G.R.I.D - Get Relevant Information on Disasters

Project Final Report

Deepti Aggarwal
Virginia Tech
Blacksburg, VA
deeptiag@vt.edu

Ananya Choudhury
Virginia Tech
Blacksburg, VA
ananya@vt.edu

ABSTRACT

In this document we propose a method to classify tweets-with-URLs related to disaster based on the 140 characters of tweet and after fetching data from those URLs, create a web archived search engine related to a specific disaster. The volume of the archived information on disasters is usually very high and filled with noise making it difficult to get information specific to a particular disaster. Hence we put forward a technique to successfully filter information related to disasters and index them thereby creating an archived search engine. We propose a method to classify tweets, expand tweet URLs, extract data from the URLs and index them in Solr. The resultant search engine, G.R.I.D will be useful for disaster management and post disaster analysis.

Categories and Subject Descriptors

[DLRL Lab, Virginia Tech]

General Terms

Theory, Final Project Report

Keywords

Disaster, Information Retrieval, Text Classification, Social Media Analysis, Microblogging

1. INTRODUCTION

Twitter has become a fantastic channel for communication during disasters. With more than 100 million users, it's microblogging service is an apt medium to receive and exchange information. With brevity guaranteed by a 140- character-message limitation and the popularity of Twitter mobile applications, users tweet and retweet instantly. Everyday, nearly 340 million tweets are created and redistributed by all these active users.

As tweets are created in real time, in recent years this social media platform has acted as an active communication channel in times of emergency as a result of which voluminous amount of data is generated. Processing such big data to obtain relevant information involves multiple challenges including handling information overload, filtering credible information and categorizing data into different classes. Another significant challenge twitter data analyst face today is the filtering of data between relevant and non-relevant as the number of spam tweets have increased exponentially with the increase of Twitters popularity. Also, there is no single data streamlining system that concentrates only on disaster. Emergency Management organizations/Historians who analyze prior disasters are often overwhelmed by this amount of data and lack of any proper engine to sieve through the noise. The motivation of the project comes from these challenges.

In this document we propose a method to classify tweets-with-URLs related to disaster based on the 140 characters of tweets. We then extract those URLS and create a a web archived search engine G.R.I.D specific to disasters.The rest of the report is organized as follows: We define our problem statement in section 2. Section 3 discusses the proposed method and we have provided details of our dataset, preprocessing done on the dataset and data acquisition in Section 4, 5 and 6 respectively. Each of our project stages elaborated in section 7, 8, 9, 10. Related work is discussed in Section 11 and Problems Faced in section 12. Our final experiment and results are put forth in section 13. We finally conclude the report on G.R.I.D in section 14.

2. PROBLEM STATEMENT

In recent years, Twitter has been used to spread news about casualties and damages, donation efforts and alerts, including multimedia information such as videos and photos [9]. In November 2012, Twitter revealed that over 2 million tweets had been posted during Hurricane Sandy. In 2011, over 5,500 tweets were posted every second following the tsunami and earthquake in Japan (Anderson, 2012). Users posted over 2 million tweets about Haiti following the earthquake in January 2010 ' [7].In general, a significant amount of data gets generated during times of a disaster, some of which are extremely valuable for any post-disaster analysis. We provide an efficient way to retrieve and search information related to a crisis as the messages generated during the disaster vary greatly and a majority does not contribute to the disaster awareness.

OUR APPROACH:

We propose a 5-step method for disaster related information archiving:

- Step 1: Classification of twitter tweets into relevant and not-relevant categories using the Naive Bayes classifier.
- Step 2: Expansion of the URLs in the relevant tweets using Map Reduce

- Step 3: Extraction of text from the expanded links (webpages)
- Step 4: Index the data in Apache Solr.
- Step 5: Build a simple interface "G.R.I.D" for query search.

3. PROPOSED METHOD

3.1 Intuition

Twitter data analysis is a prime research area in NLP. Unfortunately, most state-of-the-art research on twitter data analysis is mostly limited to sentiment analysis. Although few research related to disaster are going on, they mainly concentrate on extracting information only from the tweets at real time. We deviate slightly here. Our classifier analyzes tweets to identify webpages that containto webpages that contain more information about the disaster. And we have to classify tweets of disasters that have already occurred. We take our project further to create a corpus of only relevant data associated with a disaster and finally present an efficient way to retrieve these information. There has been no work that we know of which extracts information from webpages extracted from twitter dataset and presented this entire pipeline of information retrieval.

