A SUPERVISED ROAD TRAFFIC ACCIDENTS DETECTION WITH LOCATION EXTRACTION FROM MICROBLOGS

Edwin Kabi Njenga

June 2016

A SUPERVISED ROAD TRAFFIC ACCIDENTS DETECTION WITH LOCATION EXTRACTION FROM MICROBLOGS

Candidate:	Edwin kabi Njenga
School of Study:	School of Software
Student's Number:	2820140078
Supervisor:	Prof. Jianyun Shang
Chair of Defense Committee:	Prof. Chi Liu
Degree Applied:	MSc. Software Engineering
Major:	Software Engineering
Date of Defense:	June 5th 2016

Declaration of Originality

I hereby declare that this thesis is the result of an independent research that I have made under the supervision of my supervisor. To the best of my knowledge and belief, it does not contain any other published or unpublished research work of others, or any material which has been accepted for the award of another degree or diploma at Beijing Institute of Technology or other educational institutions, except where due acknowledgment has been made in the text. Whoever has contributed to this study is explicitly identified and appreciated in the Acknowledgements of the thesis.

Signature of Author:

Date:

Authorization Statement

Acknowledgements

First I want to thank God for good health and countless blessings that he has provided to me before and now.

My deepest gratitude and appreciation goes to my supervisor Prof. Jianyun Shang and Prof. Huaping Zhang for their guidance, patience, support and advice throughout my study at Beijing Institute of Technology (BIT) which led to the accomplishment of this thesis. I am also very grateful to my lecturers and the staff at the school of software and also the international student center for their support.

To my wife Jane, thank you for undying prayers, love, moral support, encouragement and being there for our kids (Ivy and Meles). Also without forgetting my sister and parents relentless support and encouragement. Last but not least I want to thank all my friends and colleagues both in Kenya and Beijing for their constant encouragement during this period.

Abstract

Microblogging is a very popular way of sharing information about what is happening near real-time. Microblogs have been used to report about individuals daily life, advertisement or real world events as they are happening. This data is unstructured, voluminous with constantly changing topics that needs to be analyzed to make sense out of, in a most effective and efficient way which can be used during emergency cases. Large scale events like disasters or planned events like a head of state visiting another country, will normally generate a lot of tweets making it much simpler to detect. Small scale events like traffic updates, fires, road traffic accidents do not receive a lot of attention from microblog users hence more difficult to detect.

Firstly reported on microblogs by road users, road traffic accidents are common in our roads which cause death and traffic congestion if not responded to in a timely manner. In this paper, text mining through machine learning and NLP techniques was proposed to detect road traffic accidents which includes, pedestrian, motorcycle, vehicles and other accident incidents on the roads or highways.

In text classification for accident related tweets, SVM was adopted yielding a precision of 90.2% and a recall of 89.1 %. Location extraction from content proposed method, used syntactic analysis through Noun phrases and N-gram based matching with integration to local and external location databases giving a precision of 71.8% and a recall of 83.9%. An application called Traffic Accident Detection System (TADS) was developed based on the proposed methods where users can locate accidents in real-time by querying the system, ultimately visualizing the location of the event on the map for Nairobi, Kenya.

Key Terms: Microblogs, Event Detection, SVM, Text mining, NLP.

摘要

微博是一种非常流行的分享近期发生事情的信息的方式。微博内容是用来展示个人生活，广告或正在发生的真实事件。这个数据是非结构化的，有着大量的不断变化的主题，因此在发生突发的紧急事件时需要用有效且有效率的的方法从这些数据中分析出有意义的部分。大规模的事件，如灾难，或计划好的事件，如国家元首访问的其他国家，通常会引起大量群众发微博，对这样的事件检测起来会更加容易。而诸如交通线路更新，火灾，交通事故等小规模的事件不太被微博用户关注，因此更难以检测。

道路交通事故在生活中时有发生，通常遇到的旁观者会发布在微博上，如果这些事故未得到及时响应，会造成交通拥堵，甚至人员伤亡。在本文中，提出了一种基于机器学习和自然语言处理技术的文本挖掘方法，用于从微博中及时检测道路交通事故，包括行人事故、摩托车事故、机动车辆事故和公路/**高速公路上的任何其他的事故。**

在对与交通事故有关的推特进行文本分类时，采用的SVM **方法具有平均**90%**的精度和**89.1%**的召回率，而位置提取方法使用句法分析通**过名词短语和基于N-gram**的匹配地**图从微博内容中获取事故发生的地点，这一方法具有71.8%**的精度和**83.9%**的召回率。基于上述方法，我**们开发了一个名为交通事故检测系统（TADS）**的手机**应用程序，用户可以通过查询即可定位事件发生地，系统最终会在肯尼亚内罗毕地图上显示该事件的发生位置.

关键词：**微博，事件**检测，SVM，**文本挖掘，自然**语言处理.

Table of Content

List of Abbreviations

KNN - K Nearest Neighbor

HMM – Hidden Markov Model

CRF – Conditional Random Fields

SVM – Support Vector Machine

SM – Social Media

KDD – Knowledge Discovery in Databases

IE – Information Extraction

CRISP - Cross Industry Standard Process for Data mining

NER – Named Entity Recognition

POS – Part of Speech Tagging

TDT – Topic Detection and Tracking

LDA – Latent Dirichlet Allocation

ROI – Region of Interest

API – Application Programming Interface

OSN – Online Social Media

TADS – Traffic accident detection system

UGC – User Generate Content

RBF – Radial Basis Function

RTA – Road traffic accident

List of Tables

List of Figures

1. INTRODUCTION

1.1 Introduction

Microblogs allow registered users to post short messages about personal or real world events as they happen in real-time, also follow what is trending online. Social platforms like Twitter, Tumblr, Instagram, Flickr, Sino Webo are some of the popular forms of microblogs. They are easy to use as they are portable in our mobile devices (smartphones and tablets) which make it very convenient to communicate. These platforms have made it easier to spread information faster than the traditional methods.

Due to continuous flow of updates, Twitter has been used to report events whether it's a disaster or social in nature. They can be global and large scale for example earthquakes and typhoons[1], disease[2], or local and small scale events for example traffic updates[3][4],elections[5],Crime[6].But with tweets, comes a lot of challenges than other text from traditional media like newswire, news websites as seen in the consecutive chapters.

Taking advantage of microblogs, situation awareness and incident management on road traffic accidents can be positively impacted. Traffic management goal is to monitor traffic flow for a road network while it compliments incident management whose goal is timely response to an occurrence of an incident for example a car crash and also alerting other interested parties like road users and rescue crew. To make a quick decision, detailed information regarding the incident is required for example the location, severity, persons or vehicles involved.

This work emphasizes on small scale event detection specifically road traffic accidents using microblogs (Twitter). The study tries to answer a complimentary research question *'How can we effectively detect and visualize traffic accidents on public highways and roads from microblogs, given the small amount of tweets generated for such events?'* Human observers are used as social sensors to report such incidences, later this information can be automatically detected visualized and disseminated faster to interested parties like rescue crew, hospitals, road users, police or ordinary citizens for situation awareness and further action.

Generally local events are small scale events like road traffic accidents, characterized by a narrower geographical and temporal coverage and also a small amount of message sharing due to local coverage, hence more problematic to capture these tweets in an ocean of personal updates, advertisement and spam status updates. According to [7]Only 4% of all tweets are informative or rather contains any news, for traffic accidents the percentage is much lower.

Road traffic accidents detection from microblogs has been framed as a text mining problem, where the focus is extraction of interesting information from unstructured data or text. This will include text data acquisition, text categorization by classification to separate tweets that are related to traffic accident incidences from non-accident, information extraction through Named Entity Recognition (NER) to extract location mentions. All achieved by heavily relying on using machine learning and natural language processing (syntactic analysis) techniques. To further visualize the results, locations are geocoded and projected on the map. The proposed methods are used to develop a web application TADS to detect traffic accidents in Nairobi, Kenya.

1.2 Motivation

Microblog is a popular way of how people exchange or communicate with each other in real-time acting as social sensors. Faster information sharing is very vital during emergency events especially after a disaster for example an earthquake, forest fires, car accidents etc. or events that are relevant to our daily lives like concerts, crime or traffic updates. Road traffic accidents are common phenomena on our roads causing death, injuries or traffic congestion. This could be motor vehicle crash, a pedestrian hit by a car, cyclist accidents, an injured passenger etc. Online communities share information faster even before the relevant authorities are notified. Twitter is a kind of microblog that has spatial, temporal and content dimensionality which can be used to accurately detect these events, on space and near real-time. The messages are short and unstructured including other noisy data. Road traffic accidents detection system (TADS) is needed to automatically capture this data, filter, classify, and detect such events. Once detected, dissemination to interested parties like the Police, rescue team and other road users, can be notified as these events are happening near real-time, hence faster situation awareness.

1.3 Problem Statement

Even though studies have been conducted on traffic management from microblogs[3][4], machine learning approaches has sparsely been used while location extraction from content is an active research area. Road traffic accidents on the roads has caused so many deaths or permanent injuries which can be minimized through better incidence management. While information from microblogs is rich, it is scattered since it is generated from different users. For example in Twitter, the users generally post what they have seen on the scene and also tag their friends through mention. A more robust approach is required to collect and aggregate these sources and filter out noise. In the proposed system a supervised classification algorithm is adopted to automatically categorize accident related tweets while a separate module extracts locations to be visualized on a map.

1.4 Research Questions

Event detection from small scale events in microblog is not trivial. Road traffic accidents is a small scale event that is reported in small geographical area and for a shorter period of time. So it is buried in other microblog noise or non-eventful data streams.

Challenges for local and small scale events detection from microblogs like Twitter include the following:

1) Microblogs are short limited to 140 characters.
2) Large volumes of text streams with a lot of meaningless “babbles” or noise.

3) Small events do not attract a lot of tweets.
4) Detecting locations where these events are happening is challenging and non-trivial as most tweets are not geocoded.

After considering these challenges, road traffic accidents are small scale. To detect such events from microblogs this research will answer the following question.

'How do we effectively detect and visualize road traffic accidents events from microblogs?'

To simplify the process of answering the main research question, it has been broken into the following sub problems.

1) What is event detection and the state of the art approaches to event detections from microblogs?
2) What techniques do we use to effectively crawl road traffic accidents events from Twitter?
3) How can we automatically classify road traffic accidents related tweets from non-accident ones?
4) How can location mentions be extracted from tweet content?
5) How can we develop a prototype web application system for traffic accident detection with a map visualization?

1.5 Significance of Study

The study demonstrates the importance of social media with its rich information in the area of situation awareness and incidence management when road traffic accidents happen, ensuring smooth flow of traffic on the roads and incidence management by reducing the response time. Tweets can be re-posted on other media services to increase situation awareness on public roads or highways. Through other augmented services the system can be integrated to communicate with the rescue crew or alert center, police and road users via SMS or SOA API.

The main contributions for this study are:

1) Using Machine learning approach to categorize accident related tweets
2) Proposed a framework for location extraction from microblog content and:
3) Developing a web application prototype system for traffic accident detection.

1.6 Thesis Outline

In chapter 2, we discuss the characteristics of Twitter, provide a brief key concepts on text mining, social media and event detection, we also examine related work on event detection from microblogs and characterize road traffic events.

In chapter 3, a general framework for traffic accident detection is given, it also discusses different techniques for data acquisition using Twitter API, tweet preprocessing methods, automatic classification mechanism by testing different supervised algorithms and location extraction through syntactic analysis also known as parsing.

In chapter 4, specification, development and implementation of the system is presented by describing requirement analysis (functional and non-functional requirements, use cases analysis, user interface, system environment; software and hardware requirements and constraints), system design (activity diagrams, sequence diagrams and database design) and implementation of a web application prototype for road traffic accident detection (TADS).

Finally, the conclusion and future work is presented to summarize the study by reflecting on the objectives of the research and recommendation for improvements.

2. BACKGROUND AND RELATED WORK

2.1 Introduction

In this section the topic of text-mining and the social media is introduced, related work on event detection for long text and microblogs and lastly, traffic accidents is briefly discussed.

2.2 An Overview of Text Mining

In World Wide Web, 90% of the content is unstructured and stored as text. To infer knowledge from this text, one must represent it in a way that computers would understand. Text mining as a definition is the discovery of knowledge from unstructured text first mentioned by [8]. It incorporates techniques from information retrieval [9], information extraction [10], Natural Language Processing NLP[11] and connects all this by applying KDD algorithms; Data mining, machine learning and statistics[12]. In research, text mining tackles the problem of representation, classification, clustering, and information extraction. This process includes; Text preprocessing, Text transformation or encoding, feature selection, pattern discovery, evaluation then finally deployment as shown in figure 2-1.

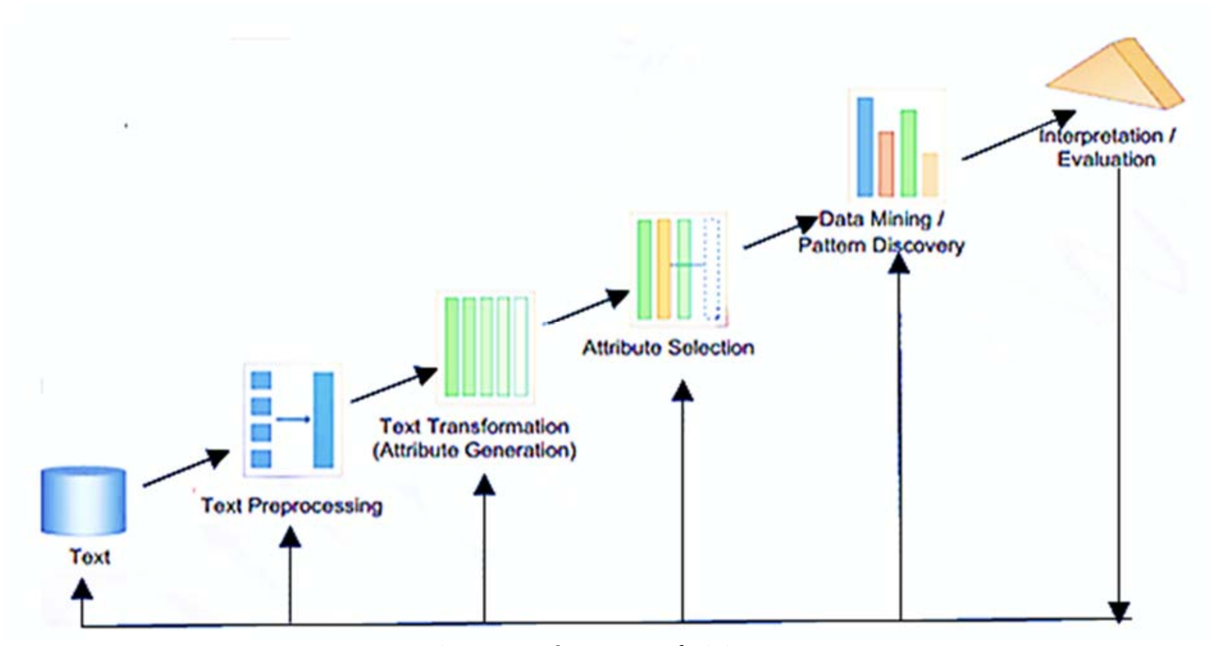

Text
Text Preprocessing
Text Transformation (Attribute Generation)
Attribute Selection
Data Mining / Pattern Discovery
Interpretation / Evaluation

Figure 2-1: The process of mining text

Text mining generally follows guidelines from Cross Industry Standard Process(CRISP) for

Data Mining and Knowledge discovery process in databases (KDD) namely; i) Business understanding ii)data understanding iii) data preparation iv) Modelling 5) Evaluation and finally Deployment demonstrated in figure 2-2 . KDD is a process of finding interesting hidden meaning and usefulness of data in structured data by applying data mining techniques[13].

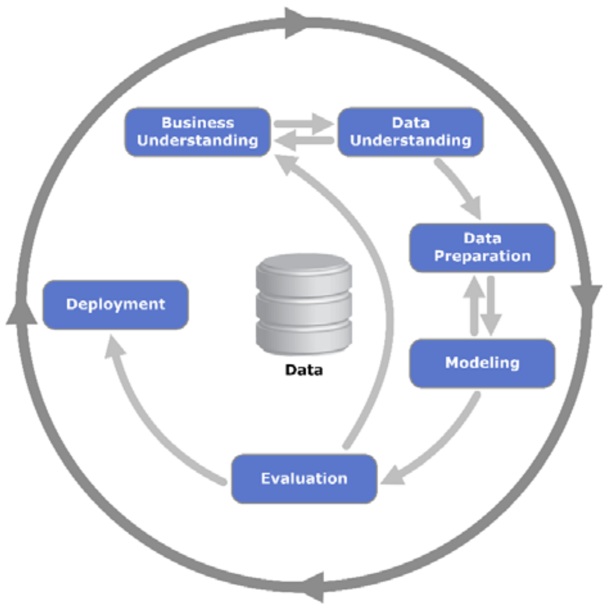


Figure 2-2: Knowledge discovery process Source CRISP

2.2.1 Text Mining Processes

Text preprocessing

This is normally the first step in text mining which involves transforming text into a more structured format for further processing to be done. Preprocessing of text by tokenization, filtering, lemmatization and stemming[12]. Furthermore linguistics processing[14] can be applied depending on the task at hand. This includes part of speech tagging (POS), Text chunking, word sense disambiguation (WSD) and parsing.

Text Transformation

For any machine learning algorithm to understand text, it must be encoded into numerical values. The most applied technique is vector space model[15], which generates a dictionary of a document collection also known as the "the bag of words" approach where a text document is represented by the set of words in it. These words are set as vectors where numerical importance, weight or value is stored for every word in the bag.

Feature Selection

In very large text collection more effort is required to reduce dimensionality of text, and selecting the most representative features while discarding the irrelevant ones in the corpus. This process allows a smaller dataset size hence less computation to traverse the search space. These techniques include Entropy - a measure of importance of a word in a given collection by computing keyword or index term selection[16]. To train a classifier with most relevant features or most informative text, feature selection is done using ranking score known as Information Gain (IG). For example, given a document with 100,000 words, we would want to select the most informative terms that represent this document and discard less representative when a classification task is presented, consequently reducing complexity of the algorithm.

2.2.2 Data mining (DM) Techniques for Text

DM helps to structure a collection by either keywords through classification, clustering, topic detection, summarization of a document, or topic extraction.

Supervised Learning (Classification)

This aims at assigning predefined classes to a document. An example would be to automatically label each incoming news story with a topic like "accident ", or "non-accident". It begins with a *training set* $D = (d_1, . . . , d_n)$ of documents that are already labelled with a class $L \in \mathrm{L}$ (e.g. accident, non-accident). The task is then to determine a classification model which is able to assign the correct class to a new document *d* of the domain. These algorithms include; Naïve Bayes[17], Support Vector Machine (SVM) [18] and K- Nearest Neighbor (K-NN).

Unsupervised Learning (Clustering)

Clustering method can be used in order to find groups of documents with similar content. Each cluster consists of a number of documents *d*. Objects — in our case documents — of a cluster should be similar and dissimilar to documents of other clusters. The quality of cluster is more accurate and better if the contents of the documents within one cluster are more similar and more dissimilar between clusters. Clustering algorithms compute the clusters based on the attributes of the data and measures of (dis)similarity. These clustering algorithms include hierarchical clustering approaches, *k*-means, bi-section-*k*-means, self-organizing maps and the EM-algorithm.

2.2.3 Information Extraction (IE)

The main objective or purpose of IE is the ability of an algorithm to sift through text in a document and identify specific information from the text, which is ultimately stored in a database for further use[10]. Extracting parts of the text and assigning specific attributes to it is the main task.

An example is to extract accident information from a tweet. *'A pedestrian been knocked down by a car around Cool Shades hotel from Ngong towards Karen... Still lying on the ground'*. In this case we have to identify this information. Road segment (Ngong road, Karen road), location referencing accident place (cool shades hotel).IE processing steps include tokenization, segmentation, POS and identification of named entities; location, people names, organization and time. Normally the investigator knows what kind of information they are looking for hence most accurate IE systems are handcrafted rules.

Moreover, there have been some advances in using classification methods to identify these entities. Words are labeled using the IOB (**I**nside, **O**utside, **B**egin) framework[19] a word based tagging. Where target entity word(s) starts, gets tagged with **'B'** continuation words get tagged with **'I'** and the words outside the entity gets tagged with **'O',** repeatedly done for every entity of interest**.** *(O)A (B)pedestrian (O)been knocked down by (O)a (B)car (O)around (B)Cool (I)Shades (I)hotel (O)from*

(B)Ngong (O)towards (B)Karen... Still lying on the ground **.**This way a sequential classification problem label for each entity with surrounding words as input feature vectors hence any classification method can be applied. [20] applies word-base tagging and SVM to extract molecular biology names. Best performers are sequential classifiers, Hidden Markov Model(HMM)[21] and Conditional Random Field(CRF)[22], which considers context on predicted labels of surrounding words. In [23], they use CRF on CoNLL corpus to extract people's names, locations, organization while [24]for biological entity recognition.

2.3 The Social Media

Social Media is the use of electronic and Internet tools for the purpose of sharing and discussing information and experiences with other people in the same or different social circles[17]. These include but not limited to; publishing in Wikipedia, sharing in YouTube or Flickr, discussing in online forums, social networking in Facebook and microblogging in Twitter or Sino webo. These platforms come with a lot of challenges; data is voluminous (big data problem), a lot of noise and is unstructured, obtaining samples is not always straight forward and evaluation dilemmas especially when there is no ground truth.

2.3.1 Microblogs.

Microblogging[25] is a combination of blogging (personal publishing websites) and instant messaging, which allows registered users to post short messages and follow what's trending online or world events as they happen in real-time. Social platforms like Twitter, Tumblr, Instagram, Flickr, Sino webo are some of the popular forms of microblogs. They are easy to use as they are portable in our mobile devices (smartphones and tablets) which make it very convenient to communicate. Contents from this microblogs may include text, images, videos and URLs. These platforms have made it easier to spread information faster than the traditional methods. Microblogs are maintained by individuals and content can be updated multiply in a day shared publicly or within a social network[26].

Motivation behind micro blogging is diverse[26] daily chatter or status update, conversation by using the @sign, sharing information by sending tiny URLs, and also reporting news or incidences as they happen during crisis[1] . Users visit these sites to either seek information, act as information source or to connect with friends.

2.3.2 Twitter

A kind of microblog, Twitter has been used to report events that are global or large scale in nature for example earthquakes and typhoons, disease or small scale events for example elections, crime and traffic owing to its characteristics of spatial-temporal, continuous flow of updates (near real-time) and popularity.

Overview of Twitter service.

Twitter is a famous microblogging and social networking service started in early 2006[27]. It has free messaging service for sending and receiving short messages in real-time. These messages are limited to140 characters also known as tweets. It is a social networking site as people create profiles and connect with other persons also having profiles. By posting a message, you inform the world and your followers of what is happening around you. A Twitter user is either followed or is a follower of another user which forms a structure of social network. Registered users can read and post tweets while unregistered users can only view tweets.

Globally there are 650 million registered users with 350 million active users. There are 135,000 new registrations daily. There is an average of 530 million tweets per day and 368,000 tweets per minute. Tweeting, tweeted, tweet is the act of sending a message through Twitter. @username is a unique

Identity of a Twitter account denoted by @sign example @johndoe. Tweet content also include screen name, text; hyperlinks, message and hashtags which sums up to 140 characters. It also contains metadata; timestamp, geocode from GPS enabled devices, registration information; account name, profile picture, location and website link. A hashtag is any word with # symbol before it (#TianjinFire). This word is converted into a link which makes it easier to find or follow a story as related tweets are clustered under that link. Tweets action can be a reply to a tweet or a retweet that resends a tweet from someone else to your followers.

Challenges

Many real world events locally and worldwide have been reported through Twitter, which has demonstrated its efficiency and effectiveness as microphone for masses in daily life stories, disasters, social movements, for example during earthquakes in Japan and Haiti, the Arab spring, disappearance of Malaysian Airline etc. In comparison to non-user generated content, Twitter has some challenges; 1) They are Short - 140 characters, 2) Streams a lot of unimportant information or noise and 3) Heavy use of informal language and grammatical errors. Only a fraction of tweets have informative information. According to [7] 40% of a tweet is pointless babble, 38% is conversational, 9% is pass along value, 6% is self-promotion, 4% is spam while news worthy information is only 4%. This makes event detection in Twitter streams a hilly task when separating noisy (spam, promotion, pointless babble) information from interesting real world events.

2.4 Related Work

2.4.1 Introduction to Event Detection

According to[28]an event is an occurrence causing change in the volume of text data that discusses the associated topic at a specific time. It is characterized by a topic and time and associated with entities such as people and location.

Topic Detection and Tracking (TDT):

TDT program sponsored by Defense Advanced Research Projects Agency(DARPA) which was solely concerned with organization of news documents streams according to topic and detecting events whether retrospectively or new events in an online manner[28][29] [30].The motivation was to come up with tools that would monitor traditional newswire media and broadcast news while also keeping abreast with old or new developments as reported. TDT had three main tasks; news segmentation, detection and tracking. The main aim was to arrange news into understandable stories, detect new events and keeping track of previously detected events.

Event detection consist of the following major phases; data acquisition, preprocessing, data representation and data classification or clustering[29]. A number of methods have been used to identify events from tweets. It is assumed that a topic is brought about by an occurrence of an event, so detecting a topic is more or less discovering an event[31].

2.4.2 Characterization of Event Detection as Observed on Microblogs

Document-Pivot vs. Feature pivot clustering:

In document-pivot approach they cluster incoming streams as per the distribution of words[32]. This words are merged with the most suitable cluster according to their similarity computed by known similarity measures for example cosine, correlation coefficient or distance based measuring method like Jaccard. In feature pivoted approach[33] [27], events are detected from burstiness of words in a topically discovered cluster. A bursty activity is a sudden change of frequency usage of a term in a cluster in a certain period of time. These terms could be single words, bigrams, n-grams. For Twitter streams, a combination of other Twitter-centric features is important, for example social interactions (retweets, mention and replies).

Specified vs. Unspecified events detection:

If events are known a priori, event detection can be done by specifying the characteristics or the attributes (venue, keywords, location) of the event which are known. For example musical events [34] or local festival[35]. To detect earthquakes[1]they used search terms "earthquake" and "shaking" to return most relevant tweets about the event. In unspecified event detection the events are not known first hand thus, clustering of tweets is important to identify topics which discuss these events[31].

Further, events can be identified by supervised means by training a classifier to identify events from non-events.[1] used SVM while [36] implemented Naïve Bayes. Unsupervised clustering approaches have also been used[27] [30], for example hierarchical clustering, partitional clustering and threshold based online(incremental) clustering or topically relatedness of words[37] [38] example the LDA. Others combined approaches by clustering then applied classification to identity real world events clusters[27].

Large scale vs. small scale event detection:

When an event is big or popular, it attracts a large number of tweets. This event is considered to be global or large scale event. It's characterized by a large number of status updates about that particular event within that location where it is happening and away from that location. For example a popular artist or president visiting a country or a disaster like an earthquake. A local or small scale event is an event that does not attract a lot of updates and normally sent by an individual on the location. It could be a local concert, car accidents, traffic updates, crime, house fires etc.

