# Real Time Analysis of Accident Prone Areas By Mining Twitter Data

Ameya Gamre,

University of Texas at Dallas

***Abstract*—In times of accidents, vast amounts of data are generated via computer-mediated communication (CMC) that are difficult to manually cull and organize into a coherent picture. Yet valuable information is broadcast, and can provide useful insight into time- and safety-critical situations if captured and analyzed properly and rapidly. We describe an approach for automatically identifying messages communicated via Twitter that contribute to situational awareness, and explain why it is beneficial for those seeking information during accidents. We will collect Twitter messages and build a classifier to automatically detect messages that may contribute to situational awareness, utilizing a combination of hand annotated and automatically-extracted linguistic features. Our system will categorize tweets that contribute to situational awareness. The results are promising, and have the potential to aid the general public in culling and analyzing information.**

## I. Introduction

IN this Digital Age, social media has played a major role in everyday life. It has become a major source of information for people and the only medium where the masses can voice their opinions and concerns. We intend to use this data for real-time analysis of accidents in Mumbai. We extend this research by focusing on Twitter communications (tweets) generated during accidents, and show how Natural Language Processing (NLP) techniques contribute to the task of sifting through massive datasets when time is at a premium and safety of people and property is in question. So much information is now broadcast during accidents that it is infeasible for humans to effectively find it, much less organize, make sense of, and act on it. To locate useful information, computational methods must be developed and implemented to augment human efforts at information comprehension and integration. The popular microblogging service Twitter serves as an outlet for many to offer and receive useful information; it provides a way for those experiencing such emergency to gather more, or different, information than they may be able to using mainstream media and other traditional forms of information dissemination. This access provides affected populations with the possibility to make more informed decisions. The challenge, however, is in locating the right information. In addition to broadcasting valuable, actionable information via Twitter during accidents, many also send general information that is void of helpful details, or communicate empathetic, supportive messages that lack tactical information. Tweets that include tactical, actionable information contribute to situational awareness; such tweets include content that demonstrates an awareness of the scope of the crisis as well as specific details about the situation. We offer an approach for automatically locating information that has the potential to contribute to situational awareness in the multitude of tweets broadcast during accidents. Our overarching goal is to help affected populations cull and analyze pertinent information communicated via computer-mediated communication. Our assumption is that immediate, dynamic culling of tweets with information pertaining to situational awareness could be used to inform and update applications aimed at helping members of the public, formal response agencies, aid organizations and concerned outsiders understand and act accordingly during accidents.

## II. Literature Review

In this section, some related work on Twitter APIs, NLP, text mining and Google maps will be discussed.

### *A. Natural Language Processing(NLP)*

It is defined in as Natural Language Processing is a theoretically motivated range of computational techniques for analysing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications [1]. In other words it is set of techniques for the machine to understand and processing naturally occurring text in way that we want it to. Text analysis is built on NLP as it inherently uses many natural language processing techniques such as :

**Splitting Words** Words Splitting or Tokenizing is decomposing the sentences into words.

**POS tagging** POS tagging assigns to every single word a label which correspond to its part of speech e.g. noun, adjective, verb, adverb etc.

**Stopwords removing** Not all words in the sentences carry useful information for classification task and that is why it is beneficial to get rid of such useless words. For German language some of such words are das, die, der, aber, als, am, an, auch, auf, aus, bei, etc. Removing these stopwords make the classification routine easier.

**Stemming** Stemming mainly deals with removing suffixes and prefixes from the words. The procedure of stemming can be explained by two sentences below. I like to watch this movie. And the second I liked her performance, it was marvellous. Here words like and

liked would be treated by classification algorithm as two completely different words. That is why it is beneficial to stem these words to one common word liki.

*1) Feature Extraction*

Feature Extraction is an extremely basic and essential task for Sentiment Analysis. Converting a piece of text to a feature vector is the basic step in any data driven approach to Sentiment Analysis [2]. In the following we will see some commonly used features used in Sentiment Analysis and their critiques.

- **Term Presence vs. Term Frequency** Term frequency has always been considered essential in traditional Information Retrieval and Text Classification tasks. But Pang-Lee et al found that term presence is more important to Sentiment analysis than term frequency. That is, binary-valued feature vectors in which the entries merely indicate whether a term occurs (value 1) or not (value 0). This is not counter-intuitive as in the numerous examples we saw before that the presence of even a single string sentiment bearing words can reverse the polarity of the entire sentence. It has also been seen that the occurrence of rare words contain more information than frequently occurring words, a phenomenon called HapaxLegomena.
- **Term Position** Words appearing in certain positions in the text carry more sentiment or weightage than words appearing elsewhere. This is similar to Information Retrieval where words appearing in topic Titles, Subtitles or Abstracts etc given more weightage than those appearing in the body. Although the text contains positive words throughout, the presence of a negative sentiment at the end sentence plays the deciding role in determining the sentiment. Thus generally words appearing in the first few sentences and last few sentences in a text are given more weightage than those appearing elsewhere.
- **N-gram Features** N-gram models can be imagined as placing a small window over a sentence or a text, in which only n words are visible at the same time. Few commonly used N-grams are

  **Unigrams:** The simplest n-gram model is therefore a so-called unigram model. This is a model in which we only look at one word at a time. Consider the sentence "Colorless green ideas sleep furiously". The above sentence contains five unigrams: ("colorless"), ("green"), ("ideas"), ("sleep"), and ("furiously"), where we only look at one word at a time.

  **Bigrams:** In similar fashion, a bigram can be thought of as a window that shows two words at a time. The sentence "Colorless green ideas sleep furiously", for instance, contains bigrams like: ("ideas","sleep"), ("Colorless","green") etc.

