Analyzing and Visualizing Disaster Phases from Social Media Streams

Xiao Lin
Liangzhe Chen
Andrew Wood

Group ProjVizDisasters
CS5604: Information Storage & Retrieval
Virginia Tech
12/11/12

Abstract

The Four Phases of Emergency Management Model, which includes Mitigation, Preparedness, Response, and Recovery, has been widely used both by the academic researchers and emergency managers in the field. This model is a useful tool for looking into the details of event changes and activities taken during each phase, as well as for making decisions for actions. In spite of its popularity, the model has received criticism due to the discrepancy between the clear phase distinctions in the model and the complex and overlapping nature of phases in reality.

In this project, conducted during the fall 2012, under the direction of the CTRNet project, we address these critiques by harnessing the wealth of micro-blogging data associated with disasters. We created a procedure to process and categorize such tweets into the four phase model, and a visualization system to display it in a way that is modular and easily accessible.

From the data set resulting from collecting tweets about Hurricane Isaac for approximately a month (8/23-9/26, 2012), we specifically selected a subset of tweets that include major disaster organization and agency names such as FEMA, Red Cross, and Salvation Army. Our assumption was that these tweets would contain either the actions conducted by those representative disaster organizations or observations about those actions by the general public or both.

We trained three classification algorithms: multi-class SVM, multinomial Naïve Bayes, and Random Forests, each with different settings for term weighting, normalization, and stemming. We then evaluated the classification results using 10-fold cross validation. The results showed that Multi-class SVM performed the best (80.82% correctly classified) using term frequency and normalization in the process.

After classifying the selected tweets into four phases, we visualized them using a ThemeRiver™ graph, in which each stream represents a phase. The phases constitute the 'What' aspect in our interface. When a user selects a time interval (i.e., 'When') from the ThemeRiver™, tweets in that interval are displayed as a table. A user relations network graph (i.e. 'Who') and user location map ('Where') are also displayed.

Our Classification evaluation and interface use cases show that our tweet-based four phase visualization has potential to be a useful tool for disaster researchers to investigate diverse aspects (i.e., what, when, where, who) of phases in disaster-related tweets. We also hope that our study aids in refining the current disaster phase model in the long run by illustrating how phases actually occur.

Table of Contents

Table of Figures

1. Project Overview

1.1 The Problem

The Four Phase Model of Emergency Management has been widely used by both in the field by emergency managers and academics researching disasters. This model as it is most commonly defined consists of four phases which flow into one another in a cyclical fashion relative to an emergency or disaster, shown in Figure 1-1. The phases are defined in more detail in Table 1-1.

Figure 1-1: Four Phases of Emergency Management

Phase	Description
Response	Immediate and ongoing reactions following an emergency or disaster, to provide emergency assistance such as search/rescue, shelter, medical care, and food.
Recovery	The development, coordination, and execution of service and site restoration plans, including government operations and services, private and public assistance, and long term care and treatment of affected persons.
Mitigation	Activities to reduce or eliminate ongoing risks to persons or property and to lessen or avoid the impact of a disaster.
Preparedness	Deliberate tasks and activities intended to build, sustain, and improve operational capability to protect against, respond to, and recover from future emergencies and disasters.

Table 1-1: Descriptions of the four phases

This model can aid emergency agencies in handling emergencies efficiently, as each phase has its own associated activities and procedures. Having a clear understanding of the model and current phase during a

disaster assists emergency management personnel in making good decisions regarding which tasks to perform and how best to prepare for the next phase. Additionally, the model provides a frame of reference when analyzing a disaster after the fact, either for training/preparedness activities or academic research.

Despite these benefits, the model has received criticism from both emergency management practitioners and academics (Neal, 1997), mainly due to the discrepancy between the model's clearly delineated phases and the overlapping and ad hoc nature of emergency management activities in practice. The model has been redefined and extended by a number of sources (FEMA, 2006; FEMA, 2009; NFPA 2007), but the fundamental critiques remain. Thus, work is needed to better adapt the model to real world emergency situations.

1.2 Our Solution

This project, performed under the direction of the CTRNet project (described below), addresses the shortcomings of the Four Phase Model by leveraging micro-blogging data from the social media giant, Twitter. As micro-blogs, or 'tweets,' have become an increasingly viable and popular form of communication and are available in large quantity, much may be learned about an important event such as a disaster (natural or otherwise) in real-time. Our end-to-end approach provides two major contributions: (1) a pipeline for processing and classifying tweets into categories and (2) a visualization system for displaying them in a multi-view manner to quickly extract meaning across tweets ('What'), users ('Who'), locations ('Where'), and time ('When'). See Figure 1-2 for a flowchart describing the project at a high level. We started with data from Hurricane Isaac, which struck the Caribbean and Southern USA in August and September 2012, as our case study, but the approach should extend to other disasters.

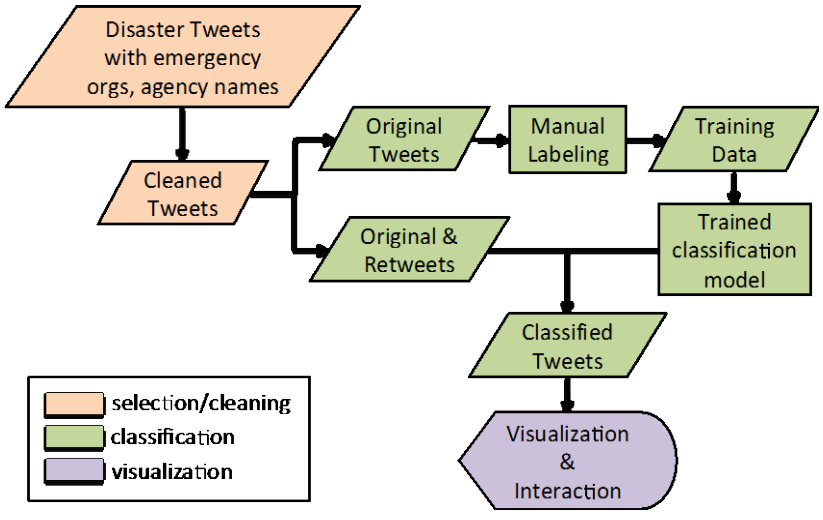


Figure 1-2: Project flowchart

1.2.1 Tweet Processing Pipeline

The four phase model captures professional and personal activities. In tweets, we observed that people do not tend to report personal activities, but are willing to share or comment on professional activities. Those

tweets often feature references to professional organizations, which make them easier to extract. So in our study, we mainly focus on analyzing professional activities of the four phases in tweets.

Our tweet processing pipeline consists of professional activity extraction, preprocessing and classification. From an archive of emergency tweets, professional activities are extracted by filtering by keywords and hashtags of professional organizations. Tweets are combined with the resource title of web pages referenced by shortened links and then stop word removal, stemming and multiple vectorization techniques are applied. Multiple classification algorithms are available for use in the classification step, and the best-performing algorithms during our test are Naïve Bayes Multinomial and Multiclass SVM, achieving an F measure of 0.78 and accuracy around 80%.

1.2.2 Web-based Visualization

The volume of tweets associated with a given emergency situation is very large; simply reading all data to understand an emergency situations is inefficient and likely intractable. Users' individual messages often involve complex combinations of information structures, including spatial (each user's geospatial location), temporal (time), network (relationships between users), and document collection (twitter message contents) structures. These different structures increase the complexity and uncertainty involved in managing these large datasets, and users may be overwhelmed and disoriented. In our visualization, we use four views to capture different features of the tweets data. The Phase View shows how the number of tweets in each given phase varies with to time and allows users to select an interval along the time range. The Tweet View shows the detail data in an Excel-like table. The Social Network View shows the user information and the @mentions in tweets (as links between different users). Finally, the Map View shows the location where tweet data was created. Breaking the data into views this way provides a method to better understand tweet data. See Figure 1-3 for an overview of the interface.