2.4.3 Large Scale Event Detection from Microblogs

As pointed earlier Twitter has a lot of noisy and meaningless information. To distinguish real world events from non-events[27], they examine different features and characteristics of Twitter and proposes an online clustering algorithm to group tweets according to similarity, then trained a SVM classifier (with features selected from clusters) to automatically identify clusters that represented real world (RW) events. For every incoming tweet a similarity is computed with all existing clusters, if this similarity meets a set threshold it is merged with the highest probability cluster, if more than one is available the cluster is chosen randomly. If the threshold is not met a new cluster is created. The classifier requires unique features which include *temporal features*, calculated by a term frequency in a cluster at time *t* to capture the ‘burstiness’ of a term which will indicate a trending topic or an event if there is a sudden increase of usage, *social features* to capture interactions(retweets, replies and mentions) of users in a cluster, *topical features* where the assumption is event clusters tend to talk about specific topics hence sharing more terms, *Twitter-centric features* to capture those clusters that might exhibit a trend but would not necessary mean it’s a real world event by looking at the number of hashtags. They use a dataset of 2.6 million tweets trained a RW-event(SVM) classifier by using the identified features with 374 clusters as training set and 300 as a test set handpicked by human annotators. They obtain an accuracy of 83%. Here the researchers did not consider geo-locating where these events were happening and also considered RW-events that are global in nature.

To detect earthquake and typhoons(surprisingly faster than geological seismic system) in real-time in Japan[1], they collect tweets by using keywords “earthquake” and “shaking” to return most relevant tweets by querying Twitter search API. SVM was used to classify tweets as earthquake or non-earthquake event and trained the classifier using features like keywords in tweets, total number of words in tweet, word before and after the query word. Location of the earthquake is assumed to be user registered location by the time of sending the tweet. By applying kalman and particle filtering technique (widely used in ubiquitous systems for location estimation) they were able to estimate and triangulate the location of the earthquake. The trajectory of a typhoon was done using a spatial-temporal characteristic of tweets to calculate the probability occurrence of an event by applying probability density function. They develop an application to alert registered users on the trajectory path of an earthquake. Also clearly seen in this study, they depend on volumes of tweets and the assumption that registered user location is where the event is happening which is not always correct. In [36], they proposed a news processing system based on Twitter, called TwitterStand, to capture tweets that correspond to late breaking news. They also implement a Naive Bayes classifier to categorize tweets containing news. An online clustering algorithm based on tf-idf and cosine similarity was deployed to form clusters of topics on the news. Furthermore, to reduce clustering errors, hashtags are used. Clusters are also associated with time information for management and for determining the clusters of interest. They also removed noise by systematically including seeders well known to propagate news, determined the topical geographical location mentions associated with the tweets through their content by extracting interesting phrases and tweets metadata on user location which is subsequently matched using a local gazetteer database.

Breaking news detection [39] collects, groups, ranks and tracks breaking news from Twitter. They first collect news tweets by sampling keywords for example, "#breakingnews" and "#breaking news". This content was then indexed using Apache Lucene for easier retrieval. To group messages that are similar, a modified TFIDF with a boost score was implemented. The groups eventually form breaking news stories. Proper nouns, hashtags and usernames are the artifacts that are boosted by a factor. Proper nouns (extracted using off-the-shelf Stanford tagger) was identified as the most useful in similarity comparison between tweets which increased system accuracy. Reliability, popularity and freshness is used to rank every group. Reliability is determined by all users posting messages in the group while popularity was measured by number of retweets. Freshness of each group was done by scoring higher groups that are receiving newer messages. An application Hotstream was developed to demonstrate potentiality of their proposed method.

2.4.4 Small Scale Events Detection from Microblogs

Most approaches in microblog event detection focuses mostly on large scale events like earthquakes, typhoons, wildfires, elections with a wider geographical, temporal and a large amount of messages or tweets is shared whereas small scale events for example road traffic accidents, local crime, factory fires etc., has a narrower geographical and temporal coverage and also a small amount of message sharing.

A novel approach to identify Twitter messages for concert events using a factor graph model, which simultaneously analyzes individual messages, clusters them according to event type, and induces a canonical value for each event property framework was proposed by[34]. The motivation is to infer a comprehensive list of musical events from Twitter (based on artist–venue pairs) to complete an existing list (e.g., city event calendar table) by discovering new musical events mentioned by Twitter users that are difficult to find in other media sources. At the message level, this approach relies on a conditional random field (CRF) to extract the artist name and location of the event. The input features to CRF model include word shape; a set of regular expressions for common emoticons, time references, and venue types; a bag of words for artist names and city venue names extracted from external source like Wikipedia. Clustering is guided by term popularity, which is an alignment score among the message term labels (artist, venue, none) and some candidate value (e.g., specific artist or venue name). To capture the large text variation in Twitter messages, this score is based on a weighted combination of term similarity measures, including complete string matching, and adjacency and equality indicators scaled by the inverse document frequency. In addition, a uniqueness factor (favoring single messages) is employed during clustering to uncover rare event messages that are dominated by the popular ones and to discourage various messages from the same events to cluster into multiple events. On the other hand, a consistent indicator is employed to discourage messages from multiple events to form a single cluster. The factor graph model is then employed to capture the interaction between all components and provide the final decision. The output of the model consists of a musical event-based clustering of messages, where each cluster is represented by an artist–venue pairs.

In [35] the authors detect geosocial local events for example local festivals, through microblogs by modelling and monitoring crowd activities in Japan, through Twitter. Geographical regularities are extracted from geotagged tweets showing usual behavior patterns of a crowd. They collected geotagged Twitter data over a long period of time in a region. This region was further subdivided into region of interest (ROI) using K-Means algorithm by creating clusters from coordinates (latitudes and longitudes). Within each ROI, geographical regularities of a crowd was estimated using three main features namely; number of users, tweets and users moving in and out of the ROI. They split time intervals of 6 hours each then applied statistical measures on the historical data to estimate a crowd behavior in ROI. Finally, unusual events in geographical area that is being monitored is detected by comparing statistics from new tweets with those of the estimated behavior. An increased user activity (moving inside or coming to an ROI) combined with an increased number of tweets is a strong indicator of local festivals.

SVM has also been used to identify traffic related tweets by [40]. They only consider tweets from official account broadcasting traffic news which are written in a specified format and proper grammar hence easily identifiable. For the other tweets sent from other users they train a classifier to identify traffic and non-traffic related tweets the positive tweets are used again to retrain the classifier for further enrichment. This helps to give higher precision.

2.4.5 Information Extraction from Microblogs

Information extraction is a key characteristic for event detection especially if they are to be geo-localized using their content. Named Entity Recognition(NER) plays an important role [12] [41]. To increase situational awareness [42], applies NER algorithm to develop ANNIE named entity extractor based on GATE[43].[44] Also used NER for tracking and filtering accidental information from tweets. Twitcident[44]is another system that uses NER to extract information from tweets.

Location identification in tweets can be extracted from different indicators from the tweet itself. User location can be extracted from user profiles, registered locations, or from the mention of locations from the tweets[45]. From August, 2009 Twitter was able to offer geotagged information by appending the location GPS coordinates from where it was sent. Geotag has latitude and longitude for identification on space. This information is not always available as users prefer to turn this feature off – only 1% of Twitter users globally turn this feature on, it might also be inaccurate as some users report the incidences away from the location of the event. So it's important to use a combination of techniques to identify locations events.

In some research they only depended on geotagged and registered locations on tweets for estimating the location. [42] used geotaggs only, [44] considered user registered location while[1] applied more techniques to triangulate the location for example used Kalman and particle filters by considering both locations geotagged and user registered location.

Location extraction from content is also another way of identifying where events are happening, [46]presents a study to identify location references from tweets by using Ling Pipe POS tagger for part of speech tagging. "Twitter tagger" is used for disambiguation of noun phrases they are compared to the USGS (US Geological Survey) database to get latitude and longitude of the location. Ambiguities rise from two perspectives; people's names can be locations and different cities can have similar names. They deploy two filters to filter out any ambiguities. Distance is measured between the user location and the tweet content location then the weights are applied on the log-linear model to calculate the most possible location.

In another research to extract location from content [3] uses syntactic analysis on the tweets to extract location names then categorize them using a custom dictionary and applying custom tags to extract traffic information. His word dictionary contains traffic information in four categories place, verb, ban and preposition which are then looked up in a local place dictionary provided by ministry of transportation in Thailand. Google Geocode is used to retrieve longitudes and latitudes of these locations.

To geo-locate Twitter user's city level based on their content of Twitter messages[47], a classification component is done to automatically identify words in tweets with local geo-scope then use a lattice–based neighborhood smoothing model for refining the estimated user location. The algorithm observes the actual distribution of some local words across cities then further processing is done to differentiate local and non-local words. Depending on the local words they estimate the current location of the user on the basis of the city level.

2.5 Detecting Road Traffic accidents from Microblogs

As seen from the discussion, Twitter can also be used to detect small scale events like traffic incidences, local festivals and concerts. In some research, they only depend on geo-tagged tweets while others only depended on tweets from official traffic update accounts or identifying traffic conditions as a general problem, not necessary accidents, also others fail to visualize this information on the maps.

An accident is defined as a traffic accident if it occurs in places like roads, highways, near buildings or crossroads to which the public have access. Examples include but not limited to; accidents which happen on the highway but which involve casualties off the highway, accidents involving the boarding and alighting of buses or coaches, accidents in which passengers already aboard a bus/coach are injured, whether or not another vehicle or pedestrian is involved . Accidents happening to pedal/motor cyclists or horse riders, where they injure themselves or a pedestrian. Road accidents can also be referred as road collisions, road crashes or road incidences.

Actors in road accidents include pedestrians, motor/pedal cyclist, drivers, passengers, vehicles, buses, trucks. Also traffic accidents are indicated by using certain adjectives, verbs, like, injured, accident, killed, lying on the ground, lifeless body etc. Prepositions are also used to identify accidents that happen in a certain location .e.g. near, between, at, in, along, before, on.

Twitter contains very little information about traffic accident incidences estimated at about 0.05% in all tweets[40]. To identify these events one needs a system that automatically identifies traffic accident related tweets, from content, extract the locations and visualize on a map. In a way, this information is reasonably more usable to road user, traffic police and rescue crews, and news agencies.

2.6 Summary

In this chapter, first, text mining concepts are introduced including preprocessing, and categorization through classification and clustering and information extraction especially named entities. Secondly, the social media, particularly microblogs (Twitter) with its key characteristics was introduced and the role text mining plays to extract meaningful information. This information can be detecting real world large scale events like earthquakes or small scale events like concerts, local crime or factory fires. From the literature, different text classification methods like SVM and Naïve Bayes and document clustering techniques like online hierarchical clustering remain to be popular for microblogs event detection.

Furthermore, from microblogs it is clear that events can be located geographically from content through NER by applying advanced information extraction like CRF and HMM models and other NLP syntactic analysis like POS, chunking and parsing which can be visualized on space by using maps. Finally we characterize traffic accidents and its key observations.

3. METHODOLOGY

3.1 Introduction

The main objective of this research is to detect road traffic accidents from microblogs with location extraction. This chapter describes and discusses different frameworks for road traffic accident data acquisition, classification tests and results and lastly, location extraction.

3.2 Conceptual Framework

The general framework gives an outlook and steps adopted to conduct this study and helps achieve the main goal of this research as shown in figure 3-1. The framework acts as a guide that describes and supports reasoning about the structures and behavior of the system.

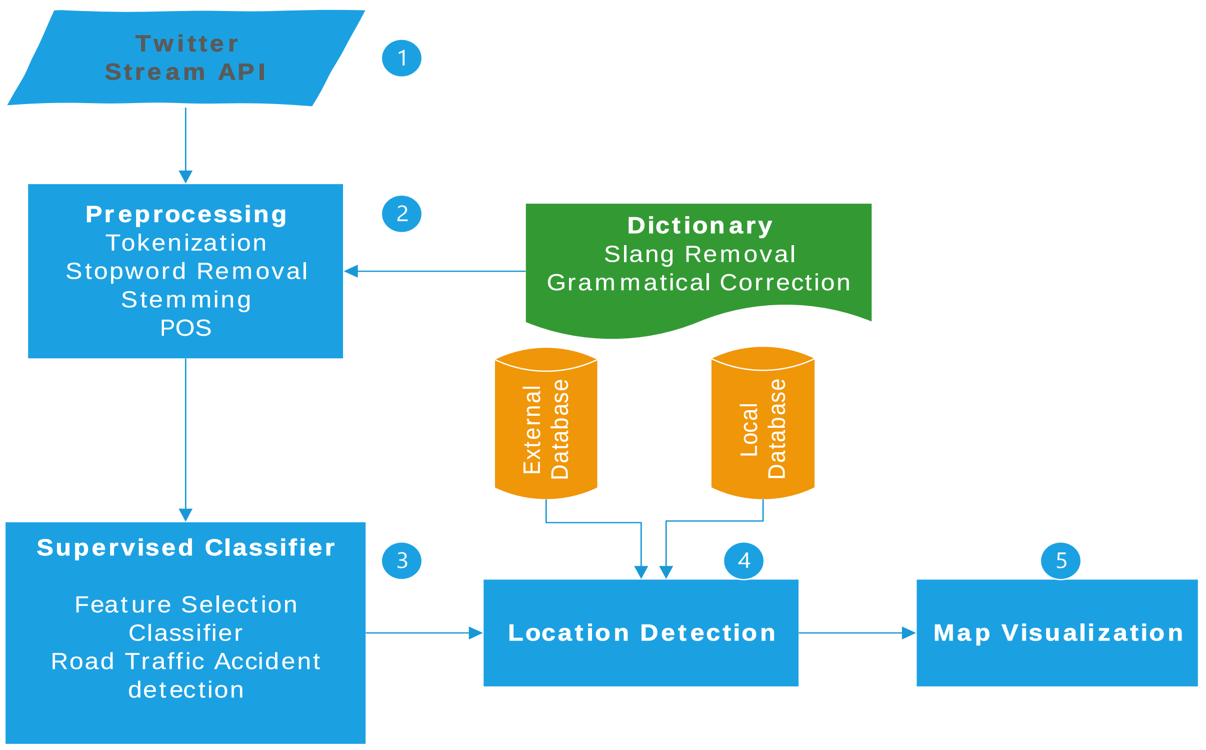


Figure 3-1: Research methodology pipeline

To detect road traffic accidents the process was divided into several stages as detailed in figure 3-1. Below is a brief elaboration of the steps.

1) Data acquisition is the initial step, the study uses Twitter API and applies keyword filtering to crawl most relevant tweets, which then are stored in the database for easier retrieval and manipulation as explained in section 3.3.

2) The next step is preprocessing of text to remove bad words, obvious misspelling mistakes, and abbreviation also known as internet slang. Other Twitter centric features are also removed like URL, punctuations and @mentions. Hashtags are kept since some locations are mentioned using the hashtag. Features are then extracted from the tweets through unigrams, bigrams, and Trigrams. Afterwards for text representation, Vector space model also known as a bag of words technique was used as detailed in section 3.4.1 and 3.4.2.

3) Next step a machine learning classification algorithm is used to categorize tweets into either accident or non-accident as demonstrated in section 3.4.3.

4) In location detection all tweets that are geotagged and are categorized as accident related have no further processing, rather they can be displayed on the map. For those with no geotaggs, they are first preprocessed and parsed to the location extraction module to extract place mentions from content. Here, internal and external location databases are utilized by applying methods explained later in section 3.5.

5) All locations contain latitudes and longitudes which are used to mark the accident spots on a geographical map as detailed under implementation in chapter 4.

3.3 Data Acquisition

Road Traffic tweet stream is any stream of tweets from Twitter API that has data containing accidents mentions on a public road or highway that involves pedestrians, cyclists, motor vehicles and passengers. It is paramount to acquire most the representative and relevant data to suit the case study area for traffic accidents in Kenya making it a very important step. It comprises two steps; identifying the most relevant keywords and choosing most appropriate Twitter API.

3.3.1 Data Collection using Twitter Streaming and Search API

Twitter [48]has several APIs used by researchers over the years to collect and create Twitter corpora to understand how Twitter streams can be used for event detection and other research. This API includes:-

Search API is part of REST API which lets anyone to search through a collection of popular tweets but restricted by imposed limits. It will only search a limited subset of tweets posted in the past 7 days in the main database from query keywords. The query parameters include time, location, language etc. Twitter limits the return results to 100 tweets per request. Twitter limits the request rate on the REST and Search API to 150 requests per hour by default. It previously allowed up to 20,000 search requests per hour from white-listed IPs (about six requests per second), but, unfortunately, Twitter no longer grants white-listing requests since Feb 2011. This limitation makes the REST and Search APIs unsuitable for real-time event detection.

Streaming API gives global tweet access with low latency without the REST API restrictions. It returns 1% of all of the tweets. Filtering can also be done at this level using API request parameter for example you can limit through follow, track, and location, corresponding to user ID, keyword and location, respectively. Twitter applies a User Quality Filter to remove low quality tweets such as spams from the Streaming API. The quality of service of the Streaming API is best-effort and the latency is usually within one second.

In this research only the Streaming API is considered as the study is concerned with the current tweets as they are streamed by users. The study also made the use of Tweepy, a python module that connects to Twitter Streaming API with modification to suit the study. (See appendix B for sample code).

3.3.2 Query Keywords Filtering

To return most relevant tweets on traffic accidents, keywords were used to query the Streaming API. First, 2000 tweets were collected from an official police traffic account to evaluate the relevant words to use as key words. Nouns and verbs were analyzed using frequency count (see figure 3-2). Table 3-1 has sample tweets as returned by the API while table 3-2 shows some sample query keywords.

Table 3-1 : Example raw tweets

T1: accident Eastern bypass Astrol petrol station near GSU training school utawala from tajmall to utawala not moving
T2: car reg no. Kbd 580t has hit a bicycle and the person is injured the drive has run away long kilimani rd near iqra fm
T3: A pedestrian been knocked down around Cool Shades hotel from Ngong towards Karen... Still lying on the ground

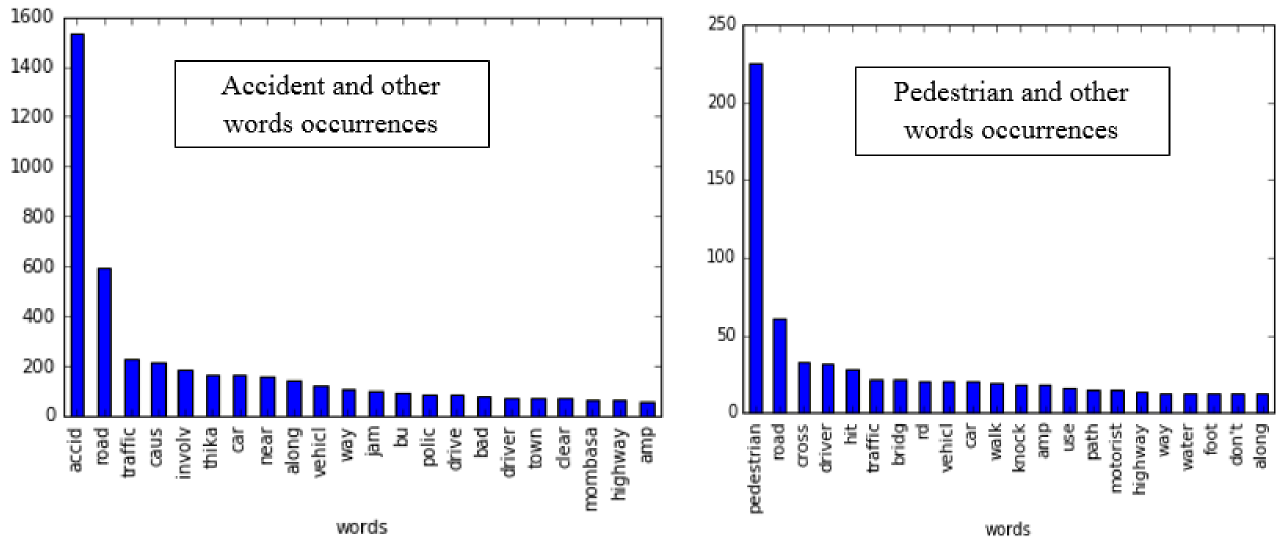


Figure 3-2: Traffic accident frequency of words used for reporting this events on Twitter

Table 3-2: Sample keywords for road traffic accident queries

Q1 "injured or hurt or wounded" AND "knocked or hit" AND "vehicle or bus or truck or lorry or van or suv or saloon or wagon" AND "accident"
Q2 "killed or wounded or injured" AND "knocked or hit" AND "car or lorry or truck bus or motorcycle or motorcyclist" AND "accident"
Q3 "collision or accident or crash" AND "pedestrian or passenger"
Q4. "Pedestrian" AND "hit" or "road" or "vehicle" or "knocked"
Q5. "Accident" AND "road or highway or vehicle or involve or traffic"

3.3.3 Study Area

There is a large number of tweets from across the globe, to test our methods for accident detection, we have to narrow down to a specific study area. We chose to do the study for Nairobi, the capital city of Kenya. This decision was partly arrived at because the researcher is familiar with the city and also it passed several criteria as listed below.

- The study area should contain a high density of state highways and secondary roads.
- The study area should contain a high density of Twitter data.
- Many incidents should occur in the study area on a daily basis.

The above criteria was studied in a short pre-analysis stage as discussed in the following section.

The study area should contain a high density of state highways and secondary roads.

Nairobi city has the most road networks that join the city and other towns with a population of about 4 million people. These roads include highways, major secondary road segments and intersections. It is also surrounded by several other major towns, namely Thika, Limuru, Naivasha, Machakos and Kiambu. (See figure 3-2).

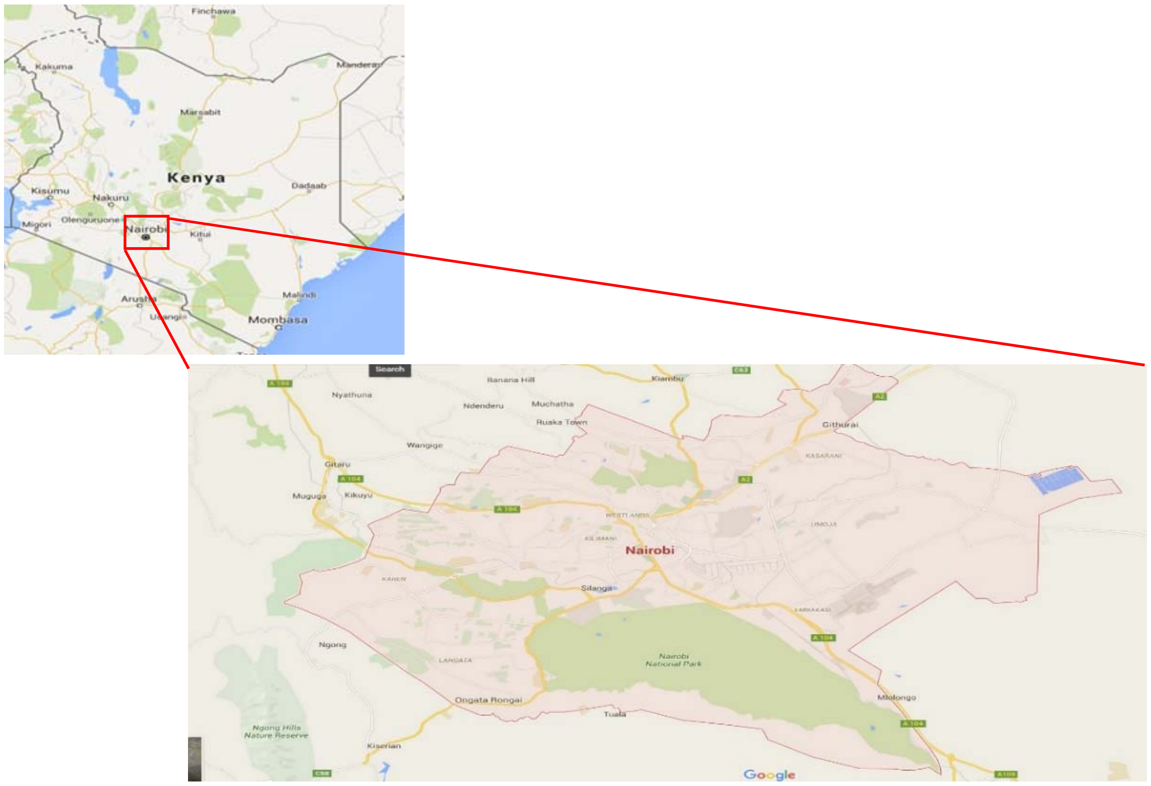


Figure 3-3: City of Nairobi, Kenya

The study area should contain a high density of Twitter data

From the data collected on 17th November 2015 – 31st December 2015, geotagged data was extracted from the tweets then projected onto a map to observe which areas were mostly dense with most tweets. Nairobi city was the most densely represented as seen in figure 3-4.

Many incidents should occur in the study area on a daily basis

Major incidences are reported from the City with close to 2500 tweets reporting about traffic updates or incidences experienced on the roads as shown on figure 3-5 and graph 3-1.

