Riskr: A Web2.0 platform to monitor and share disaster information

Jess Farber[a], Trina Myers[a,b], Jarrod Trevathan[c], Ian Atkinson[b], and Trevor Andersen[a]

[a] School of Business (Information Technology), James Cook University, Townsville, Australia.

[b] e-Research Centre, James Cook University, Townsville, Queensland, Australia.

[c] School of Information and Communication Technology, Griffith University, Brisbane, Queensland, Australia

.

***Abstract*—Disaster management that uses Web-based technology to enhance user collaboration around disasters is an emergent field. A number of dedicated 'disaster portals' exist but do not integrate large social networks such as Twitter and Facebook. These social networking sites can facilitate the analysis and sharing of collective intelligence around disaster information on a far greater scale by increasing accessibility to, and the use of, a disaster portal. This paper presents the 'Riskr' project, which applies a low-technological solution to creating a disaster portal fed by social networking messages and the strategies used in its development. The system has been implemented using Twitter and tested by users to determine the feasibility of having interoperability between social networks and disaster portals. Preliminary results suggest there is benefit in using Twitter as a middleware between users and the disaster services with 70.5% of the users able to estimate the location of a disaster (e.g., fallen powerlines, fire, etc) within a certain error margin. Furthermore, 95% of users were able to successfully adapt to using the system. The results from the Riskr project suggest that the combination of online services and interoperability between disaster portals and social networks can further enhance disaster management initiatives.**

Keywords-Web2.0, disaster management, social networks

I. Introduction

Communication is paramount in a disaster situation. Basic services and infrastructure such as electricity, food, water and the Internet, can be put under immense strain, or can even be disabled completely. These services are often taken for granted. However, in reality, the coordination and management of such services requires significant effort and timely implementation. Therefore, streamlined synchronisation for effective communication is critical.

The term 'disaster' can encompass a large number of possible events or occurrences. In this paper, the term 'disaster' refers to a significant hazard (e.g., Fallen powerlines, fire, flood, etc), affecting multiple people in terms of geographical area such as an entire suburb or district. During a disaster event, affected individuals need to voice concerns, inform others and share information relevant to their area. Likewise, emergency workers must be able to access pertinent information to prioritise high urgency situations and keep residents updated about the status of the recovery effort.

Disaster management could benefit from the information available via social networking forums. In 2011 and 2013, the Australian state of Queensland was impacted by floods and in 2011, a severe tropical cyclone. While there was significant media coverage, social networks such as Twitter and Facebook, were also inundated with posts related to the floods [3]. A government social media outlet, '@QPSmedia' was created and utilized for community interaction regarding the disasters. The creation of this outlet succeeded in nurturing community discussion and involvement and resulted in a greater ability for emergency services to communicate with the public [4]. There are potential benefits to disaster management in the use of information harvested from social networks for improved communication and dissemination to the wider community during and around disaster events.

The propagation of disaster-related information is well suited to a Web2.0 solution, particularly the use of social networks. The information shared on social networks about a disaster within 24 hours after the event has occurred, in many cases is more accurate than mainstream media [5, 6]. This phenomenon is due to the ability of social networks to facilitate collaboration between affected individuals and serve as a central point to correlate disaster information. As a result, rapid dissemination of first hand disaster-related information is made possible, which can greatly aid disaster management and control.

This paper presents the Riskr project, which provides a disaster portal fed by social networking messages. The purpose of Riskr is to explore the viability and construction of a portal for reporting and visualising disasters that harvests information from existing social networks. The Riskr framework was implemented using Twitter to demonstrate the capacity of large-scale user participation in a dynamic system with little or no administration required. User testing was conducted to evidence the merit of having interoperability between social networks and disaster portals. . The usability study showed 70.5% of the users able to estimate the location of a disaster within a certain error margin and 95% of users able to successfully adapt to the syntax of the system due to the interface design that requires low technological knowledge.

This paper is organised as follows: Section 2 presents an overview of the relevant Web2.0 technologies and the problems faced in communications during disaster situations. Section 3 discusses related work and the differentiation to this project. Section 4 describes the Riskr architecture and the technologies used in the development to enhance communication efforts in disaster situations. Section 4 outlines the results of a user study to determine the feasibility of using a social network's collective intelligence to learn more about an incident. Section 5 concludes the paper with a discussion on the implications the Riskr system for disaster management.

II. BACKGROUND - SOCIAL MEDIA

Web2.0 refers to the 'read/write Web' that promotes participation, collaboration and sharing of information among users online [7]. Social networks provide easy access to the collective intelligence of the community. The *collective intelligence* is defined here as the combined knowledge of an entire user base and user participation adds the each social network's collective knowledge [5, 8]. The recent explosion of social media and networks has seen many web sites such as Facebook, Twitter and Flickr grow to have millions of users frequently posting information. Some of this information would possibly be valuable to the management and mitigation of local disasters and crises but how can relevant information only be selected? This section describes the decisions surrounding the development of the Riskr framework based on a review of the literature and related work in the context disaster management and popular social networks. The review included analysing and comparing the available social networks that would be conducive to disaster management, specifically, Twitter and Facebook. Evidence is provided to support the Twitter social network as more suitable for analysing collective intelligence on a disaster.

A. Analysis of Twitter and Facebook capabilites to support disaster management

Twitter [9] is a popular social network that allows users to post 140 character messages publicly known as 'Tweets'. Tweets may contain links or photos [10], and have an incredibly diverse range of topics. These tweets may be available to anyone viewing the page, or show up in the feed of 'followers'. Followers are other users that have subscribed to a particular users' posts. Sometimes a tweet will have a 'hashtag' which appears as a word, preceded by a hash character '#'. These hashtags often contain a subject or word relevant to the conversation. For example, while talking about a fire in the fictional suburb "Bar" in 2011, one may post the hashtag "#BarFire2011".

Facebook [8] is another social network where users add 'friends' to subscribe to their posts and photos as opposed to Twitters 'followers'. Facebook profiles allow people to see all messages sent between subscribed users on an interactive message board known as the 'wall'. Walls are a method for users to publicly share messages and information. Facebook also incorporates 'groups' which allow users to converse and share information privately about certain topics. Both Twitter and Facebook are successful social networks, however for the purpose of this project, only one was chosen for integration with the disaster management framework presented here.

There is strong evidence that social networking is a good source for disaster information and updates during the first stages of a disaster [5, 8, 11]. The information found on social networks can often be higher quality than that available via journalism and other mainstream online media [6]. *Information quality* is defined as the degree of reliability, accuracy, relevancy and usefulness of information [6]. A number of studies have shown that during the initial 24 hours of an incident or disaster Twitter generally has higher information quality than mainstream journalism sources [5, 8, 11, 12].

Twitter and Facebook are currently two of the most popular social networks. A comparison of the relevance of these social networks to disasters was explored. Heverin et al [5] analysed Twitter responses after the shooting of several police officers in Seattle in 2009. Similarly, Vieweg et al [8] analysed Facebook responses after the 2007 at Virginia Polytechnic Institute shooting. 6013 tweets were analysed by Heverin et al [5], there was no documented size of the user base or amount of posts in the study conducted by Vieweg et al [8]. The results from these studies were compared in order to determine how appropriate Twitter and Facebook are for disseminating disaster-related information (Table 1).

TABLE I. COMPARISON OF RESPONSE BETWEEN 2007 VIRGINIA TECH SHOOTING ON FACEBOOK [8] AND THE 2009 SEATTLE POLICE SHOOTING ON TWITTER [5].

Social network	**Facebook - Virginia Tech 2007**	**Twitter - Seattle 2009**
Organisation	Organized through Facebook groups such as "I'm ok at VT"	Organized through hashtags, such as #washooting and #lakeshoot
Categorisation of posts	No categorisation of user inputs	Twitter messages (Tweets) were able to be classified into four categories: "information related", "opinion related", "emotion related" and "action related"

Categorisation of users	Categorisation of posters not possible	Posters were able to be categorised, for example "unaffiliated individual", "local media", "blog", etc.
Trust network	Trust network implied by other user's opinions - sometimes no solid proof available	No specific trust network defined, however participants included media and public service organisations
Decentralization	Almost complete disassociation from any form of control - however some users took on more responsibility in 'information broker' roles	All users were decentralized.
Effectiveness of collaboration	Very rapid deduction - the 32 deceased were accurately identified in less than 38 hours	No deduction mentioned in paper, however reported information spread quickly
Links to relevant media	Any posts including links were simply to other groups that had deducted something. Most deduction was from word-of-mouth and not solid evidence	A third of tweets included links, half of which were directed at local media websites. A large amount of users also were using URLs to post media (pictures, videos, etc)
Size of user base	No indication as to scale of user base	6013 tweets were found

Similarities and differences were identified between Facebook and Twitter from the two studies of social networking during disaster situations (Table 1). The organisation of Facebook collaborative intelligence can be accomplished through 'groups'. Alternatively, collaborative intelligence can also be organized through 'hashtags' in Twitter. Categorisation of posts and inputs was not implemented in the study, however Heverin et al [5] were successfully able to group users and tweets across a meaningful spectrum of categories.

