–90.

Olteanu, Alexandra, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. 2014. "CrisisLex: A Lexicon for Collecting and Filtering Microblogged Communications in Crises." *Proc. of the 8th International Conference on Weblogs and Social Media*: 376. http://www.aaai.org/ocs/index.php/ICWSM/ICWSM14/paper/download/8091/8138.

Olteanu, Alexandra, Sarah Vieweg, and Carlos Castillo. "What to Expect When the Unexpected Happens : Social

Media Communications Across Crises."

Palen, Leysia. 2008. "Online Social Media in Crisis Events." (3): 76–78.

Parush, Avi, Ruth Libkind, Shani Laendler, and Tal Solomon. 2016. "Simulator and Game-Based Multi-Level Training of Cognitive Skills and EMS Teamwork in Multi-Casualty Incident Management." (May 2016): 2016.

Sakaki, Takeshi, Makoto Okazaki, and Yutaka Matsuo. 2013. "Tweet Analysis for Real-Time Event Detection and Earthquake Reporting System Development." 25(4): 919–31.

Schaefer, K. E. et al. 2014. "A-Meta-Analysis-of-Factors-Influencing-the-Development-of-Trust-in-Automation-Implications-for-Human-Robot-Interaction.Pdf." *Human factors* 58(3): 377.

Walker, Warren E, and Jordan Giddings. 2011. "Training and Learning for Crisis Management Using a Virtual Simulation / Gaming Environment." : 163–73.