2017

Sentiment analysis for hate speech detection on social media: TF-IDF weighted N-Grams based approach

Sharon Kaari Mugambi
Faculty of Information Technology (FIT)
Strathmore University

2017

Sentiment analysis for hate speech detection on social media: TF-IDF weighted N-Grams based approach

Sharon Kaari Mugambi
Faculty of Information Technology (FIT)
Strathmore University

Sentiment Analysis for Hate Speech Detection on Social Media: TF-IDF Weighted N-Grams Based Approach

Mugambi Sharon Kaari

089980

Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science in Information Technology at Strathmore University.

Faculty of Information Technology

Strathmore University

Nairobi, Kenya

June, 2017

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

……

……

……

Approval

The thesis of Sharon Kaari Mugambi was reviewed and approved by the following:

Dr. Joseph Orero

Senior Lecturer, Faculty of Information Technology

Strathmore University

Dr. Joseph Orero

Dean, Faculty of Information Technology

Strathmore University

Professor Ruth Kiraka

Dean, School of Graduate Studies

Strathmore University

Abstract

Hate speech on social media has unfortunately become a common occurrence in the Kenyan online community largely due to advances in mobile computing and the internet. Incidents of hate speech on social media have the potential of quickly disseminating amidst online users and escalating into acts of violence and hate crimes due to incitement, as was the case during the 2007-2008 Post Election Violence. With the upcoming, highly contested 2017 general elections, the monitoring of hate speech on social media platforms is of critical importance to detect hate speech occurrences as soon as possible to prevent any further escalations which may result in violence.

Current efforts by the National Cohesion and Integration Commission to monitor hate speech on social media involve the use of web crawlers to collect possible instances of hate speech based on specific keywords. Human monitors then have to analyze the collected data to determine instances that are actually hate speech. This human analysis is not only time consuming and overwhelming but also introduces subjective notions of what constitutes hate speech.

This research proposed the application of machine learning techniques to build a text binary classifier to detect hate speech on twitter. Hate speech data was collected and labelled to build the corpora. A Support Vector Machine model was trained and validated based on the labelled text data using unigram features and term frequency-inverse document frequency weighting. The research employed an experimental approach to determine which combination of features, weighting schemes and classifiers gives the best performance on the collected hate speech data. Bigram features weighted using term frequency-inverse document frequency fed into a Support Vector Machine classifier gave the best classification performance at an accuracy of 76.22 percent, with an area under the curve of 0.76 for a Receiver Operating Characteristic curve.

Keywords: Hate Speech; Social Media; Machine Learning; Support Vector Machine, TF-IDF, Bigram.

Table of Contents

List of Equations

List of Figures

List of Tables

List of Abbreviations

API- Application Programming Interface

AUC- Area Under the Curve

BOW- Bag of Words

DFD- Data Flow Diagram

HTTP: HyperText Transfer Protocol

ICT- Information and Communication Technology

IDF- Inverse Document Frequency

IT- Information Technology

JSON-JavaScript Object Notation

kNN- k-Nearest Neighbor

MI- Mutual Information

NB- Naïve Bayes

NCI Act- National Cohesion and Integration Act

NCIC- National Cohesion and Integration Commission

NLP- Natural Language Processing

OAuth- Open Authentication

PEV- Post Election Violence

RAD- Rapid Application Development

REST- Representation State Transfer

ROC- Receiver Operating Characteristic

RT- Retweet

SDLC- Software Development Life Cycle

SVM- Support Vector Machine

TF- Term Frequency

TF-IDF- Term Frequency- Inverse Document Frequency

UML- Unified Modelling Language

URL- Uniform Resource Locator

VSM- Vector Space Model

Definition of Terms

Coded Language- The use of language in a manner intended to conceal the normal meanings of expressions (National Cohesion and Intergration Commission, 2013)

Pandas- An open source library providing high-performance, easy to use data structures and data analysis tools for the python programming language (Pandas, 2017)

Scikit-learn- An open source Python library that implements a range of machine learning, preprocessing, cross-validation and visualization algorithms (Pedregosa et al., 2011)

Stereotype- An entrenched generalized belief amongst a people about the typical behaviors, attributes, attitudes, abilities and weaknesses of other people such as members of other ethnic communities (National Cohesion and Intergration Commission, 2013)

Twitter- An online social networking and microblogging service that enables users to send and read short 140-character messages (Twitter, 2017)

Acknowledgement

I would like to acknowledge God for His grace, strength and good health as I undertook this research. My sincere gratitude to the members of the Faculty of Information Technology staff including: my supervisor, Dr. Joseph Orero for his continued commitment to guide and support this research from its inception to completion; Dr. Bernard Shibwabo for his readiness and willingness to advice on the research, his comments greatly improved the manuscript and to all lecturers who sat on the presentation panels for their inputs which greatly shaped and improved the work.

Special thanks to Mr. Isaac Munya, from the National Cohesion and Integration Commission, for providing expertise and insights into hate speech and availing me resources on hate speech data.

Dedication

To my dearest mother, Teju Mugambi, father, the late Dickson Mugambi, and two sisters: Lynet Makena and Nelly Kendi, thank you for your continued support and prayers always.

Chapter 1: Introduction

1.1 Background

Globally, there is no consensus on the meaning of the term hate speech. Researchers have tried to define hate speech as speech which either promotes acts of violence or creates an environment of prejudice that may eventually result in actual violent acts against a group of people (Sambuli, Morara, & Mahihu, 2013). Speech in this sense includes any kind of expression including pictures and videos (Sambuli et al., 2013). Hateful comments against an individual solely do not qualify as hate speech, this is because hateful comments can only be considered as hate speech if they target the individual as part of a group (Sambuli et al., 2013). Cohen-Almagor (2011), defines hate speech as hateful comments towards a person or group of people based on inherent attributes such as gender, ethnicity, color among others. The definition of hate speech in Kenya, emphasizes on the use of hateful words with an intention to bring about ethnic hatred, where ethnic hatred is defined as hatred against a group of people based on their color, race, nationality or ethnic origins (National Council for Law Reporting, 2008).

There exists a strong relationship between hate speech and actual hate crime (Waseem & Hovy, 2016). Widely propagated hate speech can easily result into incitement and consequent escalation into actual acts of violence against a group of people. This was clearly witnessed in the 2007-2008 Post Election Violence (PEV) in Kenya. The 2007-2008 PEV is partly blamed on widespread hate speech based on ethnic stereotypes and coded language (National Cohesion and Intergration Commission, 2013). Hate speech was widely spread through a number of channels in the times preceding and during the PEV conflict. Such kind of speech resulted in the incitement of individuals to use violence and the galvanization of groups against one another (Hirsch, 2009). This strong connection between hate speech and actual hate crime illustrates the importance of monitoring hate speech to avoid widespread incitement and potential incidents of hate crime.

Recent advances in mobile computing and the internet have resulted in an increase in use of social media to communicate, express opinions, interact with other, and to find and share information (Cohen-Almagor, 2011). While social media provides an important avenue for communication to take place easily and efficiently, it also acts as a means of spreading hate speech online. Inherent characteristics of the Internet largely contribute to the misuse of social media to transmit and propagate hate speech. Such characteristics include: affordability, ease of access,

instantaneous access from multiple points, and anonymity, amongst others (Cohen-Almagor, 2011).

In the digital era of smartphones and social media, hate messages are prevalent in the Kenyan online community as individuals spread hate messages hiding behind their screens (Daily Nation, 2017). After the 2007-2008 PEV, the Government of Kenya enacted the National Cohesion and Integration Act to promote national cohesion and integration. The Act consequently instituted the National Cohesion and Integration Commission (NCIC) to oversee and monitor content in media such as radio, television, mobile phones and television in a bid to govern hate speech (National Council for Law Reporting, 2008).

1.2 Problem Statement

Monitoring hate content in traditional mainstream media such as radio and television, is much easier than monitoring online hate speech content such as social media and microblogging sites. This is largely due to the fact that social media consists of a large amount of user generated content that would need to be monitored.

Current efforts by the NCIC to monitor hate speech on social media involve the use of web crawlers to collect text from social media platforms and human monitors to analyze the collected text. The NCIC's research department provides keywords of most frequently occurring terms in hate speech text, most of which are based on common stereotypes and coded language. Web crawlers search social media platforms collecting text matching the keywords. Once collected, human monitors have to go through all collected text to identify which ones are hate speech and which ones are not (Munya, 2017). This human processing of collected text is inadequate as the amount of content on social media is huge, significantly limiting how much a human monitor can review.

This work proposed the development of a model that applies machine learning techniques to automatically classify tweets as hate speech or not. This automatic classification will significantly improve the process of detecting hate speech on social media by reducing the amount of time and human effort required.

1.3 Research Objectives

i. To investigate the existing techniques used in hate speech detection in social media,
ii. To review the current machine learning techniques applied in hate speech detection,
iii. To develop a model for hate speech detection on twitter,
iv. To validate the model on twitter posts.

1.4 Research Questions

i. What are the existing techniques used in hate speech detection in social media?
ii. What are the current machine learning techniques applied in hate speech detection?
iii. How will the model be designed?
iv. How will the model be validated?

1.5 Justification

Hate messages disseminated online are increasingly common, largely attributed to issues of anonymity, itinerancy, permanency and cross-jurisdiction of online content (United Nations Educational, Scientific and Cultural Organization, 2015). Notably, social media usage during the PEV was not only to promote peace and justice but also as a channel for spreading of biased information, tribal prejudices and hate speech (Makinen & Kuira, 2008).

With the upcoming highly contested 2017 general elections, the current political climate in Kenya can easily bear comparison to that which preceded the 2007-2008 PEV (Institute For Security Studies, 2017). It is therefore of critical importance to monitor and identify instances of hate speech, as soon as possible to prevent their spread and possible unfolding into acts of violence or hate crimes.

Text classification is an important technique for the handling and organization of text data with a wide range of applications in information retrieval. Currently, NCIC human monitors have to sift through numerous online content to identify hate speech in social media. This human analysis is overwhelming, time consuming and introduces personal interpretation of what is considered as hate speech. Text classification would enable categorization of the huge amounts of online data into hate speech or non-hate speech text, significantly reducing the amount of data that human monitors have to review, making the process of hate speech detection faster.

1.6 Scope and Limitations

This study limited its analysis to detecting hate speech on the social media platform Twitter and only considered tweets expressed in English and Swahili. The use of sheng', vernacular languages, memes, audios and videos within tweets were not considered.

Chapter 2: Literature Review

2.1 Introduction

This chapter reviews relevant literature to further comprehend the concept and investigate the research problem. The nature of hate speech in Kenya and current processes to monitor hate speech on social media is reviewed. Significant and relevant publications and research are further reviewed to understand the application of machine learning techniques in text classification. A conceptual framework is then presented at the completion of the literature review.

2.2 Hate Speech in Kenya

Kenya has a history of hate speech, especially in politics. In 1992, multi-party politics coupled with hate speech resulted in ethnic clashes. Similarly, in the 2007 referendum, political leaders spread hate speech to incite and promote violence (Nyambane, 2012). The culmination of hate speech in Kenya, came during the 2007-2008 PEV after the disputed general elections which led to a number of serious human rights violations (Nyambane, 2012). Reports after, found that the PEV was largely promoted by ordinary Kenyans and partly by leaders. This was done through the use of incitements and calls to violence throughout the campaign period and as the conflict unfolded. Media, short messaging services, the internet and mobile phones were used as transmitters of hate speech to incite acts of violence (Nyambane, 2012).

The National Cohesion and Integration Commission (NCIC) was instituted as a consequent of the 2007-2008 PEV to oversee and monitor content in media such as radio, television, mobile phones and television in a bid to govern hate speech (National Council for Law Reporting, 2008). According to the NCIC, a statement does not amount to hate speech unless it: causes hatred, makes a group or community look inferior, makes a community or group be viewed with contempt, degrades a group or community, or dehumanizes a group or community (National Cohesion and Integration Commission, 2011). To be quantified as hate speech, the statement should contain: threatening, abusive or insulting messages, sometimes using coded language. These messages must be directed towards a targeted group and intended to stir hatred based on the group's identity including: ethnicity, race, color or any other national origin (National Cohesion and Integration Commission, 2011).

2.3 Hate Speech Detection in Kenya

While investigating and monitoring hate speech, investigators must take into consideration five key aspects: context, ripple effect, fear, possible retaliation and violence (National Cohesion and Integration Commission, 2011). A statement can be considered hate speech in one context but not in another. Additionally, the same statement might have different levels of impact depending on the context, for example ethnic statements may have a higher impact in political environments than social settings. The second aspect, ripple effect, and third effect, fear mean that the statement should cause some discomfort and fear amongst members of the group being targeted, respectively. The fourth aspect, possible retaliation means that the statement should provoke counterattacks and finally the statement promotes acts of violence or hate crimes (National Cohesion and Intergration Commission, 2013).

Hate speech in Kenyan online forums has unfortunately become a common occurrence with the growth of the internet, social media and mobile computing in the recent past. Social media has created a new space for the dissemination of hate speech. Since 200/, the NCIC, Kenyan civil society as well as police authorities have put measures to monitor hate speech on traditional mainstream media but hate speech on social media remains to hardly monitored (Sambuli et al., 2013). However, more recently NCIC have put effort into monitoring hate speech on social media through the use of web crawlers.

2.4 Coded Language and Stereotypes in Hate Speech

Kenya is a multicultural country with over forty two ethnic tribes, each with its own unique way of communicating. Almost all ethnic communities in Kenya have some kind of stereotypes about them, these stereotypes may be positive or negative (National Cohesion and Intergration Commission, 2013). Most negative statements depict feelings of contempt and general hate towards targeted communities resulting in heightened friction and animosity among various ethnic communities. The negative statements are often expressed in coded language well known to the members of the community who use it and may or may not be known to the targeted ethnic communities (National Cohesion and Intergration Commission, 2013).

Generally, negative stereotypes about a target community embody the following scenarios: association of the community with a practice considered to be bad by other communities, disdain of the target community because of traits considered to be immoral or childish, expression of

mistrust about the target community and finally expression of inherent hate towards the target community (National Cohesion and Intergration Commission, 2013). Examples of negative stereotypes used in Kenya include: attributing members of the Kikuyu community with the term thieves, use of the word uncircumcised to refer to members of the Luo community and the association of communities with animals as depicted in Figure 2.1 (National Cohesion and Intergration Commission, 2013).

Phrase/Word in Local Language	User Community	Target Community	Meaning.
Macengi	Akamba	Mbeere	Dangerous small animal that destroys crops.
Nyamu cia ruguru	Kikuyu	People from upcountry	Animals from upcountry
Ajayee	Kamba	Maasai	Smelly
Longuulkitkit	Maasai	Kamba	Smelly armpits
Longuu	Samburu	Turkana	Smelly and unclean people.
Uriti ja mutigania wa wa kunati	Tharaka	People from Tigania	Foolish like Tigania from Kunati
Rubwa	Kikuyu	Kalenjin	People who eat dogs
Pik ap kong'oek	Kalenjin	Luhya	People who eat termites
Soqi	Samburu	Pokot	The inhuman, beasts and cattle rustlers
Emoit	Turkana	Pokot	A cattle thief
Bosyanju	Luhya	Sabaot	Arsonist.
Avalivavandu/Avas eeve.	Luhya	Kikuyu/Embu/Meru	The Kikuyu will 'eat people' the way a hyena does. This means that they can con someone until the person is financially or materially drained.

Figure 2.1: Hate speech: Association of ethnic communities with animals *(National Cohesion and Intergration Commission, 2013).*

Majority of incidents of hate speech in Kenya are based on the emphasis of negative stereotypes of different ethnic communities. Coded language is used to cause hatred and animosity towards certain ethnic communities for selfish gain, especially in the political environment (Siele, 2013). In the 2007 electioneering period in Kenya, the use of coded language and stereotypes was wide spread and partly contributed to increasing tensions which led to the PEV violence between different ethnic communities (National Cohesion and Intergration Commission, 2013).

2.5 Machine Learning Approach to Detecting Hate Speech

2.5.1 The Text Classification Problem

Given a training set D of labelled documents (d, c) where (d, c) ϵ X × C, the text classification problem is to learn a classification function γ that maps unseen documents to classes as in Equation (2.1) (Manning , Raghavan, & Schutze, 2009).

$$\gamma: X \rightarrow C \tag{2.1}$$

Where X is the document space; and C is a fixed set of classes C= $\{c_1, c_2, \ldots, c_j\}$.

The training set D of documents provided is labelled with respective classes, as such the learning is said to be supervised learning. The learning algorithm learns a classification function from the training set that maps documents to classes. The classification function is then able to map unseen documents to their respective classes as in Equation (2.1).

2.5.2 Text Preprocessing

Text classification problems should be able to handle unstructured or semi-structured data sets. As such, preprocessing unstructured data is a very important role in the text classification problem. Some preprocessing activities include: replacing special characters and punctuation marks, normalizing case, removing duplicate characters, removing stop words and stemming (Vijayarani, Ilamathi, & Nithya, 2015).

Stop words are common words which are a portion of natural language that do not add meaning to text documents but make the text appear heavier. Since they do not add meaning, stop words can be easily removed without affecting the analysis process. Their removal reduces the number of features to be considered and as such can improve the performance of a classifier. The most common stop words include: articles, prepositions and pronouns (Vijayarani et al., 2015).

Words in natural language may have a multiple variations in their suffixes, increasing the number of features to be considered in analysis. So as to accurately match words, save processing time and memory space it may be important to reduce a word to its root by remove the different suffixes. This process is known as stemming, where the root or stem of a word is identified and all variations of the word reduced to their stem. The assumption made is that words with the same base have the same meaning despite their morphological forms and as such can be reduced to the

same base. In stemming it is important to keep words that have different meaning separate (Vijayarani et al., 2015).

Term Frequency- Inverse Document Frequency (TF-IDF) is a combination of two numerical statistics: Term Frequency (TF) and Inverse Document Frequency (IDF) that shows the importance of a term in a document collection. If the frequency of occurrence of a term in a document increases, the value of the term also increases. However if the frequency of occurrence of a term in various documents in the corpus increases, then the value of the term decreases. Tf-idf is often used as a weighting factor in many information retrieval and text classification problems (Vijayarani et al., 2015).

2.5.3 Feature Selection in Text Classification

The feature space for text classification problems consists of all the unique terms that occur in a document. The number of features can therefore be quite big for a corpus that is average sized. This high dimensionality of the feature space is an inherent characteristics of text classification problems and poses a significant problem to many machine learning algorithms (Yang & Pedersen, 1997). The high dimensional feature space may result in poor accuracy results and over fitting. It is therefore important to reduce the feature space to improve performance of the learning algorithms, reduce over fitting and improve the time needed to train a model. Feature selection can be defined as the process of selecting a subset of the terms occurring in a corpus and using only this subset as features in text classification (Manning et al., 2009). This section discusses three feature selection methods: chi-square, mutual information and frequency based feature selection.

