



UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

MASTER OF SCIENCE IN DISTRIBUTED COMPUTING TECHNOLOGY

**APACHE SPARK BASED BIG DATA ANALYTICS FOR SOCIAL NETWORK
CYBERCRIME FORENSICS**

BY

SIMON MULWA KIIO

REG NO: P53/78939/2015

SUPERVISOR: DR. ELISHA O. ABADE

Project report submitted in partial fulfillment of the Master of Science Degree in Distributed Computing Technology.

NOVEMBER 2017

DECLARATION

The project, as presented in this document, is my original work and has not been presented for any other university award.

Signature: _____

Date: _____

Simon Mulwa Kiio

P53/78939/2015

The Project has been submitted in partial fulfillment of the requirements for the Master of Science Degree in Distributed Computing Technology at the University of Nairobi with my approval as the University Supervisor.

Signature: _____

Date _____

Dr. Elisha O. Abade

School of Computing and Informatics

ACKNOWLEDGEMENT

I wish to convey my appreciation and special thanks to my supervisor Dr. Elisha O. Abade for his dedication and assistance throughout the research process, the members of the panel whose knowledge and experience in this field has been of great help to my research and the whole School of Computing and Informatics for their support that made me deliver in this work.

Special thanks to my family for their love, encouragement and support towards delivery. Lastly, I would like to appreciate my colleagues who supported me to deliver this research project.

ABSTRACT

The anonymity of social networks makes its attractive for cyber criminals to mask their criminal activities online posing a challenge to law enforcers in tracking and uncovering the perpetrators as most evidence is hidden within big data. With this ever-increasing volume of data, forensic analyst faces challenges in investigations involving huge data volumes while at the same time limited by computer processor, memory and storage resources of a single computer node. With increased social media data and the high rate of production, it has become difficult to collect, store and analyze such big data using traditional forensic tools. This study involved the application of apache spark and big data analytic in forensic analysis of social network cybercrimes such as hate speech, cyberbullying and demonstrated the application of data analytics in supplementing the challenges of traditional forensic tools in investigations involving Big Data. The study developed an apache spark based forensic tool to stream and analysis social media data for hate speech and cyberbully cybercrimes while diving to investigate relevant artifacts found on Twitter social network and ways to collect, preserve and ensure authenticity of the evidence. The study employed Naïve Bayes algorithm within Spark ML API to automatically classify and categorize hate speech and cyberbullying found within Twitter social media. The study showed that by generating SHA-256 Hash key for each tweet item within DStreams and storing tweet data together with corresponding Hash key in MongoDB can be used in tweet evidence preservation and authentication. Again, by streaming full tweet Account metadata, the study revealed that such metadata can be used in authenticating the creator, source, date and time for a given hate speech tweet.

Table of Contents

DECLARATION	i
ACKNOWLEDGEMENT	ii
ABSTRACT	iii
LIST OF TABLES	vii
TABLE OF FIGURES	viii
LIST OF ABBREVIATIONS	x
CHAPTER ONE: INTRODUCTION.....	1
1.1 Background.....	1
1.2 Problem Statement	4
1.3 Objectives of the Study.....	5
1.3.1 General Objectives.....	5
1.3.2 Specific Objectives	5
1.4 Research Questions.....	6
1.5 Significance of the Study	7
1.6 Scope of the Study	8
1.7 Assumptions and Limitations of the Study.....	8
CHAPTER TWO: LITERATURE REVIEW.....	9
2 Introduction	9
2.1 Digital Forensics	9
2.1.1 The Digital Forensics Process.....	10
2.2 Big Data Forensics	11
2.2.1 Big Data Attributes	13
2.2.2 Big Data Architecture	15
2.3 Data Mining and Machine Learning.....	16
2.3.1 Data Mining Techniques.....	17
2.3.2 Data Mining Algorithms.....	18
2.4 Classification Algorithms	19
2.4.1 Naive Bayes (Multinomial) Classifier	19
2.4.2 Support Vector Machines (SVM).....	20
2.5 Apache Hadoop.....	21
2.5.1 Hadoop Core Components	22