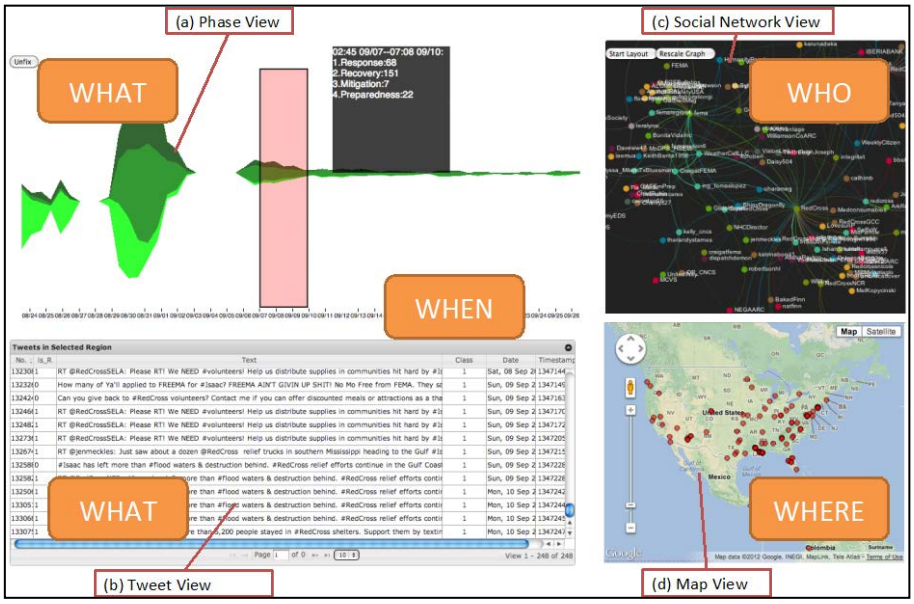


Figure 1-3: Project visualization overview

1.3 CTRNet

The NSF-funded CTRnet project archives disaster-related online data such as Twitter messages (i.e., tweets) as well as web sites in collaboration with the Internet Archive (IA). In addition to about 45 web archives, more than 114 tweet archives have been created. Tweets from both natural (e.g., hurricanes, earthquakes, floods, Tsunami, etc.) and man-made disasters (e.g., school shootings, plane crashes, political turmoil, etc.) are captured using a tool that accesses both search and streaming APIs provided by Twitter. The tool retrieves tweets based on hashtags and keywords, which are configured in advance, then stores them into a local database for research purposes.

1.4 Project Management

This project was completed as a part of *CS5604: Information Storage and Retrieval,* a graduate Computer Science course at Virginia Tech during the Fall Semester, 2012. The work was conducted for the CTRNet project, under the supervision of Seungwon Yang. In addition, we were assisted by Sunshin Lee and Haeyong Chung, who provided tweet processing and visualization/infoVis support, respectively.

Over the course of the semester, we held weekly meetings with the client to discuss our progress, refine the research direction, and to assign units of work. In general, Xiao was responsible for the tweet processing, to include pre-processing and machine learning approaches. Liangzhe was primarily responsible for the visualization component. Andrew provided support in both categories, beginning in natural language processing and entity extraction, and eventually aiding in visualization and evaluation. All group members contributed to discussions in each area during group meetings.

1.5 Challenges and Future Work

As the project evolved over the course of the semester, we ran into a number of challenges. As we already had the data on hand, the first challenge was what machine learning approach to take to categorize the tweets into a phase. We first began with clustering, but ultimately settled on a classification approach. Another challenge was how to extract meaningful information from data which was by definition extremely short. Much of the data is irrelevant to our purposes, or personal in nature. For the purposes of this project, we decided to manually extract certain professional organizations and classify only those tweets, also manually, to create the classification training data set.

In the future, the project should use a larger subset of the tweets and tackle the issue of automated entity (professional organization) extraction. Additionally, the many tweets which are of a personal nature should also be examined, as they do contain a great deal of data which could contribute to situational awareness and increase the richness of the dataset. Finally, the approach needs to be extended to other datasets, starting with similar disasters like Hurricane Sandy, and moving on to different disaster types.