3.2 Description

As shown in Figure 2 , we analyzed the data and manually labeled the tweets as relevant and not relevant. We achieved an accuracy of 85% and almost similar precision but our recall rate was less. To improve performance, we analyzed the dataset again. We created a training corpus of equal number of relevant and non-relevant tweets. We considered only boundary conditions for non-relevant tweets to create a strong distinction between the two labels and better train the classifier. For ex: "Ebola 'a Regional Threat' as Contagion Hits Guinea Capital #Health http://t.co/SQcCExqRc0" is labelled as relevant . "Guinea Ebola 'a regional threat' http://t.co/xrO 1SUoPBt" is labelled as non-relevant.

We used both Multinomial NaiveBayes and LinearSVC for classification. We noticed NaiveBayes performs better than Linear SVC. A primary reason for this is the training corpus which is comparatively smaller and NaiveBayes performs better on smaller dataset.

From the relevant tweets, we extracted URLs . In twitter the URLs are shortened. We unshortened the URLs using MapReduce programming in AWS . These URLs are unshortened using HttpClient. We then extract text from the webpages using BeautifulSoup, a module in python . We finally index all these text in Solr using sunburnt , a module in python and SimplePostTool , a tool that comes with Solr. The indexed files can be viewed using Velocity, a search based interface that works with Solr.

4. DATASET

We used dataset which contains tweets related to Ebola in multiple languages such as English, Spanish, French, etc. This tweets are collected from March 29, 2014 till Oct 22, 2014 available in http://cinnamon.dlib .vt.edu/twitter/. The entire dataset is received from Digital Libraries Research Lab, Virginia Tech. There are about 3 million tweets in the corpus. We created a sample of 5000 tweets by randomly collecting 4 sets of 1000 tweets across multiple timelines. We ran our classifier on 65000 unknown tweets and classified 22000 tweets as relevant and non-relevant. 13000 tweets are labelled as relevant. This is the input for next step which is Unshorten URL in AWS.

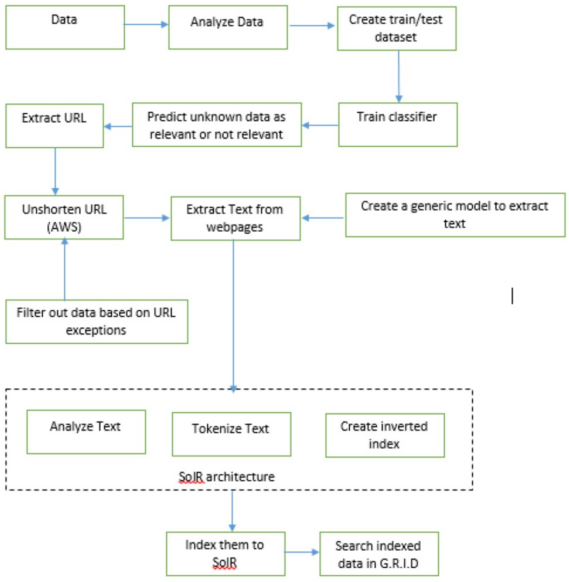

Figure 1: General framework of G.R.I.D

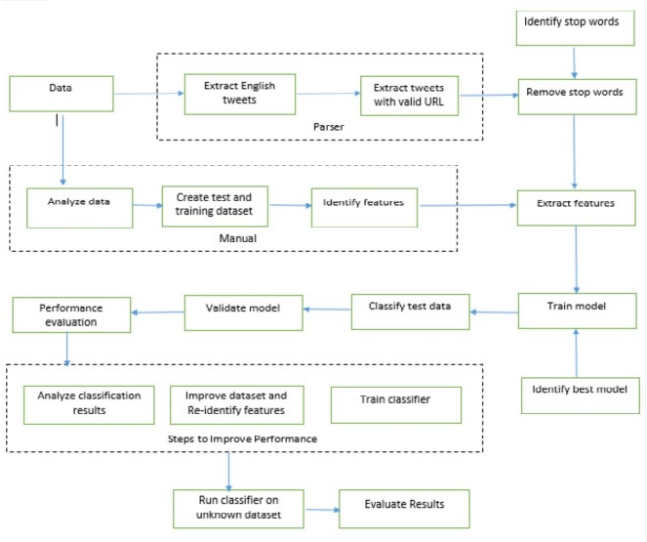

Figure 2: CLassification Pipeline

5. PREPROCESSING

The goal of the preprocessing part is to get the tweets in English language and which contains URLs. Atleast one URL (in case of multiple URLs) in the tweet should be valid. We validated the URLs by checking URL length using regular expression.

6. DATA ACQUISITION

We extracted the required information for our data modeling. The dataset is a .csv file containing following information for each tweet: archive source, text, to_user_id, from_user, id, from_user_id, iso _language_code, source, profile_image_URL, geo_type, geo_coor dinates_0, geo_coordinates_1, created_at, time. In order to get the required information we used the pandas package in python to read the file. We created a parser that identifies tweets with only valid URLs. We manually analyzed and created the learning dataset for

the classifiers by classifying the sample data of 5000 tweets into relevant and non relevant labels.