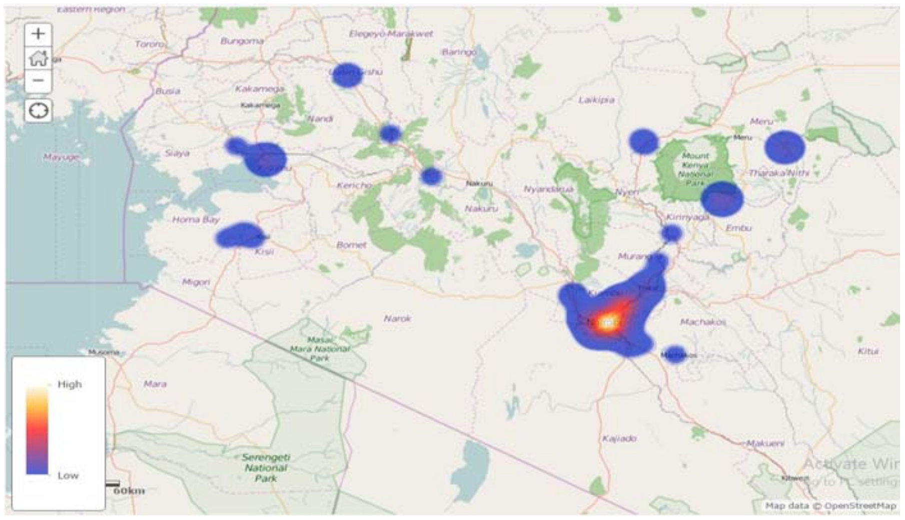
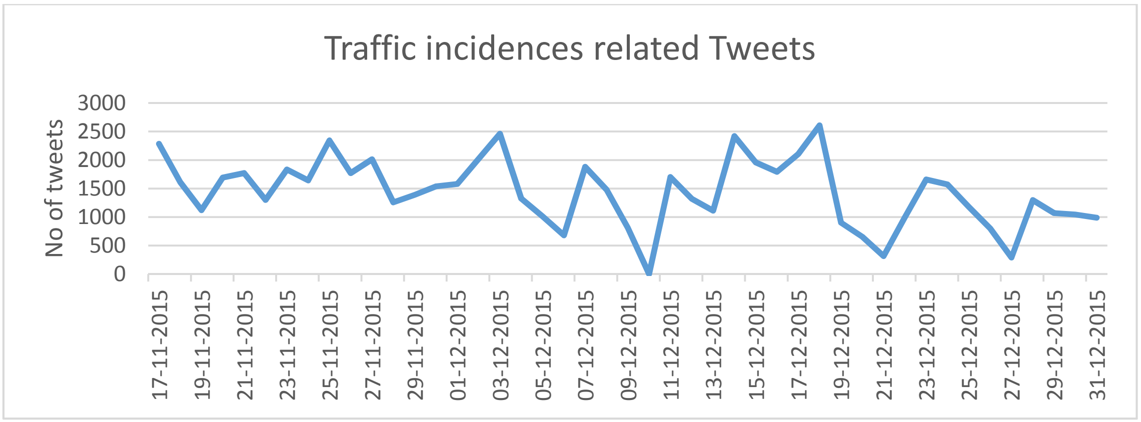


Figure 3-4: Tweets density heat map



Graph 3-1: Road traffic incidences collected for a month and a half

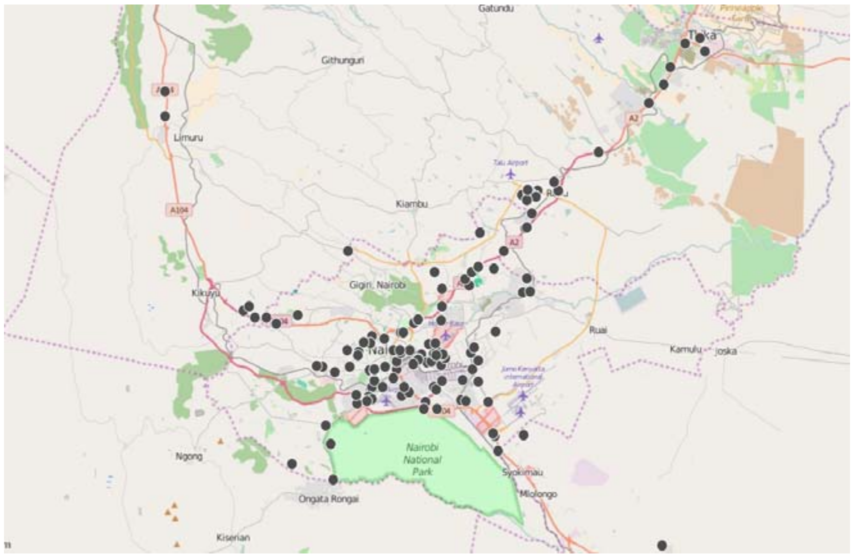


Figure 3-5: Geotagged tweets reporting traffic updates

3.3.4 Dataset

By using the set of keywords to collect traffic accident relevant tweets from streaming API, 1.2 million tweets were collected from 17th Nov. 2015 to 31st Dec. 2015. Since our case study setup was based in Nairobi, it was necessary to first get tweets which were only originating from Kenya and delete the rest. Mostly extracted from the time zone of the user, user profiles, registered location and more so, crawling data from Twitter traffic accounts in Kenya. After filtering noise, 47,000 tweets were originating from the study area. From this dataset, the training set was randomly selected – 1280 tweets. More data was collected from 1st Jan. 2016 to 1st Apr. 2016 to be used as validation for the proposed methods in the implementation of the application system. . Ma3route is one of the accounts that propagate traffic and other related information with more than 360, 000 followers

Initial data storage:

Twitter API returns queries as JSON objects. After applying the mentioned crawling techniques, results were saved to MongoDB as raw tweets. MongoDB is a non-relational database that stores data in JSON like style known as BSON and can scale very well horizontally. Due to the streaming characteristics of Twitter NoSQL type database was more convenient as it is has shown to be faster to write to or query Twitter like data.

Duplicate tweets elimination:

To reduce the number of duplicate tweets, all tweets that contains 'RT', 'via' or retweet is set as 'true' in the metadata, were directly deleted from the database. RT is an informal way that users indicate their post is a retweet or a simple forward while via is also a kind of a retweet but a modified version of the original message. (See appendix A for tweet JSON metadata). Example Duplicate Tweets included in table 3-3.

Table 3-3: Sample duplicate tweets

Original Tweet	Grisly accident at Springvale Sch. near Kithini. From Machakos towards Chumvi Junction. Police at accident scene. @KenyaRedCross @Ma3Route
Duplication Through Via	RT @Ma3Route: Grisly accident at Springvale Sch. near Kithini. From Machakos towards Chumvi Junction. Police ~more ⇢ https://t.co/MDSiGISZwq via @DavidsKE
Duplication through Retweet	RT @Ma3Route: Grisly accident at Springvale Sch. near Kithini. From Machakos towards Chumvi Junction. Police ~more ⇢ https://t.co/MDSiGISZw…

3.4 Road Traffic Accidents vs Non-road Traffic Classification

In categorization of tweets to either accident or non-accident related tweets, the problem was framed as a binary classification problem then applied machine learning classification algorithm namely support vector machine (SVM). To train the classifier, training set has to be annotated as either positive or negative class for the model to be able to predict label for unseen data. The process is broken down in figure 3-6.

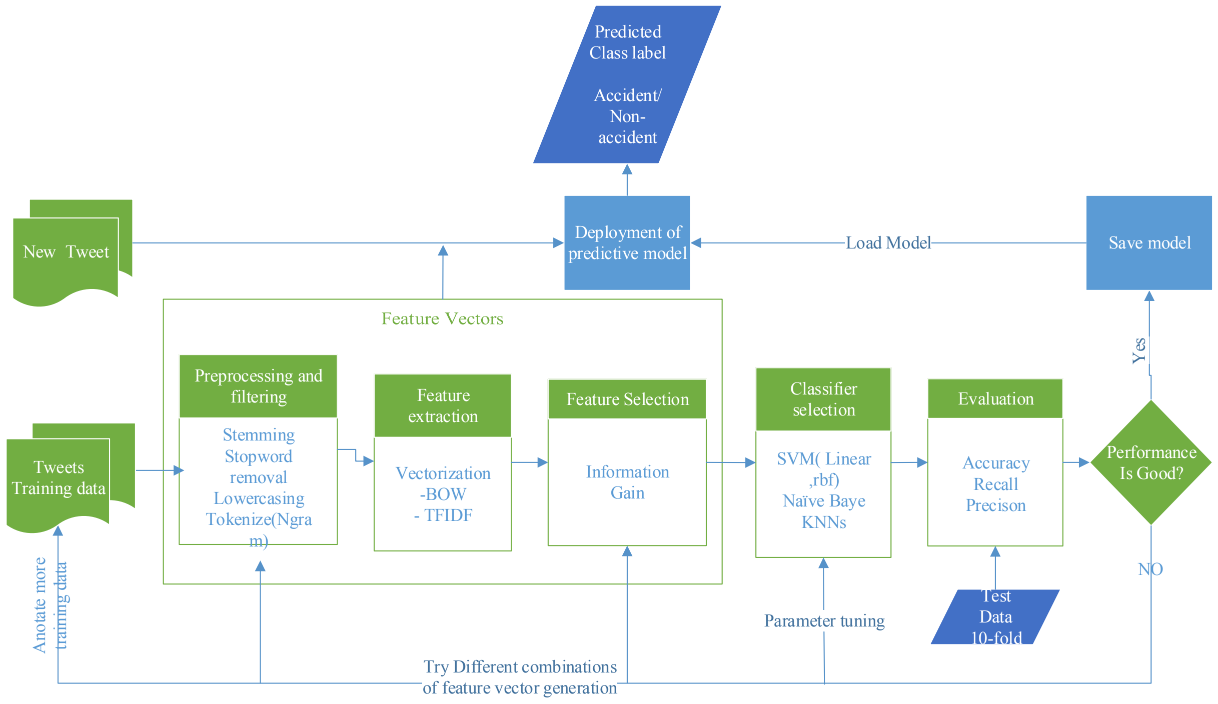


Figure 3-6: Traffic accident classification flow chart

3.4.1 Preprocessing Phase 1: Cleaning Tweets and Filtering

As mentioned earlier, microblogs are short and noisy. To overcome some of these problems, there is need to remove duplications, abbreviations, slang and bad words from the raw tweet.

1) Bad word removal

Bad-word is any word that is uncouth, abusive, offensive, vulgar or cursing in nature. User generated content (UGC) contains bad-words .To remove these words from the corpus, a dictionary of 460 possible words was compiled. These terms do not add any value to the corpus, hence they are deleted.

2) Slang and abbreviation removal

Internet slang is any English word that is abbreviated or shortened or changed in any manner that does not conform to proper grammar, for example 'btwn' meaning between. Most UGC use internet slang and abbreviations that cannot be interpreted by language dictionaries like Wordnet or any POS tagger.

To overcome this challenge, a slang dictionary containing over 5000 words was compiled. Table 3-4 lists some slang or abbreviation with the corrected text.

Table 3-4: A list of abbreviations and slang and the corrected grammar

English word	**Abbreviation or internet slang**
like	Lyk, lyk3, lyke
road	rd
between	Btw, btn, btween, btwn, btwn
Busy	bz bzy bzzy
tomorrow	2ma, 2maro, 2mmrw, 2mo, 2mora, 2moro, 2morow, 2morro, 2morrow, 2moz, 2mozz, 2mro, 2mrw, 2mw, 2mz

Other cleaning that was done include punctuation removal, Twitter centric cleaning like @mentions, URLs removal and correction of repeating words.

3.4.2 Preprocessing Phase 2: Feature Vectors Extraction

After cleaning and preprocessing the tweets, most representative features of a document in a corpus must be extracted to effectively train a classifier. To do so, text must be converted into numerical representation through vectorization usable for machine learning. Vector Space Model(VSM)[15] a method adopted from information retrieval, used to convert a document into a dictionary of words also known as bag of words, then create matrices of integers between 0 and 1 as a value occurrence for each word or term in the bag of words. In this paper each tweet is treated as a document. Below are techniques used to extract features from a tweet: -

Lowercase conversion: Since same word with different casing style will be represented independently in a vector space, all terms in a tweet are converted into lowercase to reduce the sparseness in data.

Word tokenization: Vectors in the bag-of-words can be represented in different forms as tokens of broken streams of words or characters.

Unigrams: Each distinct word is considered a feature.

Bigrams: Where every two consecutive distinct words are considered as a feature.

Trigrams: Where every three consecutive distinct words are considered as a feature.

1-to-3 grams: A combination of unigrams, bigrams and trigrams.

Stemming: A stem is a natural group of words with equal (or very similar) meaning. After the stemming process, every word is represented by its stem for example 'accidents' to 'accident', 'vehicles' to 'vehicl'. This process is done to avoid different representation of the same word on the vector space. A well-known rule based stemming algorithm was originally proposed by [9].

TF-IDF: This is a term weighting scheme developed and adopted from information retrieval domain[15]. Informed by the fact that the more a term is used in a corpus the less influential in helping the classifier in performance as it may be carrying very little meaning in identifying the document. It also applies when the term is rarely used.

$$TFIDF = tf_{t,d} * idf_t \tag{3-1}$$

$$idf_t = \ log\frac{N}{df_t} \tag{3-2}$$

Computed by multiplying TF score with IDF score for the term in a corpus (see equation 3-1). Term Frequency (TF) is the counting of occurrence of a term *t* over total terms in the document *d*. The more the count, the higher the score. On the other hand, Inverse Document Frequency (IDF) is the reverse where it tries to rank the document by scoring the terms inversely to the occurrence in a document. Thus, the frequency of the term in all documents determine the overall score for term. *N* is the total number of documents in the corpus while df is the document frequency of term *t*.

Feature Selection: In very large text collection, more effort is required to reduce dimensionality of text, and selecting the most representative features while discarding the irrelevant features in the corpus. This process allows a smaller dataset size hence less computation to traverse the search space.

These techniques includes Entropy, a measure of importance of a word in a given collection by computing keyword or index term selection[16]. To train a classifier with most relevant features or most informative text, feature selection is done using ranking score known as Information Gain (IG). For example, given a document with 100,000 words, we would want to select the most informative terms that represent this document and discard less representative when a classification task is presented, consequently reducing complexity of the algorithm.

$$IG(t_j) = \sum_{c=1}^{2} p(L_c) log_2 \frac{1}{p(L_c)} - \sum_{m=0}^{1} p(t_j = m) \sum_{c=1}^{2} p(L_c \mid t_j = m)\, log_2 \frac{1}{p(L_c \mid t_j = m)} \quad (3\text{-}3)$$

$p(L_c)$is the fraction of training documents with classes L_1 and L_2, $p(t_j = 1)$and $p(t_j = 0)$is the number of documents with or without term t_j and $p(L_c \mid t_j = m)$ is the conditional probability of classes L_1 and L_2 if term t_j is contained in the document or is missing. It measures how useful t_j is for predicting L_1. We may determine $IG(t_j)$ for all terms and remove those with very low information gain from the corpus.

3.4.3 Training the SVM Model

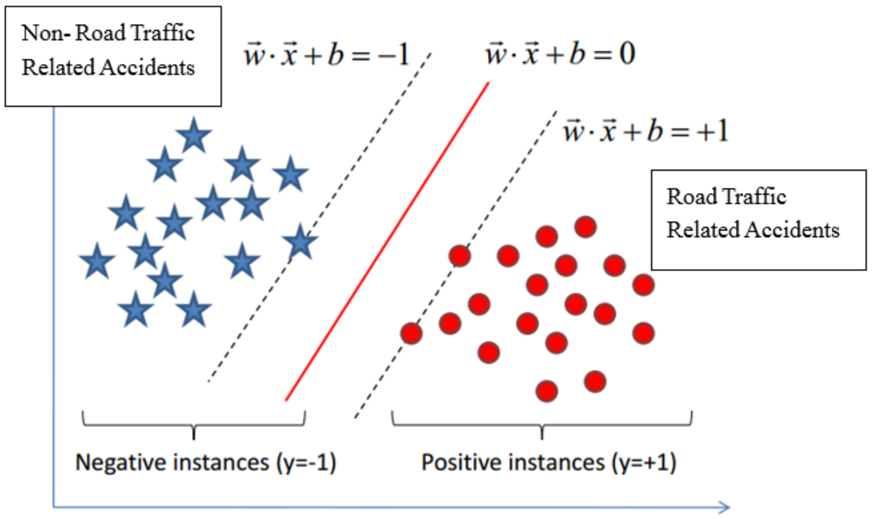

Non- Road Traffic Related Accidents
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=0$
$\vec{w}\cdot\vec{x}+b=+1$
Road Traffic Related Accidents
Negative instances (y=-1)
Positive instances (y=+1)

Figure 3-7: The case of an SVM linear classifier

Consider the problem of separating the set of training vectors belonging to two linearly separable classes,

$$(x_i, y_i),\ x_i \in R^n,\ y_i \in \{-1, +1\} \tag{3-4}$$

where $\mathbf{x}_i$ is a real-valued *n*-dimensional input vector and y_i is a label that determines the class of $\mathbf{x}_i$. A separating hyperplane is determined by an orthogonal vector **w** and a bias *b*, which identifies the points that satisfy (3-5) and predicts y_i in (3-6),

$$w.x + b = 0 \tag{3-5}$$

$$\begin{cases} w.x + b \geq +1, \text{if}\ y_i = +1 \\ w.x + b \leq -1, \text{if}\ y_i = -1 \end{cases} \tag{3-6}$$

by solving (3-7) the optimization problem with user defined constant *C* . The Lagrangian function has to be minimized with respect to **w**, *b*, and δ_i

$$\min_{w,b,\delta} \quad \frac{1}{2} w^T.w + C \sum_{i=1}^{m} \delta_i \tag{3-7}$$

Subject to:

$$y_i(w.x + b) \geq 1 - \delta_i \quad , \quad i = 1,2, \dots\dots m \tag{3-8}$$

$$\delta_i \geq 0 \quad , \quad i = 1, 2, \dots\dots m \tag{3-9}$$

To solve non-linear separable problem, SVM constructs an optimal separating hyperplane in a higher dimensional space where,

$$K\ (x_i, x_j) \equiv \theta(x_i).\ \theta(x_j) \tag{3-10}$$

(3-10) is the kernel performing mapping in non-linear feature space, radial basis function (rbf) kernel is given by (3-11) where parameter γ must be preset,

$$K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2) \tag{3-11}$$

When training an SVM with the *Radial Basis Function* (rbf) kernel, two parameters must be considered: C and gamma (γ). The parameter C, common to all SVM kernels, is a regularization parameter that trades off misclassification of training examples against simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly. Gamma (γ) is an optimization parameter that defines how much influence a single training example has. The larger gamma is, the closer other examples must be to be affected.

According to [18], the reason SVMs work well for text categorization is because they acknowledge the properties of text in the documents; its large feature spaces, the dense concept vectors, with very few irrelevant features and the sparse instance vectors. [12] Also shows the good performance of SVMs by comparison to other machine learning methods used for text categorization. LIBSVM [49] was used for classification with initial experiments done on WEKA[50] and implementation used the Scikit python machine learning toolkit. The Waikato Environment for Knowledge Analysis (WEKA) is a machine learning toolkit which helps in applying ML algorithms to real world dataset developed at the Waikato University. It includes tools for pre-processing classification, clustering, regression and visualization. It can be used as a standalone or by java through an interface.

3.4.4 Experiments and Results

In training the classifier, text must be annotated by humans. In this paper, a total of 1280 tweets were annotated as either accident or non-accident. For each class, a balanced corpus of 640 tweets labelled as belonging to the required classes with a kappa statistics of greater than 69%. Different tests were performed by comparing different attributes in extracting feature vectors and also by choosing different classifiers namely; Support Vector Machine (SVM), Naïve Bayes (NB) and K – Nearest Neighbor (K-NN). A total of six tests were performed to determine the best performing model. (See table 3-5).

Table 3-5: The list of tests and a combination of different attributes for feature vectors

ATTRIBUTE	TEST1	TEST2	TEST3	TEST4	TEST5	TEST6
1. TFIDF	Yes	Yes	Yes	Yes	Yes	Yes
2. STEMMING	No	Yes	No	No	Yes	No
3. STOP-WORD REMOVAL	Yes	Yes	No	Yes	Yes	No
4. 1-TO-3 GRAM	Yes	Yes	Yes	Yes	Yes	Yes
5. INFORMATION GAIN	Yes	Yes	No	Yes	Yes	No

3.4.5 Evaluation of Classifier

To measure the performance of a classification model, a random fraction of the labelled documents is set aside and not used for training. We may classify the documents of this *test set* with the classification model and compare the estimated labels with the true labels. The fraction of correctly classified documents in relation to the total number of documents is called *accuracy* and is a first performance measure. In this study, k-fold validation technique was used where K = 10. K-fold cross-validation[51] also known as leave-one-out validation method, the training set is partitioned into k-parts where k-1 segments are used to train the classifier while one is left out to be used as testing data. This process is done iteratively up to *k* times, finally producing the average performance. Since the test set is small, to detect over fitting other measures are included namely; precision, recall and F1 Score.

Precision measures the classifier exactness also known as Positive Predictive Value. It is a percentage of selected items that are correct. Recall measures completeness of the classifier also known as sensitivity or True Positive Rate. F-measure (F) or F1 Score is the harmonic mean of precision and recall which weights both precision and recall evenly. TP is defined as the number of true positive, FP as the number of false positive, FN as the number false negative while TN as the number of true negatives.

$$precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F = 2 \times \frac{Precision \ \times Recall}{(Precision + Recall)}$$

An abbreviation conversion using a dictionary was introduced to help address common slang in UGC, tests were performed to measure the importance of this dictionary on the accuracy of different supervised classifiers. The results are presented on table 3-6

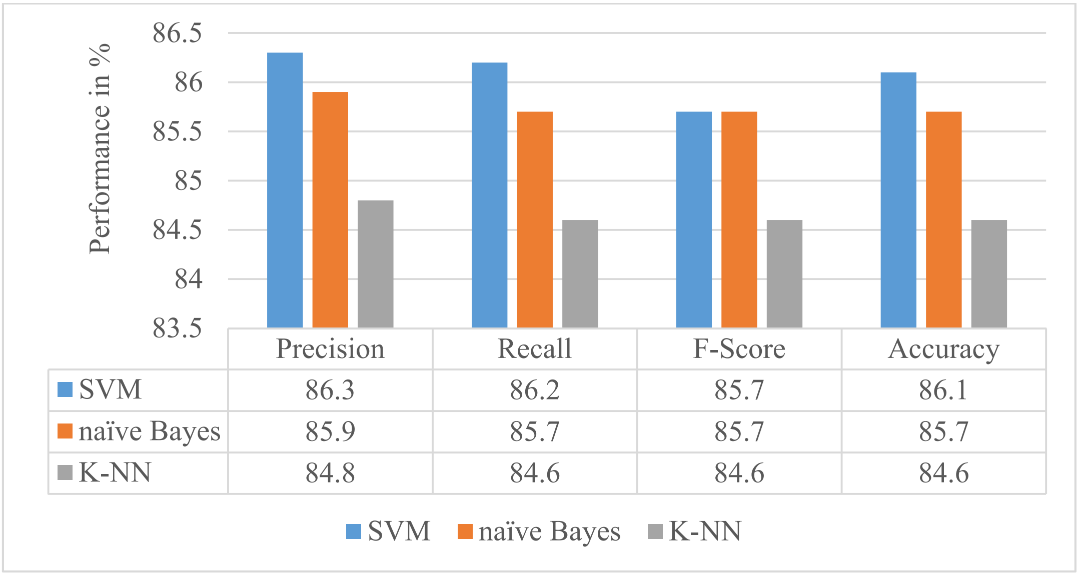
Table 3-6: Abbreviation conversion results

TEST	PRECISION	RECALL	FSCORE	ACCURACY	IMPROVEMENT	ALG
TEST1(T1) WITHOUT ABBREVIATION CONVERSION	85.6	85.7	85.2	85.6	-	SVM
	85.5	85.3	85.3	85.3	-	NB
	84.6	84.4	84.4	84.4	-	KNN
TEST1(T1) WITH ABBREVIATION CONVERSION	86.3	86.2	85.7	86.1	+0.5	SVM
	85.9	85.7	85.7	85.7	+0.4	NB
	84.8	84.6	84.6	84.6	+ 0.2	KNN

From the results, the abbreviation conversion affects all classifiers positively by performing better on an average of 0.36%; SVM +0.5%, NB +0.4%, and KNN +0.2%. Hence, the dictionary was included in all the other experiments and tests.

Test 1(T1): SVM, Naïve Bayes and K-NN = 3 training done with TFIDF, stop-word removal, 1-to-3 gram and information gain. SVM Kernel = Linear, C=10.(See table 3-7).

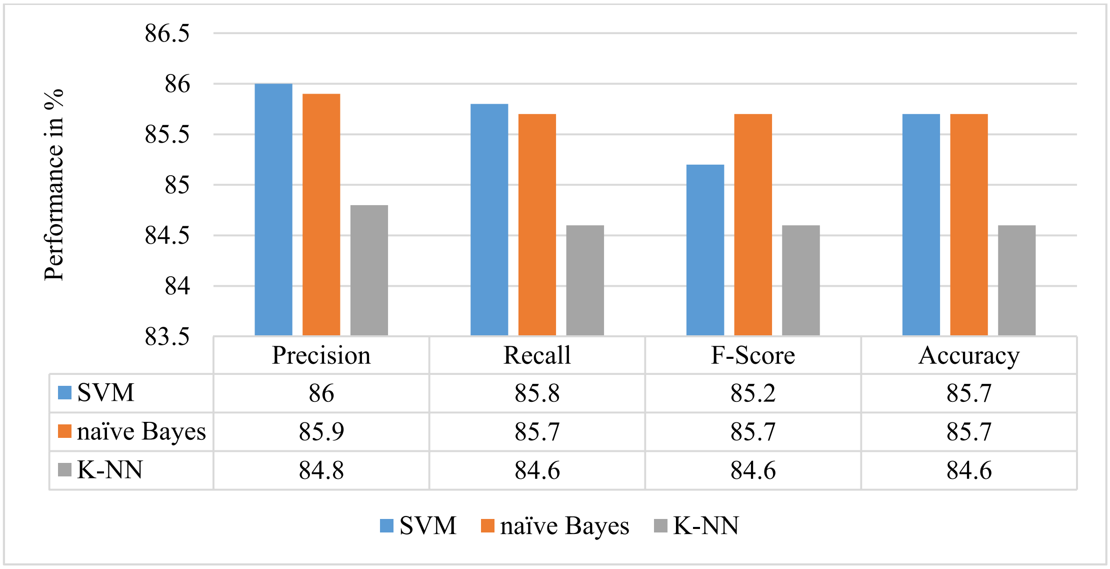
Table 3-7: Test 1 – SVM, Naïve Bayes and K-NN = 3 training done with TFIDF, stop-word removal, 1-to-3 gram and information gain



	Precision	Recall	F-Score	Accuracy
SVM	86.3	86.2	85.7	86.1
naïve Bayes	85.9	85.7	85.7	85.7
K-NN	84.8	84.6	84.6	84.6

Test 2(T2): SVM, NB, K-NN = 3 training done with TFIDF, stemming, stop-word removal, 1-to-3 gram and information gain. Kernel = Linear, C=100.(See table 3-8).

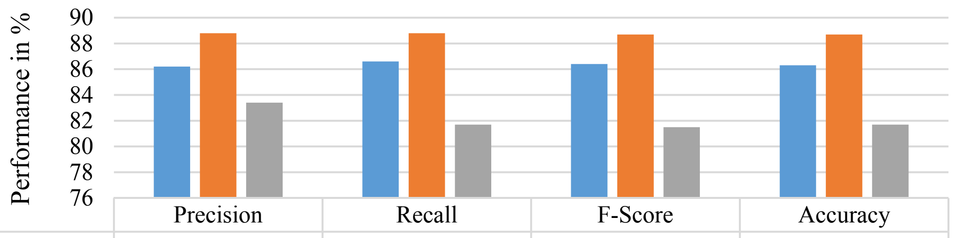
Table 3-8: Test 2 - SVM, NB, K-NN = 3 training done with TFIDF, stemming, stop-word removal, 1-to-3 gram and information gain.



	Precision	Recall	F-Score	Accuracy
SVM	86	85.8	85.2	85.7
naïve Bayes	85.9	85.7	85.7	85.7
K-NN	84.8	84.6	84.6	84.6

Test 3 (T3): SVM, NB, K-NN = 3 training done with TFIDF, 1-to-3 gram. Kernel = Linear, C=1000. (See table 3-9).

Table 3-9 : Test 3 - Test 3: SVM, NB, K-NN = 3 training done with TFIDF, 1-to-3 gram. Kernel = Linear, C=1000



	Precision	Recall	F-Score	Accuracy
■ SVM	86.2	86.6	86.4	86.3
■ naïve Bayes	88.8	88.8	88.7	88.7
■ K-NN	83.4	81.7	81.5	81.7

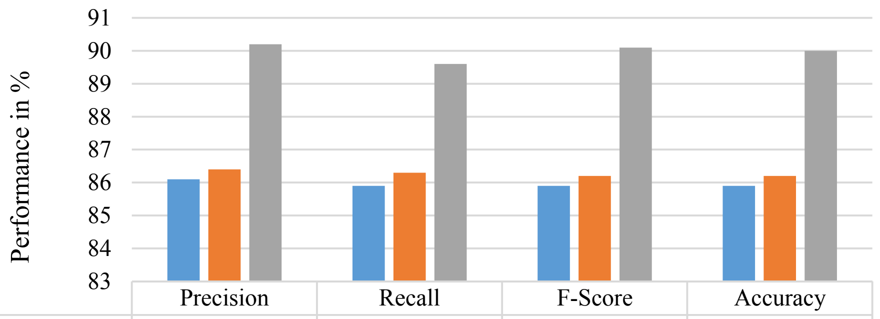
■ SVM ■ naïve Bayes ■ K-NN

Test 4(T4): SVM training done with TFIDF, stop-word removal, 1-to-3 gram and information gain. Kernel = rbf, C=1000, gamma = 0.001. Accuracy = 85.9% (see table 3-10).