2. User's Manual

2.1 Four-Phase Classification

2.1.1 Overview

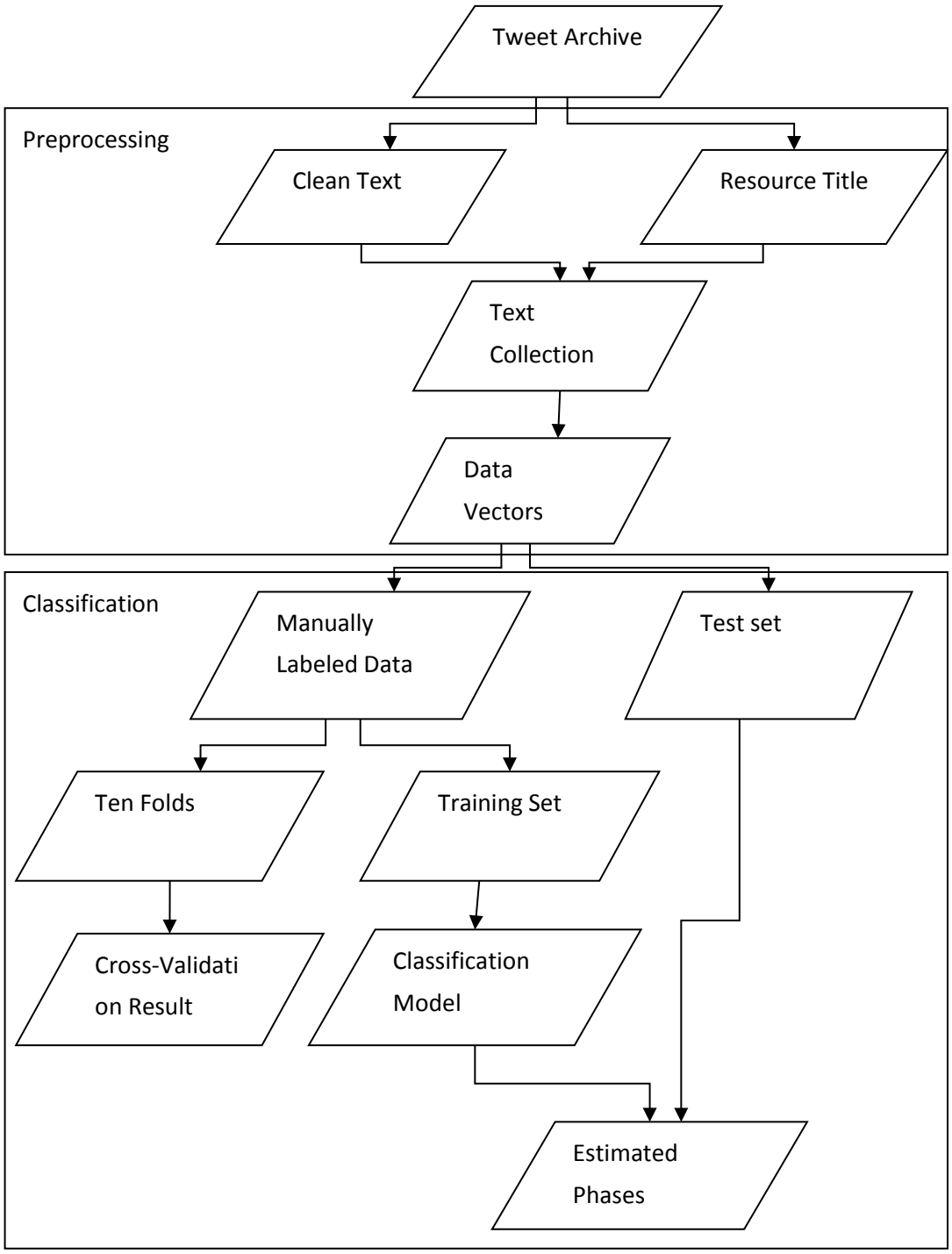


Figure 2-1: Dataflow of our four phase classification pipeline

One important task in our project is to classify tweets into four phases, to provide a timely response to incoming tweets. The dataflow of our processing pipeline is shown in Figure 2-1 . The tweet archive we use is introduced in Section 2.1.2. The preprocessing step and the classification step are introduced in Section 2.1.3 and Section 2.1.4 respectively.

2.1.2 Tweet Archives

In this project, we start with tweets with keywords or hashtags referring to a specific type of disaster. The CTRnet project has a server http://spare05.dlib.vt.edu/index.php dedicated to archiving tweets on specific topics. Archived tweets can be downloaded in multiple formats, including html, excel, simple table and json. The prototype dataset we used in our project is based on the hurricane Issac archive. The corresponding hashtag is #Issac.

Tweets are often too short to carry rich information. But fortunately, tweets often come with shortened links. The titles of the shortened links are also extracted for better classification performance.

Tweets related to professional activities are extracted by filtering with keywords and hashtags of professional organizations in the original text and resource title fields. This filtering method is not perfect. There are many tweets mentioning professional organizations purely for attention, and some giving opinions. A better filtering technique relies on the study of features of tweets reporting professional activities. We can see the examples in Section 2.1.6.

Regarding the training set, in this project we labeled the training set ourselves by hand, assigning explicit labels to one of the four phases, or marked as not clear

2.1.3 Preprocessing

Punctuation, stop words and hashtags removal are applied to the original tweets to get the clean text of tweets. Each item of clean text and corresponding resource title are concatenated. That forms the text collection we use for vectorization. An example of goes as Table 2-1.

Original tweet text	Mitigation specialists are offering free rebuilding tips in five parishes. http://t.co/hwXajm6X #Isaac
Article	FEMA Advisers Offer Home Repair Tips at Local Stores **Release date:** SEPTEMBER 18, 2012 **Release Number:** DR-4080-059 **BATON ROUGE, La.** – Teams of hazard mitigation specialists with the Federal Emergency Management Agency (improvement centers in several parishes to offer free consultations to survivors who are rebuilding after Hurricane I The FEMA advisers can offer tips and techniques on how to protect homes from future disaster-related damage an stronger and safer; they also have advice on topics such as: • emergency preparedness • roof repair • rebuilding flooded homes

Shortened link title	FEMA Advisers Offer Home Repair Tips at Local Stores
Clean text	mitigation specialists offering free rebuilding tips parishes
Concatenated text	mitigation specialists offering free rebuilding tips parishes FEMA Advisers Offer Home Repair Tips at Local Stores

Table 2-1: Example of tweet processing

Vectorization can be done by using functions in the Weka toolkit (Hall et al., 2009). Porter stemmer (Porter, 1980), normalization, tf transform, and idf transform are some of the available options in the StringToWordVector filter within Weka. Some tests performed on different options imply that stemming, normalization, and tf transform might be a good set of options to set, as shown in Section 2.1.6.

After vectorization, Weka will output an arff file containing the dictionary entries and the word vectors.

Table 2-2 shows examples of dictionary entries and word vectors.

Dictionary	@attribute cross numeric
Word vector	{0 1.0774,1 2.513626,2 2.890133,3 4.135621,4 4.135621,5 1.081041,6 1.792503,7 3.477604,8 2.43866}

Table 2-2: Examples of dictionary entries and word vectors in an arff file

2.1.4 Classification

Training a classifier

In this project, a training set is a set of word vectors with labels explicitly to one of the four phases. Response, recovery, mitigation and preparedness are labeled as 1, 2, 3 and 4 respectively.

The classification algorithms we tested include Naïve Bayes, Naïve Bayes Multinomial, Random Forest and a multi-class SVM algorithm (Crammer et al., 2002). Weka has implementations of Naïve Bayes, Naïve Bayes Multinomial and Random Forest. For multi-class SVM we used another implementation, the SVM-Multiclass (Joachims, 2008). Section 2.1.5 introduces a middleware to convert word vector file in arff format to the format required by Weka and SVM-Multiclass.

Performing classification

In contrast to most classification practices in which one simply throws the vectors into classifiers, because the test set may not share the same dictionary, we need to represent the test set using the training set's dictionary before classification can be performed. A middleware is written to do this. It is introduced in detail in Section 2.1.5.

After that we convert the test set to the required formats, and then run the classification programs with models from the training step to get estimated labels of phases.

Cross-validation

Ten-fold cross-validation is a popular technique to evaluate classifier performance. In 10-fold cross-validation, the training set is randomly partitioned into 10 folds, and each fold acts as test set in turn, while the remaining 9 folds are used for training.

Weka supports 10-fold cross-validation as a feature, while a middleware that partitions the training set into 10 folds for SVM-Multiclass is introduced in Section 2.1.5.

2.1.5 Middleware

A collection of middlewares are written to handle the format conversions in the classification dataflow.

MakeTrainingData

Overview

MakeTrainingData combines class labels with the vectors, dividing them randomly into n folds, and return the training and test data for each fold in SVM-Light format.

It can be used to convert vectorized text from Weka arff format into SVM-Light format or build n-folds training and test data for n-fold cross validation. The MakeTrainingData manual entry is shown in Figure 2-2.

```
Usage: MakeTrainingData [output file] [classlabel file] [vector file] [number of folds]

output file       Training set in SVM-Light format. The actual file names of training
                  and test files for each fold will look like [output
                  file]_fold00_train.dat and [output file]_fold00_test.dat

classlabel file   Specifies the file containing class labels.

vector file       Specifies the file containing data vectors. The number of vectors
                  and the number of class labels should agree.

number of folds   Specifies how many folds to make. For 10-fold cross-validation, set
                  this to 10. For normal training and test sets, set this to 1.
```

Figure 2-2: MakeTrainingData Usage Manual

If the parameters are not provided correctly, the program will show a prompt and try to read them from `stdin` one by one.

Note that the output file cannot be directly used for test. Weka requires the class labels to be an enum. This can be done by loading the SVM-Light format training set into Weka, saving as arff format, and changing the class labels to an enum struct. For example

@attribute class {1,2,3,4}

Input format

Classlabel file

Classlabel file consists of tab, space or newline delimitated numbers only. The first number should always be 1. The second number is the count of class label entries. The rest are the class labels.

Note that class labels cannot be 0, as required by the SVM-Light format.

An example of the classlabel file can be:

```
1 4 1 2 3 4
```

It specifies four entries, 1, 2, 3 and 4.