7. CLASSIFICATION

7.1 Feature Selection

Feature selection is an important component of classification. A flow chart of the various steps in classification process is added in Figure[1]. In this project, we classify tweets into two categories: relevant and non-relevant. We convert the collection of document into a matrix of TF-IDF features. Maximum features that can be extracted is 15000. From this set, we selected best 1500 tweets using SelectKBest method. The statistic used for feature selection is Chi-square.

Chi-square ($\chi 2$) test measures dependence between stochastic variables and does not consider features that are the most likely to be independent of the class and therefore irrelevant of classification. This statistic measure the difference between the observed counts n_i and the expected counts e_i.

$$\chi^2 = \sum \frac{(n_i - e_i)^2}{e_i} = \sum_{i_1=1}^{m_1} \sum_{i_2=1}^{m_2} \cdots \sum_{i_d=1}^{m_d} \frac{\left(n_{i_1,i_2...,i_d} - e_{i_1,i_2...,i_d}\right)^2}{e_{i_1,i_2...,i_d}} \quad (1)$$

We manually analyzed our sample 4000 tweets and identified 68 features which are relevant to classify text into relevant and non relevant. We added these 68 terms into the previous 1500 terms identified as features.

We considered retweets (tweets with RT tag in the beginning) as one of the features. We also considered the hashtags of relevant terms such as (# Ebola, #outbreak, #disaster, etc.) or hashtag of mostly affected regions (# Africa, # Liberia, etc) or twitter account of corporate houses (reuters , cnn, etc.) as few of the features. Unigrams, Bigrams and Trigrams are also extracted as features. All text in the corpus are converted into lowercase to avoid repetition of features in the TfIdfVectorizer method.

7.2 Cross-validation

K-fold Cross-validation: K-fold cross validation divides the dataset into K equal-sized parts called folds, namely $D_1, D_2, ..., D_K$. Each fold D_i is, in turn, treated as the testing set, with the remaining folds comprising the training set D $D_i = \cup_{j \neq i} D_j$. After training the model M_i on D D_i, we assess its performance on the testing set D_i to obtain the i-th estimate on $_i$. The expected value of the performance measure can then be estimated as

$$\hat{\mu} = E[\theta] = \frac{1}{K} \sum_{i=1}^{K} \theta_i \quad (2)$$

Usually K is chosen between 5 and 10. The testing set comprises a single point and the remaining data is used for training purposes. We enabled shuffling such that the folds are different each time.

We noticed in our case, the accuracy is the best for K=10. So the model is trained on 9 of the folds and the resulting model is validated on remaining part of the data. The performance measure reported by 10-fold cross-validation is then the average of the values computed in the loop.

We used StratifiedKFold cross validation iterator of sklearn package to split the data which is just a variation of KFold that returns stratified folds.

Train Test Split: This cross-validation algorithm splits arrays into random train and test subsets . This is a single split with a user defined split percentage. We mentioned the split percentage as 10% which implies 90% of the dataset is considered for training and the rest for test. We used train_test_split cross validation of sklearn package to split the data.

7.3 NAIVE BAYES CLASSIFIER:

For classification of tweets, we chose Naive Bayes because of its ease of use, simple design and good classification performance. Naive Bayes classifier is based on the model of probability. It focuses on the probability that a tweet belongs to a particular category C = $\{C_1, C_2, ..., C_K\}$. Here we use two categories (relevant and not-relevant). Each tweet is considered as a sequence of word tokens and each token will be labeled as relevant or not relevant. We calculate the posterior probability that the tweet belongs to either of the categories and chose the category with the highest probability.

The posterior probability of that a tweet t_i belongs to a category c_j can be calculated in formula:

$$p(c_j \mid t_i) = \frac{p(c_j)p(t_i)}{\sum_{k=1}^{|c|} p(c_k)p(t_i)} \quad (3)$$

$$Relevance(c_j, t_i) = log \frac{p(c_j \mid t_i)}{p(c_j^- \mid t_i)} \quad (4)$$

In order to evaluate the above we need to first calculate the probability of a category c_j i.e. $p(c_j)$ and the probability of the tweet t_i i.e. $p(t_i)$. $p(c_j)$ is given as the ratio of the number of tweets that fall into the category c_j and the total number of tweets.

$$p(c_j) = \frac{n(c_j)}{n\left(\sum_{j=1}^{n} c_j\right)} \quad (5)$$

Here, $n(c_j)$ is the total number of tweets which belong to category c_j.