  N-grams are capable of capturing context to some extent and are widely used in Natural Language Processing tasks. Whether higher order n-grams are useful is a matter of debate. Pang et al. (2002) reported that unigrams outperform bigrams when classifying movie reviews by sentiment polarity, but Dave et al. (2003) found that in some settings, bigrams and trigrams perform better.

*B. Bayesian Classification*

**What are Bayesian classifiers?**
Bayesian classifiers are statistical classifiers. They can predict class membership probabilities, such as the probability that a given tuple belongs to a particular class. Bayesian classification is based on Bayes theorem, described below. Studies comparing classification algorithms have found a simple Bayesian classifier known as the naive Bayesian classifier to be comparable in performance with decision tree and selected neural network classifiers. Bayesian classifiers have also exhibited high accuracy and speed when applied to large databases. Naive Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence. It is made to simplify the computations involved and, in this sense, is considered naive. Bayesian belief networks are graphical models, which unlike naive Bayesian classifiers, allow the representation of dependencies among subsets of attributes. Bayesian belief networks can also be used for classification.

*C. Mining the Twitter Data*

Establishing a connection to the streaming APIs means making a very long lived HTTP request, and parsing the response incrementally. The streaming process gets the input Tweets and performs any parsing, filtering, and/or aggregation needed before storing the result to a data store. The HTTP handling process queries the data store for results in response to user requests.

*1) Preprocessing the Twitter Data*

Extracting the data. Preprocessing the data. Removing Stop Words. Filtering the data based on hash tags, locations and users. Raw Tweet- FIR led against ArvindKejriwal and AAP workers for damaging public property during his Mumbai visit #AAPChaos http://www.timesnow.tv/videoshow/4449958.cms Pre processed tweet - fir led against arvindkejriwal and aap workers for damaging public property during his mumbai visit aapchaos URL

*D. Maps API*

Hash table will be implemented in Python which will contain the names of the places and their respective latitudinal and longitudinal co-ordinates. Using NLP, we will map the places mentioned in the tweets to their corresponding co-ordinates. We will also get the description of the accident using NLP. To add markers to a map, create a marker object and set the latitudinal and longitudinal co-ordinates. To add a title to the marker which will be displayed on hovering over the marker, put the title inside title attribute as a string. We need to create an

infoWindow to show the detailed information on clicking the marker and put the description inside the content attribute. To customize a map marker, specify the icon option on MarkerOptions to change the markers icon. The icon option can be either a string (the URL to the marker icon), or an Icon Object. For e.g.

```
var marker = new google.maps.Marker({
position: myLatLng,
map: map,
icon: iconBase + 'marker_maps.png'
});
```

*E. Related Work*

In current study by [3], a mechanism to extract accident related information such as congestion and incidents from textual data from the internet is proposed. The current source of data is Twitter, however, the same mechanism can be extended to any kind of text available on the internet. As the data being considered is extremely large in size automated models are developed to stream, download, and mine the data in real-time. Furthermore, if any tweet has accident related information then the models should be able to infer and extract this data. These models are designed in such a way that they are able to detect the accident congestion and accident incidents from the Twitter stream at any location. In the first step tweets are streamed from Twitter for a set of urban areas. These tweets are analyzed to determine whether they contain any accident related information. This task is achieved by using machine learning classification models [3]. Once accident related tweets are identified, they are then stored in the database for offline analysis. The process to generate the final database containing accident related tweets consists of several steps. First all geotagged tweets are logged from the twitter stream. Next the tweets with the string "accident" are segregated into a separate database. These tweets are then further filtered for tweets related to accident based on the removal of tweets with the string "accident" that are unrelated to vehicle accident (e.g. accidentking) and removing accident tweets generated by professional accident monitoring services. Also, accident related words (e.g. arterial, accident, etc.) present in the accident tweets are determined. As stated [3], a tweet database is created to handle the streams of information being collected in real-time. Every time a tweet is extracted using the Twitter API the database is updated. Finally, as part of the initial analysis it was seen that there are some professional accounts on Twitter which only publish traffic related tweets. These accounts tend to maintain a standard format of the tweets and purposefully send out the tweets as news to help the commuters in a city. These professional users tweet a significant number of traffic related tweets. For developing classification models, such professional tweets are excluded to remove potential bias to the locations with higher volumes of these professional accounts.

False positives are of greater concern, since they represent noise that could be misleading [4]. To address them, we will explore incorporating user feedback; increasing the weight of tweets that have been retweeted; and the effects of using different limits in our machine learning algorithms. Because Twitter has a high degree of redundancy, it is less likely that all tweets that represent the same information and are written in different styles will be misclassified. To measure classifier accuracy, we tested a sample of manually annotated tweets [4]. As a deployed system, it will continuously classify incoming tweets based on models built on data from previous similar events.

The most important contribution this study makes to social media research is to demonstrate that using sentiment analysis to learn from customers is likely less effective than humans reading streams of consumer chatter. This result [5] is invaluable for improving social media monitoring practices. Empirical proof that an NLP approach is potentially superior to SA suggests that efforts to build an information system based on NLP techniques are a worthwhile and beneficial goal. NLP-based software promises the potential to substantially increase the knowledge firms may glean from tapping into customer-to customer exchanges and enhance the effectiveness with which they monitor and respond to customer to-firm communications [5].