Facebook contained no implemented trust network. A crudely yet effective form of group trust was enforced through collaboration of user's opinions. Twitter was also found to have no trust network, but did include trustworthy contributions from members of the media and public service organisations. Although Facebook showed decentralisation, individual users can self appoint themselves to a higher responsibility in organising and culling information in the Facebook group. Conversely, Twitter posters were subject to no hierarchical control and all tweets were from individual, sometimes disassociated users.

A comparison of the quality and validity of the posted information showed both Facebook and Twitter were effective at deducing and reporting correct information at a faster rate than that of the mainstream media. Users of the Facebook group posted links and information to other groups, but the validity and importance of these links were questionable because many links were to other groups with similar user bases. Further, most deduction came from a centralized control, which worked on the assumption that information shared by the users was honest and correct. Comparatively, a third of tweets included links, half of these were to local media websites, and a considerable number of the remaining tweets were URLs linking to media such as photos or videos.

Twitter is efficient for providing information that is not found on normal media outlets. It is not recommended for use as a backbone for a emergency management system that emergency services and civilians must rely on because a third party social network cannot guarantee the reliability required in an emergency situation [6]. However, Twitter is suitable for civilian use to find useful information such as local news outlets and relevant disaster information services. Advantages found of Twitter include the fast and easy user interface, scalability, mobility and diversity of information [6]. Disadvantages included less reliability than government or media sources due to unfiltered and unaccountable information.

At the time of this study, Facebook did not support GIS functionality but Twitter did; although the capabilities were somewhat limited. GIS capability allows for a larger amount of information to be derived from short messages. An effective example is shown by Sakaki et al [11] in their research in real-time event detection by social sensors, where an earthquake’s epicentre was calculated using user’s geolocation and the time of each post. The application of geolocation is relevant to disasters as it can provide context to a message with sparse information.

There is strong evidence that social networking is a good source for disaster information and updates, particularly within the first stages of a disaster occurring [6, 7]. While both Facebook and Twitter were demonstrated as effective tools for communication during disasters, the support for Twitter as a system to support disasters has been shown as more effective. Twitter supports geolocation and classification of users and when combined with the decentralized nature of the network makes it superior as a disaster management and response system. For these reasons, Twitter was chosen as the social network platform for the implementation of the Riskr framework.

B. Government and organisational use of social networks

As more people use social networks as a medium to report and send information about disasters, government and/or public service groups would benefit from the available information. Valuable two-way interactions are possible between these disaster management organisations and the public over these networks. Mills et al [6] identified large relief organisations, local and municipal governments benefited by the updates and information the public sent to their subscribers on Twitter for relief operations. In many cases, individual social media users are classified as media outlets and public services due to their continual newsfeeds and updates [5].

Government services and organisations appear active on Twitter [6, 12], while conversely, Vieweg et al [8] implied that government and public service involvement in the users' Facebook groups was minimal. The significant observable activity from government, media and public services on Twitter is most likely due to the ability to use an organisation wide Twitter account as a social media outlet. The government services activity was not as excessive on Facebook as the collaborative deduction was based around citizen-organised groups with little to no government input [8].

C. *Information reliability*

Studies indicate a trend where the most important information posted on social networks of an event is created in the first stages of the occurrence. The quality of information is defined here as its reliability, accuracy, relevance and usefulness. Mills et al [12] established that Twitter has a higher degree of quality information than journalism/mainstream online media within the 24 hour period directly after an incident or disaster. Likewise, Vieweg et al [6] describes a similar duration of 24 hours from the shooting incident where users accurately reported the number of victims and the circumstances., Heverin et al [8] adds support to these claims where they found that the tweets sent in the first twelve hours were the ones classed as the highest information quality. In the first 12 hours, the number of opinion-driven tweets were a low 6.8% (6.8%) but increased to 21.3% after the 26 hour period s. Opinion-driven posts are those that consist of disagreements, name-calling and personal attacks, which are irrelevant information for use by a disaster portal. Sakaki et al[5] created a system that relies entirely on users' immediate response to incidents with a precision of hours, supporting the importance of timely information. This evidence indicates that any data mining or presentation of disaster social networking must allow and encourage fast transmission and immediate access to information.

D. *Techniques of retrieving useful information from Twitter*

During a disaster situation, some data is relevant and others are meaningless so deriving useful information can be complicated. As of 2011, Twitter reports that there are over 1 billion tweets per week[11] so prioritising and sorting relevant information is important for a disaster portal. Sorting through continual stream of posts would be highly labour intensive so automated retrieval of quality information is required for the efficient capture of disaster information[13].

Barbosa et al[14] proposed a novel way to derive sentiments (positive and negative) from tweets, by creating abstract representations of Twitter messages, and combining several noisy and biased data sources as training sets. They found their method could accurately determine sentiment with an error rate of approximately 20%. Castillo et al[15] also successfully classified tweets as credible or not by classifying messages as news or conversation and then analysing the trends and relationships between the words included in the Tweet. This solution can help in spam detection and identifying trusted sources.

There is a large amount of research into deriving tweet sentiments [16] from Twitter, however there have also been successes in categorising tweets in more content-specific means. Benevenuto et al [15, 17] display a novel approach to classifying spammers on Twitter using classification. Caragea et al [18] have created a system that includes a component that they have proven to successfully categorise short tweets sent during the Haiti earthquake.

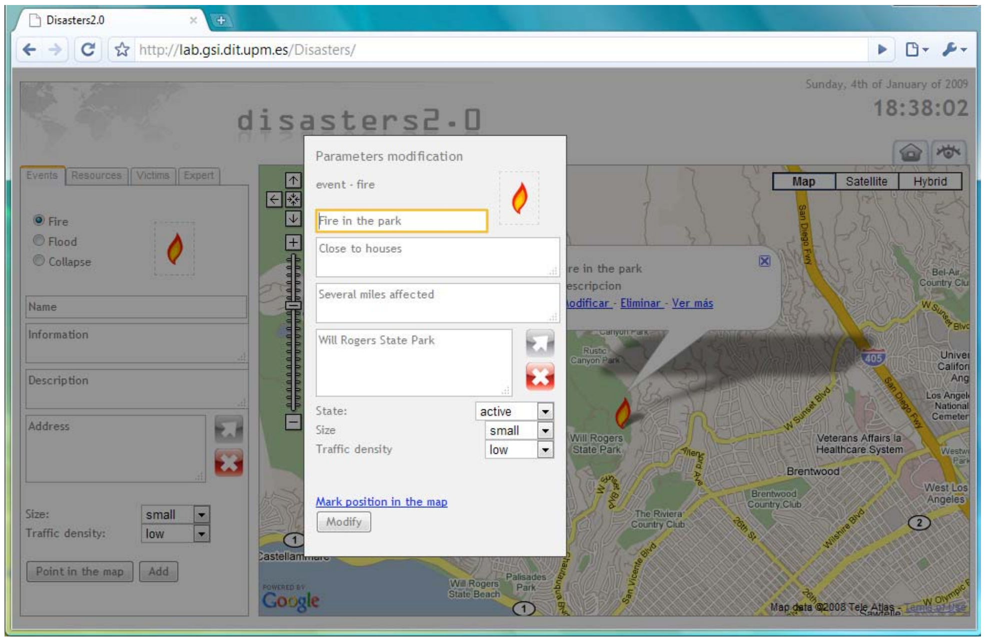
Although not published, Karandikar's thesis [19] provides an interesting alternative to classification, a clustering method to allow similar topics to be clustered together. While this is a novel approach and can be applied to certain situations, as discussed earlier a disaster portal must be updated constantly and quickly. The nature of clustering does not allow for quick placement of tweets into relevant categories. Sankaranarayanan et al [20] have created a system for a similar goal of clustering topics in Twitterstand, and displaying Twitter as a news feed. In their study, they mention although this is presented as a news service, there are many other applications for their methods.

Mills et al[21] noted that a possible method of finding location could be managed by transmitting GPS coordinates in tweets, and Abrol[6] also proposed a system to use Twitter's geo-content to order news search results. Twitterstand[22] includes a system to geographically locate tweets and group news items through this method.