2.5.3.1 Chi-square Feature Selection

In statistics, the χ^2 test is applied to test the independence of two events (Manning et al., 2009). In feature selection for text classification, the two events are the occurrence of a term and occurrence of a class. The terms are then ranked according to Equation (2.2) below (Manning et al., 2009):

$$X^2(\mathbb{D}, t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}} \tag{2.2}$$

Where:

e_t expresses whether the document contains term t or not

e_c expresses whether the document belongs to class c or not

N represents the observed frequency in document D

E represents the expected frequency

χ^2 is a measure of how much the expected counts E and observed counts N deviate from each other. A higher value of χ^2 indicates that the hypothesis of independence is incorrect (Manning et al., 2009). In text classification, χ^2 measure can be used to rank features with respect to their usefulness, and choosing the k terms with the highest χ^2 value.

2.5.3.2 Mutual Information

This feature selection approach computes A (t, c) as the expected mutual information of term t and class c (Manning et al., 2009). Mutual Information (MI) measures how much information the presence or absence of a term contributes to making the correct classification decision on c as depicted in Equation (2.3) below (Manning , Raghavan, & Schutze, 2009):

$$I(U;C) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} P(U = e_t, C = e_c) \log_2 \frac{P(U = e_t, C = e_c)}{P(U = e_t)P(C = e_c)}, \quad (2.3)$$

Where:

U is a random variable that takes values e_t=1 (the document contains term t) and e_t=0 (the document does not contain t)

C is a random variable that takes values e_c=1 (the document is in class c) and e_c=0 (the document is not in class c).

2.5.3.3 Frequency-based Feature Selection

This approach is to choose the terms that occur most frequently in a class. Frequency can be defined as document or collection frequency. Collection frequency refers to the number of tokens of a term t that occur in documents in class c whereas document frequency is the number

of documents in the class c that contain term t. The major shortcoming of this approach is that it some frequent terms may not add any information to the class but still be selected (Manning et al., 2009).

2.6 Document Representation

Text data cannot be processed by machine learning algorithms as it is. It is therefore necessary to convert the text into a format that a machine learning algorithm can process. Two common approaches can be used: Bag of Words (BOW) and Vector Space Model (VSM). The BOW approach represents documents as a collection of words without any order but keeping their multiplicity. All unique words contained in the corpus make up the dictionary. Each document is then represented a vector of word frequencies. The assumptions made by this model are that: the order of words does not matter and words are independent of each other. Additionally, this model does not allow for weighting of terms in specific documents (Mazzonello, Gaglio, Augello, & Pilato, 2013).

The vector space model (VSM) is a generalization of the BOW model that represents documents in a corpus using a multi-dimensional document-term matrix. A term may consists of more than one word and each unique term represents a dimension in the matrix. Vector elements are weights of the term contained in the specific document (Mazzonello et al., 2013).

2.7 Feature Weighting

Various weighting schemes could be then used for a document-term matrix. The simplest weighting scheme being a simple Boolean 1 if the term appears in the document or 0 if it does not. It could also be based on the frequency of the term in the corpus or term frequency (number of times the term appears in the specific document). If C is the set of all classes, then TF (t, c) can be defined as the frequency of term t in class c, calculated as in Equation 2.4.) (Mazzonello et al., 2013).

$$TF(t,c) = \frac{|occurrences\ of\ t\ in\ c|}{|terms\ in\ c|} \tag{2.4}$$

Inverse Document Frequency (IDF) weighting assigns more weight to terms that are not very common in the entire corpus. IDF (t) as the percentage of documents in class c in which term t appears, calculated as in Equation 2.5 (Mazzonello et al., 2013).

$$IDF(t) = \log \frac{|C|}{\sum_{c_i \in C} \frac{|documents\ in\ c_i\ in\ which\ t\ appears}{|documents\ in\ c_i|}} \quad (2.5)$$

A common approach to weighting in text classification problems is Term Frequency-Inverse Document Frequency (TF-IDF). TF-IDF is a combination of Term Frequency (TF) and Inverse Document Frequency (IDF). TF-IDF can then be calculated as the product of Equations (2.4) and (2.5) (Mazzonello et al., 2013). Using tf-idf, the highest weight occurs when a term occurs often within a few documents, lower when it occurs a less number of times in a document or occurs in many documents and lowest when it occurs in all documents (Manning et al., 2009).

Another type of weighting is sentiment weighting, which refers to weighting terms based on their level of positivity or negativity. SentiWordNet is a lexical resource based on WordNet (a large lexical database of English synonyms). SentiWordNet associates each set of synonyms in WordNet with a numerical value, illustrating the set's objectivity, positivity and negativity (Baccianella, Esuli, & Sebastiani, 2010). This numerical values can be used to weigh terms in the text classification problem.

2.8 Machine Learning Algorithms

In a machine supervised learning approach, a classifier is built automatically by learning the properties of categories from a set of pre-classified training documents. When using machine learning techniques four main issues need to be considered: categories that will be used to classify the instances, training data, features that will be used to represent each instance and the algorithm to be used for categorization (Feldman & Sanger, 2007). This section describes the various possible algorithms that can be used in text classification.

2.8.1 Naïve Bayes

Naïve Bayes is a simple classification method based on the Bayes rule. Given a document d and a set of predefined classes $\{\ldots c_{i,\ldots}\}$, the Naïve Bayes classifier first computes the posterior probability that the document belongs to each particular class c_i $P(c_i|d)$, and then assigns the document to the class with the highest probability value. The posterior probability is computed by applying the Bayes rule as in Equation (3.4) (Bai, Nie, & Paradis, 2004).

$$P(c_i \mid d) = \frac{P(d \mid c_i)P(c_i)}{P(d)} \tag{2.6}$$

The denominator in Equation (3.4) is independent from classes, so it can be ignored for the purpose of class ranking. Therefore Equation (3.4) can be approximated as in Equation (2.7) below (Bai, Nie, & Paradis, 2004).

$$P(c_i \mid d) \propto P(d \mid c_i)P(c_i) \tag{2.7}$$

Naïve Bayes makes a conditional independence assumption, that words are independent given a class, that is, for a document $d=d_1,\ldots,d_n$, the $P(d|c_i)$ can be calculated as in Equation (2.8) (Bai, Nie, & Paradis, 2004).

$$P(d \mid c_i) = \prod_{j=1}^{m} P(d_j \mid c_i) \tag{2.8}$$

Equation (2.8) can then be expressed as in Equation (2.9) below:

$$P(c_i \mid d) \propto \prod_{j=1}^{m} P(d_j \mid c_i)P(c_i) \tag{2.9}$$

$P(c_i)$ in Equation (2.9) can be estimated by the percentage of the training examples belonging to class c_i as in Equation (2.10):

$$P(c_i) = \frac{N_i}{N} \tag{2.10}$$

where N is the total number of training documents and N_i is the number of training documents in class c_i.

$P(d_j|c_i)$ is usually determined as in Equation (3.4)below:

$$P(d_j|c_i) = \frac{1 + \text{count}(d_j, c_i)}{|V| + N_i} \tag{2.11}$$

Where count (d_j, c_i) is the number of times that word d_j occurs within the training documents of class c_i, and |V| is the total number of vocabulary. This estimation uses the Laplace

(add one) smoothing to solve the zero-probability problem (Bai, Nie, & Paradis, 2004). The zero-probability problem occurs when unseen terms in the document are encountered due to the scarcity in training data (Kilimci & Ganiz, 2015).

The large size of vocabulary inherent in text classification problems makes Naïve Bayes suitable for the text classification problem. Additionally, Naïve Bayes works well with both textual and numerical data and is also easy to implement and compute (Swamy, Hanumanthappa, & Jyothi, 2014). However, it performs poorly when features are correlated like short texts or news headlines classifications. Additionally, the conditional independence assumption is poorly violated in real world data (Rana, Khalid, & Akbar, 2014).

2.8.2 Support Vector Machines

Support Vector Machines (SVMs) are based on the structural risk minimization principle from computational learning theory, whose basic idea is to find a hypothesis h for which we can guarantee the lowest true error (Joachims, 1998). SVMs find the hypothesis h which minimizes the bound on the true error. SVMs not only have a solid theoretical foundation but also perform classification more accurately that most other algorithms, especially in applications involving high dimensional data (Joachims, 1998).

In geometrical terms, a binary SVM classifier can be seen as a hyper plane in the feature space separating the points that represent the negative instances. The classifying hyper plane is chosen during training as the unique hyper plane that separates the known positive instances from the known negative instances with the maximal margin. Notably, SVM hyper planes are fully determined by a relatively small subset of the training instances called support vectors illustrated as in Figure 2.2 (Manning et al., 2009).

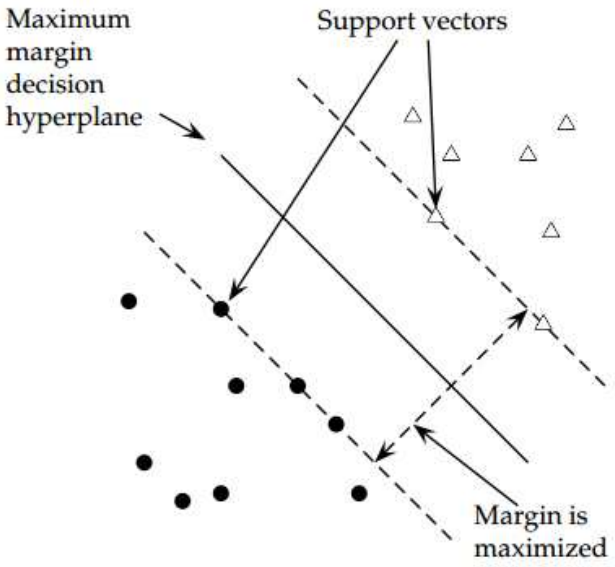


Figure 2.2: Support Vector Machine (Manning et al., 2009)

In his book, Liu (2007) describes SVM as a linear learning system that builds binary classifiers. Let the set of training examples D be $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ where $x_i = (x_{i1}, x_{i2}, \dots, x_{ir})$ is a r-dimensional input vector in a real-valued space, y_i is its class label(output value) and $y_i \in \{1, -1\}$, 1 denotes the positive class and -1 denotes the negative class. To build a classifier, SVMs find a hyperplane of the form in Equation (2.12). The hyper plane is called the decision boundary.

$$f(x) = (w.x) + b \tag{2.12}$$

So that an input vector x_i is assigned to the positive class if $f(x_i \geq 0)$, and to the negative class otherwise as expressed in Equation (2.13).

$$y_i = \begin{cases} 1 \text{ if } (w.x_i) + b \geq 0 \\ -1 \text{ if } (w.x_i) + b < 0 \end{cases} \tag{2.13}$$

The text classification problem is greatly characterized with high dimensional spaces and few irrelevant features due to the large number of terms contained in text documents. SVMs are highly applicable to text classification problems because of a number of reasons. Firstly, their learning ability is independent of the dimensionality of the feature space. SVMs use over fitting protection, which does not necessarily depend on the number of features and therefore have the

potential to handle large feature sets inherent in text classification problems. Secondly, SVMs have the ability to work with few irrelevant features. This makes them suitable for text classification since there are few irrelevant features that can be removed without loss of information. Thirdly, for each document the corresponding document vector contains only few entries which are not zero. Finally, most text categorization problems are linearly separable (Joachims, 1998).

2.8.3 Statistical n-gram Language Modeling

Language modeling has been applied successfully in information retrieval, topic detection and tracking and more recently has become mainstream in text classification (Bai, Nie, & Paradis, 2004) (Pei & Wu, 2014). Largely because it has a solid theoretical foundation in statistics. The goal of language modelling is to predict the probability of natural word sequences. Given a word sequence w_1, w_2, …, w_t, the probability of any word sequence can be calculated as in Equation (2.14) (Peng, 2003).

$$P(w_1 w_2 ... w_T) = \prod_{i=1}^{T} P(w_i | w_1 ... w_{i-1}) \tag{2.14}$$

An n-gram model approximates this probability by assuming that the only words relevant to predicting $P(w_i|w_1,\ldots, w_{i-1})$ are the previous n-1 words, i.e. it assumes the Markov n-gram independence assumption, depicted as in Equation (2.15) (Peng, 2003).

$$P(w_i | w_1 ... w_{i-1}) = P(w_i | w_{i-n+1} ... w_{i-1}) \tag{2.15}$$

A straight forward maximum likelihood estimate of n-gram probabilities from a corpus is given by the observed frequency as in Equation (2.16), where #(.) is the number of occurrences of a specified gram in the training corpus (Peng, 2003).

$$P(w_i | w_{i-n+1} ... w_{i-1}) = \frac{\#(w_{i-n+1} ... w_i)}{\#(w_{i-n+1} ... w_{i-1})} \tag{2.16}$$

An n-gram language model can therefore be applied to text classification in a similar manner to a Naïve Bayes model. A document d is categorized under a category c according to Equation (3.4) (Peng, 2003).

$$c^* = \arg\max_{c \in C} \{P(c|d)\} \tag{2.17}$$

N-gram classifiers are actually a straight forward generalization of Naïve Bayes, a unigram classifier with Laplace smoothing corresponds exactly to the traditional Naïve Bayes classifier. However, n-gram language models, for larger n, possess many advantages over Naïve Bayes, including modeling longer context and exploiting superior smoothing techniques in the presence of sparse data. In Naïve Bayes, the conditional independence assumption holds, the language modeling approach however enhances this by considering a Markov dependence between adjacent attributes (words) (Peng, 2003).

2.9 Related Work

Recently, there have been many studies applied to social media data to understand various aspects of human behavior, the physical environment and social phenomena including hate speech detection. This section reviews and discusses various related works.

2.9.1 Umati Project to Monitor Hate Speech on Social Media

Umati is a hate speech monitoring project that analyses incidents multilingual hate speech in the Kenyan online space such as blogs, forums, online newspapers, Facebook and Twitter (iHub Research, 2013). The first phase of Umati involves the use of human monitors to collect and analyze hate speech from the various online platforms. The human monitors scour the platforms for incidents of hate speech. Once a human monitor encounters a statement that is considered to be hate speech, they enter it into an online form, whilst providing additional information about the statement in the form. Finally the statement is sorted into one of three categories: offensive speech, moderately dangerous speech and extremely dangerous speech (iHub Research, 2013). This hate speech detection process is illustrated in Figure 2.4 (iHub Research, 2013).

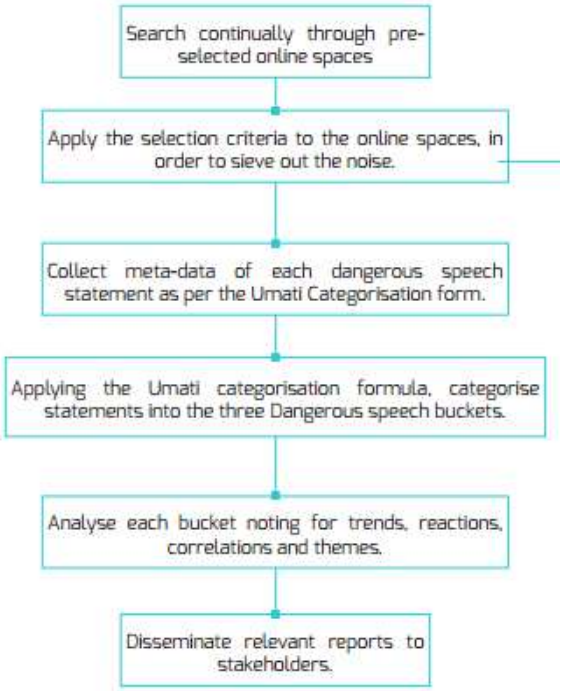


Figure 2.3: Umati Process Algorithm *(iHub Research, 2013)*

The current human processing of hate speech text by Umati phase I is time consuming, involves a lot of effort and human input. A more automated way of detecting hate speech in text would be preferable. The next phase, Umati II will involve the application of machine learning and natural language processing techniques to automatically identify instances of hate speech. This phase is however on going and not much has been achieved from it.

2.9.2 Rule Based Approach to Detecting Hate Speech

Regular expressions are an algebraic notation for specifying search strings (Daniel & James, 2000). Regular expressions can be used in classification rule builders to match a wide variety of patterns and consequent use of the matches to set classification labels. In his work, Maloba (2013) proposes the use of regular expressions to detect hate speech in multi lingual (English, Swahili and Sheng') text. In the work he notes that hate speech in Kenya is mostly dependent on ethnic grouping and uses this as the basis for formulating rules based on well-known ethnic stereotypes to match and identify hate speech.

To build the corpora, a number of correspondents were asked to come up with statements that they deem hateful and not hateful towards their own community or other communities in any

of the three languages under consideration. Each statement received in one language was translated to its equivalent in the other two languages. From the collected corpora, individual words and sentences that make the submitted statements hateful were identified and grouped according to the ethnic community being targeted. The author then builds regular expressions from the group to identify instances of hate speech (Maloba, 2013).

2.9.3 Machine Learning Approaches

In their paper, (Ogada et al., 2015), use language modeling to improve performance of a naïve Bayes classifier in detecting hate speech. The work notes that the naïve Bayes model makes strong assumptions that the words in a document are independent, and further notes that this assumption is clearly violated in natural language text. The authors try to address this problem and show that it can be solved by modeling text data differently using N-grams. The paper analyses the efficiency of n-grams as features with various machine learning algorithms and shows that bigrams have much better performance for naïve Bayes text classification. K-Nearest Neighbor (KNN) has the same accuracy for unigram, bigrams and trigrams while SVM has the highest accuracy value for bigram and trigram. The results are as depicted in Figure 2.4 below (Ogada et al., 2015):

	1-gram	2-grams	3-grams	4-grams	5-grams	6-grams	7-grams	8-grams	9-grams	10-grams
NB	90.6%	98.1%	92.5%	77.4%	66.0%	58.5%	56.6%	56.6%	56.6%	56.6%
KNN	98.1%	98.1%	98.1%	79.2%	67.9%	66.0%	60.4%	56.6%	56.6%	56.6%
SVM	96.2%	98.1%	98.1%	69.8%	67.9%	58.5%	60.4%	56.6%	56.6%	56.6%

Figure 2.4: Accuracy Values for Test Data (Ogada et al., 2015)

In their paper Gebre et al., (2013), use TF-IDF weighting with linear classifiers to improve the task of identifying the native language of a writer based on the writer's foreign language production. The native language identification problem is modeled as a classification problem where machine learning classifiers are used to assign labels of the native language to texts. They obtain the best classification accuracy when TF-IDF weighting is used with unigram and bigram terms (Gebre et al., 2013).