According to [6], an effort was made to identify the accident prone areas within Kannur District, Kerala. For this purpose, the road accident data for the years 2006, 2007 and 2008 pertaining to Kannur district was used. Accident analysis studies aim at the identification of high rate accident locations and safety deficient areas. Methodology and Materials Used

- **Data Collection** Following data were collected and used. Police stations limit map obtained from the office of superintendent of police, Kannur. Accident reports for the years 2006, 2007 and 2008. Survey of India topological map at a scale 1:50000.
- **Collection of Ground Control Points** The GCP is normally collected with the help of the GPS. Here there are eight GCPs collected from the survey of India topo sheet at various road intersections.
- **Data processing**
- **Map scanning** The Survey of India topographical map at a sale of 1:50000 were scanned as the raster input.
- **Geo-referencing** Scanned maps usually do not contain information as to where the area represented on the map fits on the surface of the earth.
- **Digitizing** Digitizing is the process of encoding the geographic features in digital form as x, y coordinates. It was carried out to create spatial data from existing hard copy maps and documents.
- **Assigning attributes** All vector data (i.e. line, polygon, point features) will contain separate attribute tables. Here each road was labeled with its corresponding name with the help of the city map obtained from the police station. Similarly the accident loca-

TABLE I
ACCIDENT PRONE LOCATIONS BY ASV

| Serial No. | Stretch(in km.) | Fatal injury | Serious injury | Minor Injury | Non Injury | Total No. of Accidents | Accident Severity Value | Ranking of Stretch |
|---|---|---|---|---|---|---|---|---|
| 1. | 98-104 | 10 | 36 | 40 | 115 | 191 | 630 | 2 |
| 2. | 105-109 | 7 | 47 | 23 | 78 | 139 | 581 | 6 |
| 3. | 110-114 | 4 | 27 | 18 | 96 | 149 | 487 | 7 |
| 4. | 115-119 | 6 | 27 | 15 | 58 | 101 | 431 | 9 |
| 5. | 120-124 | 4 | 25 | 10 | 72 | 110 | 429 | 10 |
| 6. | 125-129 | 2 | 37 | 20 | 67 | 127 | 459 | 8 |
| 7. | 130-134 | 8 | 50 | 31 | 67 | 137 | 584 | 5 |
| 8. | 135-139 | 12 | 45 | 37 | 67 | 142 | 599 | 4 |
| 9. | 140-144 | 10 | 61 | 37 | 62 | 161 | 649 | 1 |
| 10. | 145-148 | 10 | 70 | 24 | 44 | 131 | 658 | 3 |

tion attribute table contains the following data.
1. User identification Number (UID)
2. Police station limit
3. Month and date of occurrence
4. Time of occurrence
5. Exact area of occurrence
6. Type of accident
7. Type of injury
8. Type of vehicle involved

- **Mapping** Road network of the study area was digitized as line features. Accident locations are digitized as point features.

Another study carried out by [7] on road accident data of a selected stretch of NH-1 (Delhi-Ambala-Amritsar Road) measured the severity of accidents. Total accidents and accident severity value was used to rank the accident prone locations.

It is observed from Table I that maximum number of accidents take place on 98-104 km stretch followed by 140-144 km and 110-114 km. However, when we take severity of accident into consideration, stretch 140-144 km comes out to be the most critical followed by 98-104 km and 145-148 km [7].

These results were a great help to analyze the accident prone areas as they enabled the local authorities to take appropriate actions. But these details were based on past data and real time analysis of data was needed as accidents still continued to rise. There was a need for better techniques for the analysis of real time accident data to curb down these accidents.

## III. Naive Bayesian Classifier

### A. Bayes Theorem

Bayes theorem is named after Thomas Bayes, a nonconformist English clergyman who did early work in probability and decision theory during the 18th century [8]. Let X be a data tuple. In Bayesian terms, X is considered evidence. As usual, it is described by measurements made on a set of n attributes. Let H be some hypothesis, such as that the data tuple X belongs to a specified class C. For classification problems, we want to determine P(H|X), the probability that the hypothesis H holds given the evidence or observed data tuple X. In other words, we are looking for the probability that tuple X belongs to class C, given that we know the attribute description of X. P(H|X) is the posterior probability, or a posteriori probability, of H conditioned on X. For example, suppose our world of data tuples is confined to customers described by the attributes age and income, respectively, and that X is a 35-year-old customer with an income of $40,000. Suppose that H is the hypothesis that our customer will buy a computer. Then P(H|X) reflects the probability that customer X will buy a computer given that we know the customers age and income. In contrast, P(H) is the prior probability, or a priori probability, of H. For our example,this is the probability that any given customer will buy a computer, regardless of age, income, or any other information, for that matter. The posterior probability, P(H|X), is based on more information (e.g., customer information) than the prior probability, P(H), which is independent of X. Similarly, P(X|H) is the posterior probability of X conditioned on H. That is, it is the probability that a customer, X, is 35 years old and earns $40,000, given that we know the customer will buy a computer. P(X) is the prior probability of X.Using our example, it is the probability that a person from our set of customers is 35 years old and earns $40,000. P(H), P(X|H), and P(X) may be estimated from the given data, as we shall see below. Bayes theorem is useful in that it provides a way of calculating the posterior probability, P(H|X), from P(H), P(X|H), and P(X). Bayes theorem is

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)} \tag{1}$$

**Naive Bayesian Classification**

The naive Bayesian classifier, or simple Bayesian classifier, works as follows:

1. Let D be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n-dimensional attribute vector, X = (x1, x2, ... , xn), depicting n measurements made on the tuple from n attributes, respectively, A1, A2, ... , An.