Starbird et al [9] have put forth the notion of using 'prescriptive syntax' where hashtags mark information such as #city, #addy, #contact, etc. They predict that information posted in this way would be useful for both emergency services and the public, and one user's adoption of this method would encourage others to follow the same format. Twitter could be used as a middleware in a Web application with minimal analysis or data mining work to create, retrieve and group these tweets using the prescriptive syntax method. For this reason, the use of prescriptive syntax to pass messages through Twitter to the Riskr web application was chosen, as it would adequately demonstrate whether implementation of a disaster portal using social networks is feasible.

III. RELATED WORK - CURRENT ONLINE DISASTER SERVICES

Disasters2.0 [21] and the RESCUE disaster portal [2] are two examples of disaster portals that encompass Web 2.0 features. Web2.0 is the new generation of web-based services and communities characterised by participation, collaboration and sharing of information among users online [1]. Disasters2.0 provides a method for users to report and describe nearby


http://lab.gsi.dit.upm.es/Disasters/

Sunday, 4th of January of 2009

18:38:02

Parameters modification

event - fire

Fire in the park

Close to houses

Several miles affected

Will Rogers State Park

State: active

Size: small

Traffic density: low

Size: small

Traffic density: low

Figure 1 - Disasters 2.0 [2]

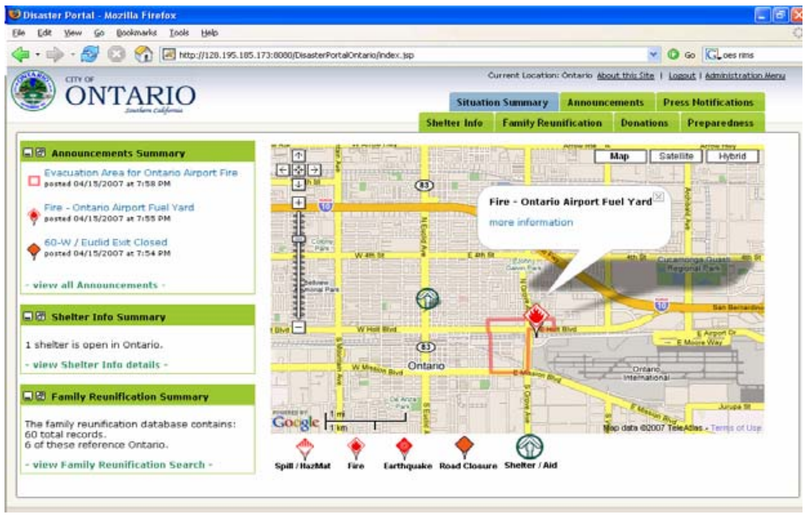
disasters (Figure 1). However, at the time of writing, this portal did not support the dexterous grouping of collective intelligence from social networks. Disasters2.0 is a disaster portal for users to post information and view information posted by government services. In contrast, the RESCUE disaster portal (Figure 2) mentions Internet information monitoring, which is claimed to result in intelligent detection of relevant news events. Both Disaster 2.0 and RESCUE have mechanisms for user feedback, where the options given to users to report incidents focus on their individual posts and responses. There is no mechanism for the conglomeration of the users' submissions to a single conclusion.

The Web application services available in Disasters 2.0 and RESCUE focus on simple interaction with emergency services to provide information to users. Similar to the social networking sites, Disasters 2.0 provides basic tools for collaboration but it does not allow collective deduction or news reporting. There is potential to extend these models to produce collaborative disaster portal frameworks based on an already implemented social network.

A. *Extending the portal with the combination of existing technologies*

The common components of the disaster portals evaluated and the benefits of extending with social networking resources include:

- *A map interface*—the aforementioned disaster portals include a map interface to visualise information geographically.


Disaster Portal - Mozilla Firefox

http://128.195.185.173:8080/DisasterPortalOntario/index.jsp

Announcements Summary

Evacuation Area for Ontario Airport Fire
posted 04/15/2007 at 7:58 PM

Fire - Ontario Airport Fuel Yard
posted 04/15/2007 at 7:55 PM

60-W / Euclid Exit Closed
posted 04/15/2007 at 7:54 PM

Shelter Info Summary

1 shelter is open in Ontario.

Family Reunification Summary

The family reunification database contains:
60 total records.
6 of these reference Ontario.

Fire - Ontario Airport Fuel Yard

Figure 2 - RESCUE disaster portal [1]

- *A news feed*—relevant news must be given to the user. The disaster portals evaluated included a news feed. A collaborative social disaster portal would also derive useful information from social networks as news information.
- *Ease of collaboration*—users' submissions must be employed to generate conclusions or add to evidence of other information. Notably, users should also be able to post anonymously or log posts through a commonly used service (e.g., Facebook, Twitter, OpenId)
- *A disaster geo-location*—the location of a disaster is incredibly important. Disasters 2.0 allows plotting of a disaster on a map. However, a versatile disaster portal should be able to take distance references and derive valid locations from a collection of posts from multiple users.
- *Accessibility*—all current disaster portal implementations are available via a Web browser and some are available via mobile applications [7, 23]. Access through social networks could allow contributions and retrieval of information without the installation of an application or visiting the portal's web site.

With these goals in mind, the approach taken by the Riskr project to harvest disaster-related information from social networks was to have Twitter tweets posted using *prescriptive syntax*. Prescriptive syntax uses hashtags to mark information in the following format:

#city, #address, #contact

Twitter was used as a middleware for the passing of the information within the prescriptive syntax. The idea was that one user's adoption of this method would predicably encourage others to follow the same format [1, 2, 24].

To achieve this, the Twitter platform offers access to the corpus of data via *Application Programming Interfaces* (API), which we can compare to the the *REpresentational State Transfer* (REST) API used in the Disasters 2.0 portal. The Twitter API enables developers to access some of the core primitives of Twitter including timelines, status updates, and user information [9]. Accurate geo-locations are a challenge if using Twitter as a middleware. Twitter, by default, only posts geo-locations refined to city or suburb, which does not allow for a truly accurate location. The location of incidents would be significantly more precise if users have the ability to post their location as content in the tweet [9, 10]. The precise location of a disaster can be derived by forming a *trilateration* from information posted by Twitter users by combining geo-location and prescriptive syntax. Trilateration determines an absolute or relative location of points by measurement of distances using geometry.

IV. THE RISKR FRAMEWORK

The development of the Riskr disaster portal incorporated the best practices found in the current portals combined with features that take advantage of the social network and map-based frameworks. An interface with minimal technical complexity was required for user accessibility and interactivity, while still offering features geared towards enhancing user collaboration.

The workflow for data from Twitter to and from the Web portal is illustrated in Figure 3. Usually, a server will communicate directly with its Web portal and mobile application. However, in this case all information is passed through

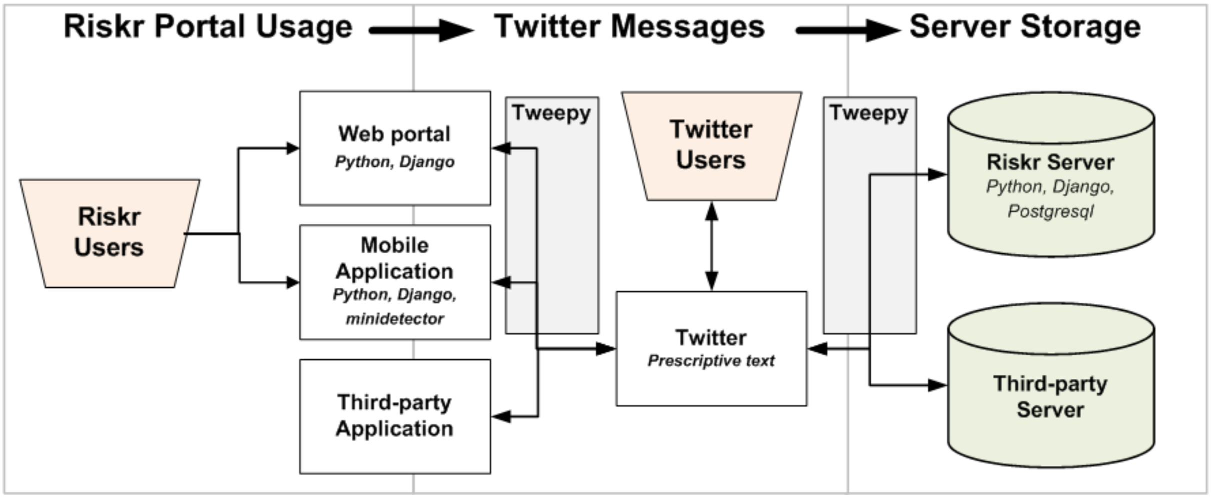


Figure 3 - Disaster portal framework and technologies.