2.10 Conceptual Framework

Based on the literature reviewed and the various gaps identified, this work proposes the following conceptual framework to detect hate speech from twitter. Hate speech relevant data will

be collected from twitter and used to create the corpus necessary for learning. The tweets will be annotated as hate speech or non-hate speech and then go through a number of steps in the preprocessing phase including: removal of punctuation marks, removal of stop words and conversion to lower case. The tweets will be represented in a document-term matrix, using unigram terms with TF-IDF feature weighting found to work well in classification problems.

For its suitability in text classification problems, SVM algorithm will then be applied to learn a model for detecting hate speech, from the training set. The model's performance will be evaluated based on the metrics: accuracy, precision, recall and the F-Score. Once the model has reached an acceptable level of performance, it can be used to detect new instances of hate speech in other tweets. A user will specify keywords to be used to retrieve unobserved tweets from the Twitter Search Application Programming Interface (API). The tweets will be preprocessed in a similar manner to that used in training and input into the SVM model created to be classified as either hate speech or non-hate speech. The classification results will then be displayed to the user. Figure 2.5 below depicts this conceptual framework of the proposed prototype.

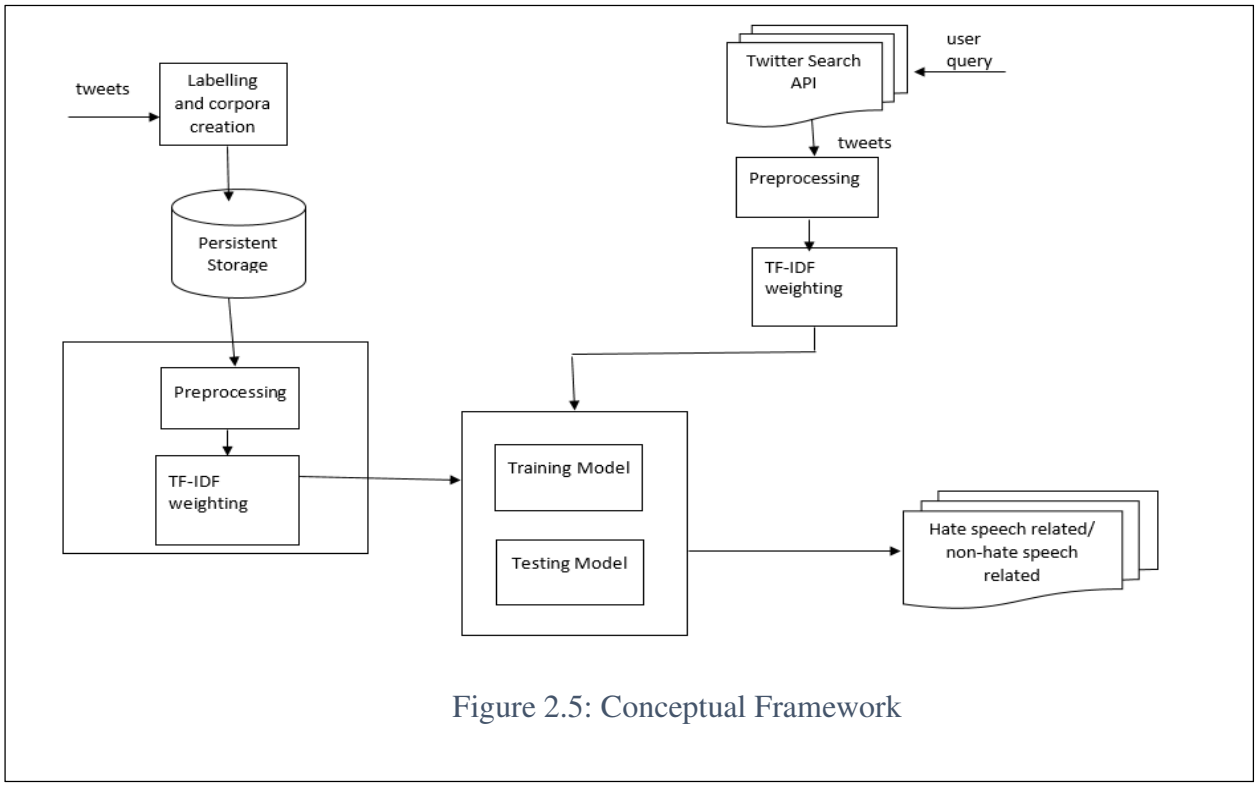


Figure 2.5: Conceptual Framework

Chapter 3: Research Methodology

3.1 Introduction

Research can be defined as the process of systematically solving problems (Bhatnagar & Singh , 2013). This section describes the various methods and procedures that were adopted in carrying out the research. This research was guided by the objectives that the author proposed to meet at the end of the research. It was greatly informed by the nature of hate speech in Kenya and research approaches that had been used in similar work reviewed in chapter 2. The research employed an applied approach to design, implement and test SVMs. Primary data in the form of historical twitter posts identified as hate speech were used to train the model and facilitate the research. Experiments were designed to validate the model and determine the best configuration of feature types, feature weighting schemes and machine learning algorithm to be used to detect hate speech on twitter.

3.2 Research Design

A research design is a blueprint describing how a research study is to be completed: operationalizing variables so they can be measured, selecting a sample of interest to study, collecting data to be used as a basis for testing hypotheses, and analyzing the results (Thyer, 1993). This research took an experimental design approach, which involved the identification of research objectives, building of the SVM model as a proof of concept and validation of the model using a number of experiments to ensure the best performance (Creswell, 2003).

3.2.1 Target Population and Sampling

A population is defined as the total number of units in a study environment from which a sample may be selected (Bryman, 2012). Twitter posts associated with hate speech in Kenya were chosen as the population of this research. Purposive sampling was applied in the research, where the sample was determined based on the judgement of the researcher with prior knowledge of characteristics of tweets that constitutes hate speech.

3.2.2 Data Collection

Interviews were used to gain additional insight on the techniques currently used by NCIC to detect hate speech on social media, to determine the user requirements of a system to detect hate speech on twitter, and to provide further guidelines on the type of keywords to be used in the mining of twitter. The guide used for the interviews is attached in Appendix B.

The NCIC provided keywords used to mine twitter for instances of hate speech. These keywords are terms frequently found in text that is determined to be hate speech. A sample of the keywords is as depicted in Table 3.1.

Table 3.1: Keywords to Search Twitter for Hate Speech Related Tweets

wabaara	malizia wao	kikuyu thief
kukuyu	ncic kenya	nyani kikuyu
tunavu	National cohesion and intergration should see this	madoadoa
wahame	Okuyu	kalenjin
tutawamaliza	Jaluo	wakamba ni wajinga
2tutawamaliza	no raila no peace	Wakikuyu ni wajinga
katakata	wajaka ni wajinga	kill duale
kill kikuyu		

3.2.3 Mining Twitter

To build the corpora, hate speech related tweets were collected from Twitter. Twitter allows developers to access tweets using two APIs: the Representational State Transfer (REST) API and the Streaming API. Both APIs require the use of Open Authentication (OAuth) to allow applications to get access to them and issue responses in JavaScript Object Notation (JSON) format. The REST API enables developers to read and write Twitter data. An important component of the REST API is the Search API which enables developers to query against indices of recent tweets up to 7 days old. The Streaming API allows developers to process tweets in real time, continuously delivering responses in JSON format over long lived HTTP connections (Twitter, 2017).

While extremely helpful, the two API have limitations in that they cannot be used to access tweets more than seven days old. To build a comprehensive corpora it was necessary to collect tweets much older than seven days. A number of online tools exist that aggregate historical tweets and provide access to them such as Gnip API (Gnip, 2017). Most of these tools are made available to developers at a cost. This work made use of Jefferson Henrique's open source code made available through GitHub to retrieve the required data from twitter. The open source code mimics the working of the search feature on twitter through a browser to retrieve the older tweets (Jefferson, 2017). It allows the collection of old tweets using various parameters as depicted in Table 3.2 (Jefferson, 2017).

Table 3.2: Jefferson's Search Parameters

Parameter		**Description**
Query search		A query text to be matched
Username		Username of a specific twitter account
Bound dates	since	The lower bound date
	until	The upper bound date
Maxtweets		The maximum number of tweets to retrieve

Using Jefferson's open source code, tweets were collected from twitter based on keywords provided by the NCIC for commonly found terms in instances of hate speech. The terms used to collect tweets included: okuyu, tunavu, nyani kikuyu, tutawamaliza, madoadoa, kill luo, kill duale, kill kikuyu, Kalenjin, kukuyu and kikuyu thief. A total number of 14055 tweets were collected based on the preceding keyword search parameters and persisted in a csv file.

3.2.4 Corpus Construction

Once collected the tweets had to be labelled as hate speech or not hate speech. The class label 1 was used for tweets found to be hate speech and -1 for tweets found to be not hate speech. The research used guidelines provided by the NCIC in the labelling process to determine what tweets are hate speech and which ones are not. These guidelines are described in Section 2.2 of this document. To further guide the labelling process, the NCIC also provided the researcher with sample hate speech text as depicted in Table 3.3.

Table 3.3: Sample Hate Speech Data

we w'll deal with the kikuyus pependicularly, already we hv sent a warning to them to vacate nyanza region
This time cord must win elections wapende wasipende kama sio hivyo kikuyu warudi kwao
These round again we should not see these happening in ur land pro. Seno have never participated in the mmu demonstration the NCIC should not try to intimidate him by all means we will support him up to ur cows and goats leave about our leaders and may God be with because he is innocen. Some people who thought we are incited by him to demand ur lectures to come to classes then ur lost and for sure no stone will be left unturned these round na tumechoka saa na watu wengine wataona.
I hate kikuyus cz all r devils hiding in sheep wool
Yes, Lamu Governor has sued us on hate speech. I have received a call to appear before National Cohesion and Integration Commission(NCIC), on allegation of hate speech.

I strongly believe that this is a ploy to #silence me, because of my firm stand on issues that matter and are of importance to the people of Bahari Ward and Lamu at large
Duale should be taken to the streets of Kisumu and be forced to abuse Raila Odinga ten times infront of the crowd. If he survives of which I doubt, he will come back and testify to his fello sychopants like kipchumba murkomen and Moses Kuria and they will change and there will be some reforms in the government.. These people should not take kenyans for a ride. We know very well these Somali people have no leadership qualities and what Duale want is this country to stumble like his country Somalia so that we can be equal. Once an alshabaab is always an alshabaab and once an ISIS is always a bomber. Duale and Kuria should stop hypocrisy
It is time CORD asks the 4 dead fools from Siaya and Kisumu to wake up and go home. If the cops are acting up lets end the morgue drama too
I blame the UON KIKUYU students for refusing to join in demo to remove Babu Owino. Kikuyus are never progressive, no wonder all your men are thieves and women are prostitutes. JIGGERS
Wajaluo wote wahame

From the 14055 collected tweets, a total of 1904 tweets were labelled, of which 785 tweets were labelled as 1(hate speech relevant) and the rest 1119 labelled as -1(not hate speech relevant) as illustrated in Table 3.4.

Table 3.4: Corpus Description

Hate Speech Tweets	Non- Hate Tweets	Total
1119	785	1904

3.2.5 Data Preprocessing

The text data collected was in an unstructured format that was not suitable for machine learning. Preprocessing of the data was done as depicted in Figure 3.1. The stop words removed from the tweets included commonly occurring terms such as hashtags, mentions and URL links.

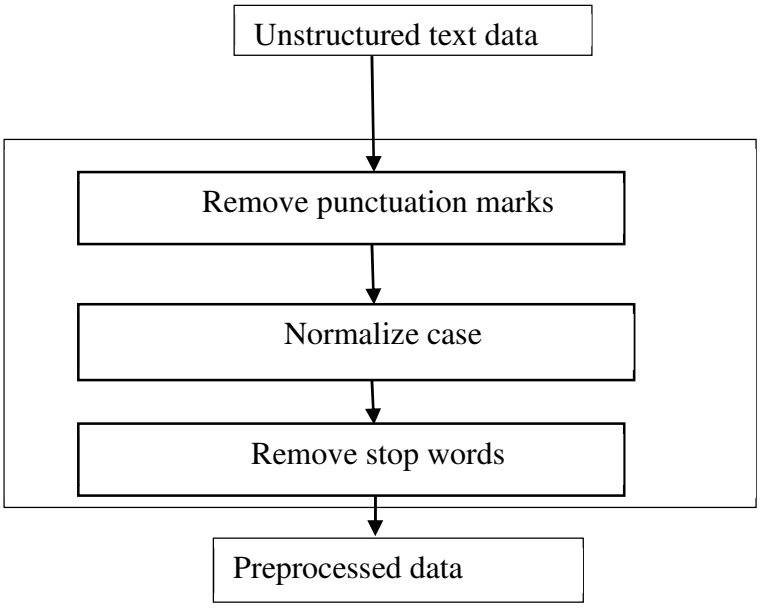


Figure 3.1: Preprocessing data

3.3 Model Training

The collected text was represented in a document-term matrix with tf-idf feature weighting to enable the application of SVM machine learning algorithm. The collected texts were split into training sets and testing sets, to be used to train and validate the model respectively. Seventy percent of the labelled data was used for training and the remaining thirty percent used to evaluate the model.

3.4 System Development Methodology

The prototype was developed following the Rapid Application Development (RAD) system development methodology which emphasizes on creation of applications in a short amount of time, sometimes with compromises in usability, features and execution speeds (Naz & Khan, 2015). Developed by James Martin, RAD accelerates the cycle of development of an application, resulting in the building of quality products faster and consequently saving valuable resources.

3.4.1 Overview of RAD Structure

The RAD phases and tasks involved in each stage can be depicted diagrammatically as in Figure 3.2 below (Orawit, 2006).

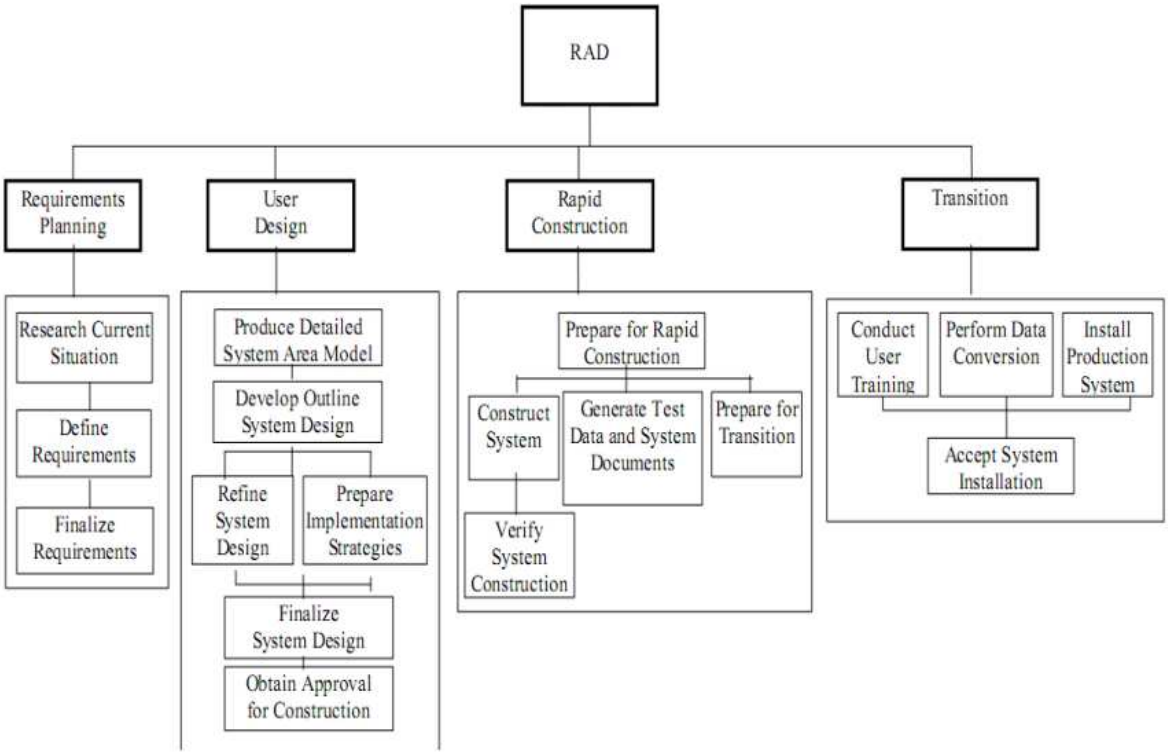


Figure 3.2: Overview of RAD *(Orawit, 2006)*

3.4.2 Phases of RAD

The James Martin approach to RAD divides the process into four distinct phases as depicted in Figure 3.3 (Orawit, 2006).

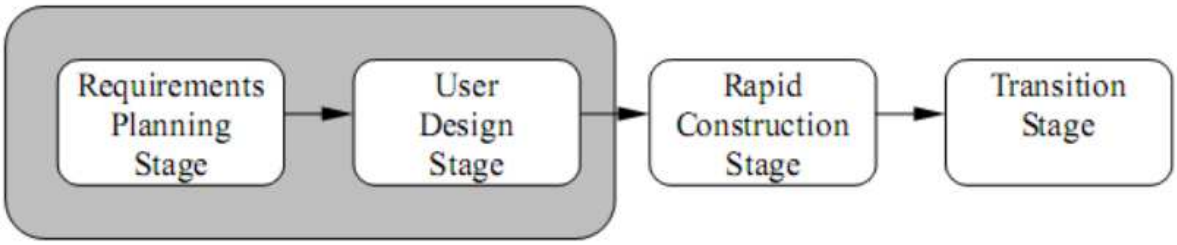


Figure 3.3: Phases of RAD *(Orawit, 2006)*.

3.4.2.1 Requirements Planning Phase

In this phase, requirements of the hate speech detection prototype were obtained through the use of interviews with an aim of establishing a general understanding of: existing systems and

processes, challenges encountered while monitoring hate speech online and the possible eventual use of the system (Orawit, 2006).

3.4.2.2 Design Phase

In this phase, the structure and architecture of the prototype was designed. Unified Modelling Language (UML) diagrams were designed to depict various components and aspects of the system including use case diagrams, context diagrams, data flow diagrams and sequence diagrams.

3.4.2.3 Construction Phase

After completion of the detailed design of the proposed system, the prototype was implemented using python as a development language. Python's scikit-learn library was used to give an implementation of the various machine learning algorithms. The pandas library was used to provide easy to use data structures that ease the data analysis and manipulation process. After implementation, testing was done to validate the model proposed by the researcher. A number of experiments were conducted to determine the best configuration of feature types, feature weights and machine learning algorithm to be used.

3.4.2.4 Transition Phase

After construction and validation of the model the prototype was deployed for prediction to monitor other unobserved instances of hate speech on twitter.