2. Suppose that there are m classes, C1, C2, ... , Cm. Given a tuple, X, the classifier will predict that X belongs to the class having the highest posterior probability, conditioned

on X. That is, the naive Bayesian classifier predicts that tuple X belongs to the class Ci if and only if

$$P(Ci|X) > P(Cj|X) for 1 <= j <= m; j \neq i \quad (2)$$

Thus we maximize P(Ci|X). The class Ci for which P(Ci|X) is maximized is called the maximum posteriori hypothesis. By Bayes theorem (Equation 1),

$$P(Ci|X) = \frac{P(X|Ci)P(Ci)}{P(X)} \quad (3)$$

3. As P(X) is constant for all classes, only P(X|Ci)P(Ci) need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, P(C1) = P(C2) = ... = P(Cm), and we would therefore maximize P(X|Ci). Otherwise, we maximize P(X|Ci)P(Ci). Note that the class prior probabilities may be estimated by P(Ci)=|Ci,D|/|D|,where |Ci,D| is the number of training tuples of class Ci in D.
4. Given data sets with many attributes, it would be extremely computationally expensive to compute P(X|Ci). In order to reduce computation in evaluating P(X|Ci), the naive assumption of class conditional independence is made. This presumes that the values of the attributes are conditionally independent of one another, given the class label of the tuple (i.e., that there are no dependence relationships among the attributes). Thus,

$$P(X|Ci) = \prod_{1}^{n} P(xk|Ci) = P(x1|Ci)x(x2|Ci)x....x(xn|Ci) \quad (4)$$

We can easily estimate the probabilities P(x1|Ci), P(x2|Ci), ... , P(xn|Ci) from the training tuples. Recall that here xk refers to the value of attribute Ak for tuple X. For each attribute, we look at whether the attribute is categorical or continuous-valued. For instance, to compute P(X|Ci), we consider the following: (a) If Ak is categorical, then P(xk|Ci) is the number of tuples of class Ci in D having the value xk for Ak, divided by |Ci,D|, the number of tuples of class Ci in D. (b) If Ak is continuous-valued, then we need to do a bit more work, but the calculation is pretty straightforward. A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean $\mu$ and standard deviation $\sigma$, defined by

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (5)$$

so that

$$P(xk|Ci) = g(xk, \mu Ci, \sigma Ci) \quad (6)$$

These equations may appear daunting, but hold on!We need to compute $\mu Ci$ and $\sigma Ci$ , which are the mean (i.e., average) and standard deviation, respectively, of the values of attribute $A_k$ for training tuples of class $C_i$. We then plug these two quantities into Equation (5), together with $x_k$, in order to estimate $P(x_k|C_i)$. For example, let X = (35, $40,000), where A1 and A2 are the attributes age and income, respectively. Let the class label attribute be buys computer. The associated class label for X is yes (i.e., buys computer = yes). Lets suppose that age has not been discretized and therefore exists as a continuous-valued attribute. Suppose that from the training set, we find that customers in D who buy a computer are $38 \pm 12$ years of age. In other words, for attribute age and this class, we have u = 38 years and s = 12.We can plug these quantities, along with x1 = 35 for our tuple X into Equation (5) in order to estimate P(age = 35|buys computer = yes).
5. In order to predict the class label of X, P(X|Ci)P(Ci) is evaluated for each class Ci. The classifier predicts that the class label of tuple X is the class Ci if and only if

$$P(X|Ci)P(Ci) > P(X|Cj)P(Cj) for 1 <= j <= m, j \neq i \quad (7)$$

In other words, the predicted class label is the class Ci for which P(X|Ci)P(Ci) is the maximum.

"How effective are Bayesian classifiers?" Various empirical studies of this classifier in comparison to decision tree and neural network classifiers have found it to be comparable in some domains. In theory, Bayesian classifiers have the minimum error rate in comparison to all other classifiers. However, in practice this is not always the case, owing to inaccuracies in the assumptions made for its use, such as class conditional independence, and the lack of available probability data. Bayesian classifiers are also useful in that they provide a theoretical justification for other classifiers that do not explicitly use Bayes' theorem. For example, under certain assumptions, it can be shown that many neural network and curve-fitting algorithms output the maximum posteriori hypothesis, as does the Naive Bayesian classifier.

### *B. Properties of Naive Bayes*

We decide class membership of a document by assigning it to the class with the probability (cf. probtheory) [9], which we compute as follows:

$$C_{map} = \operatorname*{argmax}_{c \in C} P(c|d) \quad (8)$$

$$= \operatorname*{argmax}_{c \in C} \frac{P(d|c)P(c)}{P(d)} \quad (9)$$

$$= \operatorname*{argmax}_{c \in C} P(d|c)P(c) \quad (10)$$

where Bayes' rule is applied in (9) and we drop the denominator in the last step because is the same for all classes and does not affect the argmax.

We can interpret Equation 10 as a description of the generative process we assume in Bayesian text classification. To generate a document, we first choose class with probability. The two models differ in the formalization of the second step, the generation of the document given the class, corresponding to the conditional distribution :

$$MultinomialP(d|c) = P(< t_1, ..., t_k, ..., t_{n_d} > |c) \quad (11)$$

$$BernoulliP(d|c) = P(< e_1, ..., e_i, ..., e_m > |c) \quad (12)$$

where $< t_1, ..., t_{n_d}$ is the sequence of terms as it occurs in d (minus terms that were excluded from the vocabulary)

and $< e_1, ..., e_i, ..., e_m >$ is a binary vector of dimensionality that indicates for each term whether it occurs in or not. A critical step in solving a text classification problem is to choose the document representation. and are two different document representations.

In the first case, X is the set of all term sequences (or, more precisely, sequences of term tokens). In the second case, X is $\{0, 1\}^M$.We cannot use equation (12) for text classification directly. For the Bernoulli model, we would have to estimate $2^M|C|$ different parameters, one for each possible combination of M values $e_i$ and a class. The number of parameters in the multinomial case has the same order of magnitude. This being a very large quantity, estimating these parameters reliably is infeasible.

To reduce the number of parameters, we make the Naive Bayes conditional independence assumption. We assume that attribute values are independent of each other given the class:

$$\begin{aligned} MultinomialP(d|c) &= P(< t_1, ..., t_{n_d} > |c) \\ &= \prod_{1 \le k \le n_d} P(X_k = t_k|c) \end{aligned} \tag{13}$$

$$\begin{aligned} BernoulliP(d|c) &= P(< e_1, ..., e_m > |c) \\ &= \prod_{1 \le i \le M} P(U_k = e_k|c) \end{aligned} \tag{14}$$

We have introduced two random variables here to make the two different generative models explicit. $X_k$ is the random variable for position k in the document and takes as values terms from the vocabulary. $P(X_k = t_k|c)$ is the probability that in a document of class c the term t will occur in position k. $U_i$ is the random variable for vocabulary term $i$ and takes as values *0 (absence)* and *1(presence)*. $P(U_k = e_k|c)$ is the probability that in a document of class the term will occur in any position and possibly multiple times.