Twitter. Information posted via a social network instead of just through private channels will open the service up to users of the Web portal and mobile application and any other users of the social network. The general usage of the service has the potential to expand from the niche group that uses the application or Web portal to the entire user base of the social network.

The workflow in Figure 3 begins with users posting information related to a disaster via the web portal. Then, the web portal, mobile application, or any third party application can post to Twitter using prescriptive syntax. Any Twitter user is able to see these posts, and make similar posts using prescriptive syntax. In the final step, by watching for prescriptive tweets, the Riskr server, or any other third party server can watch for information to use in disseminating meaningful information.

The complexity of the underlying framework is transparent to the user. A Web server handles the Web service requests from user interaction via a map interface and from the Twitter feeds. To provide interaction with Twitter, the Web services must communicate to the Twitter service using the Twitter APIs. In addition, a Twitter news feed was implemented in the portal's Web interface so users can view and collaborate with others. Finally, in conjunction with the map interface, an incident location algorithm is used to estimate an accurate location of incidents.

A. The web server

The Riskr Web server provides the interface and performs the transactions with the database. The most support for API libraries and most Twitter-based documentation is based in Python [12]. The Django [25] Web framework was used to convert Python easily into a functioning web service. Django allowed easy creation of the Web interface, handling of requests, database modelling and debugging. The python Twitter API library 'Tweepy' [26] was chosen to support the interaction between the server and Twitter due to its rich feature set. The feature set includes support for Twitter streaming and *Open authentication* (OAuth) [27], which is a method of authentication that allows services to securely and safely send private login details.

The database stores a large amount of information related to tweets and disasters. This information includes:

- The entire contents of all relevant tweets;
- User information;
- Different disaster types and the possible warnings, messages and keywords that may appear in tweets;
- Regional information with region geospatial location and name for use in naming incidents; and
- A log of each detected incident. Incidents are allocated a name, a hash tag, the size or warning area, a warning message, and the disaster type.

This information is stored in a PostgreSQL relational database. The PostgreSQL database engine is widely supported by the Django community. Future work will incorporate a more scalable solution (e.g., NoSQL) to increase speed and flexibility.

B. The map interface

Google maps [28], Yahoo maps [29] and MapQuest [30] APIs all have the JavaScript support required for a Web portal. They all include functions and techniques for applying intelligent mark-up to the map to provide information. There is very little difference between the map services, so the choice of Google maps here was an arbitrary decision based on the familiarity with the technology.

Google maps provide many functions to aid in the creation of the disaster Web portal. Most notably, custom markers, which are a function that conveys any image and can apply action listeners to each. Infowindows is another important function, which can be modified to pop up on events and can include Web forms. jQuery [31] was included for asynchronous mark-up of the JavaScript map. Finally, geocoding and reverse geocoding services, which are processes used to convert addressing information (e.g., street number, street name, suburb, etc.) into geographic coordinates and vice versa.

C. News feeds

A Twitter news feed displays live data from users posting relevant information. The news feed is capable of asynchronously changing its search terms, which allows users to narrow the list of tweets to ones directly related to a particular incident.

The simplest method of displaying a twitter news feed is to use a Twitter widget provided directly from Twitter. An alternative is to use a custom-created widget that draws information from the database. The Twitter widget was chosen, as the amount of functionality it affords is sufficient to provide the relevant tweets to an incident. The search terms can be narrowed down using search terms specific to the prescriptive syntax used to report incidents.

D. *Prescriptive syntax and Twitter integration*

The user base of the Riskr portal is open to all Twitter users, which is a benefit of enlisting the social network site. Posts can be pulled from Twitter to the Riskr server, and in turn, the server is capable of making posts to twitter. This way, information that is posted through the Web portal is placed into a Tweet and sent publicly through a preregistered Twitter account. In parallel, the server is used to detect relevant messages about incidents based on the hashtags and can add this information to the database.

The format of prescriptive syntax roughly follows the guidelines described in 'Tweak the Tweet' [32]. This method uses a hashtag as a variable. The server constantly reads the tweet stream and when a relevant tweet is submitted, it will filter the data into the database. The prescriptive syntax for the hash tags are shown in Table 2.

TABLE II. PRESCRIPTIVE TWEET FORMATTING AND HASHTAG DESCRIPTIONS

Information	**Description**	**Example**
Disaster Keyword	This variable consists of one hash tag representing the type of disaster	"#fire", "#flood", "#riot",
Address	This section of the tweet consists of the address tag, followed by an address.	"#address 20 Foo st, Bartown, QLD"
Distance	The distance away from the disaster in kilometres	"#distance 1"
Location	The geolocation of the tweet, in Latitude, Longitude	"#location -19,146"
Incident	The hashtag relevant to a current incident	"#BartownFire2011"

To ensure that the service is accessible, all hash tags can be omitted from the tweet by the user to encourage them to participate. Depending on the tweet content and the details omitted, the tweet may not be useful to the service for calculation of the incident location. However, participation is important and further information could possibly be derived from non-location data, such as user sentiment, tweet trends and descriptions of the disaster. A major challenge is to fit as much information as possible in the 140 character limit. Considering this short nature of tweets, hash tags should not be bound to just one format, for example instead of '#address', '#addy' could be used to save space.

Currently, the Riskr system requires only these main details from tweet. However, any other information is stored in the database as tweet contents for possible future analysis. Once the tweet has been loaded into the database, the server derives information to link a user to other users, regions and incidents. If information about a particular incident, region or user are not available, the system is capable of dynamically creating a new instance of a disaster occurrence.

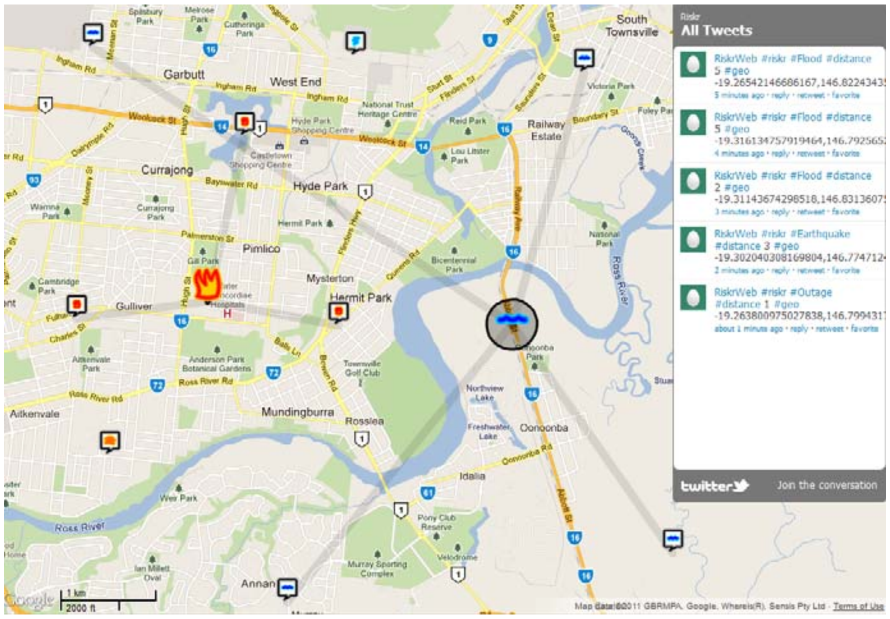
E. *Dynamic incident creation - the trilateration algorithm*

There are studies on the trilateration of incidents through estimation of at least three points and their distance[9]. However, finding an efficient and effective algorithm suitable for this solution was complicated due to the programming paradigms. An algorithm was developed for this system to find a central location and error margin from three or more inputs of location and distance (refer Appendix A). The location and error combination provides an estimate of the incident and a possible size or warning area.

Trilateration of an incident's location can be devised from the geo-location of the tweet source and the distance indicated in the tweets. The geo-location of a tweet can be found either from the tweet details, or from an address that the user posts in the content of the tweet. The more tweets about an incident, the more accurate the representation of the location of a disaster.

A simple algorithm was devised to determine the location of a disaster based on position and distance estimates. A single position and distance pair define a circle, with a centre at position, and distance as the radius. Three or more such pairs are required for trilateration. Standard trilateration, based on intersection of circles without precise data cannot be used reliably (e.g. three or more circles may fail to intersect where all the distances are underestimated). The developed algorithm employs 'hill-climbing' to find the best estimate of the disaster’s location. Given perfect position and distance information, the disaster’s location would fall on each aforementioned circle. Therefore, the best estimate is the point that minimises the sum of distances to these circles. The algorithm outputs that point, and the sum of quadratic errors of the distances. The error gives an estimate of the reliability of the data, and of the disaster’s area of influence.