3.4.3 Justification for choosing RAD

RAD was chosen as the system development methodology as it enables fast development of high quality systems at a relatively low cost. In addition to this RAD is suitable for development of the system because the project is small scale and scope is highly focused and well defined.

3.5 Research Quality

The performance of text classifiers was evaluated experimentally since the text classification problem is not well defined (Feldman & Sanger, 2007). A testing set that contained labelled examples of hate speech and non-hate speech text that had not been observed by the model during training were used to evaluate the model. The test set was fed into the model for prediction and predicted results compared to the actual target results. From this comparison four categories are identified: True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives

(FN). True positives refers to instances of hate speech text that were correctly identified as hate speech whereas true negatives are instances of non-hate speech text correctly predicted as non-hate speech. False positives are the instances of non-hate speech text incorrectly classified as hate speech whereas False Negatives are instances of hate speech text incorrectly determined to be non-hate speech (Feldman & Sanger, 2007). These four categories can be illustrated in a confusion matrix as in Table 3.4.

Table 3.5: Confusion Matrix

		Actual Class	
		Hate Speech	Non-Hate Speech
Predicted Class	Hate Speech	TP	FP
	Non-Hate Speech	FN	TN

3.5.1 Evaluation Metrics

The four categories: true positive, false positive, false negative and true negative form the basis of the metrics that were used to evaluate the classification model including accuracy, precision, recall and F-Score (Feldman & Sanger, 2007).

Accuracy measures the percentage of inputs in the test set that the model correctly labelled either as hate speech or non-hate speech. It is calculated as in Equation (3.1) (Feldman & Sanger, 2007).

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{N}} \tag{3.1}$$

where N is the size of the test set.

Precision is the ratio of correctly classified documents to the total number of documents classified under a particular category. Precision is a measure of false positives calculated as in Equation (3.2) (Feldman & Sanger, 2007).

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{3.2}$$

Recall is defined as the number of correctly classified documents among all documents belonging to that category. Recall is a measure of false negatives calculated as in Equation (3.3) (Feldman & Sanger, 2007).

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3.3}$$

F-Score is a harmonized mean of precision and recall calculated as in Equation (3.4) (Feldman & Sanger, 2007).

$$\text{F} - \text{Score} = 2 * \frac{\text{Precison} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.4}$$

3.5.2 Visualization of the Model

Tables and graphical representations were used to illustrate the model performance. The tables were used to display and give comparison of accuracy, precision, recall and F- score value in different experiments. A Receiver Operating Characteristic (ROC) curve was drawn to visualize the performance of the classifier. An ROC plots the true positive rate against the false positive rate as the output threshold is varied over the range of all possible values. The true positive rate depicts the sensitivity of the classifier whereas the false positive rate shows the sensitivity of a binary classifier (Hainard et al., 2011). The Area Under the Curve (AUC) of an ROC curve measures performance of a classifier, in terms of the ability of a model to correctly discriminate between classes. A higher AUC value, shows a good ability to classify hate speech and non-hate speech text. The AUC value is maximal at 1.00 where the classifier doesn't make any error and minimal at 0.5 where the model is considered useless and arbitrarily classifies hate speech text, in this case the ROC curve is aligned diagonally (Hainard et al., 2011).

3.6 Model Validation

To validate the researcher's approach, a number of experiments were used to determine if the best combination of feature types, feature weighting and machine learning algorithms was used to train the model for hate speech identification. The machine learning algorithms considered in the experiments were SVM, Naïve Bayes and k-Nearest Neighbor. Basic count and tf-idf were

considered for feature weighting and n-grams used as feature types, specifically unigrams, bigrams and trigrams.

3.6.1 Experiment 1: Use of Different Machine Learning Algorithms

The aim of this experiment was to compare the performance of the linear SVM model proposed by the researcher with two other machine learning algorithms (Naïve Bayes and k-Nearest Neighbor) using the same tf-idf feature weighting on the same training and testing sets.

3.6.2 Experiment 2: Use of Different Feature Types and Weighting Schemes

The aim of this experiment was to determine the effect of using different combination of feature types with different weighting schemes on the SVM model. The three features types used included: unigrams, bigrams and trigrams, whereas the feature weighting schemes considered were basic count and tf-idf. This experiment was repeated for both Naïve Bayes and kNN to determine which combination produces the best results, as experiment 3 and 4 respectively.

Chapter 4: System Design and Architecture

4.1 Introduction

This chapter describes the overall architecture and detailed design of the proposed prototype by incorporating various requirements. UML diagrams were used to: describe the overall architecture of the system; give detailed descriptions of the various components of the system and illustrate interaction between the users and various components of the system. To achieve this, various design diagrams were developed including: a depiction of the system architecture, use case diagram followed up with comprehensive use case descriptions, sequence diagrams, context diagrams and data flow diagrams.

4.2 Requirement Analysis

This research aimed at developing a model for monitoring hate speech on twitter. Based on this objective, this section outlines the various requirements to be provided for by the proposed solution.

4.2.1 Functional Requirements

i. The application should allow a user to enter keywords to be used as search parameters in the retrieval of tweets from the Search API.
ii. The application should retrieve tweets from Twitter using the Twitter Search API matching the keywords specified by the user.
iii. The application should preprocess the retrieved tweets to clean them and store them in a csv file.
iv. The application should perform feature weighting and represent the tweet in a document term matrix suitable for machine learning. The feature weighting should be done in a similar manner to the one used in training the model.
v. The application should classify the tweet as hate speech or not hate speech using the support vector machine model already trained.
vi. The application should display to the user the tweets labelled as hate speech.

4.2.2 Non-Functional Requirements

4.2.3 Usability

The intended users of the proposed solution are the Information and Communication Technology (ICT) staff at NCIC. It is intended that the interaction between these users and the model shall be simple to allow them collect data from twitter easily and view prediction results.

4.2.4 Scalability

If there is an increase in the amount of twitter posts matching user keywords searched, the proposed solution should be able to handle the extra load, collecting the tweets and predicting their labels without breaking down.

4.2.5 Persistent Storage

The system should provide permanent storage for tweets identified as hate speech. Such tweets may be used as required as evidence in prosecuting hate speech and as such must be retrievable as and when needed.

4.3 System Architecture

The system architecture shows the general layout of the twitter hate speech monitoring prototype and the components it is made up of as illustrated in Figure 4.1. The hate speech detection process begins with the user entering a keyword to be used to retrieve matching tweets. The tweets collector module receives the keyword and collects tweets matching the keyword from the Twitter Search API and stores them in a database. The tweets collected are then preprocessed to clean the tweets. The cleaned tweets are then transformed to a document-term matrix suitable for machine learning using the feature transformer module. The transformation used is the same one as the one used in training the classifier. The document-term matrix is then input into the SVM classifier module and the tweet classified as hate speech or not. Labelled tweets are then presented back to the user and the same persisted in storage.

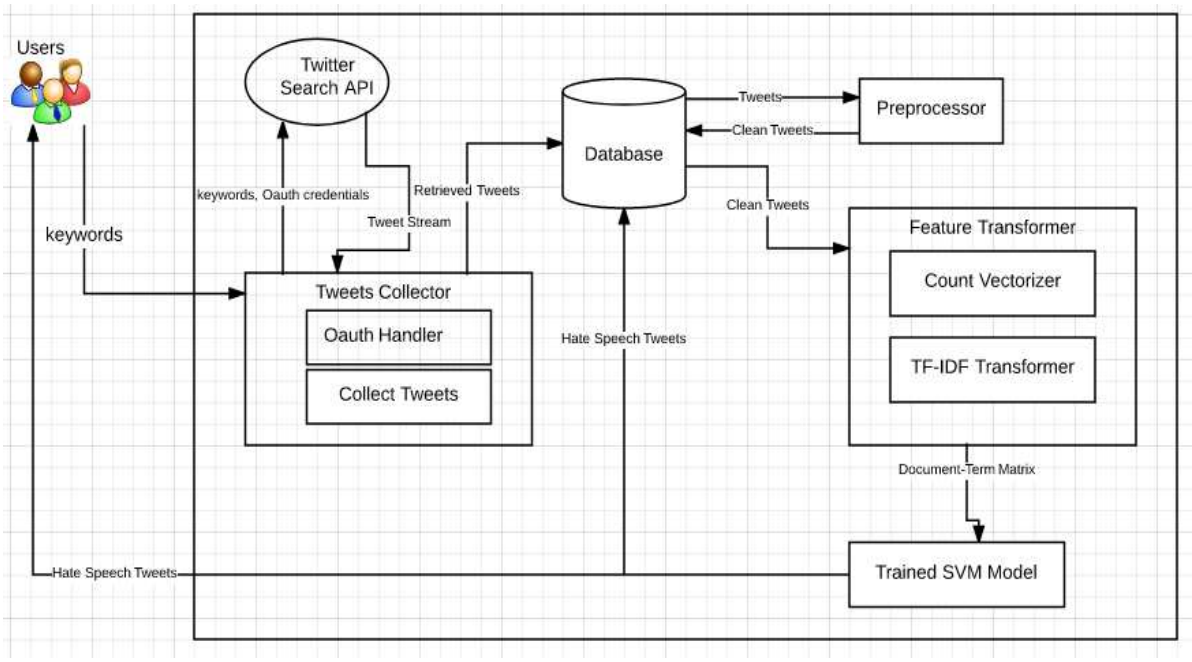


Figure 4.1: System Architecture

4.4 Use Case Diagram

Use case diagrams are used to illustrate interaction between actors and the system. Figure 4.2 illustrates these interactions between the various actors and the proposed hate speech detection prototype. The diagram also depicts the functionality that the proposed system should have.

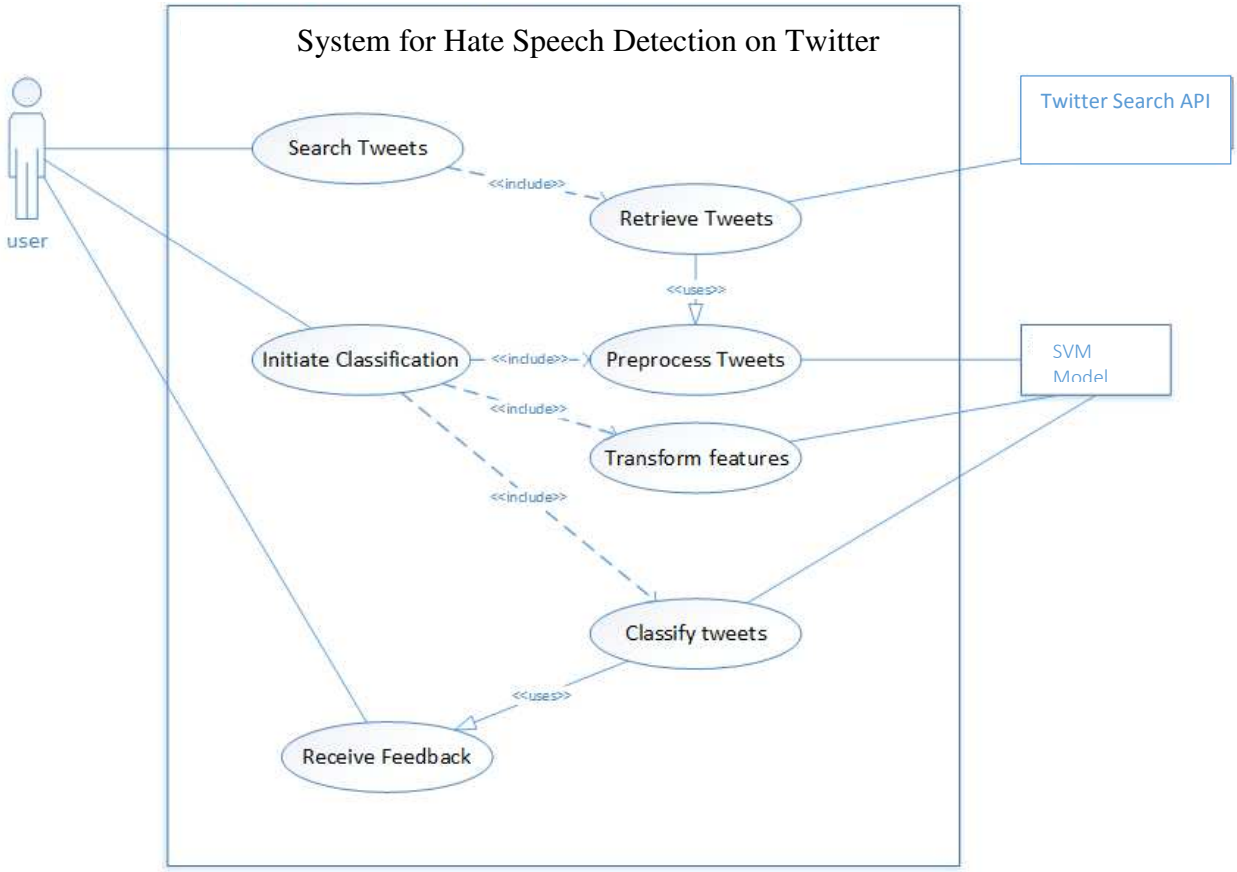


Figure 4.2: Use Case Diagram

4.4.1 Detailed Use Case Descriptions

This section provides comprehensive descriptions for the use cases in Figure 4.2 in a two-column fully dressed format.

Use case: Search Tweets, Retrieve Tweets
Primary Actors
User
Twitter Search API
Preconditions
Search Tweets use case completed successfully
User has access to internet on platform being used
Post conditions
System fetches tweets from Twitter Search API matching the keywords provided
Main Success Scenarios

Actor Intention	System Responsibility
1. User enters the keywords to be used	
	2. Pass keywords entered as parameter to be used to retrieve tweets

3. Fetch tweets from Twitter Search API using the parameters entered
4. Save tweets collected

5. View tweets collected

Extensions

At any time the system fails to retrieve tweets:

Confirm that there is internet access

Restart the system

Use case: Preprocess Tweets, Transform Features, Classify Tweets

Primary Actors

System

User

Preconditions

Tweets were retrieved and stored successfully

Post conditions

Tweet accurately classified as hate speech or not

Main Success Scenarios

	Actor Intention	**System Responsibility**
1.	User initiates classification	
2.		Preprocesses the tweets to clean them as per the SVM model
3.		Transforms tweets to document-term matrix format as per the SVM model
4.		Classify tweets as hate speech or not using SVM model

Use case: Receive Feedback

Primary Actors

User

Preconditions

Successful classification of the tweets by the system

Post conditions

User views tweets labelled as hate speech

Main Success Scenarios

	Actor Intention	**System Responsibility**
1.	User requests results of classification	
2.		Return the output of classification
3.		Display time taken
4.	Exit the system	

4.5 Sequence Diagram

The sequence diagram depicted in Figure 4.3 shows the sequence of interactions between the user and the proposed system as well as interactions between the various internal components of the system. The user enters keywords to be used as search parameters for twitter through the web platform. Once the keywords are obtained, they are passed on to the Twitter Search API which returns JSON results that are saved into a csv file. The user can then initiate classification of the retrieved tweets. In so doing, the web platform passes the message cleantweet() to the preprocessor which preprocesses the tweets and returns clean tweets. The getfeatures() message is passed to an instance of the FeatureExtractor which consequently returns a document term matrix. The document term matrix is then passed to an instance of classifier using the classifytweet () message and a classification of hate speech or non-hate speech returned. The cleantweet (), getfeatures () and classifytweet () messages are repeated for all retrieved tweets. The user can finally request for classification results for all the tweets retrieved.

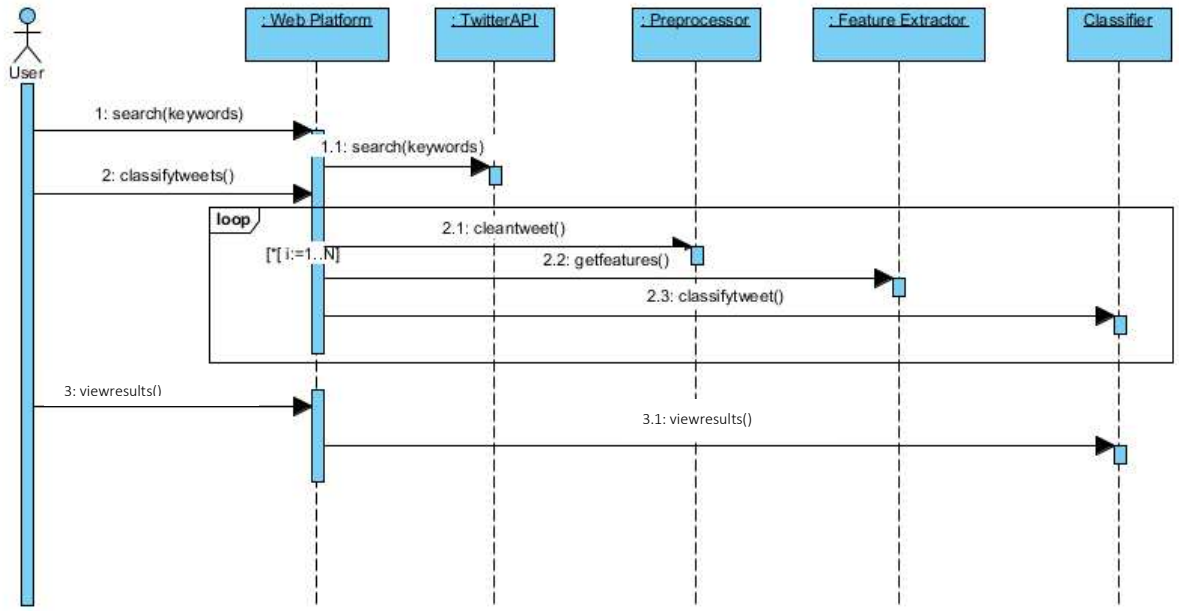


Figure 4.3: Sequence Diagram

4.6 Context Diagram

The context diagram as depicted in Figure 4.4 illustrates the boundary of the prototype, its environment and the entities that interact with it. It also shows the various inputs and outputs from the prototype to the entities. The main entities interacting with the proposed prototype are a user

and the Twitter Search API. The user gives a download request that contains keywords to be matched by the Twitter Search API. The model passes the requests to the Twitter Search API which requires verification of the application to use the API. The model provides confirmation of API details to finalize the connection. The Search API retrieves tweets matching the download request and returns them to the model, which in turn returns the retrieved tweets to the user. The user issues a prediction request to the hate speech detection model which classifies the tweets and returns the labels of the tweets back to the user.