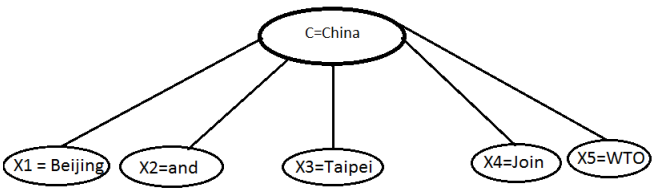



Fig. 1. Multinomial NB Model

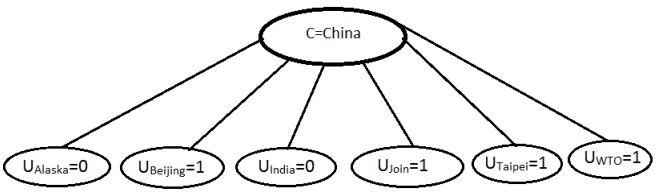



Fig. 2. Bernoulli NB Model

We illustrate the conditional independence assumption in Figure 2 . The class China generates values for each of the five term attributes (multinomial) or six binary attributes (Bernoulli) with a certain probability, independent of the values of the other attributes. The fact that a document in the class China contains the term Taipei does not make it more likely or less likely that it also contains Beijing. In reality, the conditional independence assumption does not hold for text data. Terms are conditionally dependent on each other. NB models perform well despite the conditional independence assumption. Even when assuming conditional independence, we still have too many parameters for the multinomial model if we assume a different probability distribution for each position in the document. The position of a term in a document by itself does not carry information about the class. Although there is a difference between China sues France and France sues China, the occurrence of China in position 1 versus position 3 of the document is not useful in NB classification because we look at each term separately. The conditional independence assumption commits us to this way of processing the evidence. Also, if we assumed different term distributions for each position $k$, we would have to estimate a different set of parameters for each $k$. The probability of bean appearing as the first term of a coffee document could be different from it appearing as the second term, and so on. This again causes problems in estimation owing to data sparseness. For these reasons, we make a second independence assumption for the multinomial model, *positional independence* : The conditional probabilities for a term are the same independent of position in the document.

$$P(X_{k_1} = t|c) = P(X_{k_2} = t|c) \tag{15}$$

for all positions *k1,k2*, terms $t$ and classes $c$. Thus, we have a single distribution of terms that is valid for all positions $k_i$ and we can use X as its symbol. Positional independence is equivalent to adopting the bag of words model. With conditional and positional independence assumptions, we only need to estimate $\theta(M|C|)$ parameters $P(t_k|c)$(multinomial model) or $P(e_i|c)$(Bernoulli model), one for each term-class combination, rather than a number that is at least exponential in $M$, the size of the vocabulary. The independence assumptions reduce the number of parameters to be estimated by several orders of magnitude. To summarize, we generate a document in the multinomial model 1 by first picking a class $C$=$c$ with *P(c)* where $C$ is a random variable taking values from $c$ as values. Next we generate term $t_k$ in position $k$ with $P(X_k = t_k|c)$ for each of the $n_d$ positions of the document. The $X_k$ all have the same distribution over terms for a given $c$. In the example in Figure 2 , we show the generation of $< t_1, t_2, t_3, t_4, t_5 >=< Beijing, and, Taipei, join, WTO >$, corresponding to the one-sentence document Beijing and Taipei join WTO. For a completely specified document generation model, we would also have to define a dis-

tribution $P(n_d|c)$ over lengths. Without it, the multinomial model is a token generation model rather than a document generation model. We generate a document in the Bernoulli model (Figure 2 ) by first picking a class $C{=}c$ with P(c) and then generating a binary indicator $e_i$ for each term $t_i$ of the vocabulary ($1 \leq i \leq M$). In the example in Figure 2 , we show the generation of , corresponding, again, to the one-sentence document Beijing and Taipei join WTO where we have assumed that 'and' is a stop word.

TABLE II
Multinomial vs Bernoulli Model

| | **Multinomial model** | **Bernoulli model** |
|---|---|---|
| Event Model | generation of token | generation of document |
| Random Variable(s) | X = t iff t occurs at given pos | $U_t$ iff t occurs in doc |
| Document Representation | $d =< t_1, ..., t_k, ...t_{n_d} >, t_k \in V$ | $d =< e_1, ...e_i, ..e_M >, e_i \in 0, 1$ |
| Parameter estimation | P(X=t\|c) | $P(U_i = e_i\|c)$ |
| Decision Rule:maximize | $P(c)\prod_{t_i \in V} P(U_i = e_i\|c)$ | $P(c)\prod_{1 \leq k \leq n_d} P(t_k\|c)$ |
| multiple occurrences | taken into account | ignored |
| length of docs | can handle longer docs | works best for short docs |
| # features | can handle more | works best with fewer |
| estimate for term 'the' | P(X=the\|c)=0.05 | $P(U_{the} = 1\|c) = 1.0$ |

We compare the two models in Table II, including estimation equations and decision rules. Naive Bayes is so called because the independence assumptions we have just made are indeed very naive for a model of natural language. The conditional independence assumption states that features are independent of each other given the class. This is hardly ever true for terms in documents. In many cases, the opposite is true. In addition, the multinomial model makes an assumption of positional independence. The Bernoulli model ignores positions in documents altogether because it only cares about absence or presence. This *bagofwords* model discards all information that is communicated by the order of words in natural language sentences. How can NB be a good text classifier when its model of natural language is so oversimplified?