$p(t_i)$ can be evaluated based on all the words that make the tweet. We make an assumption that the words that make up the tweets are conditionally independent. Also we use a unigram, bigram, trigram language model. We get $p(t_i)$ as:

$$p(t_i) = \prod_{i=1}^{n} p(w_i \mid c_j) \quad (6)$$

where $p(w_i)$ is the probability that a word w_i belongs to the category c_j.

The only parameters we will estimate are $p(c_j \mid t_i)$ $p(c_j^- \mid t_i)$ with the help of multivariate approach.The multinomial approach [5] specifies that a tweet is represented by the set of word occurrences from the tweet. A tweet is considered as a |V|- dimensional vector T=$(w_1, w_2, ..., w_{|V|})$ which is the result of |V| independent Bernoulli trials, where |V| is the vocabulary size w_k is a binary variable representing the occurrence or non-occurrence of the k-th word in the vocabulary. The order of the words is lost, but the number of occurrences of each word in the tweet is captured. When calculating the probability of a tweet, one multiplies the probability of the words that occur. In the multinomial approach, the formula(3) is calculated as follows

$$\log \frac{P_n(t_i|c_j)}{P_n(t_i|c_j)} = \sum_{k=1,\omega \in t_i}^{|V|} TF_{ik} . \log \frac{P_n(\omega_k|c_j)}{P_n(\omega_k|\bar{c_j})} \quad (7)$$

We used MultinomialNB class in sklearn package for classification . A Laplace smoothing parameter of 0.1 is added to eliminate zeros. This is a smoothing algorithm that simply adds one to each count.

7.4 OTHER CLASSIFIERS CONSIDERED - SVM:

Another model we used is LinearSVC class from sklearn package for classification. LinearSVC (Linear Support Vector Classification) is based on Support Vector Model classifier. LinearSVC is similar to SVC(Support Vector Classifier) but is implemented in terms of liblinear which is a linear classifier for data with millions of instances and features.. A support vector machine is a classifier which classifies two classes by constructing a hyper plane in which the margin between the two classes will be maximum.SVMs are a generally applicable tool for machine learning. Suppose we are given training examples x_i, and the target values y_i -1,1. SVM searches for a separating hyperplane, which separates positive and negative examples from each other with maximal margin, in other words, the distance of the decision surface and the closest example is maximal.

$$w^T x + b = 0 \tag{8}$$

The classification of an unseen test example x is based on the sign of $w^T x + b$. The binary classifier can be classified as

$$w^T x + b \geq 1 if y_i = +1 \tag{9}$$

$$w^T x + b \leq -1 if y_i = -1 \tag{10}$$

$K(x_i, x_j)$ denote a function that roughly speaking gives how similar two examples are. This is called a kernel-function. The decision function can be written as

$$f(x) = sign(\sum_{i=1}^{N} \alpha_i y_i . K(x_i, x_j) + b) \tag{11}$$

In general when dealing with Web collection data, SVM needs a sufficient number of positive and negative samples which need to be updated timely with increases in the size of the data set. This is a very costly process and with the given resources for the training data-set Naive Bayes seems to give better performance.In our implementation we classified data as the positive sample and consider rest of the data as negative sample. The accuracy of the system was comparatively less compared to Multinomial Naive Bayes. SVM implementation is more complex as it is time consuming to tune the internal parameters to get the best result and performance. We have chosen Naive Bayes over SVM as our mathematical model since the training time and processing time for the algorithm are comparatively less and it is fast, robust and simple to implement.

8. UNSHORTEN URL

The URLs are extracted from the tweets which are predicted as relevant by the classifier. These URLs are shortened as they are convenient for messaging technologies such as Twitter. We have used MapReduce for unshortening the URL using httpClient. The implementation of MapReduce has been done on AWS platform. A MapReduce program is composed of a Map() procedure that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a Reduce() procedure that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies).This step gives us a list of responsive unshortened unique URLs.

9. TEXT EXTRACTION FROM THE URL

We have tried to extract the relevant text from the unshortened URL by using BeautifulSoup python module. The script first extracts the URLs in the web-page and then it visits that webpage to get the relevant text from it. The URL extraction from the page has been nested to get the maximum relevant text from the webpage. We have set the lower bound of the text to be extracted as 8000 characters in order to reduce noises. We implemented this step such that we donâĂŹt extract the text if the http response for the URL is a socket-error or any other http error.

10. SOLR

Solr is an open source Java search server. The extracted data is in text format. Before the files are indexed to Solr, the schema.xml in Solr needs to be configured. The text is tokenized using Solr Standard Tokenizer Factory shown in Figure 3

The text is then converted into inverted-index as shown in Figure 4. Finally the data is indexed into Solr. The indexed data can be searched and retrieved in Velocity, a configurable search interface of Solr as shown in Figure 5.