Test 5(T5): SVM training done with TFIDF, stemming, stop-word removal, 1-to-3 gram and information gain. Kernel = rbf, C=10, gamma = 0.1, Accuracy = 86.2.5%. (See table 3-10).

Test 6(T6): SVM training done with TFIDF, 1-to-3 gram. Kernel = rbf, C=100, gamma = 0.01, Accuracy = 90%. (See table 3-10). Further a summary of all the best performance on accuracy for each classifier is listed on table 3-11. While tables 3-12, 3-13, 3-14 show confusion matrix for the different classifiers that performed best given a set of feature vectors.

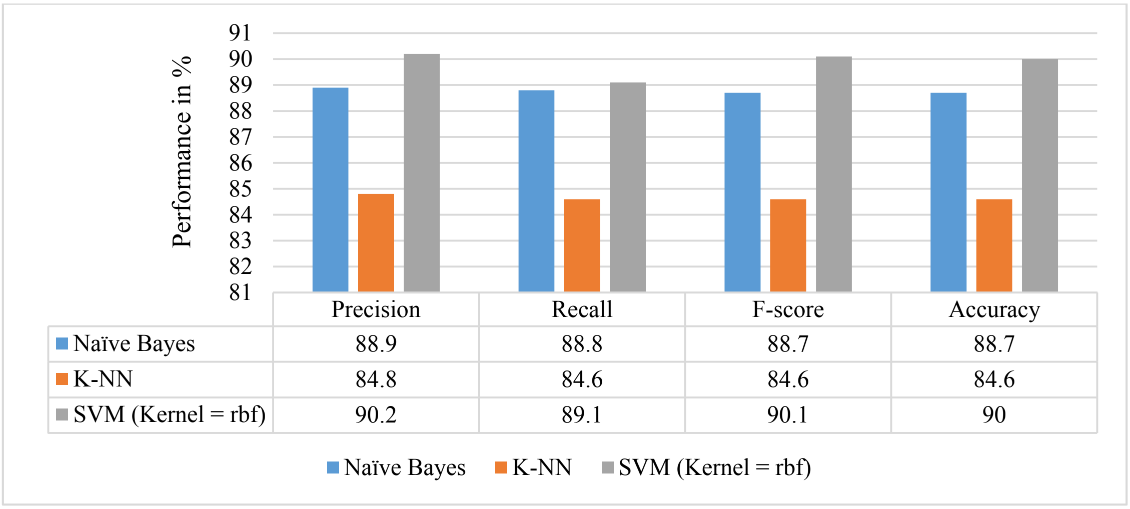
Table 3-10: Test 4 – Test 6 results table and graph



		Precision	Recall	F-Score	Accuracy
T4	SVM C=1000, gamma = 0.001	86.1	85.9	85.9	85.9
T5	SVM C=10, gamma = 0.1	86.4	86.3	86.2	86.2
T6	SVM C=100, gamma = 0.01	90.2	89.6	90.1	90

■ SVM C=1000, gamma = 0.001 ■ SVM C=10, gamma = 0.1 ■ SVM C=100, gamma = 0.01

Table 3-11: Summery of the best classifiers with the Accuracy measurement in %



	Precision	Recall	F-score	Accuracy
🟦 Naïve Bayes	88.9	88.8	88.7	88.7
🟧 K-NN	84.8	84.6	84.6	84.6
⬜ SVM (Kernel = rbf)	90.2	89.1	90.1	90

Table 3-12: Confusion matrix for NB

ACTUAL LABEL	PREDICTED LABEL	
	ACCIDENT	NON-ACCIDENT
ACCIDENT	566	74
NON-ACCIDENT	70	570
PRECISION	88.9%	
RECALL	88.8%	
F-SCORE	88.7%	
ACCURACY	88.7%	

Table 3-13: Confusion matrix for KNN

ACTUAL LABEL	PREDICTED LABEL	
	ACCIDENT	NON-ACCIDENT
ACCIDENT	515	125
NON-ACCIDENT	72	568
PRECISION	84.9%	
RECALL	84.6%	
F-SCORE	84.6%	
ACCURACY	84.6%	

Table 3-14: Confusion matrix for SVM with rbf kernel

ACTUAL LABEL	**PREDICTED LABEL**	
	ACCIDENT	**NON-ACCIDENT**
ACCIDENT	572	68
NON-ACCIDENT	67	573
PRECISION	90.2%	
RECALL	89.9%	
F-SCORE	90.1%	
ACCURACY	90%	

3.4.6 Discussion

The results varied depending on the classifier model used and also the different feature vectors. Tests T1, T2 and T3 were done using different classifiers namely SVM, NB and k-NN where k=3. Different feature attributes for each classifier gave varying results which helped to determine the best combination. In T1 to T3, NB performed best overall while SVM (Linear kernel) and k-NN followed in that order. Since NB does not have any parameter tuning, accuracy could not be further improved. SVM has more kernels and parameters that could be used to improve the classifier performance. Under this assumption, SVM was the only algorithm considered for experiment T4, T5 and T6 to test different kernels and parameter settings.

TFIDF, 1-to-3 Gram and abbreviation dictionary were the best features while the SVM with rbf Kernel was the best overall. Abbreviation dictionary increased the performance of the overall accuracy by 0.5%.

The parameters that performed well for the function were cost (C) being 100 and gamma (γ) being set as 0.01 giving an average accuracy of 90%, hence chosen for the TADS system implementation as the classification model. Other classifiers' performances were low, NB and k-NN gave an average accuracy of 88.9 % and 84.6% respectively.

3.5 Traffic Accident Location Extraction

3.5.1 Introduction to Named Entity Extraction

One of the goals in this research is extraction of locations as mentioned in a traffic accident related tweet. As indicated in the literature review, several text mining methods are used for information extraction from text. Named entity recognition (NER) [41] aims at identifying phrasal information from text which expresses named entities. These entities can be time, persons, locations, organizations or money.

A Location is defined as a place or a geographical mention which identifies where a traffic accident happened as expressed on the tweet. It can be near a building, road segment, estate or a suburb or other point of interest (POI), for example a hospital, a bank, a mall or an airport. Location extraction from text is largely a problem of NLP through information extraction.

A set of tools exist for automatic identification of named entities namely:-

StanfordNER[52] is a Java implementation of a Named Entity Recognizer to identify three major classes of named entities: PERSON, ORGANIZATION, and LOCATION. It implements Conditional Random Field (CRF) model to label sequences of words into entity types.

OpenNLP[53] is a Java based library for various natural language processing tasks, such as tokenization, part-of-speech (POS) tagging, and NER. This tool trains a Maximum Entropy model using information from the whole document to recognize entities.

T-NER[54] is a specific toolkit developed for performing natural language processing on Twitter data. It applies a supervised topic model, called LabeledLDA, together with Freebase as a source of distant supervision, to classify entity mentions in tweets. T-NER provides usage through classification, POS tagging, and chunking.

Challenges

StanfordNER and OpenNLP are trained on formal text from a news corpus. They may not scale well on Twitter data. Even though T-NER is trained on Twitter data, our problem is a bit complex for off-the shelf tools since our accident locations contain local names and streets. These tools also require sufficiently annotated training data which is expensive and takes a lot of effort. Most gazetteers like Geonames have location at country and city level which does not suit the problem at hand where street and local place names may not exist in a gazetteer database

3.5.2 Syntactic Analysis

Detecting named entities through syntactic analysis, follows an information extraction pipeline. Syntactic analysis also known as parsing is the task of recognizing a string of text and assigning structure to it, for example POS, chunking or parse tree generation as demonstrated in figure 3-8.

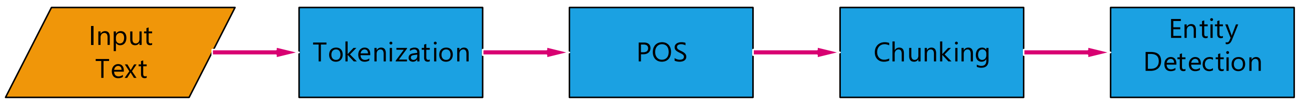


Figure 3-8: Named Entity Pipeline through NLP syntactic parse analysis

Tokenization: Breaks sentences into separated words and punctuations called tokens. These tokens can then be parsed to other modules for further processing.

Table 3-15: Penn Treebank Tagset

Tag	Description	Tag	Description	Tag	Description
CC	Coordination conjunction	JJR	Adjective, comparative	NNPS	Proper noun, plural
CD	Cardinal number	JJS	Adjective, superlative	PDT	Pre-determiner
DT	Determiner	LS	List item marker	POS	Possessive ending
EX	Existential *there*	MD	Model	PRP	Personal pronoun
FW	Foreign word	NN	Noun, singular or mass	PRP$	Possessive pronoun
IN	Preposition or subordinating conjunction	NNS	Noun, plural	RB	Adverb
JJ	Adjective	NNP	Proper noun, singular	RBS	Adverb, superlative
RP	Particle	SYM	Symbol	TO	to
UH	Interjection	VB	Verb, base form	VBD	Verb, past tense
VBG	Verb, gerund or present participle	VBN	Verb, past participle	VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present	WDT	Wh-determiner	WP	Wh-pronoun
WP$	Possessive wh-pronoun	WRB	Wh-adverb		

Part-of-Speech tagging (POS): Is an important process which aims at labelling each token with a tag indicating its syntactic role in a sentence, whether a word is noun, verb, an adjective etc. (see table 3-15 for a comprehensive tagset from the Penn Treebank WSJ). A POS tagger is a program that does this task automatically. Taggers use a large number of annotated and trained corpus to properly tag each word. Examples include OpenNLP, Stanford POS and Twitter trained T-POS[54].

Chunking: Also known as shallow parsing[55], segments and labels multi-token sequences as illustrated in figure 3-9. The smaller boxes show the word-level tokenization and part-of-speech tagging, while the large boxes show higher-level chunking. Each of these larger boxes is called a **chunk**. Like tokenization, which omits whitespaces, chunking usually selects a subset of tokens, hence grouping consecutive words that belong together. This can extract meaningful phrases by observing a particular pattern on POS tags.

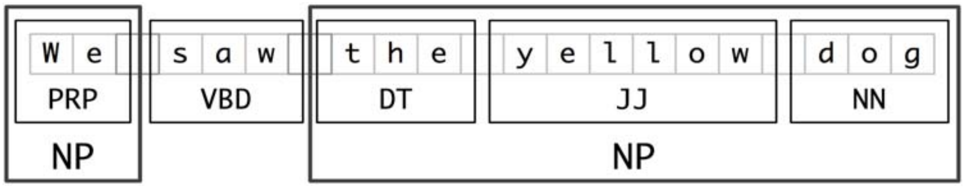


Figure 3-9: Noun Phrase Chunker (J.Perkins,2014)

Chunks can include Noun Phrases (NP) containing determiners, adjectives and nouns, Verb Phrases (VP) containing verbs and Preposition Phrases (PP) containing prepositions. Regular expressions are the most convenient way of building chunks using rule based approaches. Sometimes these chunks can be represented as trees to produce syntactic parse trees where each chunk form a constituent which can be manipulated directly (See figure 3-10).

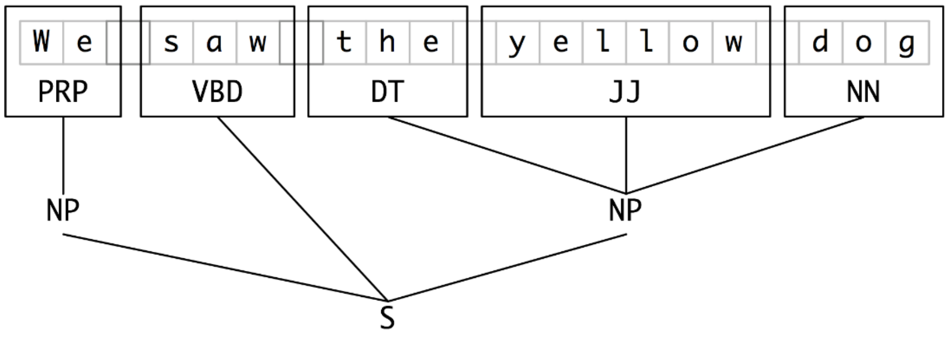


Figure 3-10: Chunk represented in a parse tree (J.Perkins,2014)

Entity Detection: Entity is a definite Noun Phrase that refers to a specific type on an individual, organization, location, date etc. For example, to identify a location, we can have a NP being matched against a dictionary with a list of locations like the Geonames Gazetteer [56]. A gazetteer is a geographical database with over 10 million geographical names updated regularly.

3.5.3 Proposed Noun Phrase with N-gram Pattern Matching for Location Extraction

Key observation on traffic accident tweet:

Observation 1: A tweet may contain more than one location: primary location and one or more secondary locations. Primary location is exact point where the accident took place while secondary location is an estimation location nearing the accident spot. A preposition is present as an indication of a spatial location on the words that follow. This preposition include at, near, along, on, in, before, after etc.

Observation 2: A tweet may not contain any preposition but contain location mentions none the less.

Observation 3: Chunker may miss to combine the right location NP. This is due to the fact that some of the tokens might be tagged with the wrong POS tag.

With the traffic accident tweets observation in mind, the study proposed a methodology, broken down into several sub-modules as depicted in figure 3-11.

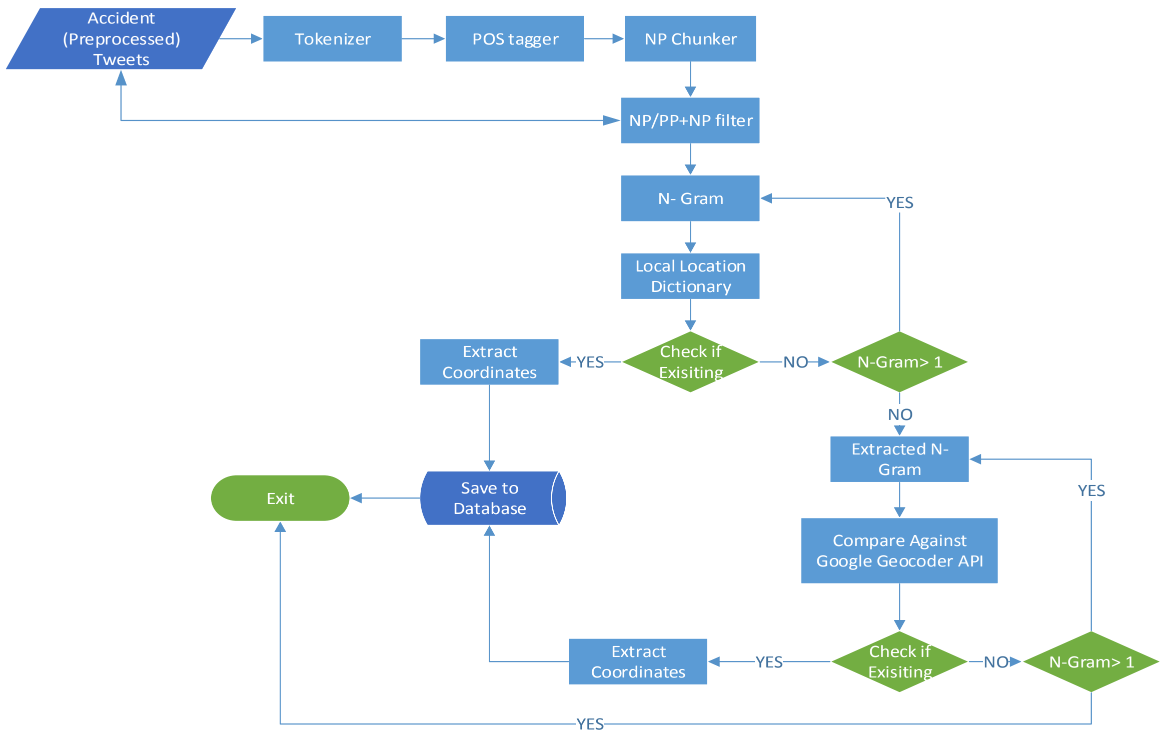


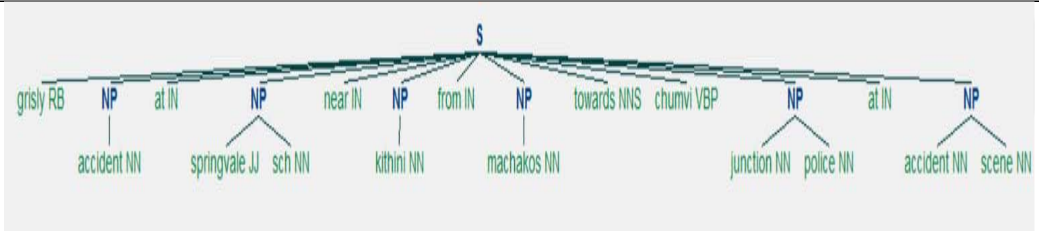
Figure 3-11: Proposed location extraction pipeline

For every accident tweet, the preprocessing module normalizes text by removing all the URLs, @mentions and other Twitter centric information not necessary for location detection. Hashtags are retained but the # symbol was removed as some of the hashtags may contain location mentions. The tweet was tokenized and tagged using NLTK POS tagger trained on Brown Corpus. A NP regular expression was then applied to extract chunks that contain adjectives and all nouns. A dictionary filter was also used to check presence of a preposition before any NP. Another regex filter was implemented to check for any location cues and extract likely mentions of these places. Each NP and its constituent N-grams are checked against a local location dictionary for existence where it returns the geo coordinates to be saved into the database. If the location does not exist in the local dictionary the external database was utilized.

3.5.4 Noun Phrase Extraction

Locative information is generally expressed using nouns hence chunking was used to combine tokens that are nouns and adjectives through a set of regular expressions. Consequently, each NP was constructed and manipulated through a parse tree (See table 3-16).

Table 3-16: A simple demonstration for syntactic analysis on an accident tweet

Process	**Description**
Tweet	grisly accident at springvale school near kithini from machakos towards chumvi junction police at accident scene
Tokens	['grisly', 'accident', 'at', 'springvale', 'school', 'near', 'kithini', 'from', 'machakos', 'towards', 'chumvi', 'junction', 'police', 'at', 'accident', 'scene']
POS	[('grisly', 'RB'),('accident', 'NN'),('at', 'IN'),('springvale', 'JJ'),('school', 'NN'),('near', 'IN'),('kithini', 'NN'),('from', 'IN'),('machakos', 'NN'),('towards', 'NNS'),('chumvi', 'VBP'),('junction', 'NN'),('police', 'NN'),('at', 'IN'),('accident', 'NN'),('scene', 'NN')]
Parse Tree structure with NP	(S grisly/RB (NP accident/NN) at/IN (NP springvale/JJ sch/NN) near/IN (NP kithini/NN) from/IN (NP machakos/NN) towards/NNS chumvi/VBP (NP junction/NN police/NN) at/IN (NP accident/NN scene/NN))
Parse tree diagram	 S grisly RB NP at IN NP near IN NP from IN NP towards NNS chumvi VBP NP at IN NP accident NN springvale JJ sch NN kithini NN machakos NN junction NN police NN accident NN scene NN

3.5.5 Filters: Preposition and Regular Expressions

1) Preposition Filter

As indicated in observation 1, most traffic accident tweets have more than one location mentions; primary and secondary locations. In this study, the assumption was that primary locations give exact point of the incident while secondary locations give proximity from the primary location. Primary locations are mostly preceded by preposition 'at', 'in', 'along', 'on' while secondary locations are preceded by 'after', 'before', 'near', 'outside', 'towards', 'opposite'. To this effect, a filter was used to first check presence of a preposition to determine the location to be assigned to the tweet after the extraction process.

2) Regular Expressions Filter

In observation 3, the POS may misstag a token, consequently the chunker may not extract the correct NP, which means a location might be incomplete for the geographical database to return a match. For example "accident thika road past KU", the tokens are tagged as follows ('accident', 'NN'),('thika', 'NN'),('road', 'VB'),('past', 'PP'), ('KU', 'NN'). From the example "road" is tagged as a verb so it is not included as a NP. Hence a sub-component address matching filter was introduced to get a location where no match was returned from the previous step. This was done by implementing a set of regular expressions which matches and extract any tokens that are indicative of an address. Address expressions include street types and abbreviations. Example of such location addresses include; overpass, underpass, bypass, road, rd, avenue, av, street, st, highway, junction, hwy etc.

3.5.6 N-gram Location Matching

For each noun phrase, an n-gram may contain both location and non-location tokens. For example, a NP "general motors outbound" has "general motors" a location and "outbound" a non-location. It would be impossible to accurately match against a location dictionary. To solve this, n-gram location matching is used. First, we match the whole NP as a single location, if an exact match is not returned, we consider all the n-gram subsets. For an NP of N tokens, break the n-gram N-1 to unigram. If n-gram location match is found, other subsets will not be considered. It is important to process this subsets in descending order so as to match the maximum NP subset.

To demonstrate this approach, taking a noun phrase "General Motors Outbound" as example, it contains location tokens and non-location token in a single phrase (see figure 3-12). First, the whole NP is matched as a location, no accurate match is returned. The second phase is to try matching 2-grams by dropping one token "General Motors ~~Outbound~~". This will return a match, hence the unigram will not be considered. The tokens are matched against another component of the system geographical location lookup databases explained later in this section.

3-gram	General Motors Outbound
2-gram	General Motors ~~Outbound~~
unigram	General ~~Motors Outbound~~

Figure 3-12: Demonstration of how the N-gram based partial matching works

3.5.7 Geographical Name Matching

1) Local location database

Faster and accurate location lookup was achieved by constructing a local location database. By using the Open Street Map dictionary containing 7000 locations from the study area, a Geojson file for possible locations including buildings, roads, highways, points of interest like airports, train stations etc. was downloaded and stored locally. This database has coordinates for visualization on a map. Since most UGC has misspellings, a fuzzy string matching fuzzywuzzy[57] was integrated into the location search. The Fuzzer makes use of Levenshtein distance algorithm which compares a word or sentences and give a percentage of how close they relate by frequency and occurrences of characters they contain.

2) Google Map V3 API

Google map has a powerful API[58] that exposes programming interfaces useful for location matching. These interfaces include geocoding, reverse-geocoding, mile marker, direction etc. In this research, geocoding was highly utilized. Geocoding is the process of transforming addresses and place names into coordinates which is used to map a location on the earth surface. When the geocoder finds a match, the result is returned in JSON format. If the location is not in the local dictionary, the noun phrase is matched against the map API service. The geocoder may return results from a different region with a similar name, to restrict results from the study area, a bounding box with coordinates for the area of interest was defined for the API.

3.5.8 Experiment and Results

A random selection of 250 tweets with location mentions was used to test the proposed method. The dataset contained 137 unique locations. Evaluation was done by counting manually how the locations were extracted and calculated the precision, recall and F-score. (Refer to section 3.4.5).

The system recognized 160 locations, 115 were correctly identified while 45 were wrongly identified as locations while 31 locations were missed. (See table 3-17).

Table 3-17: Performance of location extractor

Precision	Recall	F-score	TP	FP	FN
0.718	0.839	0.773	115	45	22

The recall was mainly affected because the extractor could not extract more than three locations, at the same time, it was not possible to extract locations with 'between' preposition. The other reasons are that some descriptions were too complex, short, with abbreviations not expressed in the regular expression or POS tagging error which led to missed or wrong extraction of a noun phrase. Precision on the other hand, identified other noun phrases that had street address cues but were really not locations for example 'Easy street'.

3.6 Summary

In this chapter, objectives are reached in several folds. During data collection, to overcome problems with streaming API returning irrelevant tweets, query keywords were used to return and filter the most relevant tweets on traffic accidents. The combination of words were derived from the most common words that are used to report this incidences. Most tweets heavily use slang and internet abbreviations, this problem was partially solved by building a dictionary of over 5000 words which corrects the slang and abbreviation useful during Part Of Speech tagging and also in maintaining a uniform distribution of words useful for classification. Since UGC also contain vulgar language, a dictionary containing over 460 words was used to remove any word that is listed as bad word.

To train a classifier, 1280 annotated data was used as a training set with a balanced 640 tweets for each representing positive and negative classes. Since tweets are short, all words were kept in a tweet except the ones that were not useful, like Twitter centric features; URL, @mentions, and other HTML tags and punctuations while preserving the original tweet for later use. A combination of word tokens 1-to-3 grams (unigram, bigrams, trigrams), TFIDF and abbreviation dictionary were observed to be best feature vectors for the classifier giving a precision of 90.2% and a recall of 89.1% by applying SVM with a choice of rbf kernel – gamma = 0.01 and C = 100.

Location is an important aspect in situation awareness. In this study, accident location was extracted from content by using NLP syntactic analysis. First, a noun phrase is extracted then each is broken into n-grams which are matched against a locally constructed location dictionary or external database on Google map API. Due to misspelling, a fuzzy string matcher implemented using Levenshtein distance algorithm was used to match n-gram with the local dictionary. Both local and external dictionaries play a major role in location identification and extraction. The proposed method achieves an accuracy of 0.718 on precision and 0.839 on recall. The average performance is attributed to the fact that tweets use free form grammar and they are short hence some accidents description was too complex, considering the assumptions made when building the algorithm.

4. WEB APPLICATION PROTOTYPE FOR TRAFFIC ACCIDENT DETECTION SYSTEM

4.1 Introduction

In this chapter, a web application prototype for road traffic detection is presented. Traffic Accident Detection System (TADS) was implemented by following the software development cycle framework; requirements analysis, system design, implementation and testing.

4.2 Requirement Analysis

Requirement analysis helps in understanding the application domain and capturing, formalizing and analyzing user requirements for TADS. These include; Functional requirements that specifies what behavior the system should offer like inputs, outputs etc. and non-functional requirements which are constraints to the system like performance, reliability and other non-behavioral characteristics.

4.2.1 Functional Requirements for the Traffic Accident Detection System (TADS)

The following are the functions that are essential for the system to work according to requirements.

1) The system should collect tweets from Twitter streaming API by using query or keyword filter terms.
2) The system should store raw JSON tweets into a database.
3) The system should preprocess all tweets in the database for further analysis.
4) The system should categorize all the tweets and classify them as accident or non-accident.
5) The system should extract locations from tweet content and also geotaggs from geocoded tweets.
6) The system should display map of the location.

4.2.2 Use cases Diagrams

As seen in figure 4-1, the use cases diagram show different ways in which different users will be interacting with the system. The user can browse accident related tweets by querying the database, view the location of the accident on space and view simple report analytics. The Admin can delete Tweets, and also retrain the classifier from the front panel.