TABLE III
Correct estimation implies accurate prediction, but accurate prediction does not imply correct estimation

| | **C1** | **C2** | **Class selected** |
|---|---|---|---|
| true probability P(c\|d) | 0.6 | 0.4 | $c_1$ |
| $P(c)\prod_{1 \leq k \leq n_d} P(t_k\|c)$ | 0.00099 | 0.00001 | $c_1$ |
| NB estimate P(c\|d) | 0.99 | 0.01 | $c_1$ |

The answer is that even though the probability estimates of NB are of low quality, its classification decisions are surprisingly good. Consider a document $d$ with true probabilities $P(c_1|d){=}0.6$ and $P(c_2|d){=}0.4$ as shown in Table III . Assume that d contains many terms that are positive indicators for *c1* and many terms that are negative indicators for *c2*. Thus, when using the multinomial model in Equation 13, $P(c_1)\prod_{1 \leq k \leq n_d} P(t_k|c1)$ will be much larger than $P(c_2)\prod_{1 \leq k \leq n_d} P(t_k|c2)$ (0.00099 vs. 0.00001 in the table). After division by 0.001 to get well-formed probabilities for *P(c/d)*, we end up with one estimate that is close to 1.0 and one that is close to 0.0. This is common: The winning class in NB classification usually has a much larger probability than the other classes and the estimates diverge very significantly from the true probabilities. But the classification decision is based on which class gets the highest score. It does not matter how accurate the estimates are. Despite the bad estimates, NB estimates a higher probability for $c_1$and therefore assigns to the correct class in Table III . Correct estimation implies accurate prediction, but accurate prediction does not imply correct estimation. NB classifiers estimate badly, but often classify well.

Even if it is not the method with the highest accuracy for text, NB has many virtues that make it a strong contender for text classification. It excels if there are many equally important features that jointly contribute to the classification decision. It is also somewhat robust to noise features and concept drift – the gradual change over time of the concept underlying a class like US president from Bill Clinton to George W. Bush. Classifiers like kNN knn can be carefully tuned to idiosyncratic properties of a particular time period. This will then hurt them when documents in the following time period have slightly different properties.

The Bernoulli model is particularly robust with respect to concept drift. The most important indicators for a class are less likely to change. Thus, a model that only relies on these features is more likely to maintain a certain level of accuracy in concept drift.

### C. Multinomial Model

In contrast to the multi-variate Bernoulli event model,the multinomial model captures word frequency information in documents. Consider, for example, the occurrence of numbers in the Reuters newswire articles; our tokenization maps all strings of digits to a common token. Since every news article is dated, and thus has a number, the number token in the multi-variate Bernoulli event model is uninformative. However, news articles about earnings tend to have a lot of numbers compared to general news articles. Thus, capturing frequency information of this token can help classification. In the multinomial model, a document is an ordered sequence of word events, drawn from the same vocabulary V . We assume that the lengths of documents are independent of class. We again make a similar naive Bayes assumption: that the probability of each word event in a document is independent of the word's context and position in the document. Thus, each document $d_i$ is drawn from a multinomial distribution of words with as many independent trials as the length of $d_i$. This yields the familiar 'bag of words' representation for documents. Define $N_{it}$ to be the count of the number of times word $w_t$ occurs in document $d_i$. Then, the probability of a document given its class is simply the multinomial distribution:

$$P(d_i|c_j;\theta) = P(|d_i|)|d_i|! \prod_{t=1}^{|V|} \frac{P(w_t|c_j;\theta)^{N_{it}}}{N_{it!}} \tag{16}$$

The parameters of the generative component for each class are the probabilities for each word, written $\theta_{w_t|c_j} = P(w_t|c_j;\theta)$, where $0 \leq \theta w_t|c_j \leq 1$ and $\sum_t \theta_{w_t|c_j} = 1$. Again, we can calculate Bayes-optimal estimates for these parameters from a set of labeled training data. Here, the estimate of the probability of word $w_t$ in class $c_j$ is:

$$\theta_{w_t|c_j} = P(w_t|c_j;\theta) = \frac{1 + \sum_{i=1}^{|D|} N_{it} P(c_j|d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{is} P(c_j|d_i)} \tag{17}$$

### D. Classifier Model

There are two different ways we can set up an NB classifier. The model we introduced in the previous section is the multinomial model. It generates one term from the vocabulary in each position of the document, where we assume a generative model. An alternative to the multinomial model is the multivariate Bernoulli model or Bernoulli model . It is equivalent to the binary independence model, which generates an indicator for each term of the vocabulary, either *1* indicating presence of the term in the document or *0* indicating absence. The Bernoulli model has the same time complexity as the multinomial model.

TrainBernoulliNB(C,D)

```
1: V ← ExtractVocabulary(D)
2: N ← CountDocs(D)
3: for each c ∈ C do
4:    N_c ← CountDocsInClass(D,c)
5:    prior[c] ← N_c/N
6:    for each t ∈ V do
7:       N_ct ← CountDocsInClassContainingTerm(D,c,t)
8:    end for
9: end for
10: condprob[t][c] ← (N_ct + 1)/(N_c + 2)
11: return V, prior, condprob
```

ApplyBernoulliNB(C,V,prior,condprob,d)