11. RELATED WORK

This paper [1] analyses tweets to extract information about disasters so that the information can be use at real time. They have used a variety of classification models like sLDA, SVM and logistic regression. They did classification, clustering and finally extraction to retrieve information from tweets. Unlike the idea in this paper, we use tweets just to identify webpages that contains relevant information about the disaster.

Twitter, as one of the most popular microblogging services, provides large quantities of fresh information including real time news, comments, conversation,pointless babble and advertisements.Twitter presents tweets in chronological order. The work of Duan et al. [3] proposes a new ranking strategy which uses not only the content relevance of a tweet, but also the account authority and tweet - specific features such as whether a URL link is included in the tweet .They used learning to rank algorithms to determine the best set of features with a series of experiments. It is demonstrated that whether a tweet contains URL or not , length of tweet and account authority are the best conjunction .

Cross-domain text classification aims to automatically train a precise text classifier for a target domain by using labeled text data from a related source domain. Most existing methods do not explore the duality of the marginal distribution of examples and the conditional distribution of class labels given labeled training examples in the source domain.In this paper [2],Bao proposes a model called Partially Supervised Cross-Collection LDA topic model [4] (PSCCLDA) for cross-domain learning with the purpose of addressing these two issues in a unified way. Experimental results on nine datasets show that their model outperforms two standard classifiers and four state-of-the art methods, which demonstrates the effectiveness of their proposed model.

A framework for summarizing and analyzing twitter feeds, [10]proposes a novel technique to analyze and summarize twitter data. The proposed method is light weight because it is an incremental summary construction which is efficient. It also enables low reconstruction error.

Yiming and Liu [11] represented a controlled study with statistical

```
<fieldtype name="phonetic" stored="false" indexed="true"
class="solr.TextField" >
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.DoubleMetaphoneFilterFactory"
inject="false"/>
  </analyzer>
</fieldtype>
```

Figure 3: Configuration Setup

```
<field name="id" type="text" indexed="true" stored="true"
multiValued="true"/>
```