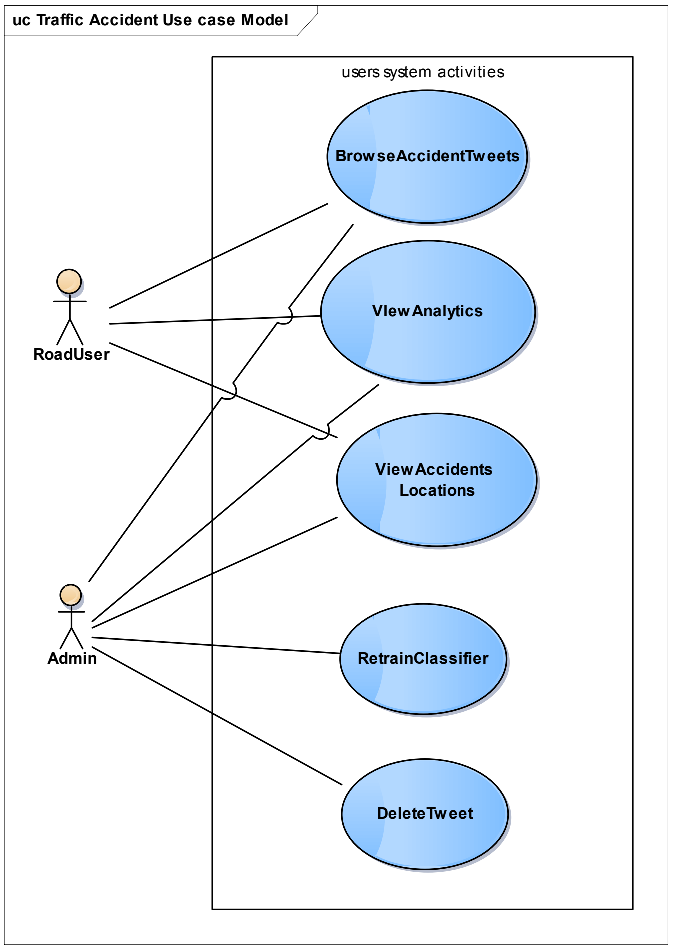


Figure 4-1: Use case Diagram for a user

4.2.3 Non-functional Requirements

1) The system must display accidents on the dates that have been queried by the system user.

2) Before any processing, all tweets with the selected metadata must exist in the relational database.

3) The user should have real-time access to the database using the system UI.

4) All tweets that are geotagged must be extracted reverse-geocoded for display of location name.

4.2.4 System Environment Requirements

Software Requirements - Different software components are required to run TADS;

1) PHP 5
2) Apache web server
3) MongoDB
4) Python
5) MySQL
6) JQUERY
7) Twitter Bootstrap
8) Google Maps API(JavaScript and Geocode API)

JQUERY is feature-rich JavaScript library. It makes it easier for HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. PHP is a web scripting language that can be used to manipulate databases. It is also used to render HTML for the user interface. Twitter Bootstrap is a free and open-source collection of tools for creating websites and web applications as a front end framework. It contains HTML and CSS based design templates for typography, forms, buttons, navigation and other interface components. It eases the development of dynamic websites and web applications. Google Maps API is a JavaScript Library for displaying locations on a map.

User Interface Requirements - User interaction is very important. It must be easier for user to navigate on the system and view related information as stored in the database. Different technologies were combined and integrated including bootstrap, jQuery and Google maps API for JavaScript.

Hardware requirements - To demonstrate TADS, the following are the minimum hardware requirements. (see table 4-1).

Table 4-1 : Hardware requirements

PROCESSOR	INTEL CORE I5
HARD DISK	500 GB
RAM	4 GB
NETWORK CAPABILITY	LAN

4.2.5 Constraints

Some elements of the TADS were not developed as planned due to different limitations.

1) Due to time limitations some modules were not developed and hence recommended for future work.
2) Due to Twitter usage limitation by the host country, tweets could not have been collected in an online manner. Hence crawling script was executed in the Amazon EC2 cloud and stored the tweets in JSON file format for each day, later the files were loaded manually to MongoDB on a local machine.

4.3 System Design

4.3.1 Activity Diagrams

SVM classification activity (Figure 4-2) is triggered every time a new tweet is received in the DB. Firstly, the new tweet is preprocessed by removing all unnecessary elements like URLs, hashtags, and also grammar correction, then the tweet is passed to the SVM classifier from the python scikit library which extracts features, loads the already trained and saved model and finally predicts the label of the tweet as either accident or non-accident related tweet.

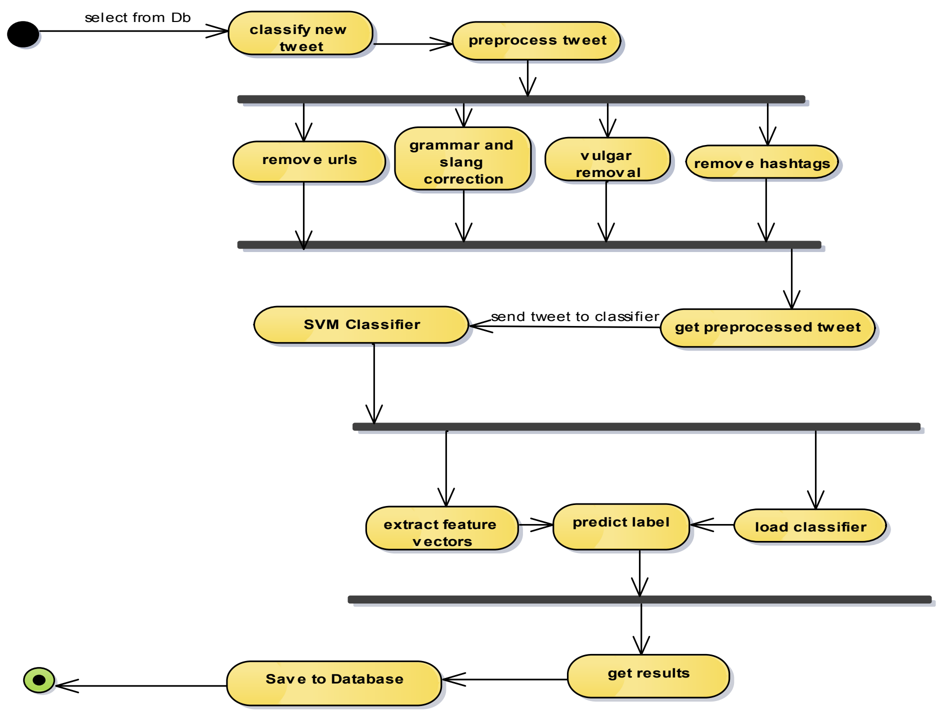


Figure 4-2: Tweets accident classification with SVM activity diagram

The user must submit a search with a date range on interested period of time to view traffic accident related tweets as shown on view activity diagram (figure 4-3). The system fetches the results and displays results back to the user. Also location links are displayed. If a user clicks on the link, a pop up with a map pointing to the location of the incident is displayed.

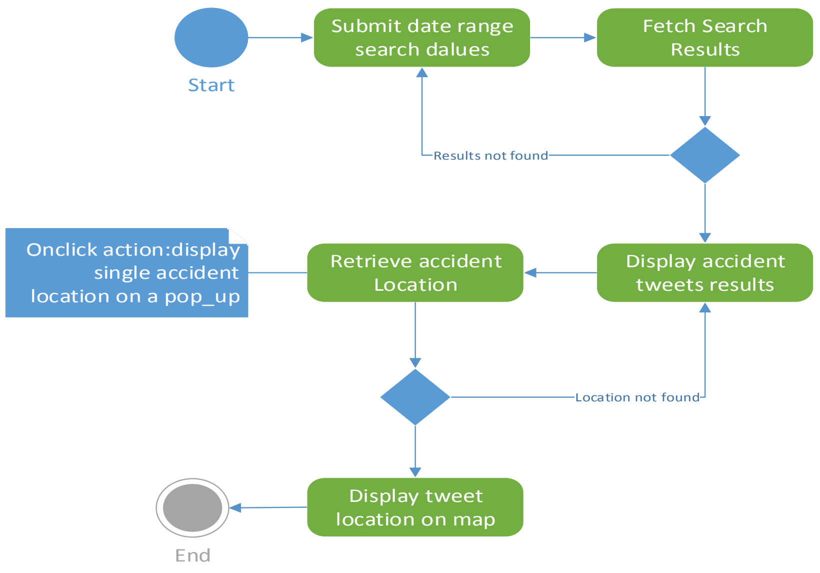

Start
Submit date range search dalues
Fetch Search Results
Results not found
Onclick action:display single accident location on a pop_up
Retrieve accident Location
Display accident tweets results
Location not found
Display tweet location on map
End

Figure 4-3: Traffic accident retrieval by date range search activity diagram

In traffic accident map location activity diagram, the user submits a search by date to retrieve accident related tweets. All locations for each tweet is retrieved and displayed on a map indicated by a marker. For each marker, a place holder with details for each tweet is displayed on a pop-up. These include sender, date of incident, tweet describing the incident and a brief accident history for that location if it exists. (See fig 4-4).

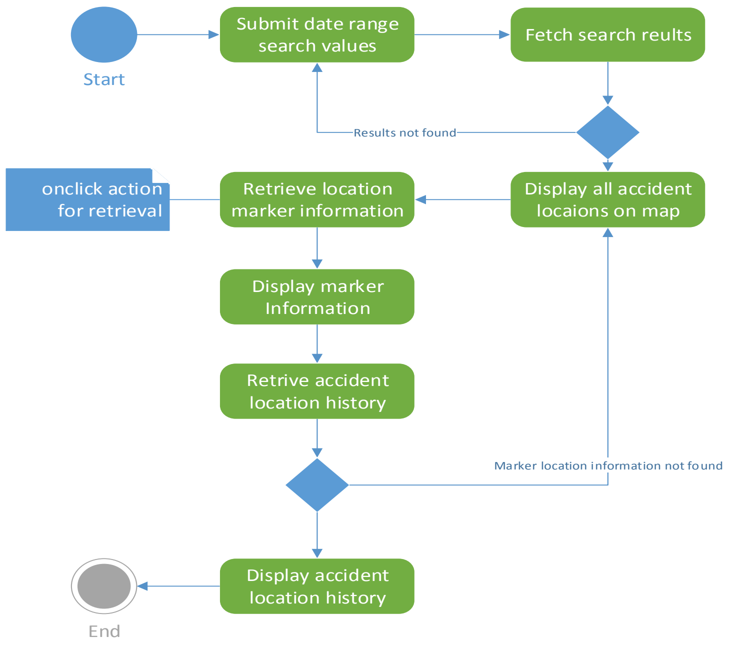

Start
Submit date range search values
Fetch search reults
Results not found
onclick action for retrieval
Retrieve location marker information
Display all accident locaions on map
Display marker Information
Retrive accident location history
Marker location information not found
Display accident location history
End

Figure 4-4: Traffic accident Locations map by date range search activity diagram

4.3.1 Sequence Diagram

In categorizing tweets, the SVM classification sequence diagram (Figure 4-5) shows how different messages are passed by different modules in the system. The System will select a new tweet from the DB, then pass the tweet to the preprocessor module which returns the preprocessed tweet back to the system eventually sending it to the classifier which extracts feature vectors, loads the trained model, predicts the label for the tweet and returns the tweet and the label back to the system which finally saves this information to the database.

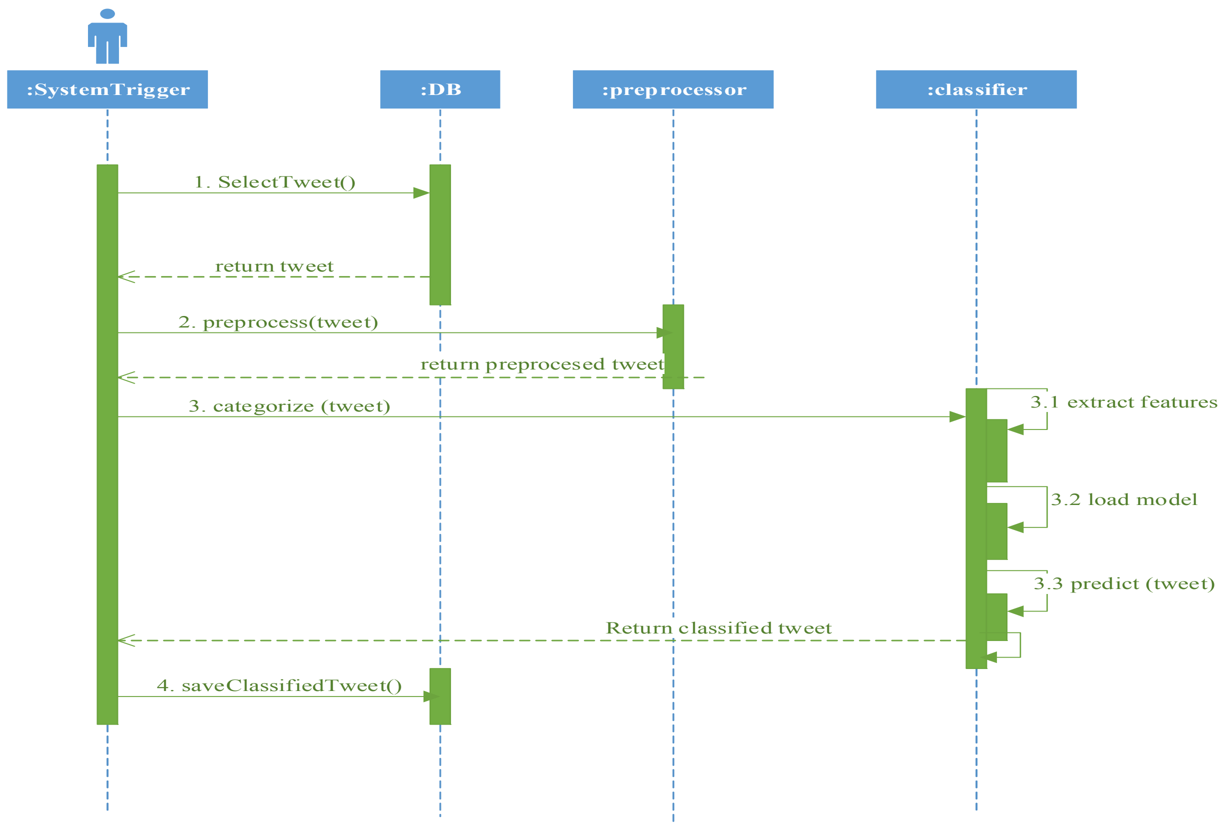


Figure 4-5: Classification Sequence Diagram

4.3.2 Class Diagram

The class model shows different classes that exist in the system (see figure 4-6). They include:-

1) **TweetCollector**: This class is responsible for collecting tweets through the Twitter Streaming API and saving into a non-relational database.
2) **TweetManager** Deduplicate by removing retweets, manage and populate the main relational database with the required Twitter metadata and information.
3) **Classifier** class has the SVM functions that train, save, load and predict traffic accidents by categorizing accident and non-accident.
4) **NlpProcessor** class contains all functions for natural language processing, POS, stemmer, tokenizer and stop word remover and maintains interaction with the main database.

5) **TweetCleaner** is a class used to strip a tweet off all unwanted characters, URLs, punctuations, numbers, spaces, mentions, hashtags and HTML tags.
6) **ExtractLocation** is a class responsible for location extraction from content and geotaggs from tweet metadata. It preprocess a tweet, chunks, creates Noun Phrases and N-grams and also it has connection to local and external location database and main database.

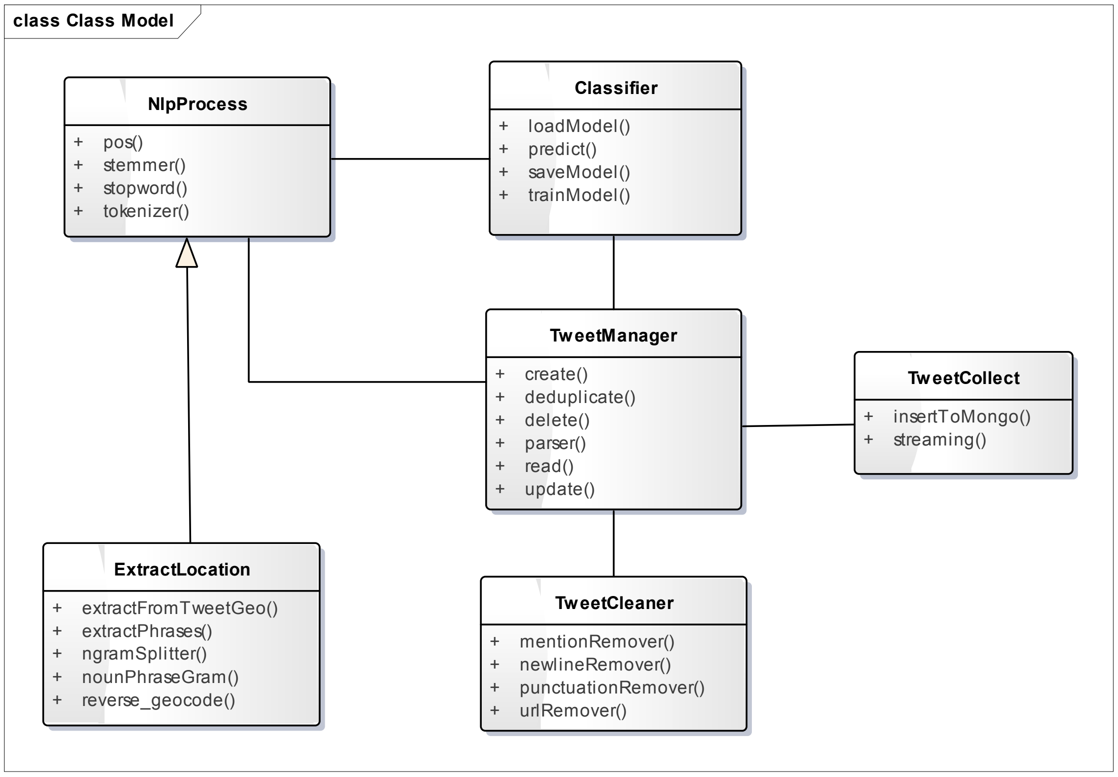


Figure 4-6: Class Diagram

4.3.3 Database Design

The database has five(5) main entities; raw_tweet, getag_location, content_location, classified_tweet and user (see figure 4-7). Their description is as follows:

1) **Raw_tweets:** It is the main table with selected metadata from unclassified tweets parsed from MongoDB. Every tweet here is stored in its original form.
2) **Geo_location:** Maintains all geotags with coordinates extracted from tweet metadata and also name of location obtained after reverse geocoding.

3) **Content_location:** Contains all location names and coordinates extracted from tweet content.
4) **Classified_tweet:** Maintains all classified tweets. Each tweet has a label distinguishing it as as accident or non-accident as categorized by the classifier.

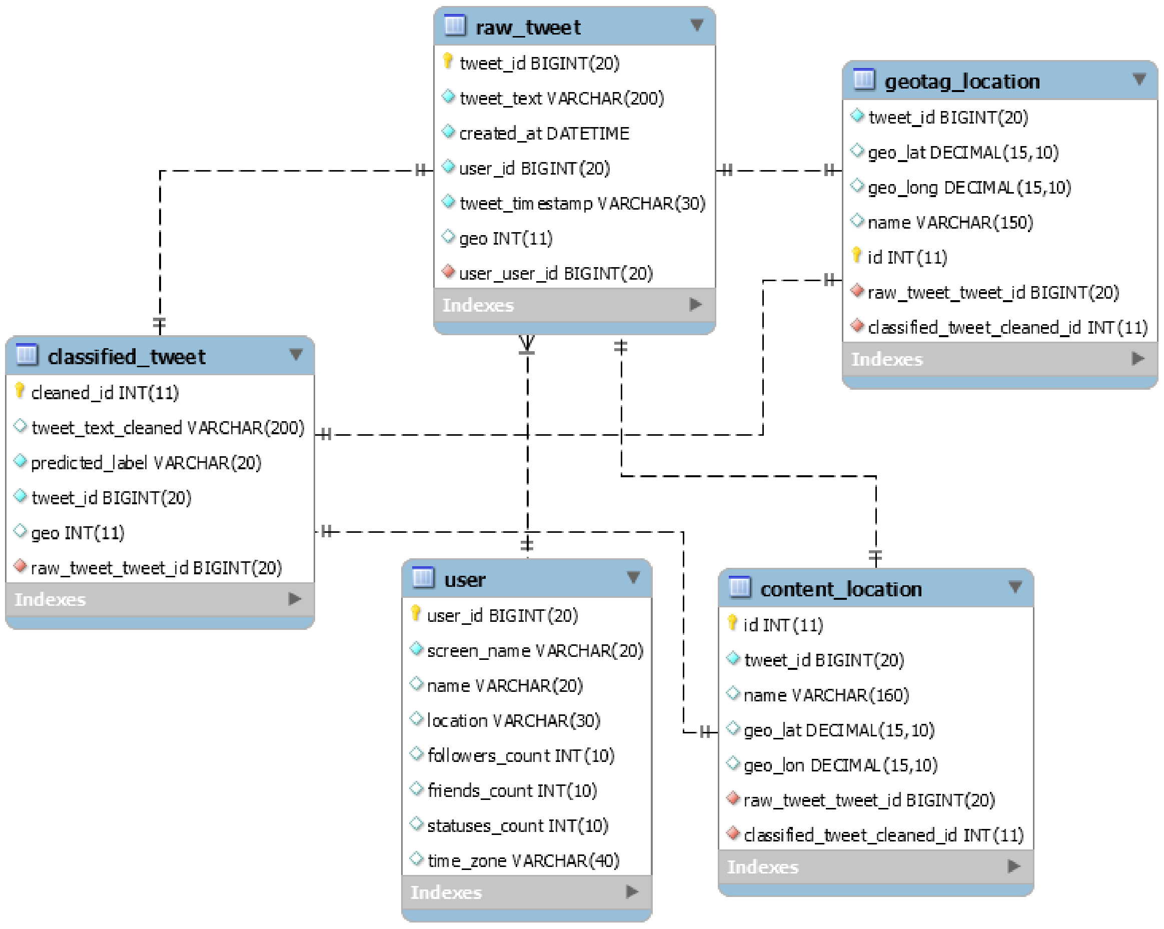


Figure 4-7: TADS database Design

4.4 Implementation

This section describes different elements that implemented TADS, thus demonstrating the proposed methods. The system architecture adopts the Model View Control (MVC) for information retrieval and manipulation. PHP, JQUERY and Bootstrap are used by the user to view different elements of the system by querying the database. Classification through SVM and location extraction was implemented using python which populates the database with the required information as demonstrated on figure 4-9.

The system comprises backend and frontend implementation, each with different elements and technologies as shown on figure 4-8. On the backend, the system contains python scripts for Tweet collection API, preprocessing, location extraction module, classification algorithms (LIBSVM), Geopy and Google Maps API for geocoding and reverse-geocoding, MySQL is the main database for all system information storage, while MongoDB holds tweets temporarily before using a script to copy the data to MySQL. On the frontend the User Interface is represented by the web browser heavily depending on Bootstrap comprising JQUERY, CSS, HTML, JavaScript and Google maps API.

4.4.1 Backend Implementation

1) Database Physical Architecture

The framework allows collection of tweets in an online manner by using a python wrapper Tweepy API to connect to the Twitter Streaming API (figure 4-9). These tweets are in JSON format and hence stored in MongoDB, a non-relational Database. A parse script is used to populate the main raw tweets table on the MySQL database.

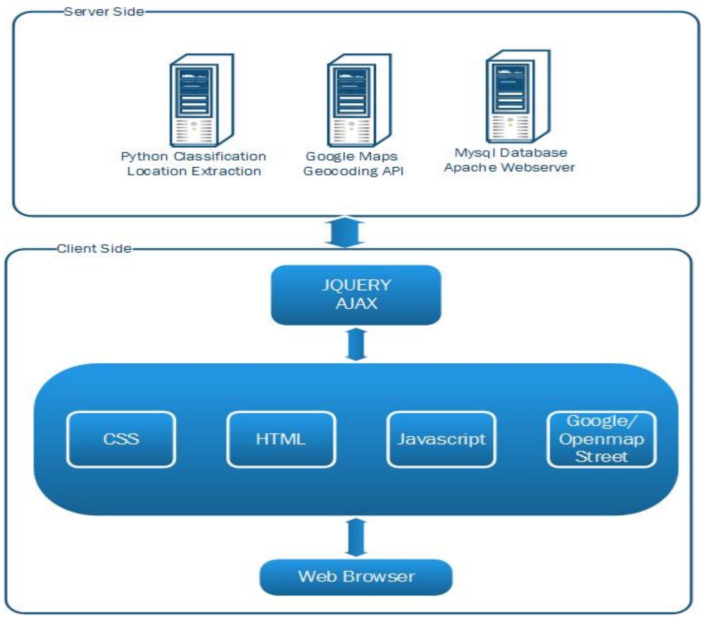

Server Side
Python Classification Location Extraction
Google Maps Geocoding API
Mysql Database Apache Webserver
Client Side
JQUERY AJAX
CSS
HTML
Javascript
Google/ Openmap Street
Web Browser

Figure 4-8: Physical system architecture

A tweet json file contains metadata (see Appendix A). The most important metadata is extracted for TADS including; Text, Tweet_Id, Name, GeoCoordinates, Created at, Timestamp, Screen_name, Followers, Place, Timezone, User_Id, Registered_location. These elements are parsed to different tables for easier retrieval, manipulation and further processing.

2) Tweets collection

Collection of tweets was done by setting up a server on amazon EC2 cloud to continuously utilize the Streaming API without interruption, using Tweepy. Tweepy is a helper module in python that connects to Twitter streaming API. Several modifications were made to realize the main goal of collecting traffic accident related tweets. This include handling errors, defining key query words for filtering tweet and saving the tweets JSON file to MongoDB. (See Appendix B).

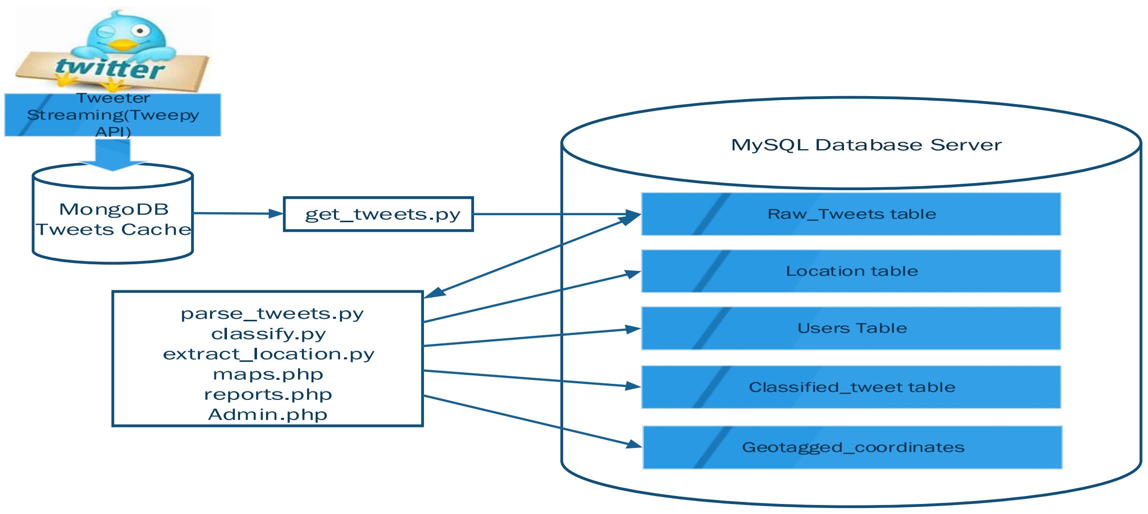


Figure 4-9: Tweets collection and parsing architecture

3) Classifier

After the tests were done on WEKA toolkit and the best model chosen, the classifier was implemented using LIBSVM in scikit learn, a python toolkit for Machine Learning. An SVM model was trained and saved using the already prepared training set. For every new raw tweet stored in the database, a call function, preprocesses and classifies the tweet by giving a label as either accident or non-accident, then stores it in a different table in the database. (Sample code under appendix B).