A user's post will not necessarily relate to a nearby incident or tweet, which must be considered in the trilateration algorithm of Tweet distances. To ensure that separate incidents can be distinguished an error resistance variable is used, which still allows users a degree of flexibility. The error resistance variable allows a user’s guess to be incorrect. However, if the distance is below their guess divided by the resistance (currently set at a constant of three), or above their guess multiplied by the resistance the system will stop considering a relationship between their tweet and other tweets or incidents. In the current implementation, users can be incorrect to a factor of three. This constant is appropriate for a proof of concept such as that described in this paper, a clustering algorithm solution will be explored in future work.



Riskr
All Tweets

RiskrWeb #riskr #Flood #distance 5 #geo -19.26542146686167,146.8224343
5 minutes ago · reply · retweet · favorite

RiskrWeb #riskr #Flood #distance 5 #geo -19.316134757919464,146.792565
4 minutes ago · reply · retweet · favorite

RiskrWeb #riskr #Flood #distance 2 #geo -19.31143674298518,146.831360
3 minutes ago · reply · retweet · favorite

RiskrWeb #riskr #Earthquake #distance 3 #geo -19.302040308169804,146.774712
2 minutes ago · reply · retweet · favorite

RiskrWeb #riskr #Outage #distance 1 #geo -19.263800975027838,146.799431
about 1 minute ago · reply · retweet · favorite

Figure 4 - Riskr Interface with user tweets for two incidents (a fire and a flood) and the location approximated from the tweet information.

Incident hashtags/names are dynamically assigned once the incident has been located, to ensure users can all collaborate on a similar hashtag. This is achieved by applying the region name, disaster type, month and year as one string. In the event that a disaster type occurs more than once in the same region in a particular year, the hashtag can be amalgamated into region name, disaster type, 's' and year as one string. For example, BartownFireNov2011 will become BartownFiresNov2011, and still maintain its relevance when being used for posts. Incidents will appear separate on the server and on the map but users only need to refer to the one hashtag on Twitter.

1) Region

The Google maps geocoding service also includes a function called 'reverse geocoding', which allows a geolocation to be transformed into an address. The origination of a region name a particular tweet or incident can be found with this function. The latitude and longitude bounds of the region can then be found by geocoding the region name again. Then, by adding this information to the system, it can check if a geolocation is inside a particular region. To reduce API calls, when a tweet or incident with a geolocation is processed, the program checks to see if the location exists within the latitude and longitude bounds of any stored region. If the location exists, the system can link the object with the region, if not, the Google API is used to find the name and bounds of the region for storage in the database.

F. The Riskr interface

The Riskr system supports both a Web (Figure 4) and mobile interface. A universal hashtag was required for easy identification of the Riskr disaster service to the end-users. The length of the service name was significant as Twitter has a character limit, so the short name 'Riskr' was chosen. The name is original so a person using the hashtag will be referring to the correct Web service. The name also has a 'Web2.0' style name, similar to the successful Websites Tumblr [33, 34] or Flickr [35]. A reference to this service uses the format #Riskr in prescriptive tweets.

A mobile interface was developed for mobile and PDA devices, which is a simplified version of the Web interface. The mobile interface is easy to read and use on a small screen. Importantly, a mobile environment allows the use of a GPS system to find a user's geo-location.

G. Database implementation

A postgresql database was developed for the Riskr proof-of-concept to store the information derived from Twitter. The database schema contains seven entities; *DisasterType, DisasterKeyword, Region, IncidentRegion, Incident, User,* and *Tweet* (Figure 5). *DisasterType* represents any disaster that would be supported in the service. It contains the name of the disaster, fields for descriptions, messages and warnings, and a URL of a relevant image representing the disaster. *DisasterKeyword* represents keywords that could represent different disaster types, which is particularly useful for using as hashtags. Besides the disaster keyword, this entity also contains a reference to the disaster type and a Boolean value to indicate the preferred keyword used for describing a specific disaster. The *Region* entity stores the region's name, location, state, country, postcode, any relevant region information, and the geolocation bounds of the region, for use in naming disasters. The *Incident* table stores the actual occurrence's date of reporting, dynamically generated name and hashtag, location, size, warning area and

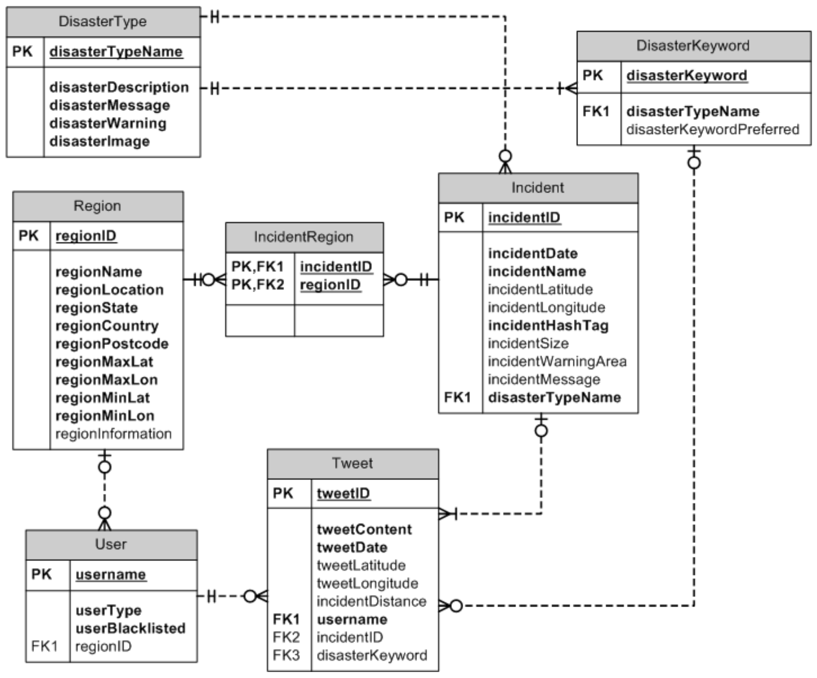


Figure 5: Entity Relationship Diagram

associated messages. The *IncidentRegion* entity bridges the Incidents and Regions entities. The *User* entity stores the username, the type of user (e.g., normal, disaster response, energy service, etc), whether the user is blacklisted, and optionally the region the user is located. The *Tweet* entity contains every tweet that is read by the service. It contains the raw content of the tweet, the date of posting, the location if provided, and the distance from an incident if provided. Tweets will always be linked to a username. Each tweet may be linked to an incident by the specific hashtag or if provided with a location and distance, which deems it relevant to a nearby occurrence. Commonly, the tweet will include a keyword hashtag that will link it through *DisasterKeyword* to *DisasterType*.

H. Asynchronous markup

Asynchronous updates to the current Twitter feed was required to allow for live updating of the map, so a periodic update solution was chosen for this pilot study. An important element to a Web 2.0 portal is live updating. In a scenario where updates happen constantly, the web site must be able to serve the user with updated information without refreshing the page. A format for a periodic update extension of jQuery proved[36] successful in loading new information dynamically. The updater loads information from a URL and checks if the information has been updated since the last checkpoint time. If the information is not different, the updater doubles its time between updates, up to a maximum amount of 8000ms. If the information has been updated, the update interval is reset to its minimum of 3000ms. Although the current values make little difference to performance, in a global implementation, this means that if a particular area has little to no activity, the updater will not be using as much load on the server.

The Riskr implementation of the periodic update queries the server for new tweets or incidents. With this information, Riskr uses Google's map to place the relevant icon in the correct position and attaches an Infowindow that contains content of the tweet and details of the incident. The map is also able to visually link tweets to an incident if they are related, and draws a circle to indicate the parameter of the incident.

There is no such limit on the amount of incident markers shown on the map, but to minimise clutter and improve performance, the service has been programmed in such a way to limit the amount of tweets shown at any one time. The Google maps function 'getBounds' can gauge the current viewing area, which ensures that the user is only downloading markup relevant to the viewable map. The Twitter sidebar is capable of showing Tweets related to the hashtag '#Riskr' and posts directed to '@RiskrWeb'.

V. Usability, results and discussion

An analysis of useability, accessibility and functionality was conducted after the project was tested and all services were working adequately. The sample of the population adopted for user testing comprised of a demographic breakdown of Twitter users. The testing methodology involved a questionnaire and interview for both quantitative and qualitative analyses. Specific elements of feedback that were important to the goals of the Riskr project were the result of the users' feedback and actions during the trials. There were two distinct tests, one to determine functionality and usability and the second to observe the adoption of prescriptive syntax during an event.