Figure 4: Inverted Index

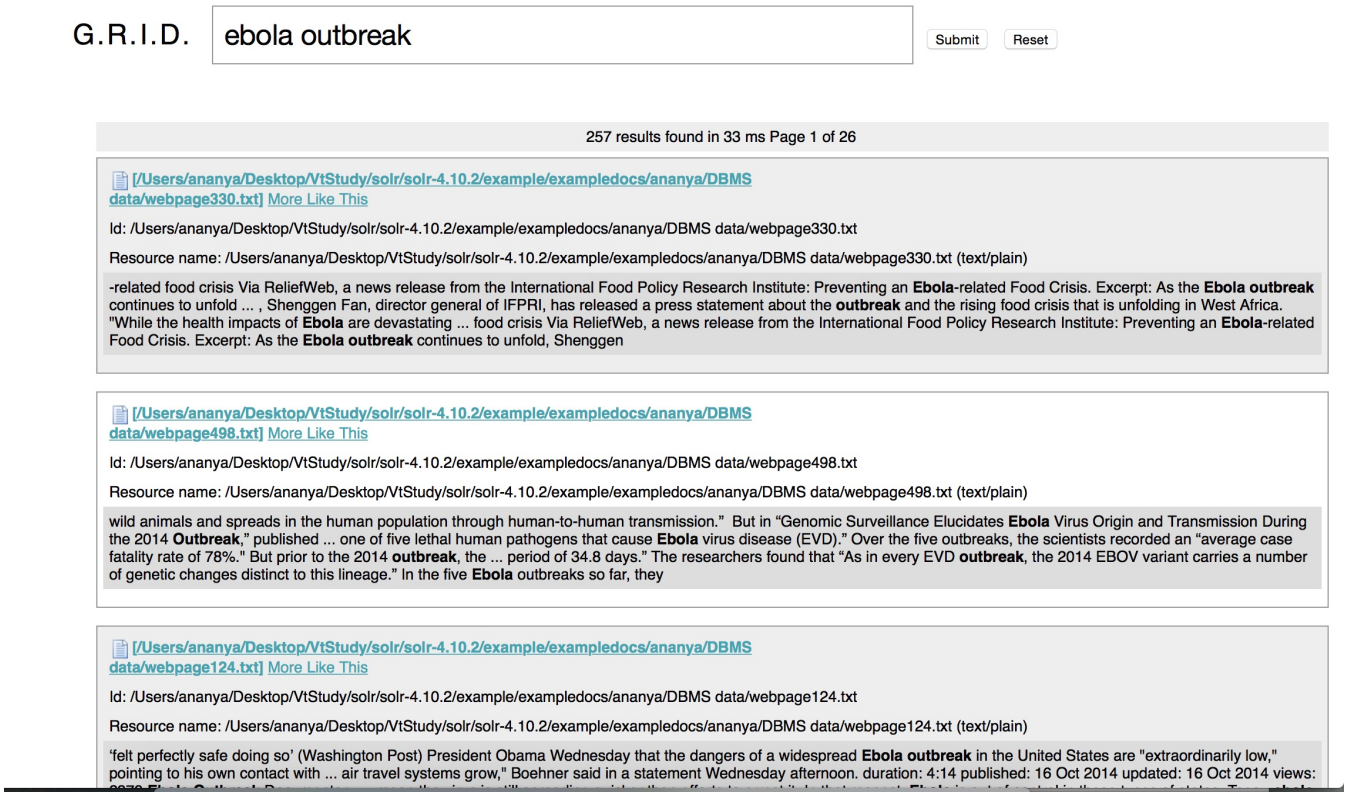

Figure 5: G.R.I.D search interface

significance tests on five text categorization methods: the Support Vector Machines (SVM), a k-Nearest Neighbor (kNN) classifier [6], a neural network (NNet) approach, the Linear Least-squares Fit (LLSF) mapping and a Naive Bayes (NB) classifier.The paper focuses on the robustness of these methods in dealing with a skewed category distribution, and their performance as function of the training set category frequency [8].They show that SVM, kNN and LLSF significantly outperform NNet and NB when the number of positive training instances per category are small, and that all the methods perform comparably when the categories are sufficiently common.

Lan, [4] represents an empirical study of top-k learning to rank training in this paper . The challenge in learning to rank algorithms is getting reliable training data. So top k learning algorithm proposes to utilize top-k ground truth for training where k is usually small. The underlying assumption is that training on top-k grade truth is as good as full-order ranking list. They study whether this underlying assumption of top-k learning ground-truth is sufficient for training holds.

12. PROBLEMS FACED

We have faced several challenges during project implementation.
1. Identifying non-relevant datasets. We wanted to create a strong support vector based on boundary conditions that accurately demarcates relevant and non-relevant tweets.
2. As the length of the text is small (140 characters), identifying features for correct classification and improving Recall was extremely tough.
3. Unshortening of the URLs was a very time consuming process as we had to handle several runtime http exceptions thrown even when the URL was valid. The dynamics of the program kept changing with the increases in input data size i.e. number of URLs that needed to be expanded.
4. The http client used by AWS is an older version which has a bug that results in the unwarranted termination of our program.
5. Each webpage has a unique parent-child relationship. So to identify a generic model to fetch relevant information from the webpages was tricky and time-consuming.
6. There is very less documentation available online on Solr and Velocity, which made indexing text to Solr and modifying the search interface (Velocity) challenging.

13. EXPERIMENT AND RESULTS

13.1 Testbed

We trained our system on a part of hum-provided labels (we manually labelled the training set) and 'tested the classifier on the remaining part.. We evaluated our system by comparing outputs to the expected responses. We measured the below two aspects which are related to the sensitivity and the specificity of our system.

Hit ratio/ precision rate (PR): Precision rate measures the fraction of examples for which our system found something, and that something could be considered correct by humans. We will consider the output correct if it overlaps in at least one word with the given human label.

$$PR = \frac{num\ of\ rightly\ classified\ and\ retrieved\ data}{total\ retrieved\ data\ in\ the\ category} \quad (12)$$

Detection rate / recall rate (RR): Recall rate will measure the fraction of examples in which humans found a relevant piece of information, and our system also found something.

$$RR = \frac{num\ of\ rightly\ classified\ and\ retrieved\ data}{rightly\ classified\ data\ in\ the\ category} \quad (13)$$

F-1 score: F-1 score is the weighted average of the precision and recall where an F_1 score best value is 1 and the worst score is 0.

$$f-1\ score = 2.\frac{precision\ recall}{Precision + recall} \quad (14)$$

The extraction of text using BeautifulSoup has been evaluated manually before indexing the text to Solr. The evaluation is to verify if only the relevant text has been extracted.