4) Location extraction

Locations from a tweet classified as accident were extracted in two ways, coordinates from metadata of a geotagged tweet as well as from the content through NLP methods. Geotaggs are extracted then reverse-geocoded using google map API to get the location names. Location from content was extracted through an implementation of several modules. Noun phrases extractor, n-gram generator, filters with regular expressions, preposition matching using dictionaries and geographical matching on either local location database or external geocoding database provided by Google maps API. (See appendix B).

4.4.2 Front End

1) Data engine

Data engine handles the data processing and manipulation by the search form on the user interface. The architecture used Model View Control (MVC). Data engine receives the search form entries, validates parameters by sorting and stripping off unwanted characters for the query builder. The query builder constructs statements for querying the database or dictionary for required data. Query handler executes the actual retrieval of data while presentation is handled by the view module which displays results to the user. (Sample code under appendix C appendix D for the system interfaces).

2) Search form

The search form captures user input and builds search parameters. It has a date picker module that accepts start date and end date which is parsed to the data engine. The viewer panel calls template builder and displays accident tweets.

3) Map visualization engine

Map visualization engine handles the processing of locations. Coordinates processor manipulates latitudes and longitudes then submits to the location processor which calls the Google maps API which draws the map canvas with filtered location while the marker placement displays selected information on the marker when clicked for example location name, original tweet and date and time.

4.5 System Testing

System testing is testing that is performed on an already integrated system to evaluate its completeness according to specified requirements also known as black-box testing. Test cases are performed by giving an input to the system and the tester validates the output according to scenarios. To test TADS functionality, different scenarios were observed and checked against expected results. The functions include view accident tweets, view single location, view multiple locations on a map, change classification label and view analytics as summarized in the tables 4-2, 4-3, 4-4 and 4-5.

Table 4-2 Tests the date picker and the search form

Table 4-2: Test Case 1 for the search form

Test Case: 1 Test Case Name: Display accident tweets Subsystem: Search Form Description: Check if the date picker and accident display functions			
Precondition: - The user is on the dashboard page - The most recent accident tweets are displayed			
Step	Action	Expected Results	Pass
1	Choose start date	A date picker displays date menu	✓
	Choose end date	A date picker displays date menu	✓
	Click on submit button	The system queries the database and displays at least first 10 items in descending order and a next button	✓
Post condition: - All accident tweets and related information are displayed			

Table 4-3 Tests the single map display

Table 4-3: Test Case 2 Displaying a single location on a map from pop up window

Test Case: 2 Test Case Name: Single map Location Subsystem: Maps Description: check if a tweet location link is clicked a pop up map will display the location			
Precondition: - The user has displayed the accident tweets - Location coordinates exist			
Step	Action	Expected Results	Pass
1	Click on View location on map	A pop up shows a map and details of the location with a marker pointing on the location	✓
2	Click on the map marker	A context menu pops up from marker and displays location name	✓
Post condition: - A map is displayed on a pop up window showing a point of accident location			

Table 4-4 tests multiple accident location display on a map

Table 4-4: Test case 3 Display Multiple accident locations on map

Test Case: 3 Test Case Name: Multiple accident locations on map Subsystem: Location map Description: Test if all selected accident dates can be displayed on the map			
Precondition: - The map page is displayed - The user has entered the test			
Step	Action	Expected Results	Pass
1	Choose start date	A date picker displays date menu	✓
	Choose end date	A date picker displays date menu	✓
	Click on submit button	The system queries the database and displays at least first 10 items in descending order and a next button	✓
Post condition: - All accident tweets and related information are displayed			

Table 4-5 Tests the analytics function for accident location analysis

Table 4-5: Test case 4 Displaying location analysis

Test Case: 4 Test Case Name: accident location analytics Subsystem: Analytics Description: Test to display a pie chart with accident analysis

Step	Action	Expected Results	Pass
Precondition: - The user is on the analytics page			
1	Enter Start and End date	Date picker menu	✓
2	The user clicks on Submit	The system displays a pie chart with related information	✓
Post condition: - A pie chart with accidents in % for the selected dates is displayed			

4.6 Summary

In this chapter, a web application, Traffic Accident Detection System (TADS) was proposed. The process followed a software development cycle from requirements analysis, system design, implementation and testing. Under requirement analysis, functional and non-functional requirements, uses case diagrams, system environment and constraints was presented. System design elaborated on class, activity, sequence and database design diagrams. Coding and component integration was presented under implementation. Lastly, testing of different functionalities was discussed.

CONCLUSION AND FUTURE WORK

This section aims at giving a final overview of the project by giving a summary of how research questions were answered to fulfill objectives of this study.

Summary

In this study the main objective was to automatically detect road traffic accidents from microblogs by answering the main research question: **"How can we effectively detect and visualize road traffic accidents from microblogs"**. To answer the main question, complimentary questions were asked to gain insights to the objectives of the study as follows:

1) **What is event detection and the state of the art approaches to event detections from microblogs?**

Described in chapter 2, First, text mining was surveyed to understand the concepts in event detection; Text preprocessing, classification, clustering and information extraction. Then, by reviewing the literature on microblogs and event detection whether large scale or small scale from both long and short unstructured text, different techniques were observed and knowledge gaps identified.

2) **What techniques do we use to effectively crawl road traffic accidents events from Twitter?**

By using Twitter Streaming API, Two (2) strategies were implemented; following Twitter traffic accounts from the study area and also using relevant keywords to filter and save traffic accident related tweets as elaborated in chapter 3, section 3.3. Keywords were generated from the most frequent nouns and verbs extracted from a sample 2000 tweets from traffic police account. Over 1.2 million tweets were collected for analysis and testing of the proposed methods.

3) How can we automatically classify road traffic accidents related tweets from non-accident ones?

This has been achieved in chapter 3, section 3.4 by testing different supervised machine learning classification algorithms namely SVM, KNN and Naïve Bayes. From the results, SVM yielded the best accuracy of 90%. Tweets were collected overtime and 1280 were chosen randomly as a training set, with a balanced 640 tweets for positive accident class and 640 tweets for negative non-accident class. A combination of unigrams, bigrams and trigrams with abbreviation dictionary and TFIDF gave best performance while information gain, stemming and stop word removal did not improve the accuracy of the classifier. Abbreviation dictionary showed its importance by increasing the accuracy of SVM by 0.5%.

4) How can location mentions be extracted from tweet content?

In extracting location mentions from tweets content(See chapter 3, section 3.5), a systematic NLP syntactic analysis also known as parsing was proposed and implemented by extracting noun phrases and matching n-grams against a local and external location databases with geocoding capabilities. Further filtering, using preposition and regular expressions was included as sub-modules that identify n-grams and noun phrases with obvious location cues. The proposed method achieves an accuracy of 0.718 on precision and 0.839 on recall.

5) How can we develop a prototype web application system for traffic accident detection with map visualization?

To demonstrate the efficacy of the proposed methods, the researcher developed a web based application, Traffic Accident Detection System (TADS) as explained in chapter 4. Based on tweets from Nairobi the capital city of Kenya. The study area was chosen after passing a predetermined criteria. The application collects, classify and display traffic accidents tweets and locations on a map. Visualization of these accidents on a map was achieved by extracting coordinates (latitudes and longitudes) from location databases. For spatial display on a map' Google Map V3 API was used.

Tools used to develop the prototype include, Python, PHP, MySQL, Twitter bootstrap, jQuery and Google maps API.

Conclusion

Overall the study demonstrated the importance of social media and microblogs in small scale event detection and its usefulness in situation awareness during emergencies.

Road traffic accident is a small scale event that is common on our roads and highways that causes death, injuries and contributes to traffic congestions especially if not responded to in a timely manner. The proposed system prototype for Nairobi Kenya, automatically categorizes accident related tweets with high recall and precision using supervised machine learning techniques, extract location by NLP and visualization of these incidences on a map. This system is useful to road users, traffic management and control and also emergency response departments. Further, with proper integration of an API, it can be used in fleet management systems.

Future work

Due to either limitations introduced by the model or the scope of the research, future work to improve the model and system is as follows:

Design and development of an ontology based information extraction for easier interpretation of traffic tweets related to incidences including accident, traffic status or condition, actors and summarization.

TADS could not detect tweets reporting about the same incidence, hence, through clustering or similarity comparison algorithms, these tweets can be merged as one incident, which can give more accurate report on the number of incidences.

Free form location descriptions from tweet content was not very successful, attributing to the fact that the POS could not tag all tokens correctly, while the regular expressions missed to extract location matches for complex incident descriptions. But with advances in deep text analysis, the performance can be improved.

References

[1] 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*, 2010, pp. 851–860.

[2] H. Achrekar, A. Gandhe, R. Lazarus, Ssu-Hsin Yu, and B. Liu, "Predicting Flu Trends using Twitter data," 2011, pp. 702–707.

[3] N. Wanichayapong, W. Pruthipunyaskul, W. Pattara-Atikom, and P. Chaovalit, "Social-based traffic information extraction and classification," in *ITS Telecommunications (ITST), 2011 11th International Conference on*, 2011, pp. 107–112.

[4] R. Kosala and E. Adi, "Harvesting real-time traffic information from Twitter," *Procedia Eng.*, vol. 50, pp. 1–11, 2012.

[5] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe, "Predicting elections with Twitter: What 140 characters reveal about political sentiment," in *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, 2010, pp. 178–185.

[6] M. S. Gerber, "Predicting crime using Twitter and kernel density estimation," *Decis. Support Syst.*, vol. 61, pp. 115 – 125, 2014.

[7] R. Kelly, "Twitter study reveals interesting results about usage," 2009. [Online]. Available: http://pearanalytics.com/wp-content/uploads/2009/08/Twitter-Study-August-2009.pdf. [Accessed: 07-Oct-2015].

[8] R. Feldman and I. Dagan, "Knowledge Discovery in Textual Databases (KDT).," in *KDD*, 1995, vol. 95, pp. 112–117.

[9] C. D. Manning, P. Raghavan, H. Schütze, and others, *Introduction to information retrieval*, vol. 1. Cambridge university press Cambridge, 2008.

[10] Y. Wilks, "Information extraction as a core language technology," in *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology*, Springer, 1997, pp. 1–9.

[11] Y. Kodratoff, "Knowledge discovery in texts: a definition, and applications," in *Foundations of Intelligent Systems*, Springer, 1999, pp. 16–29.

[12] A. Hotho, A. Nürnberger, and G. Paa\s s, "A Brief Survey of Text Mining.," in *Ldv Forum*, 2005, vol. 20, pp. 19–62.

[13] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and others, "Knowledge Discovery and Data Mining: Towards a Unifying Framework.," in *KDD*, 1996, vol. 96, pp. 82–88.

[14] C. D. Manning and H. Schütze, *Foundations of statistical natural language processing*. MIT press, 1999.

[15] G. Salton, J. Allan, and C. Buckley, "Automatic structuring and retrieval of large text files," *Commun. ACM*, vol. 37, no. 2, pp. 97–108, 1994.

[16] K. E. Lochbaum and L. A. Streeter, "Comparing and combining the effectiveness of latent semantic indexing and the ordinary vector space model for information retrieval," *Inf. Process. Manag.*, vol. 25, no. 6, pp. 665–676, 1989.

[17] R. Zafarani, M. A. Abbasi, and H. Liu, *Social media mining: an introduction*. Cambridge University Press, 2014.

[18] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," in *ECML-98, 10th European Conference on Machine Learning*, 1998, pp. 137–142.

[19] L. A. Ramshaw and M. P. Marcus, "Text chunking using transformation-based learning," *ArXiv Prepr. Cmp-Lg9505040*, 1995.

[20] K. Takeuchi and N. Collier, "Use of Support Vector Machines in Extended Named Entity Recognition," in *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, Stroudsburg, PA, USA, 2002, pp. 1–7.

[21] D. M. Bikel, R. Schwartz, and R. M. Weischedel, "An Algorithm that Learns What's in a Name," *Mach. Learn.*, vol. 34, no. 1, pp. 211–231.

[22] H. M. Wallach, "Conditional random fields: An introduction," *Tech. Rep. CIS*, p. 22, 2004.

[23] A. McCallum, "Efficiently Inducing Features of Conditional Random Fields," in *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, USA, 2003, pp. 403–410.

[24] J.-D. Kim, T. Ohta, Y. Tsuruoka, Y. Tateisi, and N. Collier, "Introduction to the Bio-entity Recognition Task at JNLPBA," in *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications*, Stroudsburg, PA, USA, 2004, pp. 70–75.

[25] M. Efron, "Information search and retrieval in microblogs," *J. Am. Soc. Inf. Sci. Technol.*, vol. 62, no. 6, pp. 996–1008, 2011.

[26] A. Java, X. Song, T. Finin, and B. Tseng, "Why we Twitter: understanding microblogging usage and communities," in *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, 2007, pp. 56–65.

[27] H. Becker, M. Naaman, and L. Gravano, "Beyond Trending Topics: Real-World Event Identification on Twitter.," *ICWSM*, vol. 11, pp. 438–441, 2011.

[28] W. Dou, K. Wang, W. Ribarsky, and M. Zhou, "Event detection in social media data," *IEEE VisWeek Workshop Interact. Vis. Text Anal.-Task Driven Anal. Soc. Media Content*, pp. 971–980, 2012.

[29] J. Allan, *Topic Detection and Tracking: Event-Based Information Organization*. Norwell, MA, USA: Kluwer Academic Publishers, 2002.

[30] J. Allan, R. Papka, and V. Lavrenko, "On-line new event detection and tracking," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 1998, pp. 37–45.

[31] J. Weng and B.-S. Lee, "Event Detection in Twitter.," *ICWSM*, vol. 11, pp. 401–408, 2011.

[32] S. Petrović, M. Osborne, and V. Lavrenko, "Streaming first story detection with application to Twitter," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2010, pp. 181–189.

[33] F. Kunneman and A. van den Bosch, "Event detection in Twitter: A machine-learning approach based on term pivoting," 2014.

[34] E. Benson, A. Haghighi, and R. Barzilay, "Event discovery in social media feeds," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 2011, pp. 389–398.

[35] R. Lee and K. Sumiya, "Measuring Geographical Regularities of Crowd Behaviors for Twitter-based Geo-social Event Detection," in *Proceedings of the 2Nd ACM SIGSPATIAL International Workshop on Location Based Social Networks*, New York, NY, USA, 2010, pp. 1–10.

[36] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling, "Twitterstand: news in tweets," in *Proceedings of the 17th acm sigspatial international conference on advances in geographic information systems*, 2009, pp. 42–51.

[37] T. Cheng and T. Wicks, "Event detection using Twitter: a spatio-temporal approach," 2014.

[38] D. Ramage, S. T. Dumais, and D. J. Liebling, "Characterizing Microblogs with Topic Models.," in *ICWSM*, 2010.

[39] S. Phuvipadawat and T. Murata, "Breaking news detection and tracking in Twitter," in *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, 2010, vol. 3, pp. 120–123.

[40] S. F. L. de Carvalho and others, “Real-time sensing of traffic information in Twitter messages,” 2012.

[41] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011.

[42] A. M. MacEachren, A. Jaiswal, A. C. Robinson, S. Pezanowski, A. Savelyev, P. Mitra, X. Zhang, and J. Blanford, “Senseplace2: GeoTwitter analytics support for situational awareness,” in *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, 2011, pp. 181–190.

[43] K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham, “Evolving GATE to meet new challenges in language engineering,” *Nat. Lang. Eng.*, vol. 10, no. 3–4, pp. 349–373, 2004.

[44] F. Abel, C. Hauff, G.-J. Houben, R. Stronkman, and K. Tao, “Twitcident: fighting fire with information from social web streams,” in *Proceedings of the 21st international conference companion on World Wide Web*, 2012, pp. 305–308.

[45] P. Zhang and Y. Cao, “Estimating the Locations of Emergency Events from Twitter Streams.,” in *ITQM*, 2014, vol. 31, pp. 731–739.

[46] S. M. Paradesi, “Geotagging Tweets Using Their Content.,” in *FLAIRS Conference*, 2011.

[47] Z. Cheng, J. Caverlee, and K. Lee, “You are where you tweet: a content-based approach to geo-locating Twitter users,” in *Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010, pp. 759–768.

[48] “Twitter,” 2015. [Online]. Available: http://www.Twitter.com.

[49] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Trans. Intell. Syst. Technol. TIST*, vol. 2, no. 3, p. 27, 2011.

[50] S. R. Garner, “WEKA: The Waikato Environment for Knowledge Analysis,” in *In Proc. of the New Zealand Computer Science Research Students Conference*, 1995, pp. 57–64.

[51] J. Langford, “The Cross Validation Problem,” in *Learning Theory: 18th Annual Conference on Learning Theory, COLT 2005, Bertinoro, Italy, June 27-30, 2005. Proceedings*, P. Auer and R. Meir, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 687–688.

[52] J. R. Finkel, T. Grenager, and C. Manning, “Incorporating non-local information into information extraction systems by gibbs sampling,” in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, 2005, pp. 363–370.

[53] G. B., Joern Kottmann, Jason Baldridge Thomas Morton, “OpenNLP: A Java-based NLP Toolkit,” 2005.

[54] A. Ritter, S. Clark, O. Etzioni, and others, “Named entity recognition in tweets: an experimental study,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011, pp. 1524–1534.

[55] J. Perkins, *Python 3 Text Processing with NLTK 3 Cookbook*. Packt Publishing Ltd, 2014.

[56] “GeoNames API,” 2015. [Online]. Available: http://www.geonames.org. [Accessed: 25-Oct-2015].

[57] A. Cohen, “FuzzyWuzzy: Fuzzy String Matching in Python,” 07-Aug-2011. .

[58] “Google Map API,” 2015. [Online]. Available: https://developers.google.com/maps/.

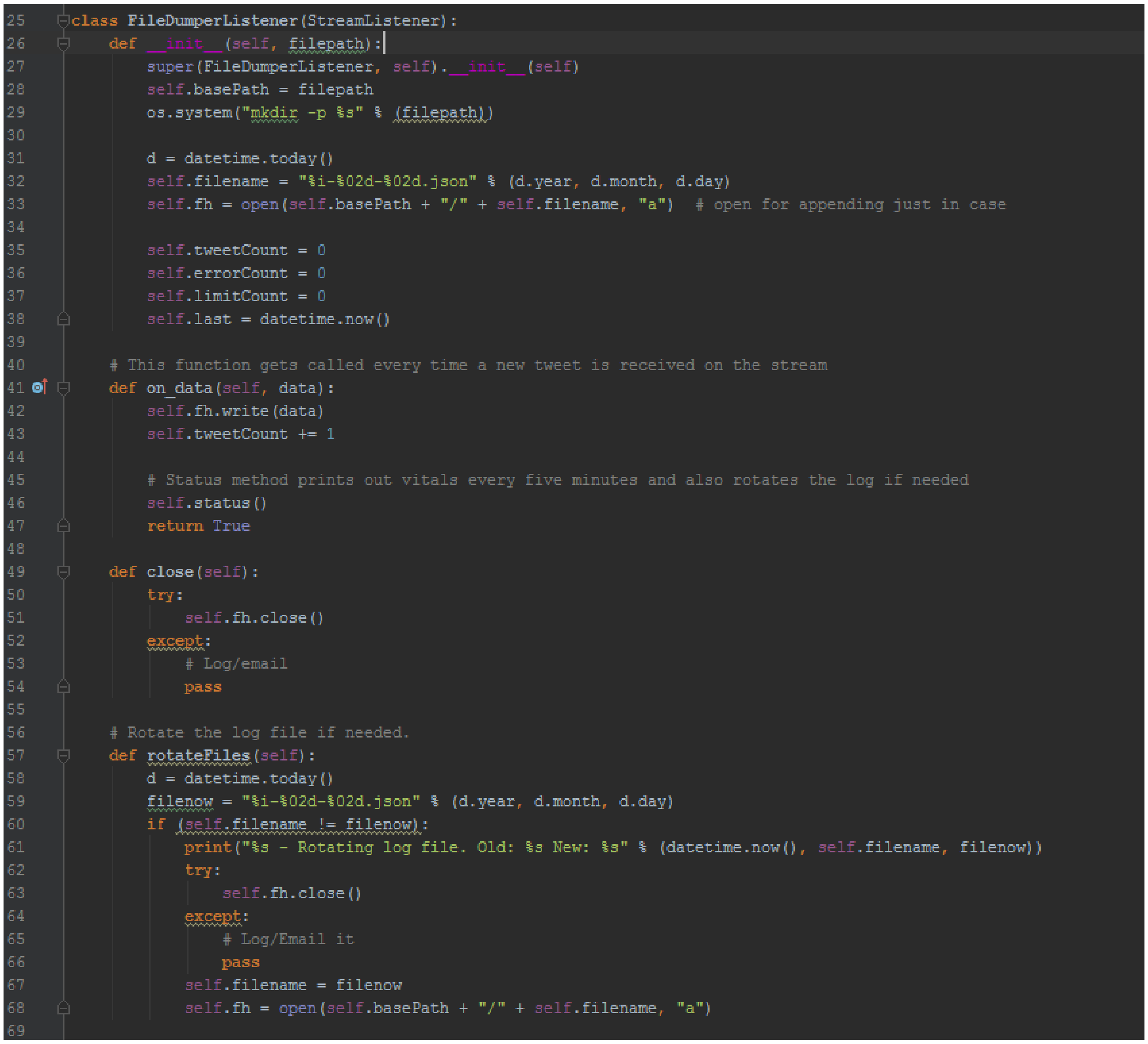
Appendices

Appendix A: Tweet JSON Metadata

```
{
        "contributors" : null,
        "truncated" : false,
        "text" : "@Ma3Route accident globe underpass matatu and saloon car.police on site.",
        "is_quote_status" : false,
        "in_reply_to_status_id" : null,
        "id" : NumberLong("666469583461687296"),
        "favorite_count" : 0,
        "source" : "<a href=\"https://Twitter.com/download/android\" rel=\"nofollow\">Twitter for  Android</a>",
        "retweeted" : false,
        "coordinates" : null,
        "timestamp_ms" : "1447733705760",
        "entities" : {
                "user_mentions" : [
                        {
                                "id" : 585372692,
                                "indices" : [
                                        0,
                                        9
                                ],
                                "id_str" : "585372692",
                                "screen_name" : "Ma3Route",
                                "name" : "Ma3Route"
                        }
                ],
                "symbols" : [ ],
                "hashtags" : [ ],
                "urls" : [ ]
        },
        "in_reply_to_screen_name" : "Ma3Route",
        "id_str" : "666469583461687296",
        "retweet_count" : 0,
        "in_reply_to_user_id" : 585372692,
        "favorited" : false,
        "user" : {
                "follow_request_sent" : null,
                "profile_use_background_image" : true,
                "default_profile_image" : false,
                "id" : 405384865,
                "verified" : false,
                "profile_image_url_https" :
"https://pbs.twimg.com/profile_images/648579203311529985/32i1hWsy_normal.jpg",
                "profile_sidebar_fill_color" : "DDEEF6",
                "profile_text_color" : "333333",
                "followers_count" : 536,
                "profile_sidebar_border_color" : "C0DEED",
                "id_str" : "405384865",
                "profile_background_color" : "C0DEED",
                "listed_count" : 1,
                "profile_background_image_url_https" : "https://abs.twimg.com/images/themes/theme1/bg.png",
                "utc_offset" : 10800,
            "statuses_count" : 4601,
            "description" : "A person who loves life,loves peace.",
            "friends_count" : 721,
            "location" : "Nairobi,Kenya",
            "profile_link_color" : "0084B4",
            "profile_image_url" : "http://pbs.twimg.com/profile_images/648579203311529985/32i1hWsy_normal.jpg",
            "following" : null,
            "geo_enabled" : true,
            "profile_banner_url" : "https://pbs.twimg.com/profile_banners/405384865/1388838159",
            "profile_background_image_url" : "http://abs.twimg.com/images/themes/theme1/bg.png",
            "name" : "Vincent Awange",
            "lang" : "en",
            "profile_background_tile" : false,
            "favourites_count" : 1279,
            "screen_name" : "VincentAwange",
            "notifications" : null,
            "url" : null,
            "created_at" : "Sat Nov 05 07:26:21 +0000 2011",
            "contributors_enabled" : false,
            "time_zone" : "Nairobi",
            "protected" : false,
            "default_profile" : true,
            "is_translator" : false
      },
      "geo" : null,
      "in_reply_to_user_id_str" : "585372692",
      "lang" : "en",
      "created_at" : "Tue Nov 17 04:15:05 +0000 2015",
      "filter_level" : "low",
      "in_reply_to_status_id_str" : null,
      "place" : null
}
```