Vector file

Vector file contains vectors one in a row, each in json format. The vectors can be extracted directly from the Weka arff file. Opening an arff file as a text file, the vectors are right after the @data caption

An example of vector file is:

```
{0 1,1 1,2 1,3 1,4 1,5 1,6 1,7 1,8 1,9 1,10 1,11 1}
{12 1,13 1,14 1,15 1,16 1,17 1,18 1,19 1,20 1,21 1,22 1,23 1,24 1,25 1}
{0 1,2 1,26 1,27 1,28 1,29 1,30 1,31 1,32 1,33 1,34 1,35 1}
{1 1,7 1,36 1,37 1,38 1,39 1,40 1,41 1,42 1,43 1}
```

Output format

The format of the output training sets and test sets are in the format required by SVM-Multiclass. An example looks like:

```
4 1:1 2:1 3:1 4:1 5:1 6:1 7:1 8:1 9:1 10:1 11:1 12:1
4 13:1 14:1 15:1 16:1 17:1 18:1 19:1 20:1 21:1 22:1 23:1 24:1 25:1 26:1
4 1:1 3:1 27:1 28:1 29:1 30:1 31:1 32:1 33:1 34:1 35:1 36:1
4 2:1 8:1 37:1 38:1 39:1 40:1 41:1 42:1 43:1 44:1
```

MakeTestData

Overview

MakeTestData try to map data vectors in one dictionary to another dictionary, add a label to the vectors and save the result in SVM-Light format.

It provides support to mapping test datasets into the dictionary of training datasets, and format conversion from json vectors into SVM-Light format. See Figure 2-3 for usage information.

```
Usage: MakeTestData [output file] [classlabel file] [original vectors] [original
dictionary] [training dictionary] [do normalization]

output file           Test dataset in SVM-Light format.

classlabel file       The file containing class labels for the test dataset. Usually
                      the test dataset does not need a label. But a classlabel file
                      with arbitrary labels for each of the vectors is still required

original vectors      Specifies the file containing data vectors of the test dataset.

original dictionary   Specifies the file containing the dictionary of the test
                      dataset.

training dictionary   Specifies the file containing the dictionary of the training
                      set.

do normalization      Re-normalize or not. 1=normalize, 0=don't normalize. Check this
                      when normalization is applied during vectorization.
```

Figure 2-3: MakeTestData usage manual

If the parameters are not provided correctly, the program will show a prompt and try to read them from `stdin` one by one.

Note that the output file cannot be directly used for test. Weka requires the class labels to be an `enum`. This can be done by loading the SVM-Light format test set into Weka, saving as arff format, and changing the class labels to an enum struct. For example

```
@attribute class {1,2,3,4}
```

Input format

Classlabel file

The format of the classlabel file is the same as middleware MakeTrainingData.

Original vectors

The file containing the original vectors shares the same format as middleware MakeTrainingData. It can be extracted in the same way from the Weka arff file.

Dictionary files

The first row of the dictionary file is the number of tokens. Following it are the tokens in the dictionary, one for each row.

An example dictionary file goes:

4
American
Cross
Emergenc
Hurrican

The tokens in the dictionary can be extracted from the Weka arff files. After vectorization, Weka will name each attribute of the data vector with the corresponding token. So the dictionary can be extracted from the attributes directly. Some examples goes as Table 2-3.

Example in arff file	Corresponding token
@attribute NULL numeric	NULL
@attribute cross numeric	cross
@attribute florida numeric	florida

Table 2-3: Finding tokens in arff files

Output format

The format of the output file is the SVM-Light fomat. It is the same as in middleware MakeTrainingData.

2.1.6 Preliminary Results and Analysis

Overview of the Hurricane Issac dataset

In our preliminary study, we built the Hurricane Issac dataset. It features:

- About 56,000 English tweets during hurricane Issac
- 5,677 tweets with reference to FEMA, Red Cross or Salvation Army
- 1,453 without re-tweets
- With re-tweets removed, 1,121 tweets are manually labeled explicitly with one of the four phases: response, recovery, mitigation or preparedness

With the Issac dataset, we carried out cross-validation analysis and four phase labeling attempts.

Cross-validation analysis

Ten-fold cross-validation of tuned classifiers

Algorithm (tf+normalization+stemming)	Accuracy	Weighted F Measure
Naïve Bayes	70.47%	0.723
Naïve Bayes Multinomial	77.87%	0.782
Random Forest	76.27%	0.754
SVM Multiclass	80.82%	0.770

Table 2-4: Cross-validation result of tuned classifiers

We performed 10-fold cross validation on the Issac dataset, with four well-tuned classifiers. We measured the raw accuracy and the weighted F-Measure on them. The result is shown in Table 2-4. Weighted F Measure is the most popular measure of performance of multi-class classifiers. Naïve Bayes Multinomial did the best. SVM is doing the best in terms of raw accuracy.

Accuracy versus different vectorization techniques

Tf	Idf	Normalization	Stemming	Naïve Bayes Multinomial	SVM Multiclass
			X	76%	80.1%
		X	X	77%	80.4%
	X		X	60%	78.8%
	X	X	X		78.1%
X			X	75%	80.4%
X		X	X	78%	80.8%
X	X		X	63%	78.9%
X	X	X	X		79.0%

Table 2-5: Accuracy of algorithms under different vectorization settings

We evaluated the performance of the two well-performing algorithms, Naïve bayes multinomial and Multiclass SVM, under different vectorization settings. The result is in Table 2-5. We can infer that idf might not be helpful for our task.

For tweet classification, 80% is already a good result. Based on our observation that the state-of-the-art classifiers are performing pretty much the same, we need to extract more information from tweets in order to push the classification accuracy to a higher stage.

SVM accuracy in different settings

We tested the 10-fold cross validation accuracy of SVM-Multiclass under different settings of c, a parameter in the SVM-Multiclass algorithm, with or without an additional class for unlabeled tweets, different idf, tf normalization and stemming settings. The result is shown in Table 2-6.

The setting tf+normalization+stemming with c=10000 outperforms others slightly.

Four-phase labeling with re-tweets

We performed four-phase classification with the SVM-Multiclass algorithm. Of all 5677 tweets, the result dataset has 1356 tweets labeled as response, 2552 labeled as recovery, 23 labeled as mitigation and 1746 labeled as preparedness. The results are later used for visualization demonstrations in Table 2-6.

With non -relavant class	Idf	Tf	Normali -zation	Stemming	Accuracy, c=100	Accuracy, c=10000	Accuracy, c=1000000
					0.668153434	0.803746655	0.786797502
				X	0.682426405	0.801070473	0.768956289
			X		0.667261374	0.795718109	0.779661017
			X	X	0.679750223	0.803746655	0.778768956
		X			0.645851918	0.804638715	0.785905442
		X		X	0.647636039	0.801070473	0.787689563
		X	X		0.658340767	0.804638715	0.78322926
		X	X	X	0.659232828	0.808206958	0.775200714
	X				0.79750223	0.781445138	0.759143622
	X			X	0.799286351	0.787689563	0.768064228
	X		X		0.786797502	0.802854594	0.795718109
	X		X	X	0.800178412	0.780553078	0.780553078
	X	X			0.776092774	0.781445138	0.754683318
	X	X		X	0.785905442	0.789473684	0.761819804
	X	X	X		0.773416592	0.801070473	0.781445138
	X	X	X	X	0.779661017	0.790365745	0.76984835
X					0.60199005	0.739161336	0.722814499
X				X	0.60909737	0.748400853	0.697228145
X			X		0.604122246	0.747690121	0.722103767
X			X	X	0.611940299	0.752665245	0.714996446
X		X			0.597725657	0.731343284	0.719971571
X		X		X	0.599147122	0.742004264	0.721393035
X		X	X		0.602700782	0.724946695	0.717128643
X		X	X	X	0.616915423	0.743425729	0.704335466
X	X				0.702914001	0.707178394	0.699360341
X	X			X	0.717839375	0.697228145	0.680881308
X	X		X		0.721393035	0.719260839	0.707889126
X	X		X	X	0.717839375	0.692253021	0.693674485
X	X	X			0.683724236	0.70931059	0.690831557
X	X	X		X	0.697938877	0.705046198	0.67803838
X	X	X	X		0.692963753	0.714285714	0.70575693
X	X	X	X	X	0.702914001	0.705046198	0.692963753

Table 2-6: Accuracy of Multiclass SVM under different settings

2.2 Visualization

2.2.1 Purpose

Despite their very short length, tweets contain a great deal of information: the phase the tweet is associated with, the text of the tweet itself, users' location (for certain users), and references to other users. The aggregation of a large number of tweets quickly makes it difficult to process all of this information directly. This visualization, therefore, breaks down this aggregate information into categories, which the user can then explore easily and independently.