13.2 Results and Discussion

We used Kfold cross validation for both models Figure[2,3].For Naive Bayes ,we notice that the performance increases steadily with the increase in k. It is the highest when k=10. This is because with the increase of K, the size of training data increases which means more information to the classifier when it is trained. Also, the size of test data decreases which means less data to be classified that is minimum error ratio. However this does not signify that the performance will continue to increase infinitely as k increases. With increase in the volume of training data, the classifier tends to over-fit information which leads to more failures. So heuristics suggest that performance is optimal when k=10.
We have implemented the k-fold cross validation for the SVM classifier. It has been observed that SVM accuracy, precision, recall is less than the Naive Bayes.The less accuracy is due to the more number of false negative as in the learning data only the positive tweets has been classified and the remaining tweets are considered as a negative sample.
We also used train_test_split cross validation on both models (Multinomial Naive Bayes and Linear SVC)Figure[4,5]. For Naive Bayes classifier, we notice that precision is optimal when test size is 0.15 but recall decreases. This is a classical recall-precision trade-off where precision decreases while recall increases. We need to find a well-defined maximum which happens at test_size=0.20.We observe that for the train_test_split the accuracy of the SVM is approximately same as that of the Naive Bayes.
We have done the manual verification of the predicted labeled data and tried to analyze the reason for wrong classification.The analysis is to manually check whether the classification done by the classifier is wrong or the corresponding input label is wrong. This helped us in improving the classifier precision to 85% and recall to 80%. The URLs extracted from the data predicted as relevant i.e. 13,000 URLs by the classfier are unshortened using MapReduce which gives us unique responsive unshortened URLs which is further used for the text extraction using BeautifulSoup module. We observed that the implemenation of MapReduce for unshortening URL decreases the number of URLs to 50 % i.e. approx. 5238 of the input data indicative redundancy in URLS.The BeautifulSoup gives us relevant text contained in the corresponding webpages which has been verified manually.It has been observed that the text extracted using BeautifulSoup is redundant although the URLs are different. This can be justified as the relevant data remains same for different links. For example a news remains same, it is just available on different websites.

14. CONCLUSION

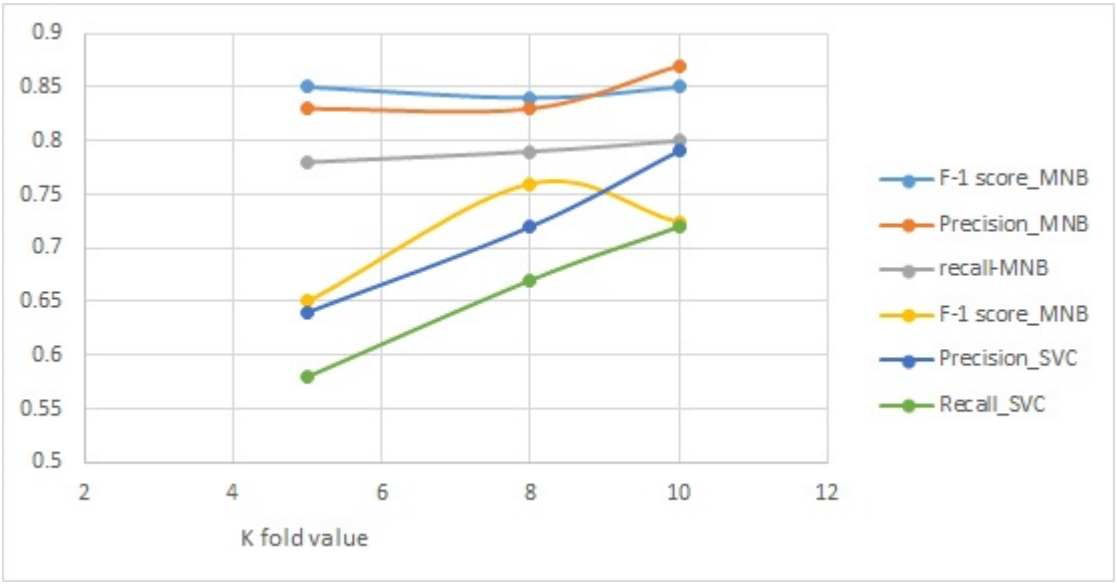

Figure 6: CrossValidation KFold graph with k=5,8,10 for Linear SVC and Multinomial Naive Bayes models

Model	Kfold	accuracy	precision	recall
Linear SVC	k=10	0.72	0.79	0.72
Linear SVC	k=8	0.76	0.79	0.45
Linear SVC	k=5	0.6	0.79	0.58
MultinomialNB	k=10	0.79	0.78	0.73
MultinomialNB	k=8	0.79	0.78	0.69
MultinomialNB	k=5	0.79	0.78	0.6

Figure 7: CrossValidation KFold data table with k=5,8,10 for Linear SVC and Multinomial Naive Bayes models