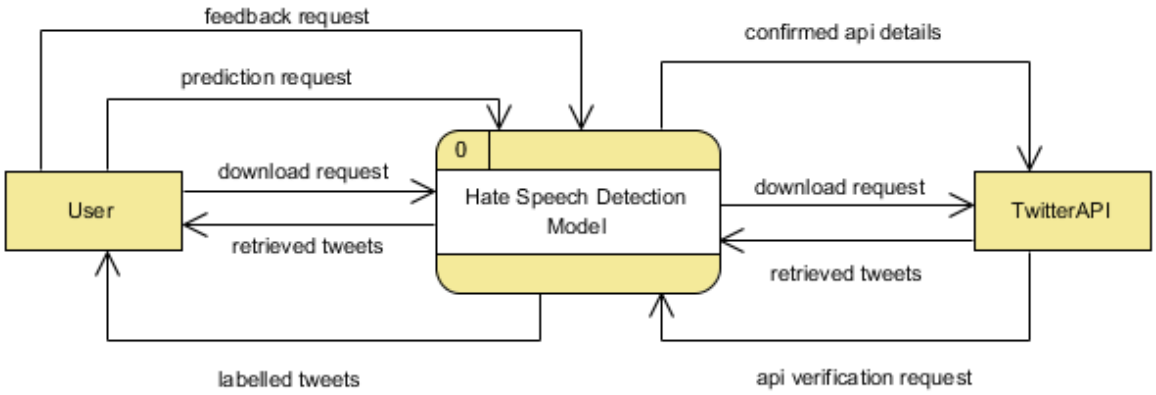


Figure 4.4: Context Diagram

4.7 Level 0 Data Flow Diagram

The level 0 Data Flow Diagram (DFD) depicted in Figure 4.5 gives a more detailed view of the prototype by illustrating the various processes contained in the module, data stores and entities. Arrows depict the flow of data among various components of the DFD. Process 1 called Collect Tweets receives a download request from the user and passes the request to the Twitter Search API entity. The API upon successful validation of API credentials returns retrieved tweets matching the download request. Process 1 stores the retrieved tweets in data store D1 called retrieved tweets. Process 2 (Clean Tweets) receives a prediction request from a user, reads tweets from data store D1 and cleans them. The cleaned tweets are then stored in data store D2 known as clean tweets. Process 3 known as Extract Features reads clean tweets from data store D2 (clean tweets), extracts and weighs the tweet producing a document term matrix that is suitable for machine learning. The document term matrix is fed into process 4: Classify Tweets which labels

the tweets as hate speech or not hate speech and writes the labelled tweets to data store D3 called labelled tweets. Process 5 (Give Feedback) receives a feedback request from the user, reads the labelled tweets from data store D3 and outputs the labelled tweets to the user.

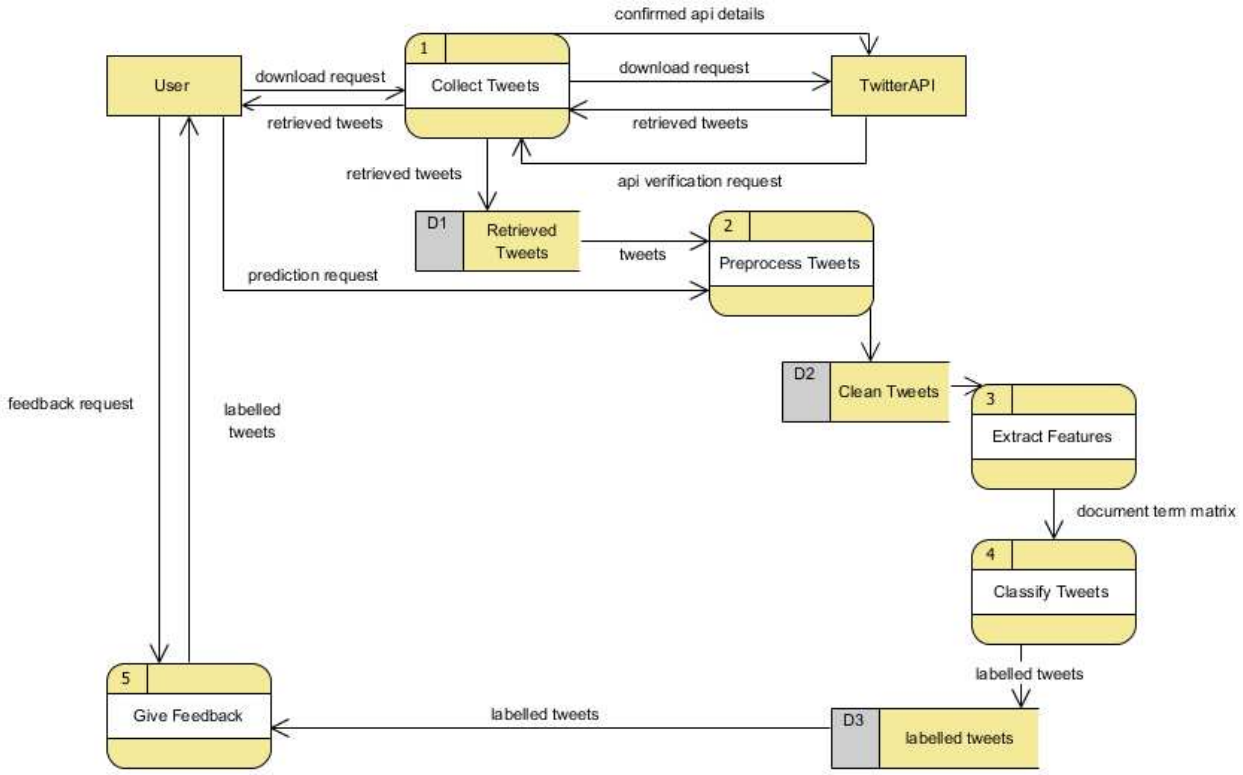


Figure 4.5: Level 0 DFD

Chapter 5: System Implementation and Testing

5.1 Introduction

This chapter describes how the prototype was implemented, tested and validated. It begins by describing the process of building a hate speech corpus for machine learning. The preprocessing process is then discussed after which the model can be trained. The model is then tested against the test set and obtains an accuracy of

To further validate the researcher's approach experiments described in Chapter 3 were implemented to determine the best configuration of feature types, feature weighting and machine learning algorithm for detecting hate speech on twitter. The best performance is achieved when SVM model is used with bigrams features weighted using tf-idf. The final section of this chapter describes the use of the model in predicting other unobserved tweets.

5.2 Building the Corpus

As detailed in section 0, tweets were collected using Jefferson Henrique's open source tool using keywords provided by NCIC. Figure 5.1 illustrates the collection of tweets using Jefferson Henrique's tool.

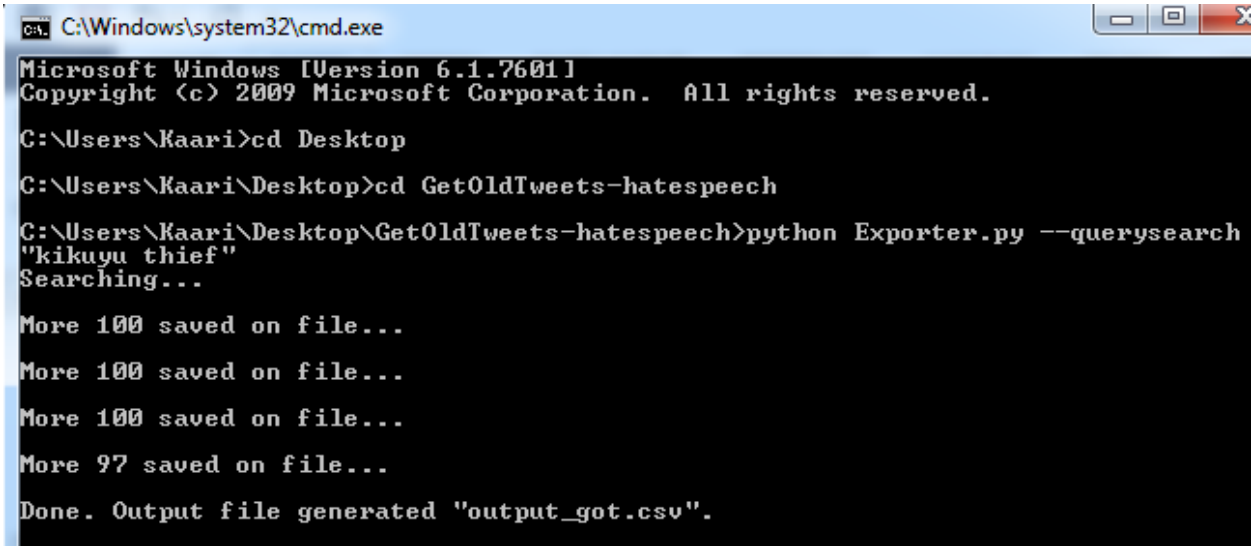
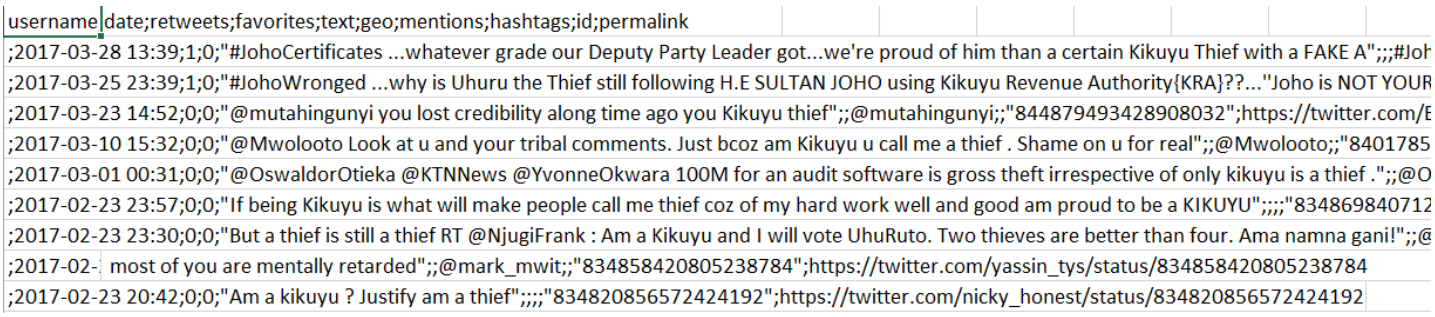


Figure 5.1: Collecting Historical Tweets from Twitter

5.3 Preprocessing

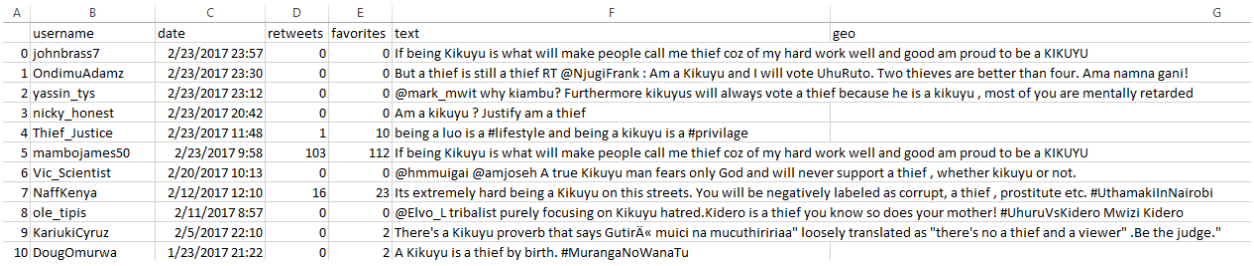
The data collected for training was in an unstructured format unsuitable for application of machine learning techniques. As such it was imperative to preprocess the data before passing it to the training model. Some of the raw unstructured data collected is as depicted in Figure 5.2 below.



```
username;date;retweets;favorites;text;geo;mentions;hashtags;id;permalink
;2017-03-28 13:39;1;0;"#JohoCertificates ...whatever grade our Deputy Party Leader got...we're proud of him than a certain Kikuyu Thief with a FAKE A";;;#Joh
;2017-03-25 23:39;1;0;"#JohoWronged ...why is Uhuru the Thief still following H.E SULTAN JOHO using Kikuyu Revenue Authority{KRA}??..."Joho is NOT YOUR
;2017-03-23 14:52;0;0;"@mutahingunyi you lost credibility along time ago you Kikuyu thief";;@mutahingunyi;;"844879493428908032";https://twitter.com/E
;2017-03-10 15:32;0;0;"@Mwolooto Look at u and your tribal comments. Just bcoz am Kikuyu u call me a thief . Shame on u for real";;@Mwolooto;;"8401785
;2017-03-01 00:31;0;0;"@OswaldorOtieka @KTNNews @YvonneOkwara 100M for an audit software is gross theft irrespective of only kikuyu is a thief .";;@O
;2017-02-23 23:57;0;0;"If being Kikuyu is what will make people call me thief coz of my hard work well and good am proud to be a KIKUYU";;;;"834869840712
;2017-02-23 23:30;0;0;"But a thief is still a thief RT @NjugiFrank : Am a Kikuyu and I will vote UhuRuto. Two thieves are better than four. Ama namna gani!";;@
;2017-02- most of you are mentally retarded";;@mark_mwit;;"834858420805238784";https://twitter.com/yassin_tys/status/834858420805238784
;2017-02-23 20:42;0;0;"Am a kikuyu ? Justify am a thief";;;;"834820856572424192";https://twitter.com/nicky_honest/status/834820856572424192
```

Figure 5.2: Sample Tweets Collected

Tweets often contain a number of different types of information including usernames, date, retweets, favorites, text, geo, mentions, hashtags, id and permalink. The research was only interested in the text part of the tweet to build the corpora. The collected tweets were therefore processed to separate the various parts as illustrated in Figure 5.3 so as to easily extract the text part of the tweet that is relevant to this study.



	A	B	C	D	E	F	G
		username	date	retweets	favorites	text	geo
0		johnbrass7	2/23/2017 23:57	0	0	If being Kikuyu is what will make people call me thief coz of my hard work well and good am proud to be a KIKUYU	
1		OndimuAdamz	2/23/2017 23:30	0	0	But a thief is still a thief RT @NjugiFrank : Am a Kikuyu and I will vote UhuRuto. Two thieves are better than four. Ama namna gani!	
2		yassin_tys	2/23/2017 23:12	0	0	@mark_mwit why kiambu? Furthermore kikuyus will always vote a thief because he is a kikuyu , most of you are mentally retarded	
3		nicky_honest	2/23/2017 20:42	0	0	Am a kikuyu ? Justify am a thief	
4		Thief_Justice	2/23/2017 11:48	1	10	being a luo is a #lifestyle and being a kikuyu is a #privilage	
5		mambojames50	2/23/2017 9:58	103	112	If being Kikuyu is what will make people call me thief coz of my hard work well and good am proud to be a KIKUYU	
6		Vic_Scientist	2/20/2017 10:13	0	0	@hmmuigai @amjoseh A true Kikuyu man fears only God and will never support a thief , whether kikuyu or not.	
7		NaffKenya	2/12/2017 12:10	16	23	Its extremely hard being a Kikuyu on this streets. You will be negatively labeled as corrupt, a thief , prostitute etc. #UthamakiInNairobi	
8		ole_tipis	2/11/2017 8:57	0	0	@Elvo_L tribalist purely focusing on Kikuyu hatred.Kidero is a thief you know so does your mother! #UhuruVsKidero Mwizi Kidero	
9		KariukiCyruz	2/5/2017 22:10	0	2	There's a Kikuyu proverb that says GutirÃ« muici na mucuthiririaa" loosely translated as "there's no a thief and a viewer" .Be the judge."	
10		DougOmurwa	1/23/2017 21:22	0	2	A Kikuyu is a thief by birth. #MurangaNoWanaTu	

Figure 5.3: Collected Tweets with Separated Columns

On analysis of the text contained in the collected tweets a number of twitter specific terms existed that were not informative in the context of the research. Such observed terms included retweets (RT), mentions (@username), hashtags (#), and URL links and non-utf8 characters in the tweets. These occurrences were unnecessary for building the corpora and were therefore removed using regular expressions as depicted in the code snippet in Figure 5.4.

```
def clean_tweets():
    #reading csv file into panda dataframe
    df=pd.read_csv("kikuyu thief.csv",sep=";", error_bad_lines=False)
    print("CSV file read")

    print("Removing usernames from tweets")
    df.text=df.text.str.replace("@\w*\s?","")#remove usernames from tweets

    print("Removing urls")
    df.text=df.text.str.replace("https?:\/\/.*[\r\n]*","")

    print("Removing hashtags")
    df.text=df.text.str.replace("#\w*","")#removing hashtags

    print("Removing non utf8 characters")
    df.text=df.text.str.replace("[^\x00-\x7F]+","")  #remove non utf8 characters

    print("Removing RTs")
    df.text=df.text.str.replace("RT","",False)

    print("Writing clean CSV")
    df.to_csv("hate\clean_kikuyu thief.csv",encoding="utf8")
```

Figure 5.4: Regex Removal of Common Twitter Terms

Further preprocessing was done to remove punctuation and lower case the tweets. A sample of the clean tweets is as depicted in Figure 5.5. The tweets were then labelled as 1 for hate speech tweet and -1 for non-hate speech tweets to facilitate supervised machine learning as depicted in Figure 5.6.

	username	date	retweets	favorites	text	geo
0	johnbrass	2/23/2017 23:57	0	0	if being kikuyu is what will make people call me thief coz of my hard work well and good am proud to be a kikuyu	
1	OndimuA	2/23/2017 23:30	0	0	but a thief is still a thief am a kikuyu and i will vote uhuruto two thieves are better than four ama namna gani	
2	yassin_tys	2/23/2017 23:12	0	0	why kiambu furthermore kikuyus will always vote a thief because he is a kikuyu most of you are mentally retarded	
3	nicky_hor	2/23/2017 20:42	0	0	am a kikuyu justify am a thief	
4	Thief_Just	2/23/2017 11:48	1	10	being a luo is a and being a kikuyu is a	
5	mambojar	2/23/2017 9:58	103	112	if being kikuyu is what will make people call me thief coz of my hard work well and good am proud to be a kikuyu	
6	Vic_Scient	2/20/2017 10:13	0	0	a true kikuyu man fears only god and will never support a thief whether kikuyu or not	
7	NaffKenya	2/12/2017 12:10	16	23	its extremely hard being a kikuyu on this streets you will be negatively labeled as corrupt a thief prostitute etc	
8	ole_tipis	2/11/2017 8:57	0	0	tribalist purely focusing on kikuyu hatredkidero is a thief you know so does your mother mwizi kidero	
9	KariukiCyr	2/5/2017 22:10	0	2	theres a kikuyu proverb that says gutir muici na mucuthiririaa loosely translated as theres no a thief and a viewer be the judge	
10	DougOmu	1/23/2017 21:22	0	2	a kikuyu is a thief by birth	
11	DavidWar	1/21/2017 21:40	1	0	before today id never heard ordinary kikuyu calling uhuru a thief himself not his government i guess folks really are fed up	
12	TheReUpS	1/17/2017 18:11	6	3	quote of the day there is difference between a thief and his accomplice kikuyu proverb on exclusive to	
13	ElayneOka	1/17/2017 14:54	4	2	stop being blinded by your obsession with your tribe or clan a thief is a thief be heshe luo kikuyu kamba luhya or anything else	
14	Thief_Just	1/3/2017 1:08	0	0	ndikwenda gahakwo muhu nie ngoorera tuhui hope i got that kikuyu right	
15	Olwana	12/28/2016 13:58	15	9	now that you have been crowned a kikuyu chief or thief whichever why dont you vie in central	
16	abedydiko	12/6/2016 8:47	0	0	that does not justify anything a thief is a thief be a kikuyu or non kikuyu	

Figure 5.5: Sample cleaned tweets

a senior police officer is on record warning that tutawamaliza this isnt the first time	1
allah akbar tutawamaliza nyote muslims	1
majority of politicians have madoadoa ya moi was once their teacher he predicted that kanu will rule for a 100 years	1
the man who propagated the phrase toa madoadoa and 41tribes vs 1 is the architect of this hatred by other tribes	1
in nasa dp ruto hakuna madoadoa n tribalism from two tribes nasa is there for change to kenya n kenyans through powers	-1
all our stadiums have madoadoa what are our boys and girls being exposed to really	-1
let me remind u odm called 4 peaceful demos others took the chance kuondoa madoadoa others decided to revenge	-1
bora nione rangi ya orange kwa ballot box madoadoa hapana tambua	1
what do you mean kama kuna madoadoa huyo mtu aende nyumbani ata kama ni governor 047	1
the truth is kalenjins will vote for uhuru pale juu lakini hapa chini kutakuwa na madoadoa ya ccm kanu nasa	1

Figure 5.6: Sample Labelled Tweets

5.4 Training the model

Once the tweets were preprocessed and evaluated, training of the model could proceed. To train the model, the csv file containing preprocessed and labelled tweets is read into a pandas Data Frame. A Data Frame is a two-dimensional labeled pandas data structure made up of columns of potentially different types (Pandas, 2017). The corpus was then randomly split into two sets using the train_test_split method of the scikit-learn library; 70 percent of the corpus was used to train the model whereas the remaining 30 percent was used to test its performance. Before the training data could be passed to the SVM classifier for training, it had to be converted into a document term matrix suitable for learning. A count vectorizer was used to convert the text into a matrix of token counts. Since no dictionary had been defined beforehand and no feature selection method was used, the number of features used was equal to the vocabulary size found by analyzing the data. Tf-idf weighting was then applied to the matrix to transform the counts into tf-idf weights. An SVM classifier could then be applied to the document term matrix.