2.2.2 The Four Views

Before we discuss how to interact with the system, let us first explore the interface. The main window in the browser is divided into four areas, or *views*, each presenting a unique perspective into the tweet information. The views are distinct, but also synchronized by time; that is, each displays different kinds of data from the same time interval. Note that most of the views are not initialized until an initial time selection is made.

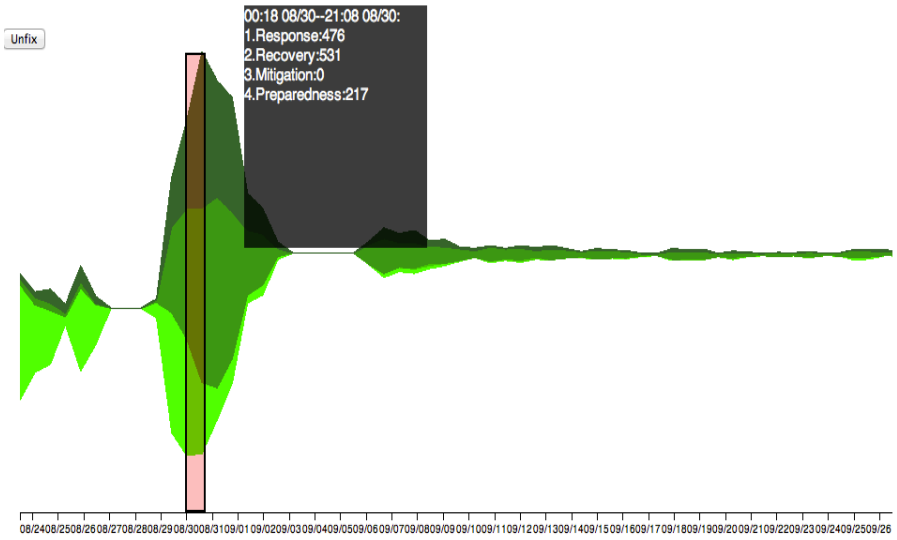
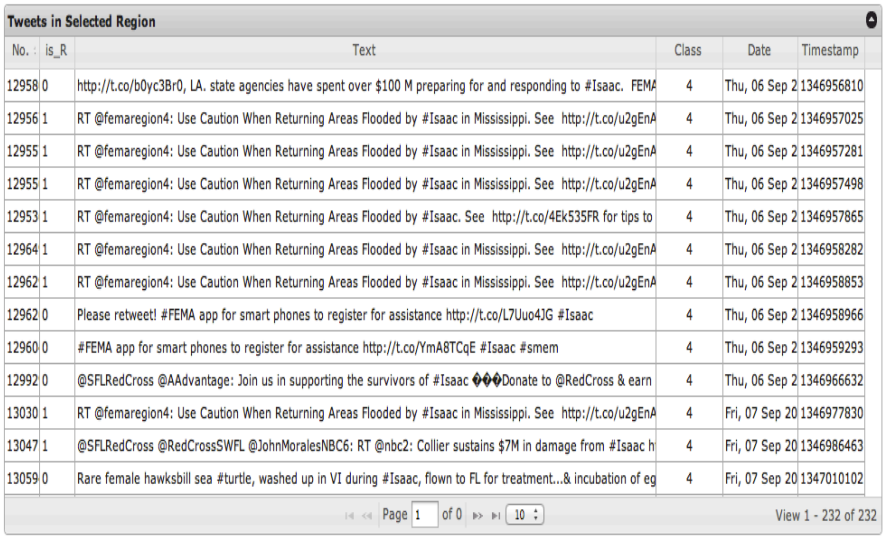


Figure 2-4: Phase View

Phase View: If any of the views can be considered the 'main' or 'primary' view, it is the Phase View (Figure 2-4). It is this view which displays the tweets in the context of the four phases and allows the user to select a time frame. The view is dominated by a ThemeRiver graph, which shows how the number of tweets in each phase (y-axis) changes over time (x-axis). Individual phases are indicated by color. If the user mouses over the graph, or selects a time period, detailed information about the number of tweets in each phase are given textually in the grey box.



Tweets in Selected Region

No.	is_R	Text	Class	Date	Timestamp
12958	0	http://t.co/b0yc3Br0, LA. state agencies have spent over $100 M preparing for and responding to #Isaac. FEMA	4	Thu, 06 Sep 2	1346956810
12956	1	RT @femaregion4: Use Caution When Returning Areas Flooded by #Isaac in Mississippi. See http://t.co/u2gEnA	4	Thu, 06 Sep 2	1346957025
12955	1	RT @femaregion4: Use Caution When Returning Areas Flooded by #Isaac in Mississippi. See http://t.co/u2gEnA	4	Thu, 06 Sep 2	1346957281
12955	1	RT @femaregion4: Use Caution When Returning Areas Flooded by #Isaac in Mississippi. See http://t.co/u2gEnA	4	Thu, 06 Sep 2	1346957498
12953	1	RT @femaregion4: Use Caution When Returning Areas Flooded by #Isaac. See http://t.co/4Ek535FR for tips to	4	Thu, 06 Sep 2	1346957865
12964	1	RT @femaregion4: Use Caution When Returning Areas Flooded by #Isaac in Mississippi. See http://t.co/u2gEnA	4	Thu, 06 Sep 2	1346958282
12962	1	RT @femaregion4: Use Caution When Returning Areas Flooded by #Isaac in Mississippi. See http://t.co/u2gEnA	4	Thu, 06 Sep 2	1346958853
12962	0	Please retweet! #FEMA app for smart phones to register for assistance http://t.co/L7Uuo4JG #Isaac	4	Thu, 06 Sep 2	1346958966
12960	0	#FEMA app for smart phones to register for assistance http://t.co/YmA8TCqE #Isaac #smem	4	Thu, 06 Sep 2	1346959293
12992	0	@SFLRedCross @AAdvantage: Join us in supporting the survivors of #Isaac ���Donate to @RedCross & earn	4	Thu, 06 Sep 2	1346966632
13030	1	RT @femaregion4: Use Caution When Returning Areas Flooded by #Isaac in Mississippi. See http://t.co/u2gEnA	4	Fri, 07 Sep 20	1346977830
13047	1	@SFLRedCross @RedCrossSWFL @JohnMoralesNBC6: RT @nbc2: Collier sustains $7M in damage from #Isaac h	4	Fri, 07 Sep 20	1346986463
13059	0	Rare female hawksbill sea #turtle, washed up in VI during #Isaac, flown to FL for treatment...& incubation of eg	4	Fri, 07 Sep 20	1347010102

Page 1 of 0 10 View 1 - 232 of 232

Figure 2-5: Tweet View

Tweet View: Directly below the PhaseView lies the Tweet View (bottom left), shown in Figure 2-5. It provides a detailed look at tweet content in tabular form. The tweets shown here are the subset corresponding to the currently selected time interval.