A. *Analysis of usability and functionality*

Table 3 shows the Twitter user age demographics extracted from the 2009 statistics recorded by Sysmosis [37]. Also available was a statistical breakdown on the gender of Twitter users. These statistics show male users account for 47 percent of the Twitter user base and female users account for 53 percent. The age and gender figures were used to derive a sample population for the purpose of this study on usability, as they are indicative of the Twitter user population.

TABLE III. AGE DEMOGRAPHICS OF TWITTER USERS (2009) [38]

Age group	Percentage of Twitter users
15-19	31
20-24	35
25-29	15
30+	19

TABLE IV. AGE DEMOGRAPHICS OF TEST PARTICIPANTS

Age Group	Number of female test users	Number of male test users
15-19	3	3
20-24	4	4
25-29	1	1
30+	2	2

Table 4 shows the sample size of twenty test users that were divided into groups relative to age and gender. The personal identifiers recorded for each testing subject were their age group, computer skill, and if they were a Twitter user. Each individual trial involved a fifteen-minute testing session at one of three locations on the Townsville campus of James Cook University (Queensland, Australia) (Figure 6). The buildings at each location were within a five hundred meter radius of each other and were familiar to the test subjects. Two usage trials were conducted, one for the Web interface and one for the mobile interface.

On arrival at the alpha testing session, participants were given an information sheet on the hashtags. They were then instructed to tweet a 'fire' incident in a specific place on the map from via the Web interface. Following this task, notes were

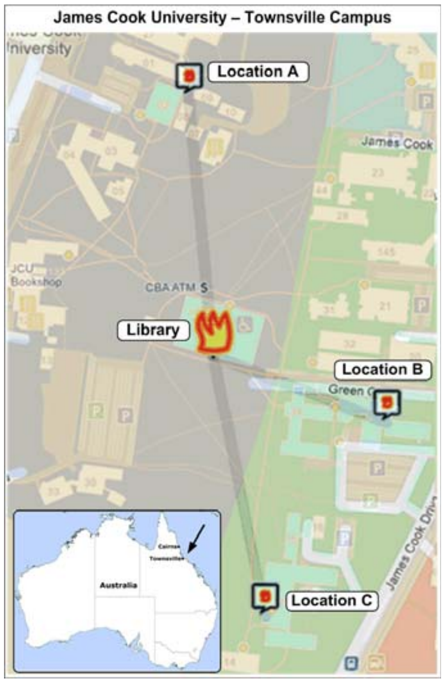


Figure 6 - Correct geolocation of the hypothetical 'library fire' using 200m as input. The Riskr interface is overlaid on top of the JCU campus map to indicate geolocation of rooms.

taken of how easily the user was able to pick up and use the service with no explanation. Details of the inputs the participants used when reporting the incident were also documented along with all queries or confusion from them on how to use the service.

The users were given a mobile device for the second trial. The users were instructed to report a 'fire' incident in the James Cook University library after the mobile had detected their geo-location. If the test user was unsure of the location of the library, they were shown an approximate location relative to their current location on the mobile map. Notes were taken on the ease of use of the mobile application. Also, the arbitrary distance from the library the users logged. The users completed a questionnaire and then interviewed about the experience.

The user testing sessions derived information on usability and accuracy. This information included how the users could easily recognize and mimic prescriptive syntax, the users' ability to accurately judge distances, and a gauge on the ease of access and use of the portal's interfaces. The users gave feedback on how useful and accessible they perceived the portal to be. In addition, the accuracy of the location information was assessed to determine the geo-location of the incident the users logged was close to the actual geo-location of the library.

The usability of the portal was determined from observations of the users' interactions with the portal and the qualitative review of their experience. Table 5 describes the most notable observations. Specifically, the twenty users rated themselves between five and ten on a scale of one to ten on computer capability; the average was seven. 55% of the users already had an existing Twitter account.

TABLE V. SIGNIFICANT OBSERVATIONS

Trends	Comments
55% of users had a twitter account	The users are mixed in their experience with the twitter social network, which allows us to gauge the effectiveness of the portal on inexperienced users.
41% of users made scale errors (200m instead of 0.2km)	Many users made errors in scale, using meters instead of km. This indicates that the portal would benefit from support for different units of measurement.
70.5% of users were close in their estimates of the incident location	There is indication that there is some merit in using user’s estimates to find disaster locations, however further work must be considered to bring users’ accuracy up.
After reading text sent in prescriptive syntax, 95% of users were able to correctly transpose their own message in prescriptive syntax	Almost all users were able to read, understand, and write their own prescriptive syntax, in a situation that would be presented to Twitter users if they were following someone using the Riskr service. This shows that the additional accessibility using prescriptive syntax in social networks is realistic and important.

The distance input relies on an arbitrary figure logged by the user, which proved to be problematic. Many users had an issue with scale because the reporting window only supported kilometre input. Some users specified the units of distance incorrectly, for example stating 100 metres instead of one kilometre. Other users logged incorrect distances due to lack of knowledge of the scale (e.g., stating a figure only of 100). Approximately forty-one percent made an error in distance.

The three chosen test locations were all approximately 200 meters away from the James Cook University library (refer to locations A, B and C in Figure 3). The trilateration algorithm considers an error resistance with a factor of three to judge an estimate as related to an incident, which for this experiment means an estimate was deemed as ‘close’ if it was between 66.67m and 600m. Many users made errors due to uncertainty over the units they were reporting, for example attempting to report 200m instead of 0.2km. Out of the users that did estimate a distance from the library, approximately 47.1% were in the ‘close’ range. However, if the aforementioned scale issues are taken into account, and the intended distance is considered, 70.5% of users were able to estimate the distance in a ‘close’ margin. Due to the inaccuracy of the users’ posts, an accurate trilateration of the incident was not produced.

Most qualitative feedback was positive (Table 6). In the interview, 100% of users indicated if given the chance, they would use at least one of the services again. However, 10% of users noted that they would only use the disaster logging service if it had Facebook integration. Some users indicated dislike of the small text limit on messages.

TABLE VI. QUALITATIVE RESPONSES

Question	Average Response (1-10)	Comments
I found this service easy to use	9	The users generally indicated that the service was easy to use and accessible given not much information about it.
The information given was sufficient to allow me to use the service	8.85	
This service would be useful during disaster situations	9.55	Users generally perceived the service to be useful both during and after disaster situations
This service would be useful after disaster situations	8.85	
I would use this service again if it were available	9.05	Most users strongly indicated they would use this service and get more

This service would encourage me to become more involved in discussion of a disaster or incident	8.7	involved if it was available to them, some would only use it if it was implemented with their current social network of choice.
This service is not necessary	1.35	Users generally did not find the service unnecessary.

The respondents indicated that users would be drawn to this kind of service during a disaster. The participant interviews found mostly positive feedback on the experience with the portal and all users indicated they were interested in using this service again. This observation indicates that using social networks to transmit information regarding disasters is an important part of a social networking disaster framework. Further, there are implications that any disaster-related information system available on the Internet would benefit from greater social network interoperability. A portal for reporting and viewing disasters using existing social networks can be a successful and viable enhancement to disaster management.

B. The prescriptive syntax case study

The users were given a prescriptive syntax case study, where they play the role of someone who has seen a post with prescriptive syntax, but do not know details of the service. Users were given the text

#riskr #fire #address 16 Alice St, Townsville, QLD #distance 4

The users were then asked to write down information about another disaster (A flood 2 kilometres away from 24 Oscar St, Townsville QLD) in the same format. The correct response for this question was:

#riskr #flood #address 24 Oscar St, Townsville QLD #distance 2

95% of users answered this question correctly. The issue with scale (using 2km after #distance instead of 2) was the cause of the one incorrect response.

The users understanding and application of prescriptive syntax proved positive. Specifically, posts of this nature are readable by the public, even if they are not Twitter users. The format is easily mimicked, which shows the feasibility of using Twitter as a middleware to encourage more user participation. The user's ability to reformat the prescriptive syntax indicates that a social network as a middleware is beneficial for any disaster portal.

VI. Conclusion

A citizen’s need for communication and information is important in any disaster situation. The collective knowledge of a social network can be more beneficial in the initial stages of an emergent event than that of the mainstream media [38]. This collective knowledge provides an opportunity to harness more information from large populations to facilitate and nurture collaborative efforts in disaster situations.