This work made use of an implementation of SVMs provided by Python's scikit-learn machine learning library. Scikit-learn's CountVectorizer module was used to transform the training data into a matrix of token counts and the TfIdfTransformer used to transform the counts using tf-idf weighting. The vectorization and transformation steps can be combined into a single scikit-learn pipeline. Scikit-learn's pipeline was therefore used to chain the three processes of vectorization, transforming and specifying the classifier into one pipeline. Figure 5.7 shows the SVM implementation with count vectorizer and tf-idf weighting, the function receives the training data X with the labels y, learns and returns an SVM model.

```
104 def createSVM(X,y):
105     svm_clf=Pipeline([('vect',CountVectorizer()),('tfidf',TfidfTransformer()),('svm',SVC(kernel="linear",C=1))])
106     svm_clf=svm_clf.fit(X,y)
107     return svm_clf
108
```

Figure 5.7: SVM Implementation

5.5 Testing the Model

The test set (30 percent of the labelled data) was used to validate the model. The test set was passed to the learnt model to be predicted and the results of the predicted compared with the actual labels on the test data. Consequently, a confusion matrix was output to describe the performance of the implemented model. The confusion matrix is depicted in Table 5.1.

Table 5.1: Confusion Matrix for Implemented Model

	Actual -1	Actual 1
Predicted -1	258	83
Predicted 1	56	175

The values for True positive, true negative, false positive and false negative were determined from the confusion matrix as illustrated in Table 5.2.

Table 5.2: Values from the confusion matrix

True Positive	175
False Positive	56
True Negative	258
False Negative	83

The metrics: accuracy, recall, precision and f-score can then be calculated from the values in Table 5.2 and summarized as in Table 5.3. Accuracy is determined to be 75.69 percent calculated as in Equation (3.2).

Table 5.3: Performance of the SVM model

	Precision	Recall	F-Score	Support
-1	0.82	0.76	0.79	341
1	0.68	0.76	0.72	231
Total/ Average	0.76	0.76	0.76	572

Additionally, an ROC curve was drawn to visualize the performance of the classifier as illustrated in Figure 5.7.

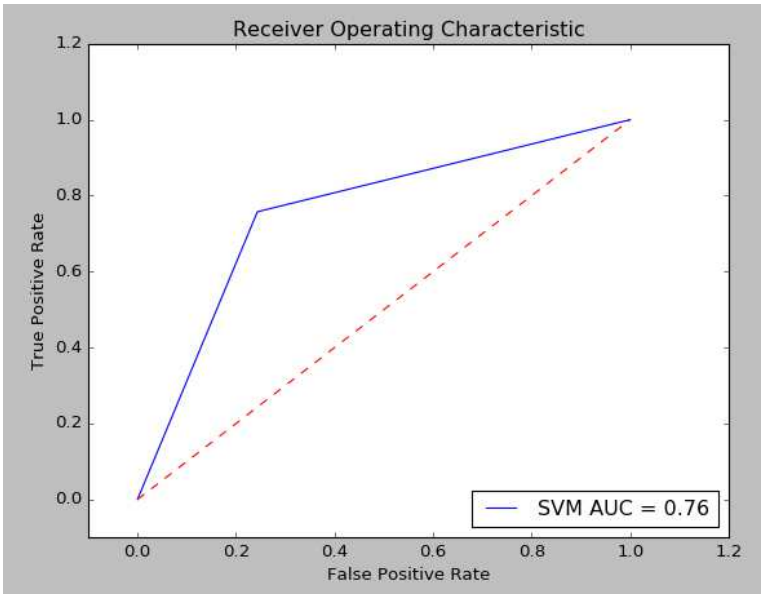


Figure 5.8: ROC curve for the SVM model

5.6 Using the Model in Prediction

Using the model to predict other tweets involves the user entering keywords which are used to retrieve tweets from the Twitter Search API and passing the tweets to the built model for prediction. To avoid retraining the model every time it was required for prediction, the model had to be persisted, this involved dumping the model in a pickle file done using the joblib module in sklearn using the code illustrated in Figure 5.9. The model then needs to be loaded when required to predict retrieved tweets from the Twitter Search API.

```
79 from sklearn.externals import joblib
80 def savemodel(clf):
81     joblib.dump(clf,'model.pkl')
```

Figure 5.9: Persisting the learnt model

To retrieve tweets the Twitter Search API was used. An application was created on twitter to obtain the necessary keys for OAuth as depicted in Figure 5.9. Upon successful registration, the user is issued with the following keys to be used in authentication: access key, access secret key,

consumer key and consumer secret key. The prediction module can use the keys to request access to the search API to facilitate the collection of tweets.

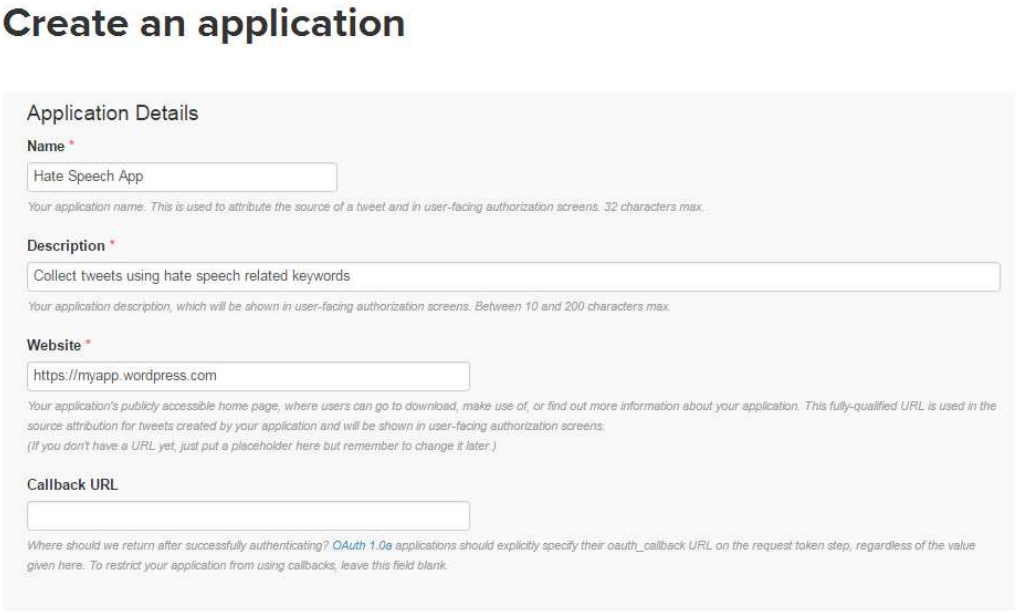


Figure 5.10: Application Registration on Twitter

5.6.1 Collecting Tweets

A user interacts with the application by entering the keywords to be used to retrieve tweets from Twitter using the Search API. Once the user enters the keywords they are passed to the collect tweets module in Appendix C. The module retrieves tweets from the Search API in JSON format and saves them in a text file. Figure 5.11 illustrates the user interface for the web platform to receive keywords from the user.

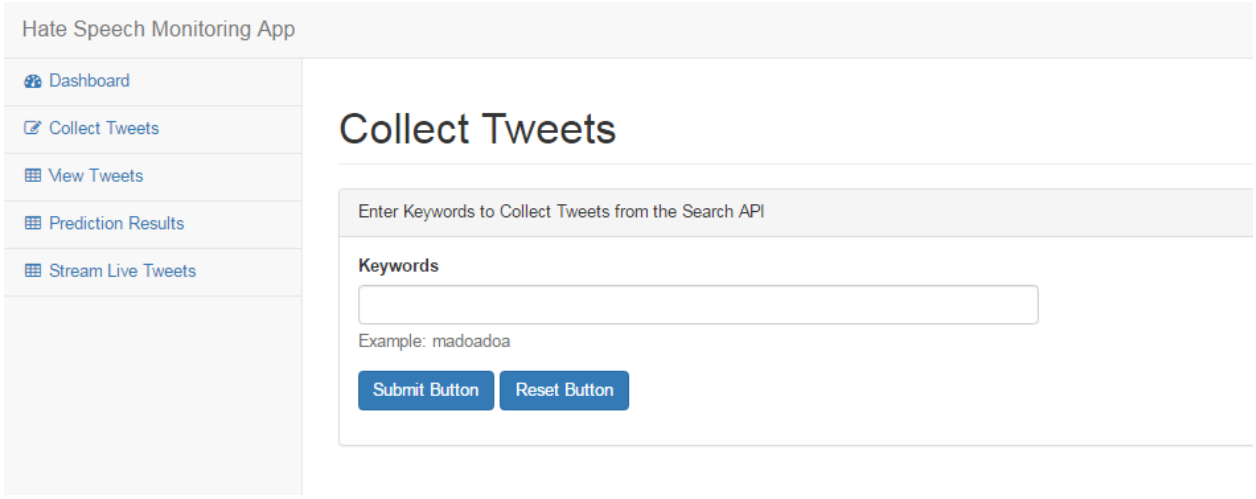


Figure 5.11: User Interface to obtain keywords from the user

5.6.2 Preprocessing Tweets

The collected tweets have to go through a number of preprocessing steps similar to the ones done when training the module. The text file containing the read files is read, the text part of tweets extracted and cleaned to remove hashtags, mentions, URL links and non utf8 characters. The cleaned tweets are then stored in a csv file ready for prediction.

5.6.3 Predicting Labels for Tweets

In this phase, cleaned tweets are read from the csv file and passed to the persisted trained model that needs to be loaded from a pickle file. The model labels the tweets as hate speech or not and returns the results to the user. Figure 5.12 shows a sample of predicted tweets by the system.

Predicted Tweets

Green: Non-Hate Speech... Red: Hate Speech

ID	Text	Label
271	ngmbe ya mama githogori ilipotelea forest ina madoadoa mekundu na	Non-Hate
272	huku kwetu ni suti sita mgombea huru asubiri hadi mwaka wa 2022hatutaki madoadoa	Non-Hate
273	naona mashemeji derby kogallo inatoa madoadoa ya leopards kwa muosho mmoja tu	Non-Hate
274	they were like usikatiwe na madoadoa wtf	Non-Hate
275	hawa waschana wakale hapendi kugawa kwa madoadoa banasijui ka unanielewa	Non-Hate
276	hawa waschana wakale hapendi kugawa kwa madoadoa banasijui ka unanielewa	Non-Hate
277	ulisema hutaki madoadoakuliendaje	Non-Hate
278	chui hawezi kubadili madoadoa yake pia mkushi hawezi kuwa mweupe	Non-Hate
279	kuna mbuzi wa maziwa wa maziwa rangi nyekundu na madoadoa meupi na ako na pembe moja ukimwona nijulishe	Non-Hate
280	out of 10 kikuyu women 13 can potentially kill their current or future husbands john evans michuki	Hate
282	you people who preach kikuyu women kill their husbands stop marrying murderers and putting it on entire community	Hate
303	out of 10 kikuyu women 13 can potentially kill their current or future husbands john evans michuki	Hate
304	raila used kales to kill kikuyuwhen he said no raila no peace then fi	Hate
305	fyi they did not kill the luoluhya economy we killed the kenyan economy voted one mastermind as a governor	Hate

Figure 5.12: Sample results returned from prediction

5.7 Implementation of Experiments

The experiments described in section 3.6 were implemented to validate the researcher's approach in detecting hate speech from twitter. The same scikit-learn library was used to implement the four scenarios: including the use of Naïve Bayes and kNN as the machine learning algorithms and use of different combination of feature types and feature weighting schemes. The code for the implementation of the experiments is in Appendix B and the results of the experiments are discussed in Chapter 6.

Chapter 6: Discussions

6.1 Introduction

This chapter discusses the results of the research in light of the objectives set out at the beginning. The objectives of this research were to develop a hate speech monitoring prototype to detect hate speech from twitter and validate it. An SVM model was created using unigram features and tf-idf weighting, and its performance tested against the test set. A number of experiments were additionally implemented to validate the researcher's approach in detecting hate speech from twitter. The experiments show that the best performance is achieved when bigram features are used with tf-idf weighting.

6.2 Experiment Results

6.2.1 Using Different Classifiers

The aim of this experiment was to compare the performance of the linear SVM model implemented with two other machine learning algorithms (Naïve Bayes and k-Nearest Neighbor) using the same tf-idf feature weighting where a term is considered to be a word (unigram). The results of the experiment are summarized in Table 6.1 and visualized using ROC curves as depicted in Figure 6.1.

Table 6.1: Performance Comparison of Different Classifiers

Classifier	Accuracy	Precision	Recall	F-Score
Linear SVM	0.7569	0.76	0.76	0.76
Multinomial Naïve Bayes	0.7342	0.73	0.73	0.73
KNN	0.7010	0.70	0.70	0.70

The results in Table 6.1 show that the linear SVM model performs better than both the Multinomial Naïve Bayes and kNN across all the metrics (accuracy, precision, recall and f-score). Similarly, ROC curves in Figure 6.1 show that the linear SVM outperforms both kNN and Naïve Bayes with an AUC of 0.76.

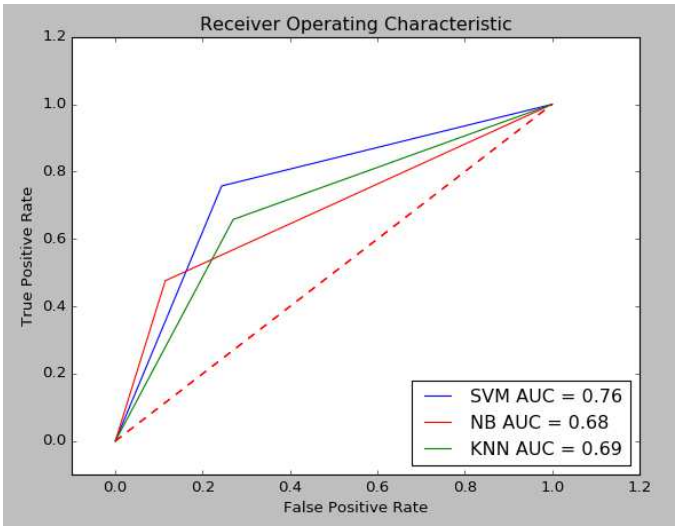


Figure 6.1: ROC Comparison of Different Classifiers

6.2.2 Experiment 2: SVM performance using various feature types

The aim of this experiment was to determine the effect of using different feature types and weighting schemes on the SVM model. The three features types considered included: unigrams, bigrams and trigrams, whereas the feature weighting schemes considered were basic count and tf-idf. The results of the experiment show that the SVM model performs best on the hate speech data when bigram features are considered with tf-idf weighting as depicted in Table 6.2. The results also show that better results are obtained when tf-idf weighting is used for each of the three features (unigram, bigram and trigram)

Table 6.2: SVM Performance Using Different Features and Weighting Schemes

Feature	Weighting	Accuracy	Precision	Recall	F-Score
Unigram	counts	0.7412	0.74	0.74	0.74
	Tf-idf	0.7569	0.76	0.76	0.76
Bigram	counts	0.7534	0.75	0.75	0.75
	Tf-idf	0.7622	0.77	0.76	0.76
Trigram	counts	0.75	0.75	0.75	0.75
	Tf-idf	0.7587	0.77	0.76	0.76

6.2.3 Experiment 3: Naïve Bayes with Different Feature Types

This experiment was similar to experiment 2 and was set up to determine the performance of a Naïve Bayes classifier on the hate speech data given different feature types and weighting schemes. The text documents were represented using unigram, bigram and trigram features and

count and tf-idf used for feature weighting. The results in Table 6.3 show that the best performance of the Naïve Bayes classifier is obtained when bigram counts are used.