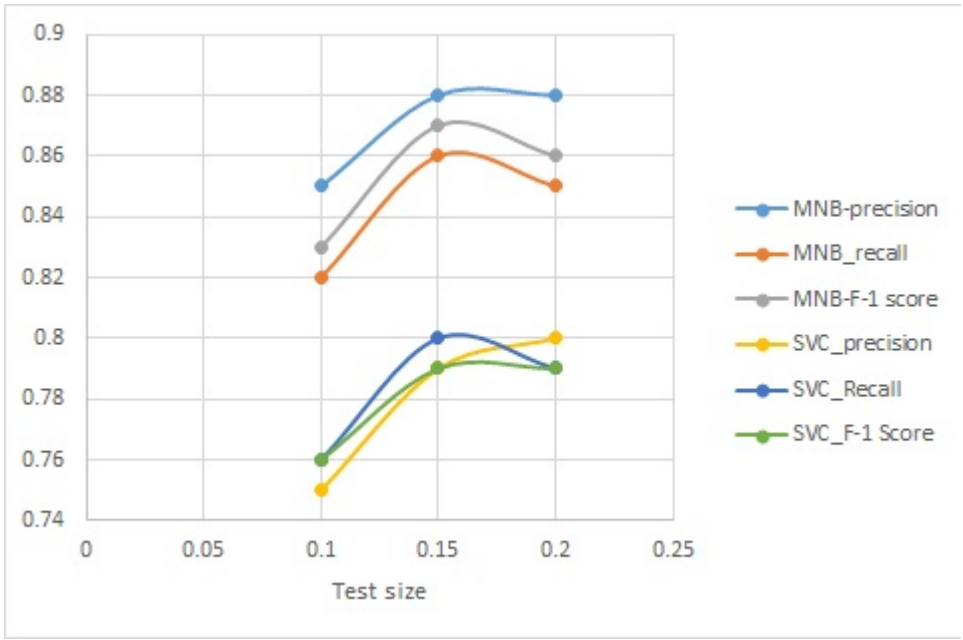

Figure 8: Train_Test_Split graph for Linear SVC and Multinomial Naive Bayes models

Model	Test data	F-1 score	precision	recall
Linear SVC	0.1	0.76	0.76	0.75
Linear SVC	0.15	0.79	0.8	0.79
Linear SVC	0.2	0.79	0.79	0.8
MultinomialNB	0.1	0.83	0.85	0.88
MultinomialNB	0.15	0.87	0.88	0.85
MultinomialNB	0.2	0.86	0.87	0.86

Figure 9: Train_Test_Split table for Linear SVC and Multinomial Naive Bayes models

Twitter has been used to spread news about casualties and damages, donation efforts and alerts, including multimedia information such as videos and photos.In general, a significant amount of data gets generated during times of a disaster, some of which are extremely valuable for any post-disaster analysis.We wanted to provide an efficient way to get the information related to tweets.In this project we have taken Ebola disaster to create an efficient web-archived search engine G.R.I.D. In the G.R.I.D. the classification takes places in two stages. The first stage classifications using approaches like Naive Bayes and the second stage exploits text mining techniques to get the relevant text from URL webpage. Before implementing the second stage of classification we try to unshorten the URL using map reduce. After classification is done we index the data into Solr and use Velocity interface to search the indexed data

15. SOFTWARES/PACKAGES USED

Python: version 2.7
Latex: version 2e
Python packages used:
NLTK : for stop words
Sklearn : for classification
Numpy : for converting text to array
Pandas: for reading .csv file
BeautifulSoup: python package for text extraction
Sunburnt: python package to index data to Solr
MapReduce: AWS
Solr: version 4.10

16. REFERENCES

[1] Z. Ashktorab, C. Brown, M. Nandi, and A. Culotta. Tweedr: Mining twitter to inform disaster response, 2014.

[2] Y. Bao, N. Collier, and A. Datta. A partially supervised cross-collection topic model for cross-domain text classification. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 239–248. ACM, 2013.

[3] Y. Duan, L. Jiang, T. Qin, M. Zhou, and H.-Y. Shum. An empirical study on learning to rank of tweets. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 295–303, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[4] Y. Lan, S. Niu, J. Guo, and X. Cheng. Is top-k sufficient for ranking? In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1261–1270. ACM, 2013.

[5] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134, 2000.

[6] F. Rousseau and M. Vazirgiannis. Graph-of-word and tw-idf: new approach to ad hoc ir. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 59–68. ACM, 2013.

[7] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.

[8] W. Shang, H. Huang, H. Zhu, Y. Lin, Y. Qu, and Z. Wang. A novel feature selection algorithm for text categorization. *Expert Systems with Applications*, 33(1):1–5, 2007.

[9] S. Vieweg, A. L. Hughes, K. Starbird, and L. Palen. Microblogging during two natural hazards events: What twitter may contribute to situational awareness. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1079–1088, New York, NY, USA, 2010. ACM.

[10] X. Yang, A. Ghoting, Y. Ruan, and S. Parthasarathy. A framework for summarizing and analyzing twitter feeds. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 370–378. ACM, 2012.

[11] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49. ACM, 1999.