The Riskr portal has been implemented to collect and disseminate information relevant to the location of an incident. The portal's framework consists of the Web server, the map API, the news feed, the prescriptive syntax and the mobile implementation. Tests on the usage and feasibility of the portal in disaster situations were conducted with positive outcomes. The users were given the mobile Web service to report the incident and their estimation ability and use of the mobile service were noted. The user testing found the service easy to use and most indicated they would be drawn to this kind of service during a disaster. The qualitative case study indicated prescriptive syntax is effectively easy to read. Almost all users were able to see and understand prescriptive posts and use the format to post their own information.

Research and development of social network based disaster reporting and management is emerging. There are new opportunities available for citizen collaboration and participation during disaster events. The Riskr disaster portal is one such initiative. The Riskr prototype and the ensuing user testing support evidence of the potential interoperability with social networks can have when applied to disaster management. The Riskr portal represents a starting point for the development of a fully autonomous disaster portal in the context of social networks.

VII. Acknowledgements

We wish to thank Professor Ian Atkinson, Tristan Davey, Dylan Mathiesen and Barry Mitchell for their helpful feedback and support.

VIII. Bibliography

[1] J. Lickfett, *et al.*, "The RESCUE disaster portal for disasters and emergency response," in *Proceedings of the 5th International ISCRAM Conference*, Washington, DC, USA, 2008.

[2] J. Camarero and C. A. Iglesias, "Disasters2.0: application of web 2.0 technologies in emergency situations," *International Journal of Emergency Management*, vol. 6, p. 261, 2009.

[3] A. Bruns, "Towards distributed citizen participation: lessons from WikiLeaks and the Queensland floods," in *CeDEM11: Proceedings of the International Conference for E-Democracy and Open Government*, Danube-University Krems, Austria, 2011, pp. 35-52.

[4] A. Bruns and J. E. Burgess, "The use of Twitter hashtags in the formation of ad hoc publics," in *6th European Consortium for Political Research General Conference*, University of Iceland, Reykjavik, 2011.

[5] T. Heverin and L. Zach, "Microblogging for Crisis Communication: Examination of Twitter Use in Response to a 2009 Violent Crisis in the Seattle-Tacoma, Washington Area," in *7th International Conference on Information Systems for Crisis Response and Management*, 2010.

[6] A. Mills, *et al.*, "Web 2.0 Emergency Applications: How Useful can Twitter be for Emergency Response," *Journal of Information Privacy & Security*, vol. 5, pp. 3-26, 2009.

[7] T. Oreilly, "What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software," *Social Science Research Network Working Paper Series*, vol. 1, p. 17, 2007.

[8] S. Vieweg, *et al.*, *Collective intelligence in disaster: examination of the phenomenon in the aftermath of the 2007 Virginia tech shooting*, 2008.

[9] K. Starbird and J. Stamberger, "Tweak the Tweet: Leveraging Microblogging Proliferation with a Prescriptive Syntax to Support Citizen Reporting," in *7th International ISCRAM Conference*, Seattle, USA, 2010.

[10] M. Efron, "Hashtag retrieval in a microblogging environment," presented at the Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval, Geneva, Switzerland, 2010.

[11] T. Sakaki, *et al.*, "Earthquake shakes Twitter users: real-time event detection by social sensors," presented at the Proceedings of the 19th international conference on World wide web, Raleigh, North Carolina, USA, 2010.

[12] B. D. Longueville, *et al.*, ""OMG, from here, I can see the flames!": a use case of mining location based social networks to acquire spatio-temporal data on forest fires," in *Proceedings of the 2009 International Workshop on Location Based Social Networks*, Seattle, Washington, 2009, pp. 73-80.

[13] C. Penner, (2011). *#numbers*. Available: http://blog.twitter.com/2011/03/numbers.html

[14] J. N. Sutton, "Twittering Tennessee: Distributed Networks and Collaboration Following a Technological Disaster," in *7th International Conference on Information Systems for Crisis Response and Management*, 2010.

[15] L. Barbosa and J. Feng, "Robust sentiment detection on Twitter from biased and noisy data," presented at the Proceedings of the 23rd International Conference on Computational Linguistics: Posters, Beijing, China, 2010.

[16] C. Castillo, *et al.*, "Information credibility on twitter," presented at the Proceedings of the 20th international conference on World wide web, Hyderabad, India, 2011.

[17] A. Go, *et al.*, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, 2009.

[18] F. Benevenuto, *et al.*, "Detecting spammers on twitter," 2010.

[19] C. Caragea, *et al.*, "Classifying Text Messages for the Haiti Earthquake," 2011.

[20] A. Karandikar, "Clustering short status messages: A topic model based approach," University of Maryland, 2010.

[21] J. Sankaranarayanan, *et al.*, "TwitterStand: news in tweets," presented at the Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, Washington, 2009.

[22] S. Abrol and L. Khan, "TWinner: understanding news queries with geo-content using Twitter," presented at the Proceedings of the 6th Workshop on Geographic Information Retrieval, Zurich, Switzerland, 2010.

[23] T. O'Reilly, "Web 2.0 Compact Definition: Trying Again," *Read*, 2006.

[24] T. Onorati, *et al.*, "Interaction Design for Web Emergency Management Information Systems," in *7th International Conference on Information Systems for Crisis Response and Management*, 2010.

[25] M. A. Russell, *21 Recipes for Mining Twitter*: Oreilly & Associates Inc, 2011.

[26] Django Software Foundation, (2011). *Django | The web framework for perfectionists with deadlines*. Available: https://www.djangoproject.com/

[27] Tweepy, (2011). *Tweepy --- A pythonist's interface to twitter*. Available: http://tweepy.github.com/

[28] Twitter, (2010). *OAuth FAQ | Twitter Developers*. Available: https://dev.twitter.com/docs/auth/oauth/faq

[29] Google, (2011). *Google Maps JavaScript API V3*. Available: http://code.google.com/apis/maps/documentation/javascript/

[30] Yahoo, (2011). *Yahoo! Maps Web Services - YDN*. Available: http://developer.yahoo.com/maps/

[31] Mapquest, (2011). *MapQuest Open Data APIs and Web Services*. Available: http://developer.mapquest.com/web/products/open

[32] The jQuery Foundation, (2010). *jQuery: The write less, Do more, JavaScript Library*. Available: http://jquery.com/

[33] Z. Yang and Y. Liu, "Quality of Trilateration: Confidence-Based Iterative Localization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, pp. 631-640, 2010.

[34] M. Moradi Zaniani, *et al.*, "Trilateration target estimation improvement using new Error Correction Algorithm," Isfahan, Iran, 2010, pp. 489-494.

[35] Tumblr, (2011). *Sign up | Tumblr*. Available: https://www.tumblr.com/

[36] Flickr, (2011). *Welcome to Flickr - Photo Sharing*. Available: http://www.flickr.com/
[37] (2009). *PeriodicalUpdater for jQuery*. Available: http://www.360innovate.co.uk/blog/2009/03/periodicalupdater-for-jquery/
[38] A. Cheng, *et al.*, (2009). *An In-Depth Look Inside the Twitter World* Available: http://www.sysomos.com/insidetwitter/