Table 6.3: Naive Bayes Performance Using Different Features and Weighting Schemes

Feature	Weighting	Accuracy	Precision	Recall	F-Score
Unigram	counts	0.7290	0.74	0.73	0.73
	Tf-idf	0.7342	0.73	0.73	0.73
Bigram	counts	0.7517	0.76	0.75	0.75
	Tf-idf	0.7202	0.72	0.72	0.71
Trigram	counts	0.75	0.75	0.75	0.75
	Tf-idf	0.7167	0.72	0.72	0.70

6.2.4 Experiment 4: KNN with Different Feature Types

Having established that tf-idf weighting improves the performance of the SVM model on the hate speech data, this experiment was set up to determine whether the same effect is had on a kNN classifier. The text documents were represented using unigram, bigram and trigram features and count and tf-idf used for feature weighting. kNN performs best when bigram features are used with tf-idf, closely followed by the use of unigram features with tf-idf as depicted in Table 6.4.

Table 6.4: kNN Performance Using Different Features and Weighting Schemes

Feature	Weighting	Accuracy	Precision	Recall	F-Score
Unigram	Counts	0.6608	0.65	0.66	0.65
	Tf-idf	0.7010	0.70	0.70	0.70
Bigram	counts	0.6726	0.64	0.63	0.55
	Tf-idf	0.7027	0.70	0.70	0.70
trigram	counts	0.6136	0.64	0.61	0.51
	Tf-idf	0.6958	0.70	0.70	0.70

6.3 Discussions

The results described in the preceding sections show that the best approach in creating a model for hate speech detection on twitter from the data collected is the use of an SVM machine learning algorithm with bigram features weighted using tf-idf. This approach yields the best accuracy of 0.7622 in the classification task. Tf-idf weighting is found to improve accuracy across all 3 feature types (unigram, bigram, trigram) for SVM. This is clearly illustrated using Figure 6.2.

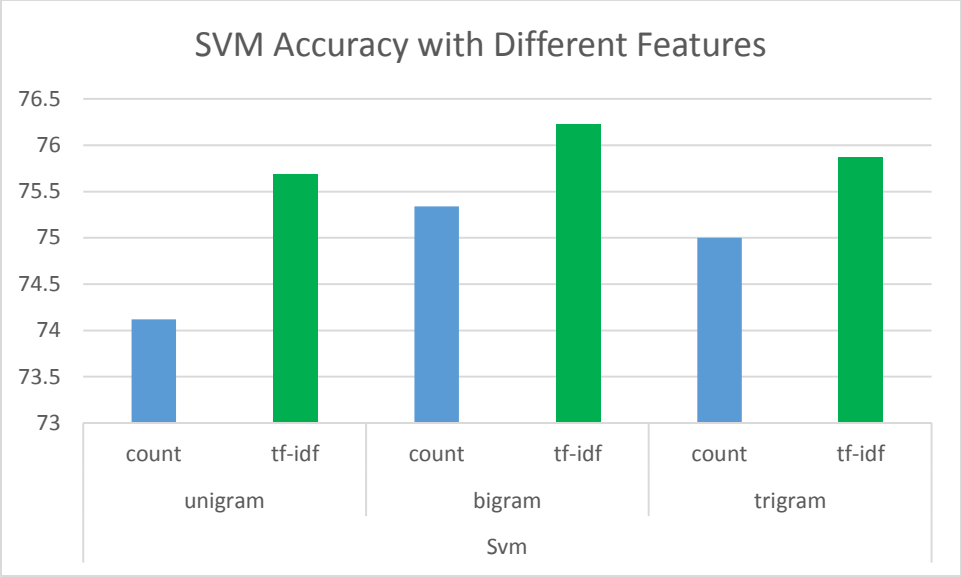


Figure 6.2: Accuracy values for different features for SVM

Naïve Bayes performs best when bigram features are used with count as illustrated in Figure 6.3. Tf-idf weighting in this case does not improve the performance of the NB classifier on the hate speech data.

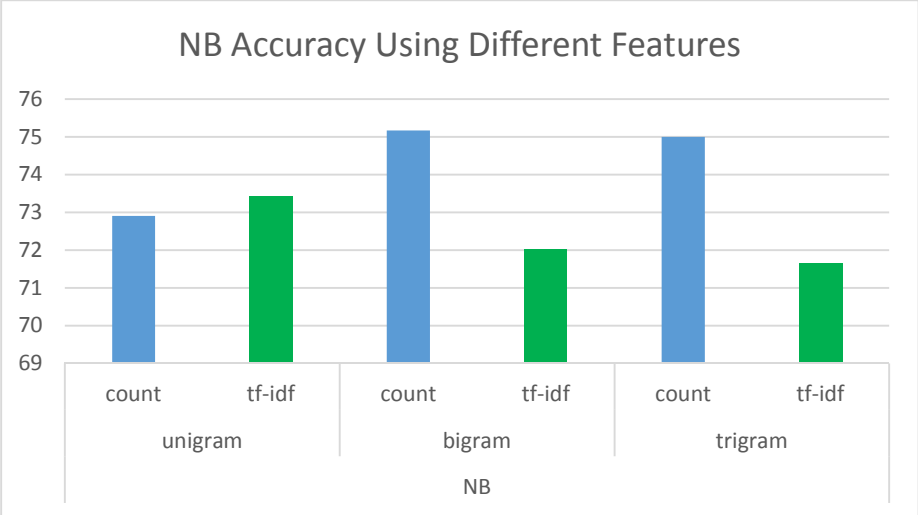


Figure 6.3: Accuracy values NB using different features

kNN classifier accuracy is optimized on the hate speech data collected when bigram features are weighted using tf-idf, similar to the SVM model. kNN accuracy levels with different feature weighting schemes is depicted in Figure 6.4.

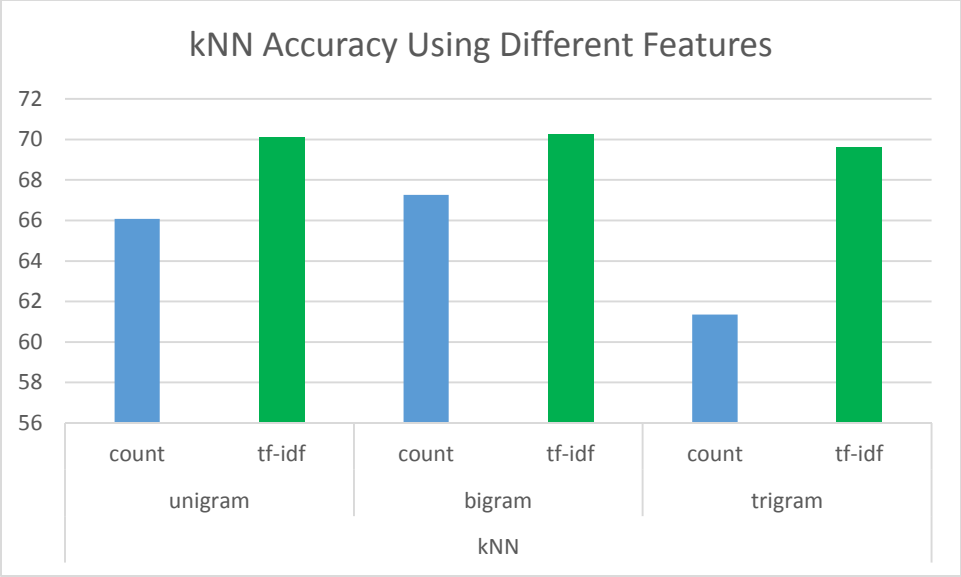


Figure 6.4: kNN accuracy comparison using different features

The results of the study show that the SVM model outperforms the NB and kNN model in detecting hate speech from twitter. Additionally, bigrams can be used to improve the performance of SVMs in detecting hate speech in tweets consistent with Ogada et al., (2015) who show that ngrams can be used to improve the perfomance of naïve bayes, SVMs and kNN. For the data collected a combination of bigrams with tf-idf weighting on an SVM machine learning algorithm have the best performance with an accuracy of 76.22. As such this combination was persisted and used in predicting other unobserved tweets collected by the user.

Chapter 7: Conclusions and Recommendations

7.1 Conclusion

This research intended to develop a tool to detect hate speech on twitter using machine learning techniques. To enable successful execution of the research it was necessary to understand what hate speech is and its occurrence and manifestation on social media platforms. To achieve this relevant literature was reviewed and experts interviewed to determine the challenges encountered in the current processes of detecting hate speech on social media. Further literature was reviewed to understand the application of various machine learning techniques in text classification and hate speech detection.

The main objective of this research was to develop a hate speech monitoring tool for twitter to automatically detect instances of hate speech based on an SVM machine learning algorithm. Hate speech data was collected, preprocessed and labelled using guidelines provided by the NCIC, and further split into training and test sets. The training set was used to train an SVM model suing unigram features with tf-idf weighting. The testing set was used to test the model which achieved an accuracy of 75.69 percent.

To validate the approach taken by the researcher, four experiments were conducted to determine the best combination of feature types, feature weighting schemes and machine learning algorithms (SVM, NB, kNN are considered) to be used to detect hate speech on twitter. The results of the experiment show that the best performance is achieved when an SVM model is used with bigram features weighted using tf-idf, with an accuracy of 76.22 percent. This model was then persisted and deployed in the prediction of unobserved live tweets obtained from twitter using the search API.

7.2 Recommendations

This work showed that the SVM model can be used to automatically detect hate speech on twitter instead of the human analysis of tweets currently employed by NCIC officers; significantly improving the amount of time taken to identify hate speech tweets and reducing the number of tweets that human monitors have to go through.

The research notes that better prediction results would have been obtained if a large data set had been used. A total number of 14055 tweets were collected but only1904 tweets were used

due to the expensive nature of the labelling process. From the 1904 labelled tweets 1332 (70 percent) tweets used to train the model and the remaining 572 used to test and validate the model. The final training data was not large, the researcher therefore recommends that the size of the training data to be increased by collecting and labelling more tweets to improve the performance of the classifier.

7.3 Future Work

Twitter limits the characters that can be used in tweets to 140. In a bid to convey messages within the 140 character limit, users on twitter have to be innovative in the words they use, which results in a number of abbreviations, and informal twitter specific terms. This nature of the language used on twitter makes the classification process difficult. Future research can focus on preprocessing and cleaning tweets to formulate complete statements before classification can be performed.

Hate speech in Kenya can be expressed on twitter in more than one language. This study limited its scope to only hate speech expressed in English and Swahili. For a more comprehensive detection tool, it is necessary to build on the corpus by including tweets expressed in other different languages. Additionally, tweets often contain images and videos which may also contain expressions of hate speech. Future research could focus on such kind of content.

In future research, sentiment weighting can be applied to the tweets to determine levels of negativity and positivity which can be used to detect hate speech. There already exists English lexical resources for sentiment weighting but none for Swahili. Future work could involve the building of such a lexical resource for Swahili to facilitate sentiment weighting.

References

Baccianella, S., Esuli, A., & Sebastiani, F. (2010). SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. *International Language Resources and Evaluation*, (pp. 2200-2204).

Bai, J., Nie, J.-Y., & Paradis, F. (2004). Using Language Models for Text Classification. *Asia Information Retrieval Symposium.* Montreal.

Bhatnagar, M., & Singh , K. (2013). Research Methodology as SDLC Process in Image Processing. *International Journal of Computer Applications*, Vol 77 No 2.

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python.* Sebastopol: O'Reilly Media Inc.

Bordens, K., & Abbott, B. (2011). *Research Design and Methods: A Process Approach.* New York: McGraw-Hill.

Bryman, A. (2012). *Social Research Methods.* New York: Oxford University Press.

Cohen-Almagor, R. (2011). Fighting Hate and Bigotry on the Internet. *Policy & Internet*, Article 6.

Creswell, J. (2003). *Research Design: Qualitative, Quantitative and Mixed Methods Approaches.* London: SAGE Publications.

Daily Nation. (2017, January 25). *Ordinary Kenyans Spread Hate Speech in Online Groups*. Retrieved from Daily Nation: http://www.nation.co.ke/oped/Opinion/Ordinary-Kenyans-spread-hate-speech-in-online-groups-/440808-2951054-10wsl1az/index.html

Daniel, J., & James, M. (2000). *Speech and Language Processing.* New Jersey: Prentice Hall.

Ethnologue. (2017, January 20). *Kenya*. Retrieved from Ethnologue- Languages of the World: https://www.ethnologue.com/country/KE

Feldman, R., & Sanger, J. (2007). *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data.* New York: Cambridge University Press.

Gebre, B., Zampieri, M., Wittenburg, P., & Heskes, T. (2013). Improving Native Language Identification with TF-IDF weighting. *Innovative Use of NLP for Building Educational Applications* (pp. 216-223). Atlanta: Association for Computational Linguistics.

Gnip. (2017, January 5). *Gnip Historical*. Retrieved from Gnip: https://gnip.com/historical/

Hainard, A., Robin, X., Turck, N., Tiberti, N., Lisacek, F., Sanchez, J.-C., & Muller, M. (2011). pROC: An Open Source Package for R and S+ to Analyze and Compare ROC Curves. *BMC Bioninformatics*, 77.

Hirsch, S. (2009). *Putting Hate Speech in Context: Observations on Speech, Power, and Violence in Kenya.* George Mason University.

iHub Research. (2013). *Umati Final Report.* Nairobi.

Institute For Security Studies. (2017, February 3). *Hate speech and ethnic tension ahead of Kenya's 2017 elections*. Retrieved from ISSAfrica: https://issafrica.org/amp/iss-today/hate-speech-and-ethnic-tension-ahead-of-kenyas-2017-elections

Jefferson, H. (2017, January 3). *GetOldTweets-python*. Retrieved from Github: https://github.com/Jefferson-Henrique/GetOldTweets-python

Joachims, T. (1998). Text Categorization with Suport Vector Machines: Learning with Many Relevant Features. *ECML '98 Proceedings of the 10th European Conference on Machine Learning* (pp. 137-142). London: Springer-Verlag .

Joachims, T. (1998). Text Categorization with Support Vector Machine: Learning with Many Relevant Features. *Proceedings of the 10th European Conference on Machine Learning* (pp. 137-142). London: Springer-Verlag.

Jumia Kenya. (2016, May 30). *Growth of the Smartphone Market in Kenya*. Retrieved from Jumia: https://www.jumia.co.ke/blog/whitepaper-the-growth-of-the-smartphone-market-in-kenya/

Kilimci, Z. H., & Ganiz, M. C. (2015). Evaluation of classification models for language processing. *Innovations in Intelligent SysTems and Applications (INISTA).*

Liu, B. (2007). *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data.* London: Springer-Verlag .

Makinen, M., & Kuira, M. (2008). *Social Media and Post-Election Crisis in Kenya.* Scholarly Commons.

Maloba, W. (2013). *Use of Regular Expressions for MultiLingual Detection of Hate Speech in Kenya.* Strathmore University.

Manning , C., Raghavan, P., & Schutze, H. (2009). *An Introduction to Information Retrieval.* Cambridge: Cambridge University Press.

Mazzonello, V., Gaglio, S., Augello, A., & Pilato, G. (2013). A Study on Classification Methods Apllied to Sentiment Analysis. *International Conference on Semantic Computing* (pp. 426-431). IEEE.

Munya, I. (2017, February 3). Personal Interview.

National Cohesion and Integration Commission. (2011). *Police Training Manual- On the Enforcement of the Law on Hate Speech.* Nairobi: National Cohesion and Integration Commission.

National Cohesion and Intergration Commission. (2013). *The Use of Coded Language and Stereotypes among Kenyan Ethnic Communities.* Nairobi: NCIC.

National Council for Law Reporting. (2008). *National Cohesion and Integration Act.* Nairobi.

Naz, R., & Khan, M. (2015). Rapid Applications Development Techniques: A Critical Review. *International Journal of Software Engineering and Its Applications*, 163-176.

Nobata, C., Tetreault, J., Mehdad, Y., Chang , Y., & Thomas, A. (2016). Abusive Language Detection in Online User Content. *International Conference on World Wide Web*, (pp. 145-153). Montreal.

Nyambane, O. (2012). *Prosecuting Hate Speech in Kenya.* Nairobi: Academia.edu.

Ogada, K., Mwangi, W., & Cheruiyot, W. (2015). N-gram Based Text Categorization Method for Improved Data Mining. *Journal of Information Engineering and Applications, 5*(8), 35-43.

Orawit, T. (2006). *Rapid Application Development.* Chiang Mai University.

Pandas. (2017, March 21). *Intro to Data Structures*. Retrieved from pandas: http://pandas.pydata.org/pandas-docs/stable/dsintro.html#dataframe

Pandas. (2017, March 21). *Pandas*. Retrieved from Pandas: http://pandas.pydata.org/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Researcch*, 2825-2830.

Pei, M., & Wu, X. (2014). Text classification based on SMO and fuzzy model. *Information Technology and Artificial Intelligence Conference* (pp. 306-310). IEEE.

Peng, F. (2003). Augmenting Naive Bayes Classifiers with Statistical. *University of Massachusetts, Computer Science Department Faculty Publication Series*.

Rana, M., Khalid, S., & Akbar, M. (2014). News Classification Based On Their Headlines: A Review. *Multi-Topic Conference(INMIC)* (pp. 211-216). IEEE.

Sambuli, N., Morara, F., & Mahihu, C. (2013). *Monitoring Online Dangerous Speech in Kenya.* Nairobi: Umati.

Siele, V. (2013). *Coded Language as a Source of Ethnic Conflict in Africa: A Case Study of Kenya.* Nairobi: University of Nairobi.

Swamy, N., Hanumanthappa, M., & Jyothi, N. (2014). Indian Language Text Representation and Categorization using Supervised Learning Algorithm. *International Conference on Intelligent Computing Applications* (pp. 406-410). IEEE.

Thyer, B. (1993). Single-systems Research Design. In R. Grinnell, *Social Work Research and Evaluation* (pp. 94-117). Illnois: F.E. Peacock.

Twitter. (2017, January 2). *Twitter Developer Documentation.* Retrieved from Twitter: https://dev.twitter.com/docs

United Nations Educational, Scientific and Cultural Organization. (2015). *Countering online hate speech.* Paris: UNESCO.

Vijayarani, S., Ilamathi, J., & Nithya, M. (2015). Preprocessing Techniques for Text Mining- An Overview. *International Journal of Computer Science & Communication Networks*, 7-16.

Waseem, Z., & Hovy, D. (2016). Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. *NAACL-HLT* (pp. 88-93). San Diego: Association for Computational Linguistics.

Yang, Y., & Pedersen, J. (1997). A Comparative Study on Feature Selection in Text Categorization. *International Conference on Machine Learning* (pp. 412-420). San Francisco: Morgan Kaufmann Publishers Inc.

APPENDIX A: Originality Report

A Support Vector Machine Approach to Text Classification for Automatic Detection of Hate Speech in Social Media in Kenya: A Case of Twitter

ORIGINALITY REPORT

17%	14%	9%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Strathmore University Student Paper	2%
2	ai.uwaterloo.ca Internet Source	1%
3	www.iro.umontreal.ca Internet Source	1%
4	nlp.stanford.edu Internet Source	1%
5	"Supervised Learning", Web Data Mining, 2007 Publication	<1%
6	www.lri.fr Internet Source	<1%
7	fb.docs.com Internet Source	<1%
8	Submitted to Loughborough University Student Paper	<1%

APPENDIX B: Interview Guide

The following interview guide was used to in a personal interview with a staff members of the NCIC to find out the challenges faced in monitoring hate speech on social media.