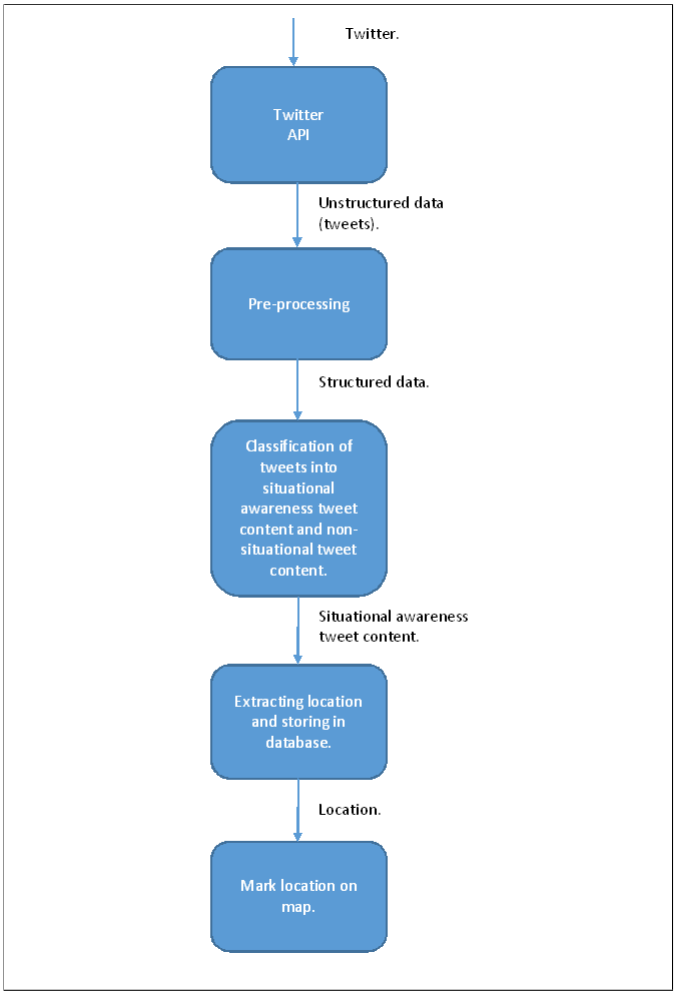
```
1: V_d ← ExtractTermsFromDoc(V,d)
2: for each c ∈ C do
3:    score[c] ← log prior[c]
4:    for each t ∈ V do
5:       if t ∈ V_d then
6:          score[c]+=log condprob[t][c]
7:       else
8:          score[c]+=log(1-condprob[t][c])
9:       end if
10:   end for
11: end for
12: return argmax_{c∈C} score[c]
```

**Algorithm 1:** Bernoulli Algorithm

The different generation models imply different estimation strategies and different classification rules. The Bernoulli model estimates P(t|c) as the fraction of documents of class that contain term. In contrast, the multinomial model estimates as the fraction of tokens or fraction of positions in documents of class $c$ that contain term $t$. When classifying a test document, the Bernoulli model uses binary occurrence information, ignoring the number of occurrences, whereas the multinomial model keeps track of multiple occurrences. As a result, the Bernoulli model typically makes many mistakes when classifying long documents. For example, it may assign an entire book to the class China because of a single occurrence of the term China. The models also differ in how non-occurring terms are used in classification. They do not affect the classification decision in the multinomial model, but in the Bernoulli model the probability of nonoccurrence is factored in when computing P(c|d). This is because only the Bernoulli NB model models absence of terms explicitly.

## IV. Proposed Algorithm

### A. Block Diagram




Step 1 Initially, Twitter data is streamed using Twitter APIs using the access keys and tokens which are needed for authentication.

Step 2 Now this data is unstructured and needs to be preprocessed. Tweet preprocessing will be done by removing the stop-words(such as the, from, here, your), emoticons, URLs and unnecessary punctuation marks.

Step 3 The Bernoulli model,is a probabilistic model which,on the basis of training data, will classify the tweets into situational and non-situational.

Step 4 If the tweets are situational, we extract the co-ordinates from the Location Table and locations from the Tweet Table.

Step 5 We finally map the individual accident with Red markers and Accident Prone Areas using Blue Markers on Google Maps.

## V. Output and Screenshots

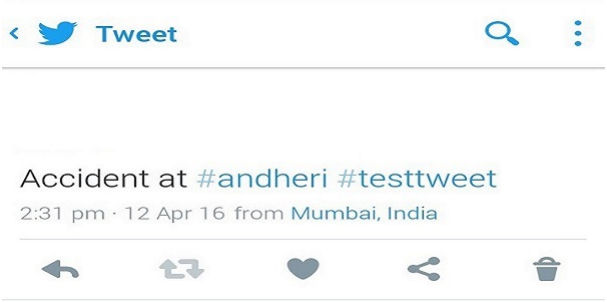



Fig. 3. Situational Tweet

```
situational
[u'andheri', u'bandra', u'borivali', u'dadar', u'jogeshwari', u'vikhroli',
['accident', 'andheri', 'ignore', 'testtweet']
Insertion complete..
```

Fig. 4. Classified : Situational


Fig. 5. Non Situational Tweet

```
non-situational
accident cat funny
Tweet classified...
```

Fig. 6. Classified : Non-Situational


Fig. 7. An example of a False Positive Tweet

```
situational
[u'andheri', u'bandra', u'borivali', u'dadar', u'jogeshwari', u'vikhroli',
['indias', 'top', 'womanbiker', 'veenupaliwal', 'killed', 'accident']
indias top womanbiker veenupaliwal killed accident
Tweet classified...
```

Fig. 8. An example of a False Positive(i.e.non-situational tweet classified as situational)