IX. Appendices

A. Trilateration Code – 'trilat.py'

```
from Tkinter import *
import random

#Code written by Trevor Andersen from JCU
#Modified slightly for application on google maps

def tupleSum(a, b):
    return tuple(a[i] + b[i] for i in xrange(min(len(a), len(b))))

def midpoint(a, b):
    return tuple(x / 2. for x in tupleSum(a, b))

def eucDistance((x, y), (a, b)):
    return ((a - x)**2 + (b - y)**2) ** .5

def quadraticError(circle, (x, y)):
    return circle.distanceToPoint((x, y)) ** 2

class Circle:
    def __init__(self, (x, y), radius, distance=eucDistance):
        self.x = x
        self.y = y
        self.radius = radius
        self.distance = distance

    def setxy(self, x, y):
        self.x = x
        self.y = y
        return

    def getx(self):
        return self.x

    def gety(self):
        return self.y

    def distanceToPoint(self, (x, y)):
        return abs(self.radius - self.distance((self.x, self.y), (x, y)))

    def __str__(self):
        return "Circle((%d, %d), %d)" % (self.x, self.y, self.radius)

def gridSearch(circles, errorFn, deltaX, deltaY, left, right, top, bottom):
    """
    Searches for the point (x, y) that minimises \Sigma errorFn(circle, (x, y))
    over all circles. Each point to the resolution of deltaX and deltaY
    (i.e. if (x, y) is checked, then (x + deltaX, y) is checked etc.)
    within the box defined by left, right, top and bottom is tested.

    Returns the best point and the sum of errors
    """
    best = (left, bottom)
    leastErrorSum = 1e+300

    x = left
    while x <= right:
        y = top
        while y <= bottom:
            errorSum = sum(errorFn(c, (x, y)) for c in circles)

            if errorSum < leastErrorSum:
                best = (x, y)
                leastErrorSum = errorSum

            y += deltaY
        x += deltaX

    return best, leastErrorSum
```

```
def hillClimbSearch(circles, errorFn, resolution, left=None, right=None, top=None, bottom=None):
    """
    Searches for the point (x, y) that minimises \Sigma errorFn(circle, (x, y))
    over all circles. The search concludes when no better point can be found
    outside of the range defined by resolution (i.e. x, y +/- resolution).

    Returns the best point found and the sum of errors
    """
    CURRENT = (0, 0)
    FORMER = (-1, -1)
    NORTH = (0, -1)
    SOUTH = (0, 1)
    EAST = (-1, 0)
    WEST = (1, 0)

    # establish bounds
    if left != None:
        left = min(c.x - c.radius for c in circles)
        right = max(c.x + c.radius for c in circles)
        top = min(c.y - c.radius for c in circles)
        bottom = max(c.y + c.radius for c in circles)

    posMap = {}
    posMap[CURRENT] = ((left + right) / 2., (top + bottom) / 2.)
    searchResolution = min((right - left) / 4., (bottom - top) / 4.)

    posMap[NORTH] = tupleSum(posMap[CURRENT], (0, -searchResolution))
    posMap[SOUTH] = tupleSum(posMap[CURRENT], (0, searchResolution))
    posMap[EAST] = tupleSum(posMap[CURRENT], (-searchResolution, 0))
    posMap[WEST] = tupleSum(posMap[CURRENT], (searchResolution, 0))

    leastError = 1e+300
    best = CURRENT

    for p in (CURRENT, NORTH, SOUTH, EAST, WEST):
        error = sum(errorFn(c, posMap[p]) for c in circles)
        if error < leastError:
            leastError = error
            best = p

    posMap[FORMER] = posMap[CURRENT]
    posMap[CURRENT] = posMap[best]

    directionToFormer = (-best[0], -best[1])

    #done = False
    while True:
        if best == CURRENT:
            searchResolution /= 2.
            if searchResolution < resolution:
                break

        #print posMap[CURRENT], leastError

        posMap[NORTH] = tupleSum(posMap[CURRENT], (0, -searchResolution))
        posMap[SOUTH] = tupleSum(posMap[CURRENT], (0, searchResolution))
        posMap[EAST] = tupleSum(posMap[CURRENT], (-searchResolution, 0))
        posMap[WEST] = tupleSum(posMap[CURRENT], (searchResolution, 0))

        posMap[directionToFormer] = midpoint(posMap[CURRENT], posMap[directionToFormer])

        leastError = 1e+300
        best = CURRENT

        for p in (CURRENT, NORTH, SOUTH, EAST, WEST):
            error = sum(errorFn(c, posMap[p]) for c in circles)
            if error < leastError:
                best = p
                leastError = error

        posMap[FORMER] = posMap[CURRENT]
        posMap[CURRENT] = posMap[best]

        directionToFormer = (-best[0], -best[1])

    return posMap[CURRENT], leastError

def multiHillClimber(circles, errorFn, resolution, gridSize):
    left = min(c.x - c.radius for c in circles)
    right = max(c.x + c.radius for c in circles)
    top = min(c.y - c.radius for c in circles)
    bottom = max(c.y + c.radius for c in circles)

    deltaX = (right - left) / float(gridSize)
    deltaY = (bottom - top) / float(gridSize)

    leastError = 1e+300
    bestPoint = (0, 0)

    for x in (left + (i + 0.5) * deltaX for i in xrange(gridSize)):
        for y in (top + (j + 0.5) * deltaY for j in xrange(gridSize)):
            point, error = hillClimbSearch(circles, errorFn, resolution, x - deltaX / 2.,
                                           x + deltaX / 2., y - deltaY / 2., y + deltaY / 2.)
            if error < leastError:
                leastError = error
                bestPoint = point

    return bestPoint, leastError

def testTownsville(circles):
    MIN_X, MAX_X = -19, -20
    MIN_Y, MAX_Y = 146, 147

    MIN_X *= 1000
    MAX_X *= 1000
    MIN_Y *= 1000
    MAX_Y *= 1000

    point = (abs(MAX_X-MIN_X) / 2, abs(MAX_Y-MIN_Y) / 2)

    errorFn = quadraticError

    #change circles origin point
    for c in circles:
        print c.x, c.y
        c.setxy(abs((c.getx()*1000-MIN_X)), abs((c.gety()*1000-MIN_Y)))
        c.radius *= 1000
        print "mod"
        print c.x, c.y

    bruteForcePoint = (0, 0)

    hillClimbPoint, hillClimbError = multiHillClimber(circles, errorFn, 0.2, 10)
    #bruteForcePoint, bruteForceError = gridSearch(circles, errorFn, 4., 4., MIN_X, MAX_X, MIN_Y, MAX_Y)

    plot(abs(MAX_X-MIN_X), abs(MAX_Y-MIN_Y), circles, point, hillClimbPoint, bruteForcePoint)

def normalTrilat(circles):
    errorFn = quadraticError
    return multiHillClimber(circles, errorFn, 0.2, 10)

def degTrilat(circles):

    #find lowest, highest x and y

    min_x, max_x = circles[0].x, circles[0].x
    min_y, max_y = circles[0].y, circles[0].y

    for c in circles:
        if c.x < min_x:
            min_x = c.x
        if c.y < min_y:
            min_y = c.y
        """
        if c.x > max_x:
            max_x = c.x
        if c.y > max_y:
            max_y = c.y
        """

    min_x -= 1
    min_y -= 1
    """
    max_x += 1
    max_y += 1
    """

    errorFn = quadraticError

    #change circles origin point
    for c in circles:
        print (c.x, c.y)
        c.setxy((c.x-min_x), (c.y-min_y))

    hillClimbPoint, hillClimbError = multiHillClimber(circles, errorFn, 0.2, 10)

    hillClimbPointNormal = (hillClimbPoint[0] + min_x, hillClimbPoint[1] + min_y)

    print hillClimbPointNormal

    return hillClimbPointNormal, hillClimbError

def test():
    MIN_X, MAX_X = 0, 800
    MIN_Y, MAX_Y = 0, 800
    MIN_RAD, MAX_RAD = 1, 30
    CIRCLES = 10

    point = (MAX_X / 2, MAX_Y / 2)

    circles = []
    for c in xrange(CIRCLES):
        x = random.randint(MIN_X + MAX_RAD, MAX_X - MAX_RAD)
        y = random.randint(MIN_Y + MAX_RAD, MAX_Y - MAX_RAD)
        if random.choice((True, False)):
            radius = eucDistance((x, y), point) * (0.5 + 0.5 * random.random())
        else:
            radius = eucDistance((x, y), point) * (1 + 1 * random.random())

        circles.append(Circle((x, y), radius))

    errorFn = quadraticError

    bruteForcePoint = (0, 0)

    hillClimbPoint, hillClimbError = multiHillClimber(circles, errorFn, 0.2, 10)
    #bruteForcePoint, bruteForceError = gridSearch(circles, errorFn, 4., 4., MIN_X, MAX_X, MIN_Y, MAX_Y)

    plot(MAX_X, MAX_Y, circles, point, hillClimbPoint, bruteForcePoint)

def plot(width, height, circles, point, hillClimbPoint, bruteForcePoint):
    tk = Tk()

    canvas = Canvas(tk, width=width, height=height)

    for circle in circles:
        canvas.create_oval(circle.x - circle.radius,
                           circle.y - circle.radius,
                           circle.x + circle.radius,
                           circle.y + circle.radius)

    canvas.create_line(hillClimbPoint[0], hillClimbPoint[1] - 4,
                       hillClimbPoint[0], hillClimbPoint[1] + 5, fill='blue')
    canvas.create_line(hillClimbPoint[0] - 4, hillClimbPoint[1],
                       hillClimbPoint[0] + 5, hillClimbPoint[1], fill='blue')
```

```
canvas.create_line(bruteForcePoint[0], bruteForcePoint[1] - 4,
                   bruteForcePoint[0], bruteForcePoint[1] + 5, fill='green')
canvas.create_line(bruteForcePoint[0] - 4, bruteForcePoint[1],
                   bruteForcePoint[0] + 5, bruteForcePoint[1], fill='green')

canvas.create_line(point[0] - 2, point[1],
                   point[0] + 3, point[1], fill='red')
canvas.create_line(point[0], point[1] - 2,
                   point[0], point[1] + 3, fill='red')

canvas.pack()
tk.mainloop()
```