1. Do you currently monitor hate speech on social media?
    a. If yes, how do you monitor hate speech on social media?
    b. If no, why do you not monitor hate speech on social media? Are there plans to start monitoring?
2. How do you monitor hate speech on social media?
3. How do you collect hate speech data from social media? Which search parameters are used?
4. Which social media sites do you monitor?
5. How often do you monitor the social media sites for hate speech?
6. How do you analyze collected data from social media?
7. What do you define as hate speech?
8. How do you deal with the multilingual nature of tweets?
9. Do you have any systems in place to help with monitoring hate speech on social media? If yes, which systems?
10. Which are the most frequent terms found in hate speech text?
11. What challenges do you face in monitoring hate speech on social media?
12. Which stakeholders are interested in the monitoring of hate speech on social media?
13. Which other organizations (governmental/non-governmental) do you collaborate with in the detection of hate speech on social media?

APPENDIX C: Python Programs

Training and Validating Module

This module contains the code for reading a csv file containing labelled data, preprocessing the text to clean it, splitting of the data into training and test sets. Converting the text into a document term matrix with tf-idf weighting and finally feeding the matrix into an SVM classifier. The learned model is then evaluated against the training set and finally persisted into a pickle file.

```
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import learning_curve
from sklearn.model_selection import cross_val_predict
from sklearn.feature_selection import mutual_info_classif
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import classification_report
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.externals import joblib

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import random
import string


def readcsv():
    df=pd.read_csv("new.csv",)#read labelled tweets
    df2=df.reindex(np.random.permutation(df.index))
    X=df2.text
    y=df2.label
    return X, y

def drawrocSVM(y_test,y_pred):
    fpr,tpr,threshold=roc_curve(y_test,y_pred)
    print("Drawing")
    roc_auc=auc(fpr,tpr)
    plt.title('Receiver Operating Characteristic')
```

```
    plt.plot(fpr,tpr,'b',label='SVM AUC = %0.2f'%
roc_auc,color='b')
    plt.legend(loc='lower right')
    plt.plot([0,1],[0,1],'r--')
    plt.xlim([-0.1,1.2])
    plt.ylim([-0.1,1.2])
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()


def savemodel(clf):
    joblib.dump(clf,'model.pkl') #persisting the model


def createSVM(X,y):

svm_clf=Pipeline([('vect',CountVectorizer(max_df=0.7)),('tfidf',
TfidfTransformer()),('svm',SVC(kernel="linear",C=1))])
    svm_clf=svm_clf.fit(X,y)
    return svm_clf

def createNB(X,y):

nb_clf=Pipeline([('vect',CountVectorizer()),('tfidf',TfidfTransf
ormer()),('nb',MultinomialNB())])
    nb_clf=nb_clf.fit(X,y)
    return nb_clf


def evaluatemodel(y_pred,y_test):
    print( metrics.confusion_matrix(y_test,y_pred))
    accuracy=metrics.accuracy_score(y_test,y_pred)
    print(accuracy)
    report=classification_report(y_test,y_pred)
    print(report)

def main():
    X,y=readcsv()

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3
)#split data into training and testing sets

    svm_clf=createSVM(X_train,y_train)
    y_pred=svm_clf.predict(X_test)

    print("SVM evaluation")
```

```
    evaluatemodel(y_pred,y_test)
    drawrocSVM(y_test,y_pred)

    savemodel(svm_clf)


if __name__=="__main__":
        main()
```

Experiments Module

This module contains the code of the various experiments carried out in this research as discussed in Chapter 6.

```
# -*- coding: utf-8 -*-
"""
Created on Mon Apr  3 17:30:01 2017

@author: Kaari
"""
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import learning_curve
from sklearn.model_selection import cross_val_predict
from sklearn.feature_selection import mutual_info_classif
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import classification_report
from sklearn.svm import SVC
from sklearn import metrics

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import random
import string

def readcsv():
    df=pd.read_csv("new.csv",)#read labelled tweets
    #df2=df.reindex(np.random.permutation(df.index))
```

```
    X=df.text
    y=df.label
    return X, y

def createSVM(X,y):

svm_clf=Pipeline([('vect',CountVectorizer(max_df=0.7)),('tfidf',
TfidfTransformer()),('svm',SVC(kernel="linear",C=1))])
    svm_clf=svm_clf.fit(X,y)
    return svm_clf

def createNB(X,y):

nb_clf=Pipeline([('vect',CountVectorizer()),('tfidf',TfidfTransf
ormer()),('nb',MultinomialNB())])
    nb_clf=nb_clf.fit(X,y)
    return nb_clf

def drawrocSVM(y_test,y_pred):
    fpr,tpr,threshold=roc_curve(y_test,y_pred)
    print("Drawing")
    roc_auc=auc(fpr,tpr)
    plt.title('Receiver Operating Characteristic')
    plt.plot(fpr,tpr,'b',label='SVM AUC = %0.2f'%
roc_auc,color='b')
    plt.legend(loc='lower right')
    plt.plot([0,1],[0,1],'r--')
    plt.xlim([-0.1,1.2])
    plt.ylim([-0.1,1.2])
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()

def drawrocNB(y_test,y_pred):
    fpr,tpr,threshold=roc_curve(y_test,y_pred)
    print("Drawing")
    roc_auc=auc(fpr,tpr)
    plt.title('Receiver Operating Characteristic')
    plt.plot(fpr,tpr,'b',label='NB AUC = %0.2f'%
roc_auc,color='r')
    plt.legend(loc='lower right')
    plt.plot([0,1],[0,1],'r--')
    plt.xlim([-0.1,1.2])
    plt.ylim([-0.1,1.2])
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()
```

```
def drawrocKNN(y_test,y_pred):
    fpr,tpr,threshold=roc_curve(y_test,y_pred)
    print("Drawing")
    roc_auc=auc(fpr,tpr)
    plt.title('Receiver Operating Characteristic')
    plt.plot(fpr,tpr,'b',label='KNN AUC = %0.2f'%
roc_auc,color='g')
    plt.legend(loc='lower right')
    plt.plot([0,1],[0,1],'r--')
    plt.xlim([-0.1,1.2])
    plt.ylim([-0.1,1.2])
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()

def experiment1(X,y):
    """Different Classifiers"""

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3
,random_state=0)
    #SVM classifier

svm=Pipeline([('vect',CountVectorizer()),('tfidf',TfidfTransform
er()),('svm',SVC(kernel="linear",C=1))])
    svm=svm.fit(X_train,y_train)
    ypred=svm.predict(X_test)
    print("SVM metrics")
    print(metrics.accuracy_score(y_test,ypred))
    print(metrics.classification_report(y_test,ypred))
    drawrocSVM(y_test,ypred)
    #NB classifier

nb=Pipeline([('vect',CountVectorizer(ngram_range=(1,2))),('tfidf
',TfidfTransformer()),('nb',MultinomialNB())])
    nb=nb.fit(X_train,y_train)
    yprednb=nb.predict(X_test)
    print("NB Metrics")
    print(metrics.accuracy_score(y_test,yprednb))
    print(metrics.classification_report(y_test,yprednb))
    drawrocNB(y_test,yprednb)
    #KNN classifier

knn=Pipeline([('vect',CountVectorizer()),('tfidf',TfidfTransform
er()),('knn',KNeighborsClassifier())])
    knn=knn.fit(X_train,y_train)
    ypredknn=knn.predict(X_test)
```

```
    print("KNN evaluation")
    print(metrics.accuracy_score(y_test,ypredknn))
    print(metrics.classification_report(y_test,ypredknn))
    drawrocKNN(y_test,ypredknn)
def experiment2(X,y):
    """Different features with SVM"""

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3
)

    svm=createSVM(X_train,y_train)
    y_pred=(svm.predict(X_test))
    print("Original Accuracy: Unigram with tf-idf")
    print(metrics.confusion_matrix(y_test,y_pred))
    print(metrics.accuracy_score(y_test,y_pred))
    print(metrics.classification_report(y_test,y_pred))
    stop=["haha","lol","lmao"]

svm2=Pipeline([('vect',CountVectorizer(stop_words=stop)),('svm',
SVC(kernel="linear",C=1))])
    svm2=svm2.fit(X_train,y_train)
    ypred2=svm2.predict(X_test)
    print("Just unigram counts Accuracy")
    print(metrics.accuracy_score(y_test,ypred2))
    print(metrics.classification_report(y_test,ypred2))

svm3=Pipeline([('vect',CountVectorizer(ngram_range=(1,2))),('svm
',SVC(kernel="linear",C=1))])
    svm3=svm3.fit(X_train,y_train)
    ypred3=svm3.predict(X_test)
    print("just bigram counts Accuracy")
    print(metrics.accuracy_score(y_test,ypred3))
    print(metrics.classification_report(y_test,ypred3))

svm4=Pipeline([('vect',CountVectorizer(ngram_range=(1,3))),('svm
',SVC(kernel="linear",C=1))])
    svm4=svm4.fit(X_train,y_train)
    ypred4=svm4.predict(X_test)
    print("Trigram counts Accuracy")
    print(metrics.accuracy_score(y_test,ypred4))
    print(metrics.classification_report(y_test,ypred4))

svm5=Pipeline([('vect',CountVectorizer(ngram_range=(1,2))),('tfi
df',TfidfTransformer()),('svm',SVC(kernel="linear",C=1))])
    svm5=svm5.fit(X_train,y_train)
    ypred5=svm5.predict(X_test)
    print("bigram with tfidf Accuracy")
```

```
    #print(metrics.confusion_matrix(y_test,ypred5))
    print(metrics.accuracy_score(y_test,ypred5))
    print(metrics.classification_report(y_test,ypred5))

svm6=Pipeline([('vect',CountVectorizer(ngram_range=(1,3))),('tfi
df',TfidfTransformer()),('svm',SVC(kernel="linear",C=1))])
    svm6=svm6.fit(X_train,y_train)
    ypred6=svm6.predict(X_test)
    print("trigram with tfidf Accuracy")
    #print(metrics.confusion_matrix(y_test,ypred6))
    print(metrics.accuracy_score(y_test,ypred6))
    print(metrics.classification_report(y_test,ypred6))


def experiment3(X,y):
    """Different Feature set with Naive Bayes"""

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3
)

    nb=createNB(X_train,y_train)
    y_pred=nb.predict(X_test)
    print("Original Accuracy")
    print(metrics.classification_report(y_test,y_pred))
    print(metrics.accuracy_score(y_test,y_pred))

nb2=Pipeline([('vect',CountVectorizer()),('nb',MultinomialNB())]
)
    nb2=nb2.fit(X_train,y_train)
    ypred2=nb2.predict(X_test)
    print("Just counts Accuracy")
    print(metrics.classification_report(y_test,ypred2))
    print(metrics.accuracy_score(y_test,ypred2))

nb3=Pipeline([('vect',CountVectorizer(ngram_range=(1,2))),('nb',
MultinomialNB())])
    nb3=nb3.fit(X_train,y_train)
    ypred3=nb3.predict(X_test)
    print("bigram counts Accuracy")
    print(metrics.accuracy_score(y_test,ypred3))
    print(metrics.classification_report(y_test,ypred3))

nb4=Pipeline([('vect',CountVectorizer(ngram_range=(1,3))),('nb',
MultinomialNB())])
    nb4=nb4.fit(X_train,y_train)
    ypred4=nb4.predict(X_test)
    print("Trigram counts Accuracy")
```

```
    print(metrics.accuracy_score(y_test,ypred4))
    print(metrics.classification_report(y_test,ypred4))

nb5=Pipeline([('vect',CountVectorizer(ngram_range=(1,2))),('tfid
f',TfidfTransformer()),('nb',MultinomialNB())])
    nb5=nb5.fit(X_train,y_train)
    ypred5=nb5.predict(X_test)
    #drawrocSVM(y_test,ypred5)
    print("bigram with tfidf Accuracy")
    print(metrics.accuracy_score(y_test,ypred5))
    print(metrics.classification_report(y_test,ypred5))

nb6=Pipeline([('vect',CountVectorizer(ngram_range=(1,3))),('tfid
f',TfidfTransformer()),('nb',MultinomialNB())])
    nb6=nb6.fit(X_train,y_train)
    ypred6=nb6.predict(X_test)
    #drawrocSVM(y_test,ypred5)
    print("trigram with tfidf Accuracy")
    print(metrics.classification_report(y_test,ypred6))
    print(metrics.accuracy_score(y_test,ypred6))



def experiment4(X,y):
    """Different feature sets with KNN"""

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3
)


knn=Pipeline([('vect',CountVectorizer()),('tfidf',TfidfTransform
er()),('knn',KNeighborsClassifier())])
    knn=knn.fit(X_train,y_train)
    ypredknn=knn.predict(X_test)
    print("Original Accuracy: Unigram tfidf")
    print(metrics.accuracy_score(y_test,ypredknn))
    print(metrics.classification_report(y_test,ypredknn))


knn=Pipeline([('vect',CountVectorizer()),('knn',KNeighborsClassi
fier())])
    knn=knn.fit(X_train,y_train)
    ypredknn=knn.predict(X_test)
    print("Unigram counts")
    print(metrics.accuracy_score(y_test,ypredknn))
    print(metrics.classification_report(y_test,ypredknn))
```

```
knn=Pipeline([('vect',CountVectorizer(ngram_range=(1,2))),('knn'
,KNeighborsClassifier())])
    knn=knn.fit(X_train,y_train)
    ypredknn=knn.predict(X_test)
    print("Bigram counts")
    print(metrics.accuracy_score(y_test,ypredknn))
    print(metrics.classification_report(y_test,ypredknn))


knn=Pipeline([('vect',CountVectorizer(ngram_range=(1,2))),('tfid
f',TfidfTransformer()),('knn',KNeighborsClassifier())])
    knn=knn.fit(X_train,y_train)
    ypredknn=knn.predict(X_test)
    print("Bigram tfidf")
    print(metrics.accuracy_score(y_test,ypredknn))
    print(metrics.classification_report(y_test,ypredknn))


knn=Pipeline([('vect',CountVectorizer(ngram_range=(1,3))),('knn'
,KNeighborsClassifier())])
    knn=knn.fit(X_train,y_train)
    ypredknn=knn.predict(X_test)
    print("trigram counts")
    print(metrics.accuracy_score(y_test,ypredknn))
    print(metrics.classification_report(y_test,ypredknn))


knn=Pipeline([('vect',CountVectorizer(ngram_range=(1,3))),('tfid
f',TfidfTransformer()),('knn',KNeighborsClassifier())])
    knn=knn.fit(X_train,y_train)
    ypredknn=knn.predict(X_test)
    print("Trigram tfidf")
    print(metrics.accuracy_score(y_test,ypredknn))
    print(metrics.classification_report(y_test,ypredknn))



def main():
    print("Hello Main method")
    X,y=readcsv()
    print("Experiment One")
    experiment1(X,y)#call Different Experiments


if __name__=="__main__":
```

```
        main()
```

Collect Tweets Module

This module collects tweets from the Twitter Search API matching keywords provided by the user.

```
# -*- coding: utf-8 -*-
"""
Created on Wed Apr  5 15:44:29 2017

@author: Kaari
"""

import tweepy
import csv

consumer_key="eCx6UFJlVXBzkgxliR3m7anK3"
consumer_secret="fAAjkFnNNyUkNxnt0yj1owMLoIoEdKmHm4KHYjKf8BTq17v
acR"
access_token="222455698-
b2sFM3Oavip4CUuR8oR9C6NuXIugelk3i0aSbMA5"
access_secret="U7mtX8StbQZz5HIS0aPSXAyk8cIIBKP7vWwZnxfRvdOep"

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)
api = tweepy.API(auth)

# Open/Create a file to append data
csvFile = open('tweets.csv', 'w')
#Use csv Writer
csvWriter = csv.writer(csvFile)

def getkeywords():
    f=open("mycriteria.dat",'r')
    keywords=[]
    text=f.readline()
    text=text.split(';')
    for t in text:
        keywords.append(t)

    return keywords;


keywords=getkeywords()
#print(keywords)
```

```
csvWriter.writerow(["text"])
print("here now")

for tweet in tweepy.Cursor(api.search,q=keywords).items():
    print (tweet.text)

    csvWriter.writerow([tweet.text.encode('utf-8')])
```

Prediction Module

Once tweets are received from the twitter search API using the collect tweets module, they are cleaned, preprocessed and predicted using the persisted model created at training.

```
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 3 15:26:04 2017

@author: Kaari
"""
import json
import pandas as pd
import string
from sklearn.externals import joblib


def readjson():
    tweets_data=[]
    file=open("tweets.txt",'r')
    for line in file:
        try:
            t=json.loads(line)
            tweets_data.append(t['text'])
        except:
            #print("error")
            continue

    print(len(tweets_data))
    #print(tweets_data)
    df=pd.DataFrame()
    df['text']=tweets_data
    print(df)
    df.to_csv("readtweets.csv",encoding="utf8")
    #df['text']=map(lambda tweet:tweet['text'],tweets_data)
    #print(df.text)
```

```
def clean_tweets():
    #reading csv file into panda dataframe
    df=pd.read_csv("readtweets.csv", error_bad_lines=False)
    print("CSV file read")
    print("Removing RTs")
    df.text=df.text.str.replace("RT","",False)
    df.text=df.text.str.lower()#change tweets to lowercase
    print("Removing usernames from tweets")
    df.text=df.text.str.replace("@\w*\s?","")#remove usernames
from tweets
    print("Removing urls")
    df.text=df.text.str.replace("https?:\/\/.*[\r\n]*","")
#remove url links froms tweets
    #removing hashtags
    print("Removing hashtags")
    df.text=df.text.str.replace("#\w*","")#removing hashtags
    #removing punctuations
    print("Removing punctuations")

df.text=df.text.str.translate(str.maketrans("","",string.punctua
tion))
    #remove non utf8 characters
    df.text=df.text.str.replace("[^\x00-\x7F]+","")

    print("Writing clean CSV")
    df.to_csv("clean_tweets.csv",encoding="utf8")

def predict():
    df=pd.read_csv("clean_tweets.csv")
    df=df.dropna(how='any')
    df=df.drop_duplicates()
    model=joblib.load("model.pkl")
    df['label']=model.predict(df.text)
    print(df.label)
    df.to_csv("predicted.csv",encoding="utf8")
    #print(model.predict(df.text))
    print("read")

def main():
    readjson()
    clean_tweets()
    predict()


if __name__=="__main__":
        main()
```