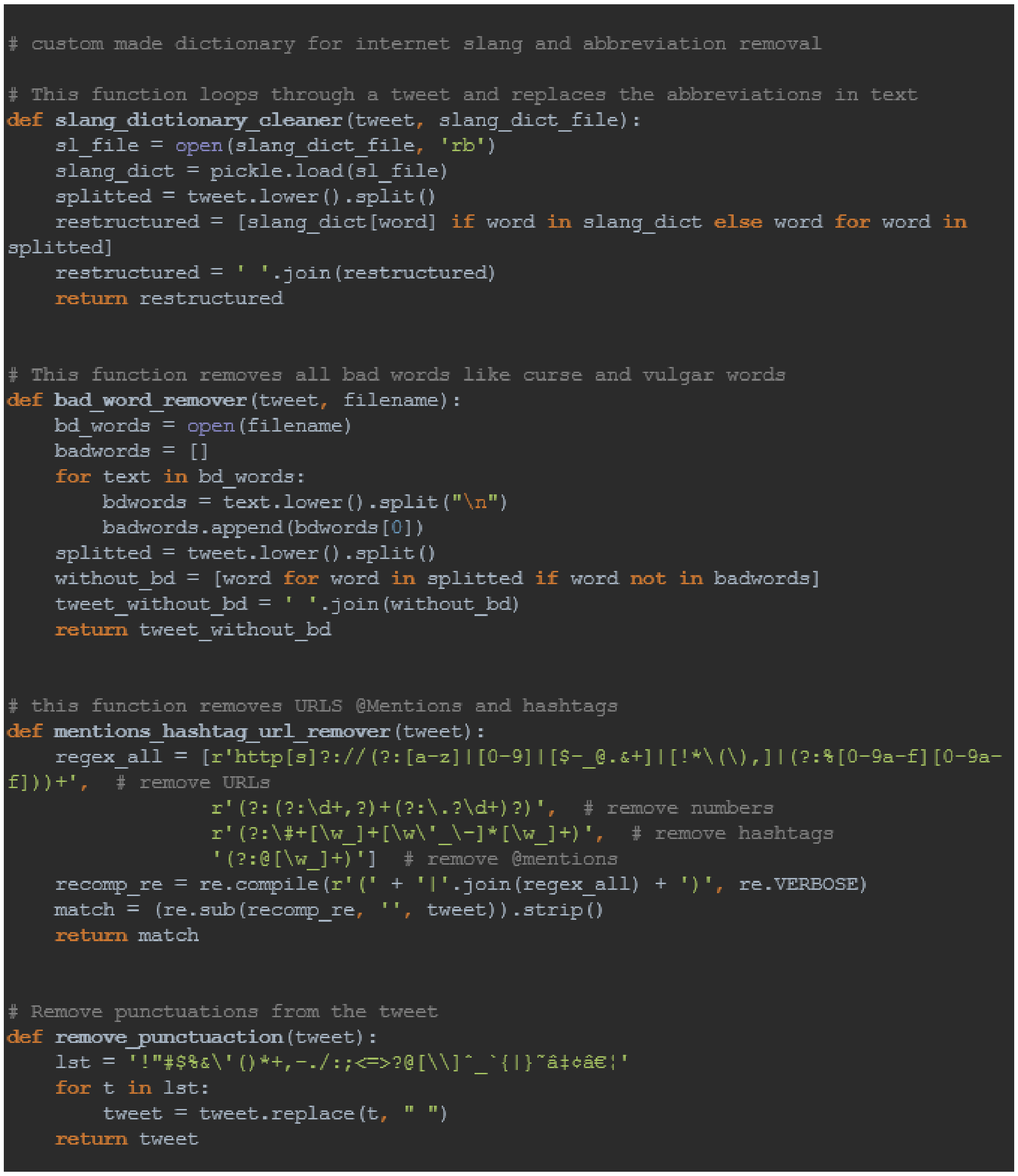
Appendix B: Sample Codes: Back End

1. Tweets acquisition through streaming API



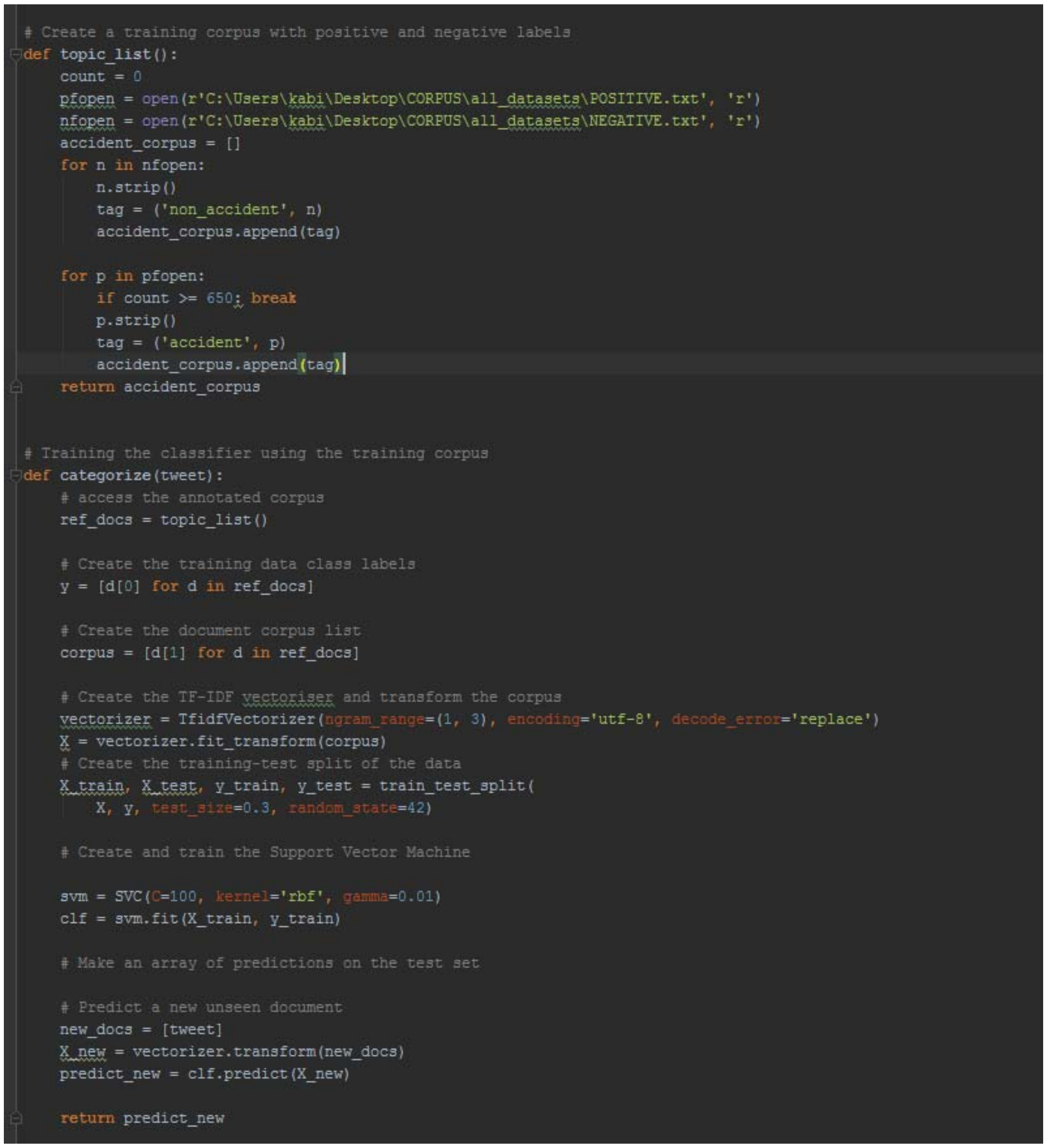
```
class FileDumperListener(StreamListener):
    def __init__(self, filepath):
        super(FileDumperListener, self).__init__(self)
        self.basePath = filepath
        os.system("mkdir -p %s" % (filepath))

        d = datetime.today()
        self.filename = "%i-%02d-%02d.json" % (d.year, d.month, d.day)
        self.fh = open(self.basePath + "/" + self.filename, "a")  # open for appending just in case

        self.tweetCount = 0
        self.errorCount = 0
        self.limitCount = 0
        self.last = datetime.now()

    # This function gets called every time a new tweet is received on the stream
    def on_data(self, data):
        self.fh.write(data)
        self.tweetCount += 1

        # Status method prints out vitals every five minutes and also rotates the log if needed
        self.status()
        return True

    def close(self):
        try:
            self.fh.close()
        except:
            # Log/email
            pass

    # Rotate the log file if needed.
    def rotateFiles(self):
        d = datetime.today()
        filenow = "%i-%02d-%02d.json" % (d.year, d.month, d.day)
        if (self.filename != filenow):
            print("%s - Rotating log file. Old: %s New: %s" % (datetime.now(), self.filename, filenow))
            try:
                self.fh.close()
            except:
                # Log/Email it
                pass
            self.filename = filenow
            self.fh = open(self.basePath + "/" + self.filename, "a")
```

2. Preprocessor Functions



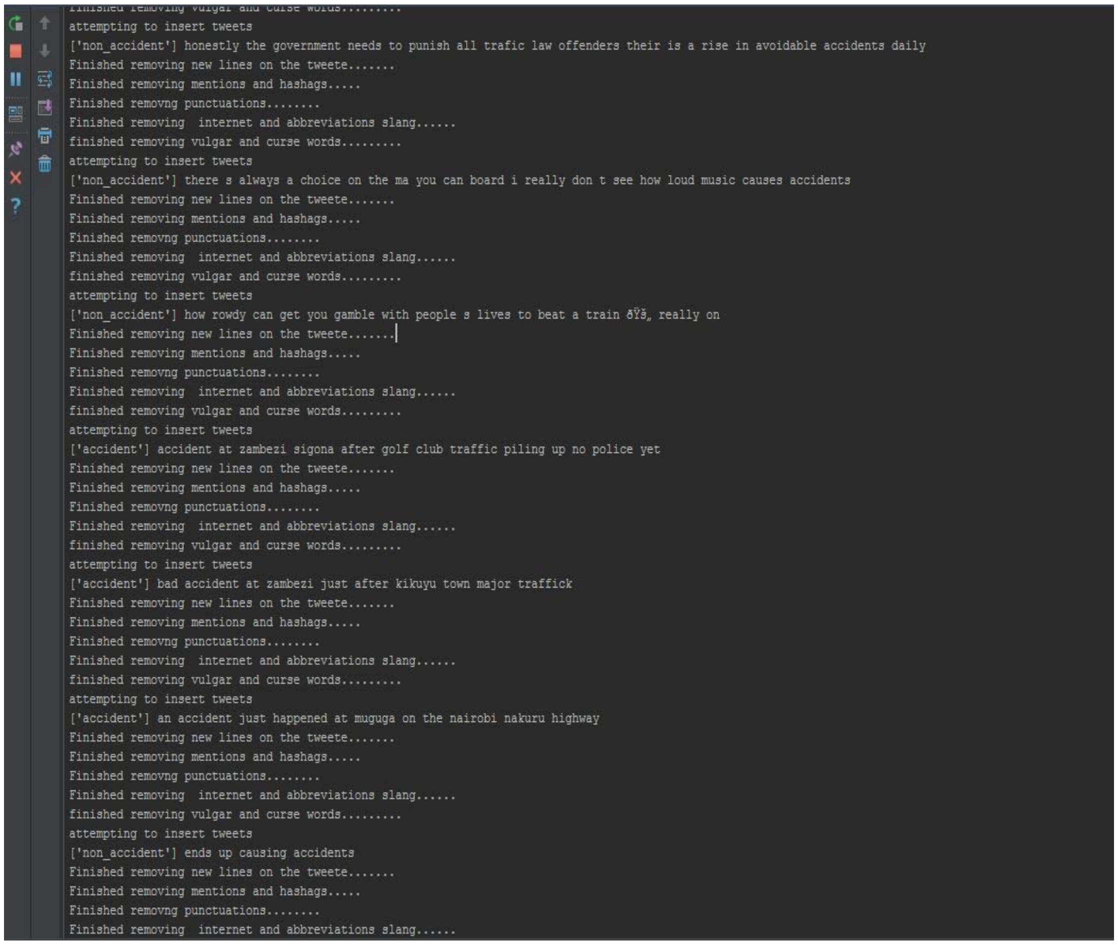
```
# custom made dictionary for internet slang and abbreviation removal

# This function loops through a tweet and replaces the abbreviations in text
def slang_dictionary_cleaner(tweet, slang_dict_file):
    sl_file = open(slang_dict_file, 'rb')
    slang_dict = pickle.load(sl_file)
    splitted = tweet.lower().split()
    restructured = [slang_dict[word] if word in slang_dict else word for word in
splitted]
    restructured = ' '.join(restructured)
    return restructured


# This function removes all bad words like curse and vulgar words
def bad_word_remover(tweet, filename):
    bd_words = open(filename)
    badwords = []
    for text in bd_words:
        bdwords = text.lower().split("\n")
        badwords.append(bdwords[0])
    splitted = tweet.lower().split()
    without_bd = [word for word in splitted if word not in badwords]
    tweet_without_bd = ' '.join(without_bd)
    return tweet_without_bd


# this function removes URLS @Mentions and hashtags
def mentions_hashtag_url_remover(tweet):
    regex_all = [r'http[s]?://(?:[a-z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-f][0-9a-
f]))+',  # remove URLs
                 r'(?:(?:\d+,?)+(?:\.?\d+)?)',  # remove numbers
                 r'(?:\#+[\w_]+[\w\'_\-]*[\w_]+)',  # remove hashtags
                 '(?:@[\w_]+)']  # remove @mentions
    recomp_re = re.compile(r'(' + '|'.join(regex_all) + ')', re.VERBOSE)
    match = (re.sub(recomp_re, '', tweet)).strip()
    return match


# Remove punctuations from the tweet
def remove_punctuaction(tweet):
    lst = '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~â‡¢â€¦'
    for t in lst:
        tweet = tweet.replace(t, " ")
    return tweet
```

3. Classifier Functions



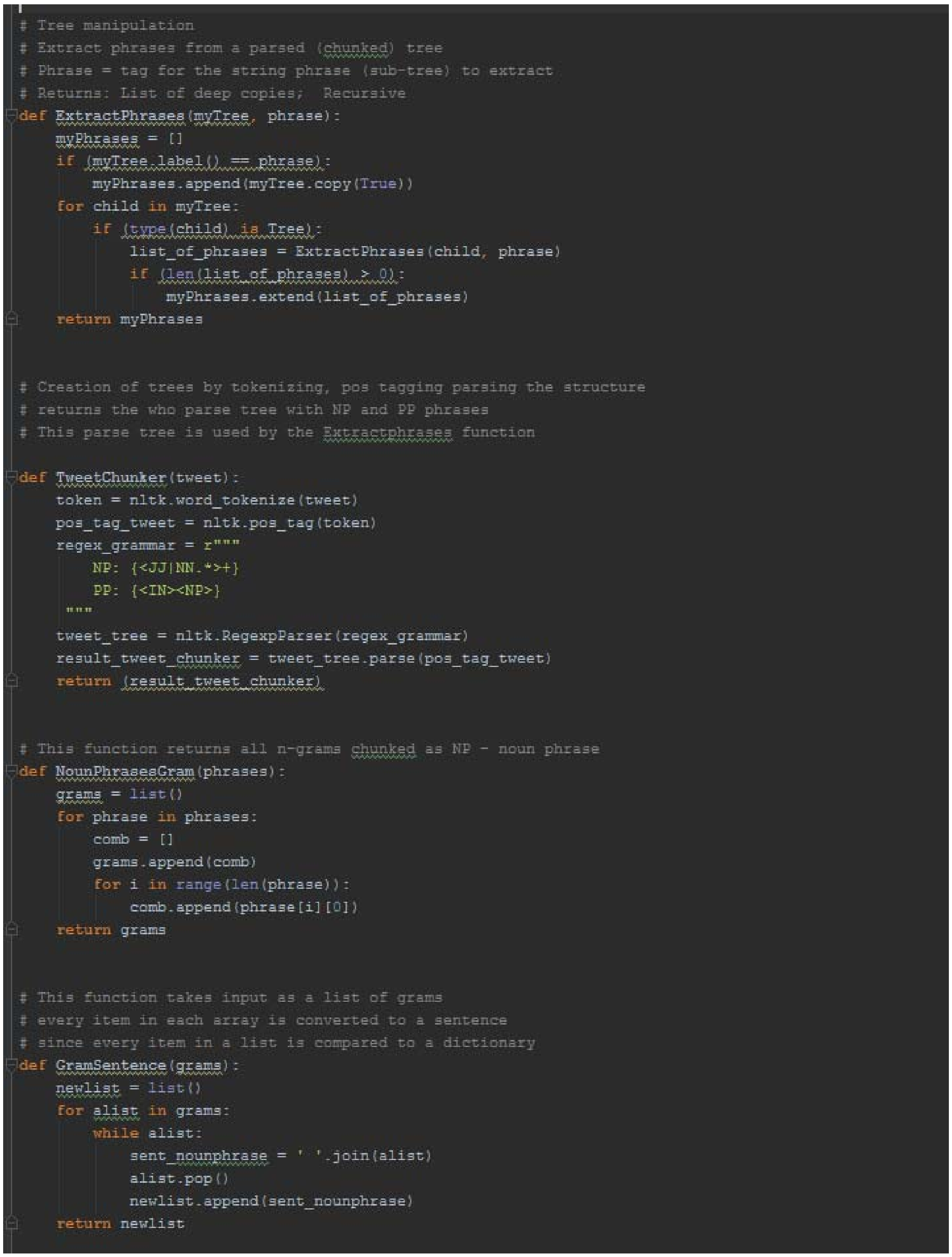
```python
# Create a training corpus with positive and negative labels
def topic_list():
    count = 0
    pfopen = open(r'C:\Users\kabi\Desktop\CORPUS\all_datasets\POSITIVE.txt', 'r')
    nfopen = open(r'C:\Users\kabi\Desktop\CORPUS\all_datasets\NEGATIVE.txt', 'r')
    accident_corpus = []
    for n in nfopen:
        n.strip()
        tag = ('non_accident', n)
        accident_corpus.append(tag)

    for p in pfopen:
        if count >= 650: break
        p.strip()
        tag = ('accident', p)
        accident_corpus.append(tag)
    return accident_corpus


# Training the classifier using the training corpus
def categorize(tweet):
    # access the annotated corpus
    ref_docs = topic_list()

    # Create the training data class labels
    y = [d[0] for d in ref_docs]

    # Create the document corpus list
    corpus = [d[1] for d in ref_docs]

    # Create the TF-IDF vectoriser and transform the corpus
    vectorizer = TfidfVectorizer(ngram_range=(1, 3), encoding='utf-8', decode_error='replace')
    X = vectorizer.fit_transform(corpus)
    # Create the training-test split of the data
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.3, random_state=42)

    # Create and train the Support Vector Machine

    svm = SVC(C=100, kernel='rbf', gamma=0.01)
    clf = svm.fit(X_train, y_train)

    # Make an array of predictions on the test set

    # Predict a new unseen document
    new_docs = [tweet]
    X_new = vectorizer.transform(new_docs)
    predict_new = clf.predict(X_new)

    return predict_new
```

4. Labelling output



```
finished removing vulgar and curse words.........
attempting to insert tweets
['non_accident'] honestly the government needs to punish all trafic law offenders their is a rise in avoidable accidents daily
Finished removing new lines on the tweete.......
Finished removing mentions and hashags.....
Finished removng punctuations........
Finished removing  internet and abbreviations slang......
finished removing vulgar and curse words.........
attempting to insert tweets
['non_accident'] there s always a choice on the ma you can board i really don t see how loud music causes accidents
Finished removing new lines on the tweete.......
Finished removing mentions and hashags.....
Finished removng punctuations........
Finished removing  internet and abbreviations slang......
finished removing vulgar and curse words.........
attempting to insert tweets
['non_accident'] how rowdy can get you gamble with people s lives to beat a train ðŸš„ really on
Finished removing new lines on the tweete.......
Finished removing mentions and hashags.....
Finished removng punctuations........
Finished removing  internet and abbreviations slang......
finished removing vulgar and curse words.........
attempting to insert tweets
['accident'] accident at zambezi sigona after golf club traffic piling up no police yet
Finished removing new lines on the tweete.......
Finished removing mentions and hashags.....
Finished removng punctuations........
Finished removing  internet and abbreviations slang......
finished removing vulgar and curse words.........
attempting to insert tweets
['accident'] bad accident at zambezi just after kikuyu town major traffick
Finished removing new lines on the tweete.......
Finished removing mentions and hashags.....
Finished removng punctuations........
Finished removing  internet and abbreviations slang......
finished removing vulgar and curse words.........
attempting to insert tweets
['accident'] an accident just happened at muguga on the nairobi nakuru highway
Finished removing new lines on the tweete.......
Finished removing mentions and hashags.....
Finished removng punctuations........
Finished removing  internet and abbreviations slang......
finished removing vulgar and curse words.........
attempting to insert tweets
['non_accident'] ends up causing accidents
Finished removing new lines on the tweete.......
Finished removing mentions and hashags.....
Finished removng punctuations........
Finished removing  internet and abbreviations slang......
```

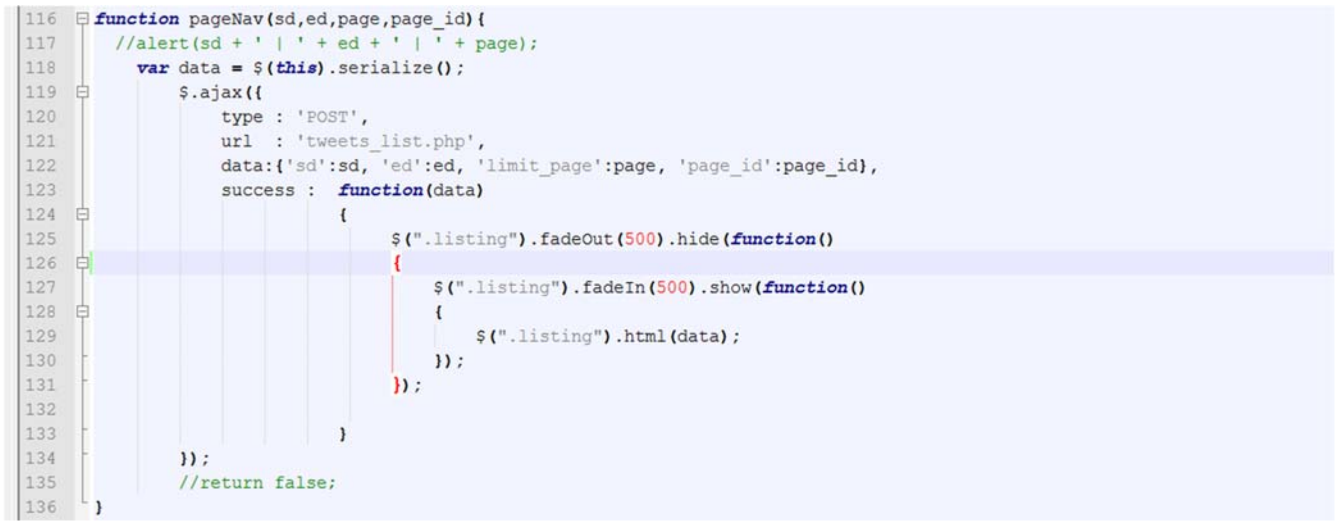
5. **Location extraction functions**



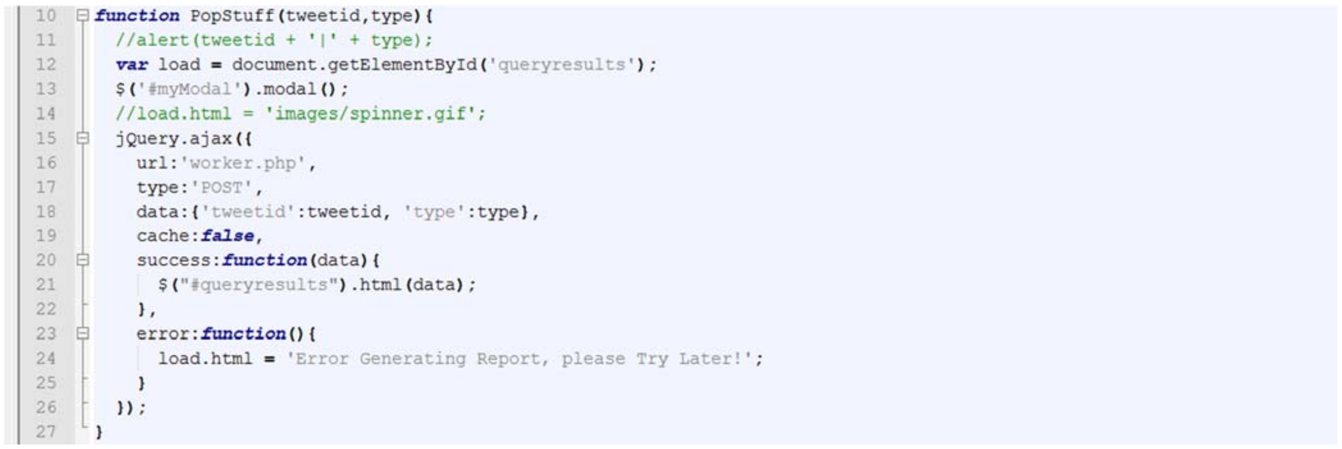
```python
# Tree manipulation
# Extract phrases from a parsed (chunked) tree
# Phrase = tag for the string phrase (sub-tree) to extract
# Returns: List of deep copies;  Recursive
def ExtractPhrases(myTree, phrase):
    myPhrases = []
    if (myTree.label() == phrase):
        myPhrases.append(myTree.copy(True))
    for child in myTree:
        if (type(child) is Tree):
            list_of_phrases = ExtractPhrases(child, phrase)
            if (len(list_of_phrases) > 0):
                myPhrases.extend(list_of_phrases)
    return myPhrases


# Creation of trees by tokenizing, pos tagging parsing the structure
# returns the who parse tree with NP and PP phrases
# This parse tree is used by the Extractphrases function

def TweetChunker(tweet):
    token = nltk.word_tokenize(tweet)
    pos_tag_tweet = nltk.pos_tag(token)
    regex_grammar = r"""
        NP: {<JJ|NN.*>+}
        PP: {<IN><NP>}
     """
    tweet_tree = nltk.RegexpParser(regex_grammar)
    result_tweet_chunker = tweet_tree.parse(pos_tag_tweet)
    return (result_tweet_chunker)


# This function returns all n-grams chunked as NP - noun phrase
def NounPhrasesGram(phrases):
    grams = list()
    for phrase in phrases:
        comb = []
        grams.append(comb)
        for i in range(len(phrase)):
            comb.append(phrase[i][0])
    return grams


# This function takes input as a list of grams
# every item in each array is converted to a sentence
# since every item in a list is compared to a dictionary
def GramSentence(grams):
    newlist = list()
    for alist in grams:
        while alist:
            sent_nounphrase = ' '.join(alist)
            alist.pop()
            newlist.append(sent_nounphrase)
    return newlist
```

Appendix C: Sample Codes: Front End

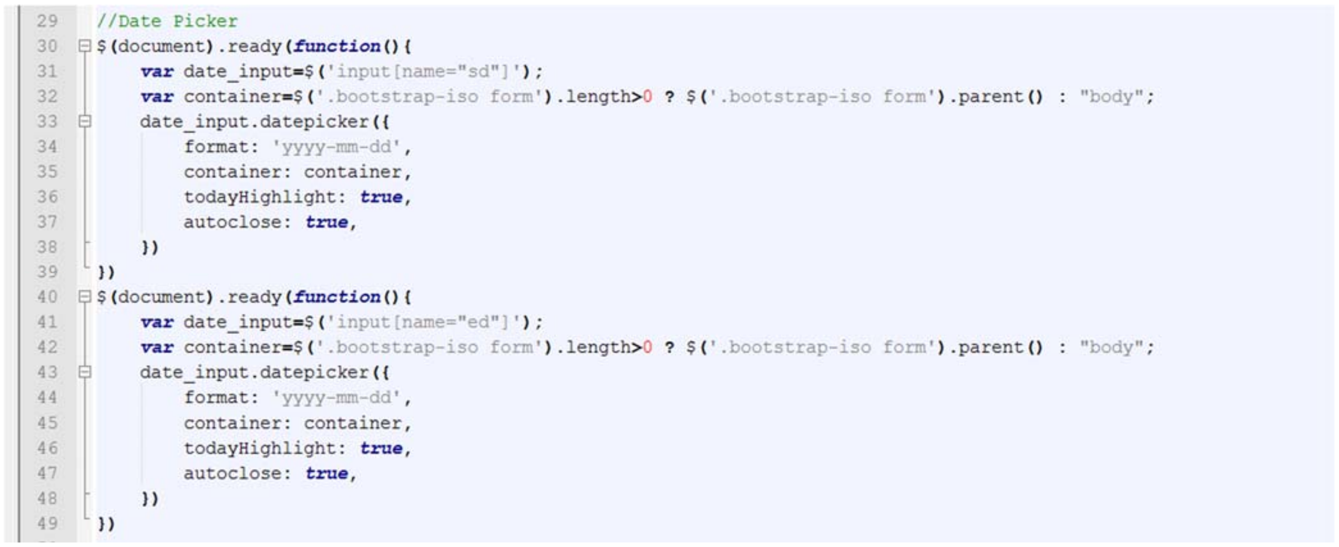
1. **JQUERY and Ajax methods –** Pass data around the various HTML sections allowing smooth responsiveness



```
function pageNav(sd,ed,page,page_id){
  //alert(sd + ' | ' + ed + ' | ' + page);
    var data = $(this).serialize();
        $.ajax({
            type : 'POST',
            url  : 'tweets_list.php',
            data:{'sd':sd, 'ed':ed, 'limit_page':page, 'page_id':page_id},
            success :  function(data)
                       {
                            $(".listing").fadeOut(500).hide(function()
                            {
                                $(".listing").fadeIn(500).show(function()
                                {
                                    $(".listing").html(data);
                                });
                            });

                       }
        });
        //return false;
}
```