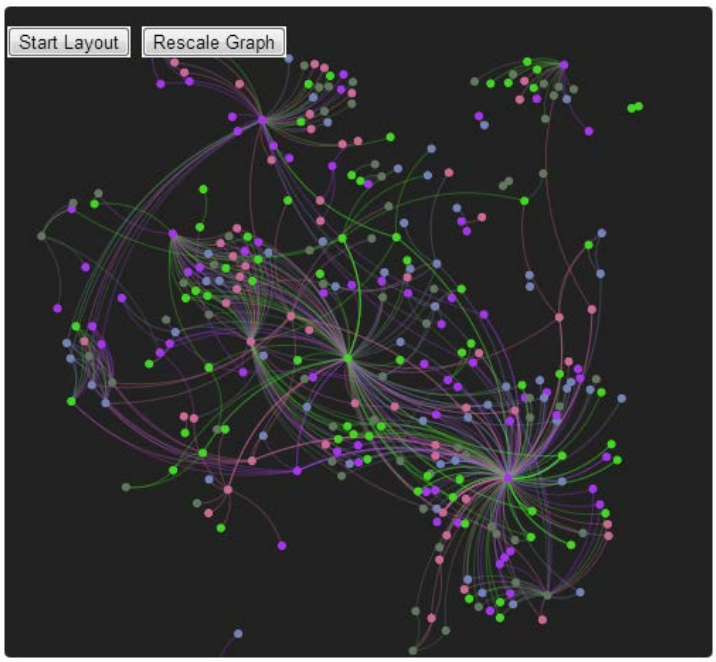


Figure 2-6: Social Network View

Social Network View: This view represents users and their tweets (Figure 2-6). In this graph, users are represented by dots, or nodes, and are connected by lines, or edges. These edges represent a reference of one user to the other via the use of the '@' tag within the text of their tweet. The edges are undirected, but colored the same as the node of the user who mentioned the other. Examining this graph can give a good understanding of how different tweets propagate across users, and which users are most active.

Figure 2-7: Map View

Map View: Illustrated in Figure 2-7, shows the geospatial distribution and overview of users and mentions which match the date query in the Phase View. Each small red circle on the map represents users located in a place and time.

2.2.3 Interaction – Using the Interface

Using the interface is quite simple. To get a quick textual overview of a short time period, simply mouse over the ThemeRiver graph. This does not, however, update the other views. To select a longer time, click and drag on the ThemeRiver. The red selection box will expand accordingly, and now all of the other views are updated. Once you are done examining a time period, simply press the 'unfix' button on the top of the Phase View display. Again, the other views are not updated until you make another selection.

When you make a selection, you'll notice that the user graph is constantly moving and changing; this is because it is iteratively refining its layout. To stop its movement, hit the 'stop layout' button. The graph will cease to move, and labels and edges will appear. You may zoom in and out of the graph with the mouse wheel, and may pan about by clicking and dragging. Clicking on individual nodes does not do anything.

The Map View may be controlled using the standard Google Maps widget and by clicking and dragging to pan and using the mouse wheel to zoom. Clicking on individual points will bring up an empty information box, which may in the future contain more pertinent information.

2.2.4 Use Case – The Interface in Action

Suppose a disaster researcher would like to examine the events of Hurricane Isaac during specific time intervals from the collected Twitter data. They first select a time interval (A), shown in Figure 2-8, by clicking and dragging, releasing to set or 'fix' the interval. A grey box displays the number of tweets within the time frame (05:25 08/24—06:02 08/26) belonging to each phase. This confirms the initial impression from the graph: the dominant phase is Preparedness (419), but Response (100) and Recovery (42) are also present.

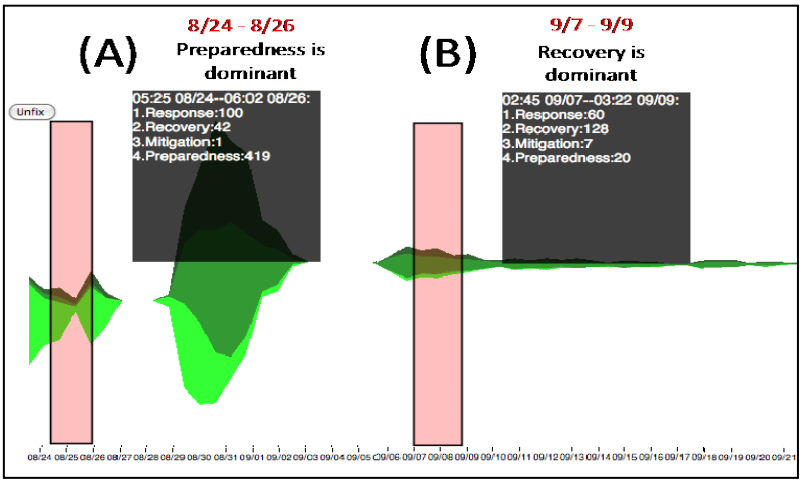


Figure 2-8: Selecte time intervals superimposed on a single ThemeRiver for simplicity of presentation. (A) the beginning of Hurricane Isaac; (B) one week after the hurricane dissipate

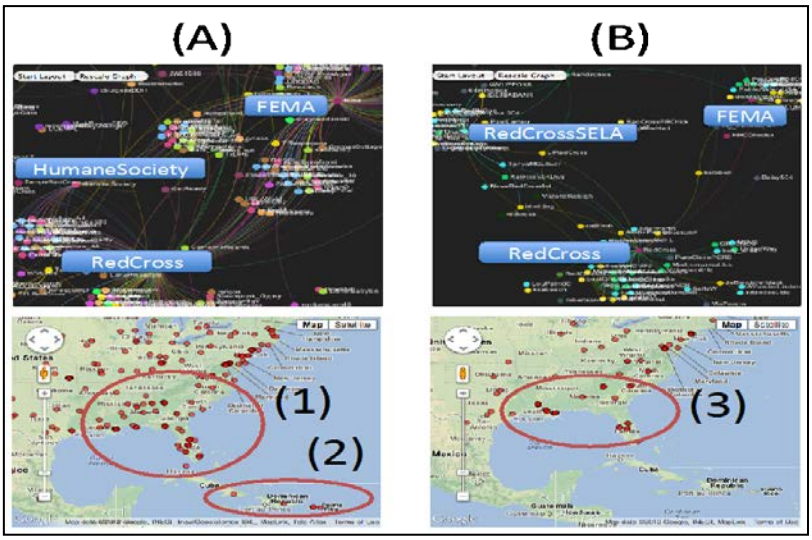


Figure 2-9: Social network and map views corresponding to the time intervals (A) and (B) in the preceding figure.

When the researcher examines the corresponding location graph (Figure 2-9(A)), they gain the insight that the Preparedness tweets are likely coming from the southeastern USA (Figure 2-9(1)) while the hurricane has already struck the countries in the Caribbean (see (2) in Figure 2-9), which are in response and recovery mode. The social network graph shows that accounts with the largest number of connections are 'FEMA,' 'RedCross,' and 'HumaneSociety'. As the presence of the Humane Society is unexpected, the researcher then reads some of their tweets, which were mostly discussing the Red Cross' opening of pet-friendly shelters.