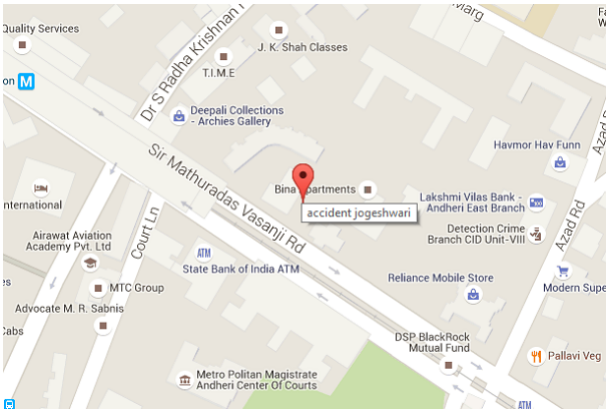



Fig. 9. Individual Accident

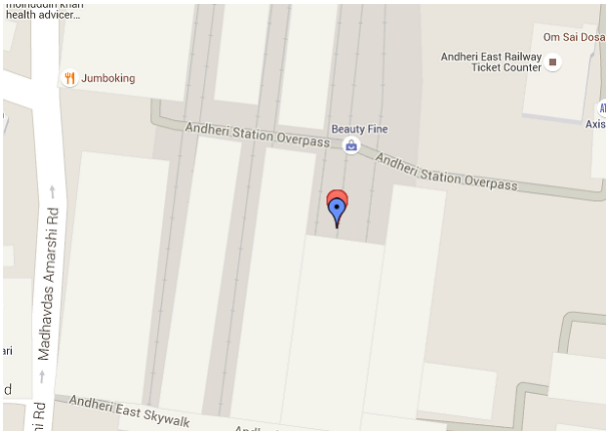



Fig. 10. Accident Prone Area

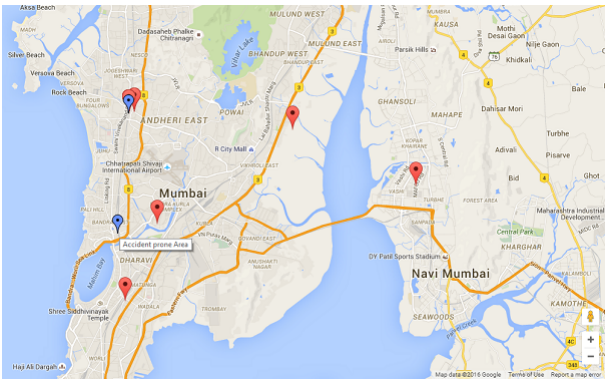



Fig. 11. Final Output

## VI. Accuracy

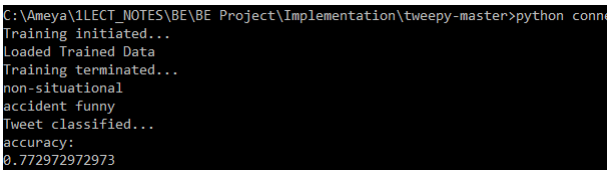



Fig. 12. Accuracy of classifier (for 182 test tweets)

TABLE IV
Accuracy Table

| No. of training tweets | No. of testing tweets | Accuracy |
|---|---|---|
| 200 | 182 | 77.29% |

Accuracy is calculated using the formula:

$$\text{Accuracy(in \%)} = \frac{No.\ of\ correctly\ classified\ Tweets}{Total\ No.\ of\ Tweets} * 100$$

## VII. Conclusion

We have learnt and implemented the following techniques: data streaming techniques, classification techniques and mapping techniques. For streaming the data in real-time from Twitter, we have written our code in python using the Tweepy framework. We have programmed the Bernoulli's model for classifying the tweets which contribute to situational awareness. Finally, we have implemented the Google Maps to map our results. We have trained the model using 100 situational tweets and 100 non-situational tweets. We have tested the model by sending real time tweets and recording the response. We have achieved the accuracy of 77.29%.

We hope to present this project as platform to keep a track on accident prone regions in Mumbai. The government agencies as well as several NGO's will now have a direct way to keep an eye on accident prone regions whereas the general public will know which route to avoid at a given time. Since the data is crowd-sourced, the entire responsibility of data is shared among users. We have implemented 2 levels of filters in order to root out the misleading tweets. However, the problem of false positives needs greater efforts and we hope to solve the same in future. The major benefits of this project include but not limited to: Low cost and crowd-sourced solution for a problem almost everyone faces in their lifetime, Saves human life and resources, Quick and real time response during times of emergency, No need of expensive hardware, sensors, trackers, cameras, etc.

## References

[1] N. P. Rishit Bhatia, Dhiraj Gurkhe, "Sentiment analysis engine using natural language processing," 2015, pp. 23–28.

[2] A. A. Sara Rosenthal and K. McKeown, "Sentiment detection of sentences and subjective phrases in social media," in *Proceedings of the 8th International Workshop on Semantic Evaluation*, 2014, pp. 198–202.

[3] W. J. C. Sudha Verma, Sarah Vieweg *et al.*, "Natural language processing to the rescue?: Extracting 'situational awareness' tweets during mass emergency," in *Proc. IEEE of the Fifth Int. AAAI Conf. on Weblogs and Social Media*, 2012.

[4] Larson and Keri, "The impact of natural language processing based textual analysis of social media interactions on decision making," in *Proc. of the 21st European Conf. on Information Systems*, 2012.

[5] C. P. Khatri, "Real-time road traffic information detection through social media," 2015.

[6] D. Jayan.K and B.Ganeshkumar, "Identification of accident hot spots: A gis based implementation for kannur district, kerala," *INTERNATIONAL JOURNAL OF GEOMATICS AND GEOSCIENCES*, vol. 1, no. 1, 2010.

[7] G. Goel and S. Sachdeva, "Identification of accident prone locations using accident severity value on a selected stretch of nh-1," in *IJERA National Conf. on Advances in Engineering and Technology*, 2014.

[8] J. Han and M. K. J. Pei, "Naive bayesian classification," in *Data Mining Concepts and Techniques, Third Edition*, 2012.

[9] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *Proc. The College of Information Sciences and Technology, The Pennsylvania State University*, 2014.

[10] H. C. S. Sang Bum Kim and H. C. Rim, "Poisson naive bayes for text classification with feature weighting," in *Proc. ACL Home Association for Computational Linguistics*.