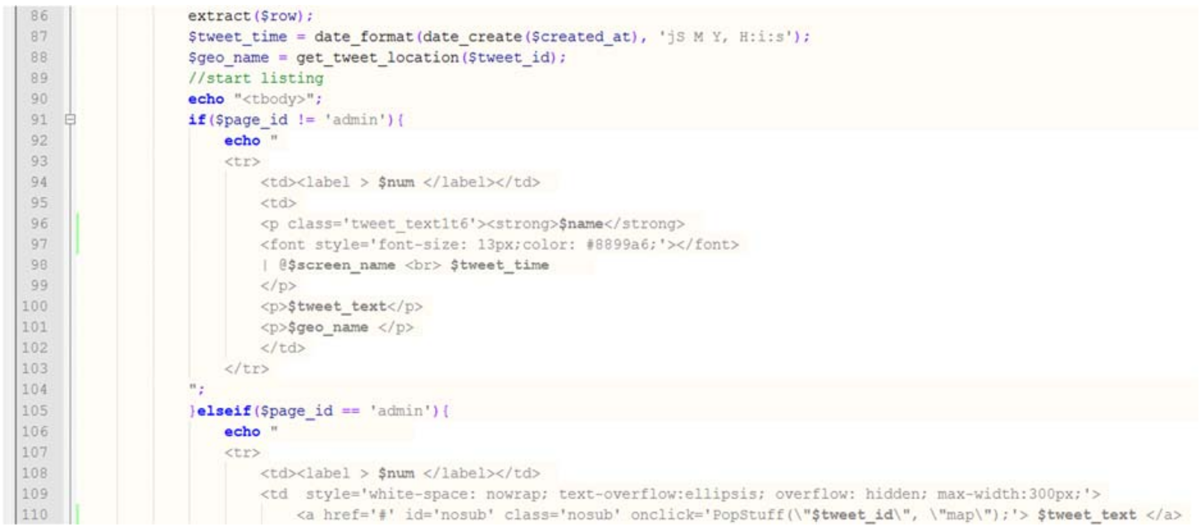


```
function PopStuff(tweetid,type){
  //alert(tweetid + '|' + type);
  var load = document.getElementById('queryresults');
  $('#myModal').modal();
  //load.html = 'images/spinner.gif';
  jQuery.ajax({
    url:'worker.php',
    type:'POST',
    data:{'tweetid':tweetid, 'type':type},
    cache:false,
    success:function(data){
      $("#queryresults").html(data);
    },
    error:function(){
      load.html = 'Error Generating Report, please Try Later!';
    }
  });
}
```

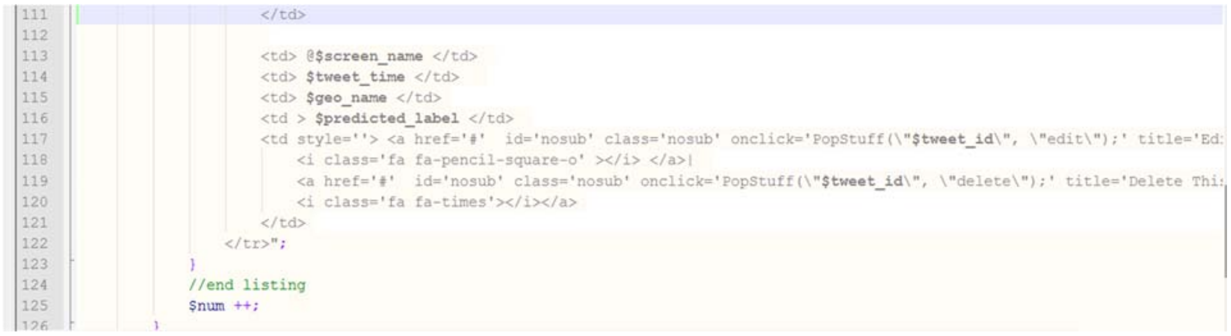


```
//Date Picker
$(document).ready(function(){
    var date_input=$('input[name="sd"]');
    var container=$('.bootstrap-iso form').length>0 ? $('.bootstrap-iso form').parent() : "body";
    date_input.datepicker({
        format: 'yyyy-mm-dd',
        container: container,
        todayHighlight: true,
        autoclose: true,
    })
})
$(document).ready(function(){
    var date_input=$('input[name="ed"]');
    var container=$('.bootstrap-iso form').length>0 ? $('.bootstrap-iso form').parent() : "body";
    date_input.datepicker({
        format: 'yyyy-mm-dd',
        container: container,
        todayHighlight: true,
        autoclose: true,
    })
})
```

2. Accidents Listing after a query Search

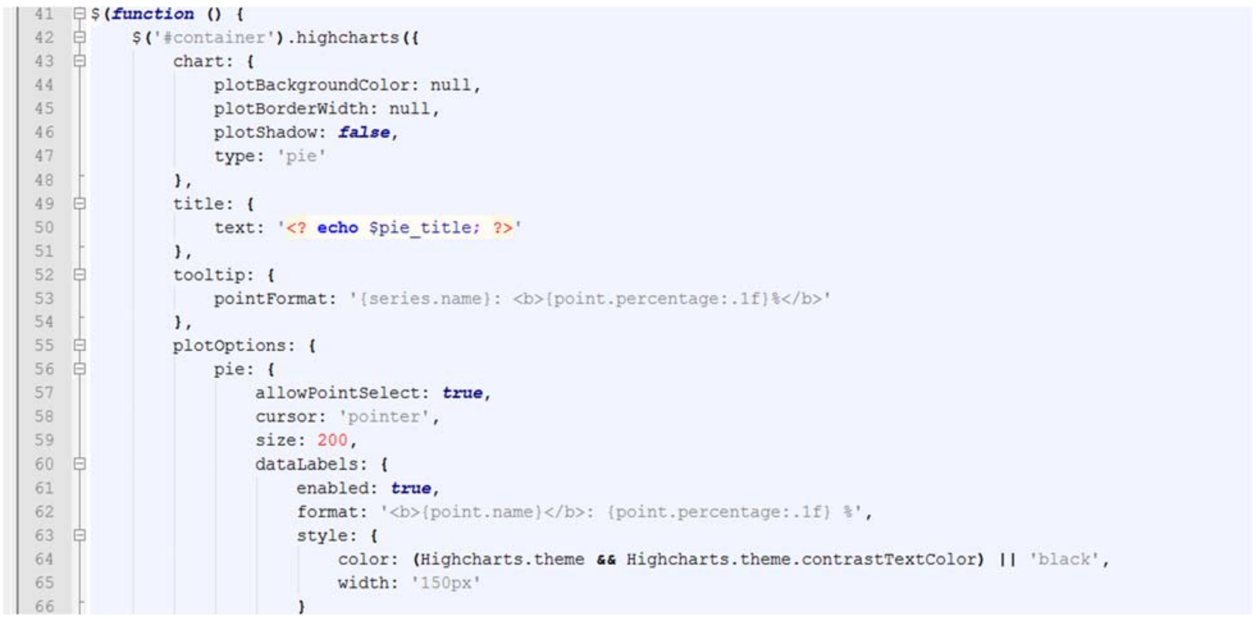


```
extract($row);
$tweet_time = date_format(date_create($created_at), 'jS M Y, H:i:s');
$geo_name = get_tweet_location($tweet_id);
//start listing
echo "<tbody>";
if($page_id != 'admin'){
    echo "
    <tr>
        <td><label > $num </label></td>
        <td>
        <p class='tweet_text1t6'><strong>$name</strong>
        <font style='font-size: 13px;color: #8899a6;'></font>
        | @$screen_name <br> $tweet_time
        </p>
        <p>$tweet_text</p>
        <p>$geo_name </p>
        </td>
    </tr>
";
}elseif($page_id == 'admin'){
    echo "
    <tr>
        <td><label > $num </label></td>
        <td  style='white-space: nowrap; text-overflow:ellipsis; overflow: hidden; max-width:300px;'>
            <a href='#' id='nosub' class='nosub' onclick='PopStuff(\"$tweet_id\", \"map\");'> $tweet_text </a>
```

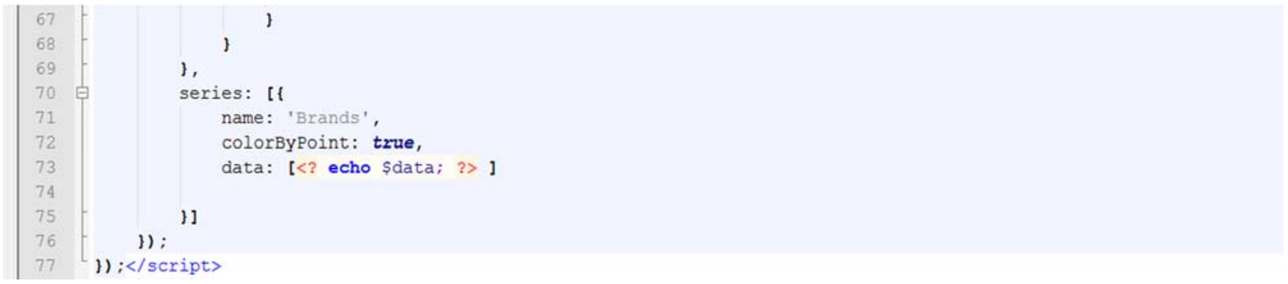


```
        </td>

        <td> @$screen_name </td>
        <td> $tweet_time </td>
        <td> $geo_name </td>
        <td > $predicted_label </td>
        <td style=''> <a href='#'  id='nosub' class='nosub' onclick='PopStuff(\"$tweet_id\", \"edit\");' title='Ed
            <i class='fa fa-pencil-square-o' ></i> </a>|
            <a href='#'  id='nosub' class='nosub' onclick='PopStuff(\"$tweet_id\", \"delete\");' title='Delete Thi
            <i class='fa fa-times'></i></a>
        </td>
    </tr>";
}
//end listing
$num ++;
```

3. Piechart Drawer for analytics page

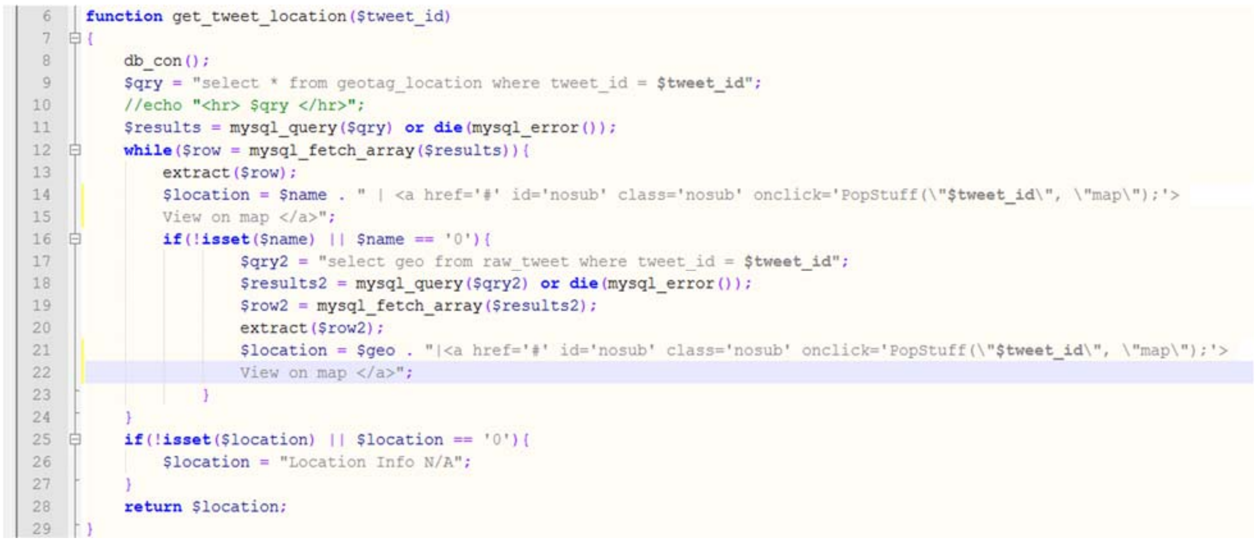


```
$(function () {
    $('#container').highcharts({
        chart: {
            plotBackgroundColor: null,
            plotBorderWidth: null,
            plotShadow: false,
            type: 'pie'
        },
        title: {
            text: '<? echo $pie_title; ?>'
        },
        tooltip: {
            pointFormat: '{series.name}: <b>{point.percentage:.1f}%</b>'
        },
        plotOptions: {
            pie: {
                allowPointSelect: true,
                cursor: 'pointer',
                size: 200,
                dataLabels: {
                    enabled: true,
                    format: '<b>{point.name}</b>: {point.percentage:.1f} %',
                    style: {
                        color: (Highcharts.theme && Highcharts.theme.contrastTextColor) || 'black',
                        width: '150px'
                    }
```

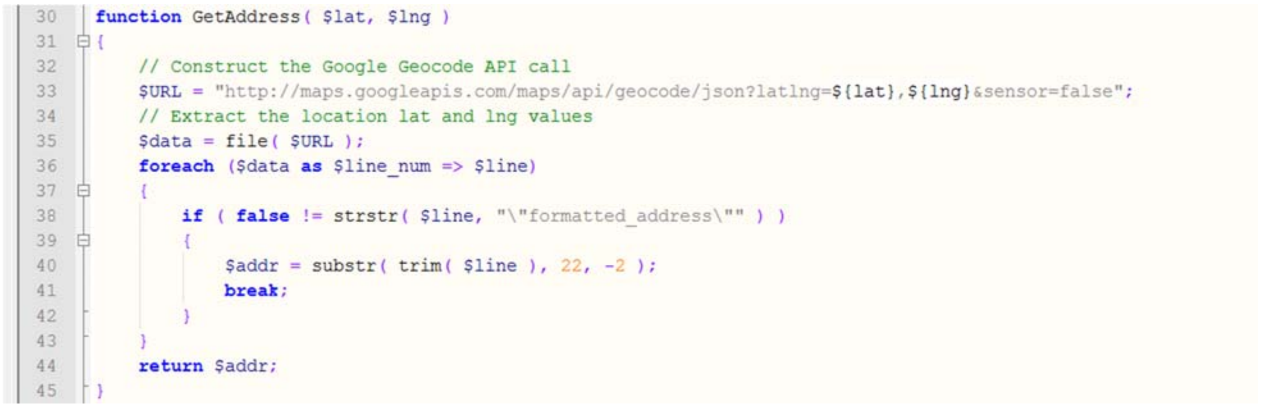


```
                }
            }
        },
        series: [{
            name: 'Brands',
            colorByPoint: true,
            data: [<? echo $data; ?> ]

        }]
    });
});</script>
```

2. Location fucntions to display on Google maps



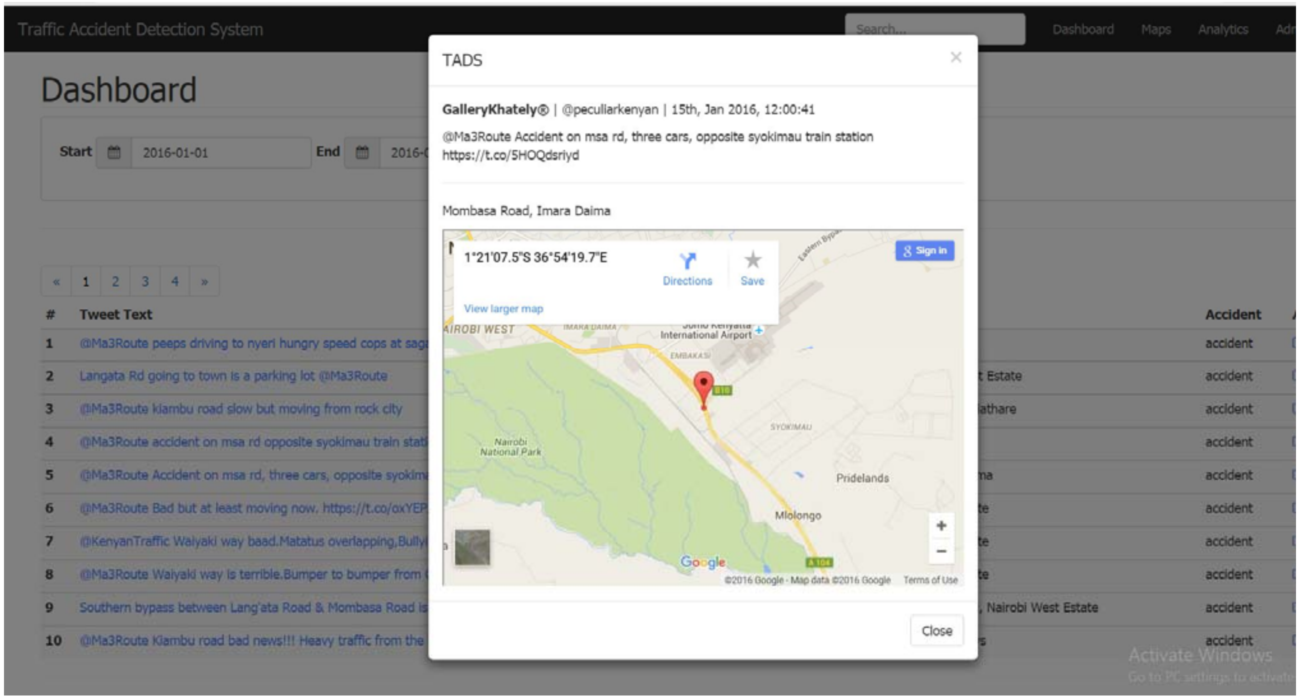
```
function get_tweet_location($tweet_id)
{
    db_con();
    $qry = "select * from geotag_location where tweet_id = $tweet_id";
    //echo "<hr> $qry </hr>";
    $results = mysql_query($qry) or die(mysql_error());
    while($row = mysql_fetch_array($results)){
        extract($row);
        $location = $name . " | <a href='#' id='nosub' class='nosub' onclick='PopStuff(\"$tweet_id\", \"map\");'>
        View on map </a>";
        if(!isset($name) || $name == '0'){
                $qry2 = "select geo from raw_tweet where tweet_id = $tweet_id";
                $results2 = mysql_query($qry2) or die(mysql_error());
                $row2 = mysql_fetch_array($results2);
                extract($row2);
                $location = $geo . "|<a href='#' id='nosub' class='nosub' onclick='PopStuff(\"$tweet_id\", \"map\");'>
                View on map </a>";
            }
    }
    if(!isset($location) || $location == '0'){
        $location = "Location Info N/A";
    }
    return $location;
}
```



```
function GetAddress( $lat, $lng )
{
    // Construct the Google Geocode API call
    $URL = "http://maps.googleapis.com/maps/api/geocode/json?latlng=${lat},${lng}&sensor=false";
    // Extract the location lat and lng values
    $data = file( $URL );
    foreach ($data as $line_num => $line)
    {
        if ( false != strstr( $line, "\"formatted_address\"" ) )
        {
            $addr = substr( trim( $line ), 22, -2 );
            break;
        }
    }
    return $addr;
}
```

Appendix D: Snapshots of TADS

1. Dashboard



GalleryKhately® | @peculiarkenyan | 15th, Jan 2016, 12:00:41

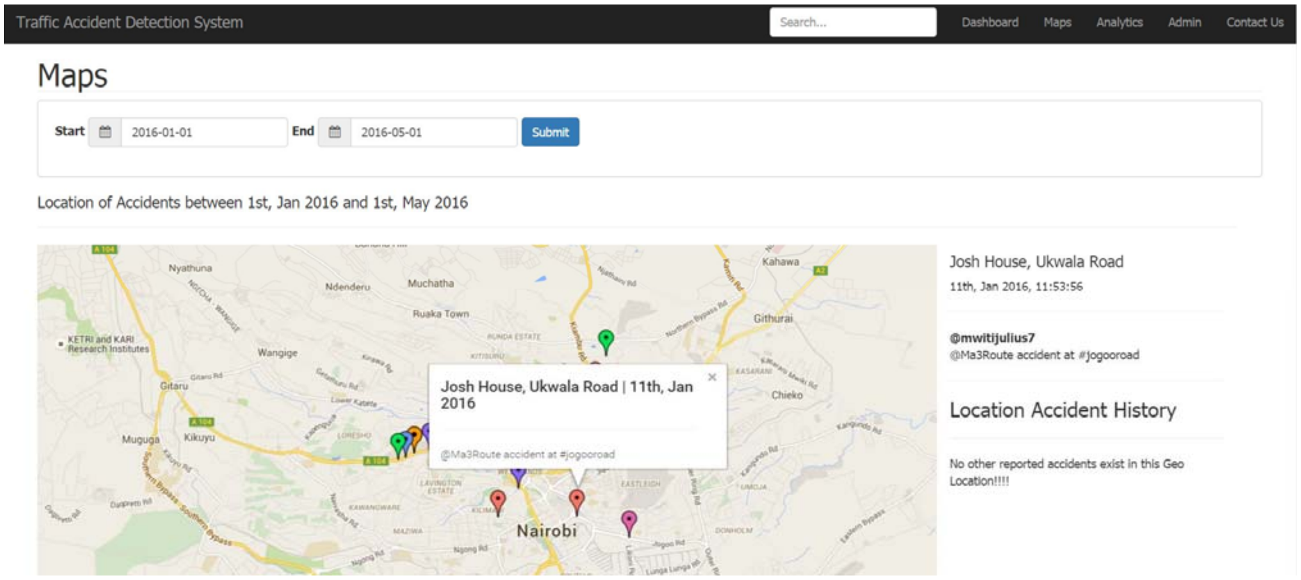
@Ma3Route Accident on msa rd, three cars, opposite syokimau train station https://t.co/5HOQdsriyd

Mombasa Road, Imara Daima

Start 2016-01-01 End 2016-

#	Tweet Text		Accident
1	@Ma3Route peeps driving to nyeri hungry speed cops at sag		accident
2	Langata Rd going to town is a parking lot @Ma3Route	t Estate	accident
3	@Ma3Route kiambu road slow but moving from rock city	athare	accident
4	@Ma3Route accident on msa rd opposite syokimau train stat		accident
5	@Ma3Route Accident on msa rd, three cars, opposite syokim	ma	accident
6	@Ma3Route Bad but at least moving now. https://t.co/oxYEP	te	accident
7	@KenyanTraffic Waiyaki way baad.Matatus overlapping,Bully	te	accident
8	@Ma3Route Waiyaki way is terrible.Bumper to bumper from	te	accident
9	Southern bypass between Lang'ata Road & Mombasa Road is	, Nairobi West Estate	accident
10	@Ma3Route Kiambu road bad news!!! Heavy traffic from the	s	accident

2. Accident Location



Start 2016-01-01 End 2016-05-01

Location of Accidents between 1st, Jan 2016 and 1st, May 2016

Josh House, Ukwala Road | 11th, Jan 2016

@Ma3Route accident at #jogooroad

Josh House, Ukwala Road

11th, Jan 2016, 11:53:56

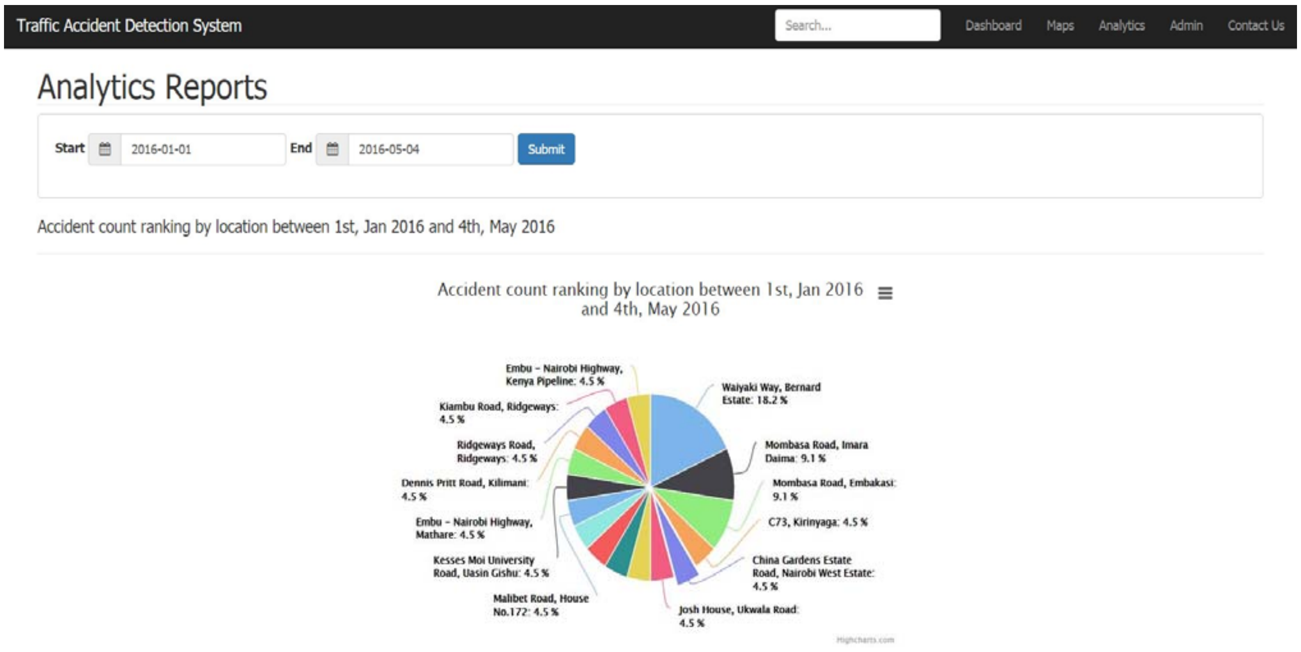
@mwitijulius7

@Ma3Route accident at #jogooroad

Location Accident History

No other reported accidents exist in this Geo Location!!!!

3. Simple Analytics on locations


Traffic Accident Detection System
Search...
Dashboard
Maps
Analytics
Admin
Contact Us
Submit

Analytics Reports

Start 2016-01-01 End 2016-05-04

Accident count ranking by location between 1st, Jan 2016 and 4th, May 2016

Accident count ranking by location between 1st, Jan 2016 and 4th, May 2016
Embu – Nairobi Highway, Kenya Pipeline: 4.5 %
Waiyaki Way, Bernard Estate: 18.2 %
Kiambu Road, Ridgeways: 4.5 %
Ridgeways Road, Ridgeways: 4.5 %
Mombasa Road, Imara Daima: 9.1 %
Dennis Pritt Road, Kilimani: 4.5 %
Mombasa Road, Embakasi: 9.1 %
Embu – Nairobi Highway, Mathare: 4.5 %
C73, Kirinyaga: 4.5 %
Kesses Moi University Road, Uasin Gishu: 4.5 %
China Gardens Estate Road, Nairobi West Estate: 4.5 %
Malibet Road, House No.172: 4.5 %
Josh House, Ukwala Road: 4.5 %
Highcharts.com