Selecting another time interval, Figure 2-8(B), shows that Recovery (128) and Response (60) are the most dominant phases between 9/7 and 9/9. This makes sense as Hurricane Isaac dissipated around the first. Examining the map and social graph views (Figure 2-9(B)) reveals that fewer people are posting, and RedCrossSELA – responsible for parishes in Southeast Louisiana – is active. This also makes sense, as this is the area hardest hit by Isaac.

3. Developer's Manual

3.1 Four-Phase Classification

3.1.1 Middleware

MakeTrainingData

Basic procedure

1. Load the vector file and the classlabel file, and read other input parameters

2. Randomly divide the vectors and class labels into n folds.

3. Format and save each fold into a training set and a test set.

Some details

- The vector file is loaded with a customized reader, reading comma delimitated pairs of numbers. The reader reads the file as a sparse matrix.
- The classlabel file is read as a matrix, with width=1 and height=number of labels.

MakeTestData

Basic procedure

1. Load the vector file, the class label file and two dictionaries, and read other input parameters.

2. Calculated the squared norm of one of the vectors, if normalization is required.

3. Figure out word to word mapping from the test dictionary to the training dictionary.

4. Transform the vectors using the word mapping.

5. Remove words that do not exist in the training dictionary. Sort each vector with dictionary index from small to large. Re-normalize the vectors if necessary

6. Format and save the result.

Some details

- The vector file is loaded with a customized reader, reading comma delimitated pairs of numbers. The reader reads the file as a sparse matrix.
- The classlabel file is read as a matrix, with width=1 and height=number of labels.
- The dictionaries are loaded as vectors of strings.
- Token mapping is calculated by exhaustive matching.
- Non-existing words are first mapped to an invalid id, then removed.

- Instead of normalizing the vectors to length of 1, Weka normalize the vectors to a certain length that changes from one setting to another. We could not figure out how it did that, so we simply calculate the length before-hand and re-normalize back to the same length.

3.1.2 Future Improvements

Despite the fact that the pipeline is in good shape, some components can be further improved. Professional activity extraction can be improved by applying a classifier to find professional activities from tweets. Currently the tweets and the resource titles are concatenated directly, but a weighting schema can be applied instead. That may be a potential improvement. And some other information sources including tweet locations, tweet user and some context information can be added for a more accurate classification.

3.2 Visualization

Our visualization is based on HTML and Javascript. It contains 4 parts: A ThemeRiver view, tweets view, social network view and map view. Several libraries and APIs are used in the different views, detailed information on how we implement the visualization will be covered in the following sections.

3.2.1 Data processing

The original data is in a Microsoft Excel table containing several columns like user_id, is_R, text, class, date, timestamp. Additional changes must be made to the code when the order or the content of the columns change. The algorithms we use to separate tweets with different classes in the program requests that the table is already sorted by the class column, which means tweets with the same class is already grouped together in the table like Figure 3-1.

id	is_RT	text	class	date_time	unix_time
185	1	RT @fema: Aug 23: If #Isaac m	4	Thu, 23 Aug 20	1345739211
128	0	Oklahoma Red Cross Voluntee	4	Thu, 23 Aug 20	1345739297
367	1	RT @infobabe: #redcross ramp	4	Thu, 23 Aug 20	1345740562
328	1	RT @redcrosscanada: Red Cro	4	Thu, 23 Aug 20	1345740615
261	1	RT @RedCross: We are movin	4	Thu, 23 Aug 20	1345740709
259	1	RT @RedCross: We are movin	4	Thu, 23 Aug 20	1345740711
251	1	RT @RedCross: We are movin	4	Thu, 23 Aug 20	1345740723
248	1	RT @redcrosscanada: Red Cro	4	Thu, 23 Aug 20	1345740725
241	1	RT @RedCross: We are movin	4	Thu, 23 Aug 20	1345740733
234	1	RT @fema: Aug 23: If #Isaac m	4	Thu, 23 Aug 20	1345740745
227	1	RT @RedCross: We are movin	4	Thu, 23 Aug 20	1345740753

Figure 3-1: Original tweets data

For example, all tweets with class 4 are grouped together; following class 4, there is a group of tweets with class 3, and so on.

We then use the following website to convert the CSV or Excel file to JSON array format:

http://shancarter.com/data_converter/

After getting the array, we create it as a variable (test_data) in a .js file for data. Before importing this .js file for use, we need to check the data manually for any possible mistakes, which is mainly caused by unpaired double quotation symbol in the text column, like those shown in Figure 3-2.

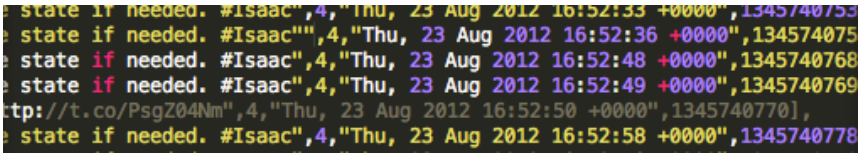


Figure 3-2: Example of mistakes in input data

The double quotation mark in the second row before the number 4 will lead to misunderstanding ",4," as a string and mess up the following data.

We see that the ThemeRiver, contains different layers of data. More strictly speaking, what d3.layout.stack()—the function we use to create the theme river—does is put one layer of data onto another. So we have to first convert the data to such form, separating data in the different classes into different layers.

The algorithm is simple. Since we assume that the original data is already sorted by the class columns, we go through each entry of the array and put that entry into a layer(a new array) until we find that the content in the class column of a new entry differs from the previous one. Thus we have our data prepared for each layer.

Because what the theme river shows is actually the number of tweets over time, we also need to do some counting in each layer. Each iteration we count the number of tweets in a window which span 75000 seconds, and then move forward 50000 seconds and count again. To make the theme river look continuous we add 0.1 to every counting result.

The input for d3.layout.stack() should contain x-coordinate and y-coordinate values for each data point that will be depicted in the graph. So after counting and smoothing the data, we have to create 2 fields: x and y for each point, where x stands for the time and y stands for the counting and smoothing results.

3.2.2 Theme River View

There are several important functions and components used to create the theme river view. We introduce them here respectively.

The function creating the Themeriver

d3.layout.stack will take several layers of data represented by an high dimension array. The data in different layers should have the same x values set. The output of this function will be passed to functions which draw paths and functions which draw areas. The paths and areas compose the final theme river.

The red vertical selecting bar

The rectangle is a div component. Making it move with the mouse is easily implemented by jQuery. Now to implement both mouse click and mouse drag is a little tricky. We need to add 2 flags to accomplish this (Figure 3-3).

```
flag1=0;//whether there's a mouse_down before the mouse_move
flag2=0;//whether it's a mouse_up for click or drag
$("svg").mouse_down(function(e){
    flag1=1;
})
$("svg").mouse_move(function(e){
    if(flag1==1){//mouse drag
        flag2=1;
    }else if(flag1==0){//mouse move
    }
})
$("svg").mouse_up(function(e){
    if(flag2==1){//mouse drag
        flag1=0;
        flag2=0;
    }else if(flag2==0){//mouse click
        flag1=0;
        flag2=0;
    }
})
```

Figure 3-3: Mouse events

The reason why we need flag2 is that the mouse_move event is executed multi times during a mouse drag. So we cannot change the flag we use to tell a mouse drag from a mouse move during the process, otherwise the mouse_move function only executes as we expect at the first time, and after the first execution the flag value is changed and become useless.

The unfix button

After a click or a drag, a time period is chosen and detail information is shown in other 3 views. Before users hit the unfix button, everything including the local information rectangle, the red vertical bar and content in other views will be fixed.

The x-axis

The x-axis can be implemented by a D3 function: d3.svg.axis(). We can further use .scale() to specify a scale to be used, .tickSubdivide() to specify the number of subdivide, .ticks() to specify the distance between main ticks, or .tickFormat() to define the format of text shown under each ticks.

The local information rectangle

The semi-transparent black rectangle moving with the mouse showing local information is also a div component. When the mouse is on the left part of the graph the rectangle appears at the right of it, and when the mouse is on the right, this information pad appears at the left, so that this local information pad will always stay inside the graph.

X-coordinate of the mouse is extracted and scaled to the exact time stamp, and then all tweets around that time (within the window we've mentioned above) will be counted again to form the statistic information at that exact time. The final result is shown in the rectangle.

Connection with other views

The theme river view is the basic of all other 3 views, and it's the only interface where users can manipulate the output of our visualization. Everytime a user click or drag to select a span of data, it returns the left and right time boundary and passes them to other 3 views.

3.2.3 Tweets View

Tweets View of the visualization is basicly an excel table. Jqgrid library is used to create the table. The creation of a table is shown in Figure 3-4.

```
//the excel form under the graph
        $("#list").jqGrid({
    url:'example.php',
    datatype: 'clientSide',
    mtype: 'GET',
    //colNames:['No.','ID', 'Text','Class','Date','Timestamp'],
    colNames:colN,
    colModel :[
      {name:colN[0], index:'No.', width:35},
      {name:colN[1], index:'is_R', width:35},
      {name:colN[2], index:'text', width:800, align:'left'},
      {name:colN[3], index:'class', width:80, align:'center'},
      {name:colN[4], index:'date', width:80, align:'center'},
      {name:colN[5], index:'timestamp', width:80,align:'center'}
    ],
    pager: '#pager',
    rowNum:10,
    rowList:[10,25,30],
    sortname: 'invid',
    sortorder: 'desc',
    viewrecords: true,
    gridview: true,
    caption: 'Correspond Tweet Data'
  });

$("#list").parents('div.ui-jqgrid-bdiv').height(300);
```

Figure 3-4: Creating a jqGrid component

RowNum limits the number of rows shown in a page. RowList offers 3 option of rowNum under the table. Some other options are also provided to adjust the table.

After we created the table we use $('#list').addRowData(row_No,row_Data) to add data into the table. And whenever a new selection of data is made in Theme River View, previous data in the table will be erased by $('#list').jqGrid('clearGridData') and new data will be loaded.

3.2.4 Social Network View

The Social Network View represents users and their tweet mentions as a ForceAtlas2 layout provided by Gephi[1]. In this graph, each circle represents a tweet user ID or a mention. Currently, it is implemented as a non-directional graph. Examining the connections exhibited by the graph allows the user to see how such mentions are propagated for a given time period. This visualization also allows users to navigate the network graph through panning and zooming.

3.2.5 Map View

The Map View shows the geospatial distribution and overview of users and mentions which match the date query in the Phase View(theme river view). Each small red circle on the map represents users located in a place and time. User IDs from the selected tweets are identified, and locations are extracted from their publicly available Twitter profile page in advance. Then, we applied geocoding using the Google Geocoding API[2] on the locations to find the latitude and longitude data. Out of 7,616 users, we were able to extract only 4,573 (60%) with latitude and longitude information. Problems might have been caused by: (1) the user did not add any location information in his/her profile; or (2) the entered location information could not be processed due to uninterpretable content (e.g., “Sumwhere in the universe”, “R.D”, “InMySkin” “http://maps.BlackBerry.com?lat”).

3.2.6 Future Improvements

There're several places we can improve in this visualization:

Now the original data has to be sorted by its class content before we can separate them to different classes. We can improve our separating algorithms to eliminate this condition on input.

We convert the excel table to an array in the theme river view and tweet view, the reason why we do that is because the input to d3.layout.stack() has to be an array. But since we have to do mistake check if we convert the original data directly to array, we can improve the program by first convert the data to json and then convert it to array in the program, so that we don't need to check the mistake manually.

Control in other 3 views. In current visualization we can only operate on the theme river view to select the time period, it would be better if we can also operate on the social network view and the map view. For example, clicking a node in the social network view to select all tweets of that user and show that in the tweets view, or clicking a location to show all tweets created in that location. More operations can be implemented to make the visualization better.

---

[1] https://gephi.org/

[2] https://developers.google.com/maps/documentation/geocoding/

Smooth function can be improved. We smooth the data once before we create the theme river view. However, from the theme river graph we see that the effect is not good enough. Multiple smoothing and parameter adjusting can make the theme river better.

Zoom in and out in the theme river.

Include data converter in the program instead of using a website. In the current version we use a website to convert excel table to json array (http://shancarter.com/data_converter/), which means everytime we use new data we have to do some part of data preprocessing manually.

3.3 Inventory of Repository Files

Included with this report is a repository of source code and other supporting documentation used over the course of this project. The contents of the repository directory include:

- FinalReport_ProjVizDisasters.pdf – This report, in PDF format
- FinalReport_ProjVizDisasters.docx –This report, in Word format
- FinalPresentation_ProjVizDisasters.pdf –The final project presentation for the course, in PDF format
- FinalPresentation_ProjVizDisasters.pptx – The final project presentation for the course, in PowerPoint format
- Isaac_Dataset.xlsx – The case study dataset in Excel format
- Middlewares.zip – Zip archive of source code and solution files for the tweet processing pipeline
- PhaseVis.zip – Zip archive of source code and associated files for the visualization system. To see the complete visualization, simply open 'index.html' in a web browser.

References

1. Baird, M. E. (2010). The “Phases” of Emergency Management. Background Paper. Prepared for the Intermodal Freight Transportation Institute, Univ. of Memphis. URL: http://www.memphis.edu/cait/pdfs/Phases_of_Emergency_Mngt_FINAL.pdf
2. Crammer, K., & Singer, Y. "On the algorithmic implementation of multiclass kernel-based vector machines." The Journal of Machine Learning Research 2 (2002): 265-292.
3. FEMA – Federal Emergency Management Agency. 2006. Principles of Emergency Management, Independent Study, IS230, Washington.
4. FEMA - Federal Emergency Management Agency. 2009. Guide to Emergency Management and Related Terms, Definitions, Concepts, Acronyms, Organizations, Programs, Guidance, Executive Orders & Legislation: A Tutorial on Emergency Management, Broadly Defined, Past and Present. http://training.fema.gov/EMIWeb/edu/docs/terms%20and%20definitions/Terms%20and%20Definitions.pdf (September 15, 2009).
5. Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. "The WEKA data mining software: an update." ACM SIGKDD Explorations Newsletter 11, no. 1 (2009): 10-18.
6. Joachims, Thorsten. 2008. "SVM-Multiclass: Multi-Class Support Vector Machine", Version: 2.20. URL: http://svmlight.joachims.org/svm_multiclass.html
7. Neal, D. M. (1997). Reconsidering the Phases of Disaster. International Journal of Mass Emergencies and Disasters. August 1997, Vol. 15, No. 2, pp. 239-264. URL: http://training.fema.gov/EMIWeb/downloads/IJEMS/ARTICLES/Reconstructing%20the%20Phases%20of%20Disaster.pdf.
8. NFPA - National Fire Protection Association. 2007. NFPA 1600, Standard on Disaster/Emergency Management and Business Continuity Programs (2007 Edition), Quincy, Massachusetts: National Fire Protection Association.
9. Porter, Martin F. "An algorithm for suffix stripping." Program: electronic library and information systems 40, no. 3 (2006): 211-218.