2.6	Apache Spark	24
2.6.1	Spark Streaming.....	25
2.6.2	Use Cases of Spark/Spark Streaming	26
2.7	MongoDB	27
2.7.1	MongoDB Document Structure	28
2.7.2	MongoDB Connector for Spark.....	30
2.8	Social Networks.....	30
2.8.1	Social Network Structure.....	31
2.8.2	Social Network Analysis (SNA).....	33
2.8.3	Social Network Sites Forensics	34
2.8.4	Legal Challenges to Social Media Evidence Authentication.....	35
2.8.5	Sentiment Analysis	37
2.9	Proposed Solution	38
2.10	Proposed Apache Spark Forensic Tool Conceptual Model	38
2.11	Literature Summary	39
CHAPTER THREE: RESEARCH METHODOLOGY		41
3	Introduction	41
3.1	Research Design.....	41
3.1.1	CRISP-DM Overview.....	42
3.2	Sources of Data and Sample Population.....	44
3.3	Data Collection and Data Collection tools.....	44
3.4	Data Preparation.....	46
3.5	Data Mining Algorithm and Sentiment Classification.....	48
3.6	Data Analysis	48
3.7	System Implementation	49
3.8	Architectural Design	50
3.9	Model Evaluation.....	52
3.10	Ethical Issues	53
3.11	Summary	54
CHAPTER FOUR: DESIGN AND IMPLEMENTATION		55
4	Introduction	55
4.1	Modeling Tools and Techniques.....	55

4.2	Spark Forensic Model Analysis	56
4.3	Forensic Tool Module Analysis.....	57
4.4	Cluster Setup and Configurations	59
4.4.1	Hadoop Yarn Configuration	60
4.4.2	Starting Hadoop Cluster Manger	62
4.4.3	Apache Spark Configuration.....	63
4.4.4	Starting Apache Spark Cluster.....	65
4.5	Twitter API Connection.....	66
4.6	Data Collections.....	69
4.7	Feature Selection.....	70
4.8	Data Preprocessing.....	72
4.9	Training Tweet Labelling	73
4.10	Social Media Evidence Identification.....	75
4.10.1	Evidence Retrieval	76
4.11	Evidence Preservation.....	80
4.11.1	SHA-256 Hash Key Verification	82
4.12	Model Design and Classification.....	85
4.13	Model Deployment	87
4.14	Model Evaluation.....	89
4.15	Model Results and Analysis.....	93
CHAPTER FIVE: CONCLUSIONS AND RECOMMENDATIONS		99
5	Conclusions	99
5.1	Limitations	100
5.2	Recommendations.....	100
5.3	Future Plan	101
REFERENCES		102
APPENDICES		105
Sample Project Code.....		105

LIST OF TABLES

Table 1: Forensic design software and Tools	56
Table 2: Twitter Account Metadata of interest in forensics	76
Table 3: Streamed Twitter JSON Sample Data	79
Table 4: Tweet SHA-256 Verification.....	85
Table 5: Forensic Model Performance Metrics	91
Table 6: Multiclass Label Metrics	91
Table 7: Sample Hate Speech Tweets.....	98

TABLE OF FIGURES

Figure 1: Traditional Digital Forensics Processes	10
Figure 2: Attributes of Big Data (DAVE 2016).....	13
Figure 3: Big Data Variety (Erl, Khattak & Buhler 2016)	14
Figure 4: Big Data Architectural Overview (Sremack 2015).	15
Figure 5: Support Vector Machines sample hyperplane.....	20
Figure 6: Example of Mapreduce Word Count Process.	23
Figure 7: Spark Components (Nandi 2015).....	25
Figure 8: Data sources for Apache Streaming (Frampton 2015).....	26
Figure 9: Spark Streaming Data flow	26
Figure 10: Embedded document data model.	28
Figure 11: Reference Document Structure.	29
Figure 12: Reference Document example.....	29
Figure 13: A Directed Graph and an Undirected Graph (Zafarani, Abbasi & Liu 2014).....	32
Figure 14: Sample Key metadata fields for individual Facebook posts (Patzakis 2012)	36
Figure 15: Proposed Apache Spark Forensic Tool Conceptual Model.....	38
Figure 16: CRISP-DM Methodology (Ncr et al. 1999)	42
Figure 17: Forensic Tool Architectural Design	50
Figure 18: Forensic Tool Module Analysis	57
Figure 19: Core-site.xml Configurations	60
Figure 20: Yarn-site.xml Configurations.....	60
Figure 21: hdfs-site.xml Configurations.....	61
Figure 22: Slaves.xml Configurations	61
Figure 23: Starting Hadoop Cluster Manger.....	62
Figure 24: Hadoop/HDFS cluster resource manager	62
Figure 25: Hadoop HDFS file system.....	63
Figure 26: Spark-defaults configuration	63
Figure 27: Spark-env.sh Configuration.....	64
Figure 28: Spark log4j.properties Configuration	64
Figure 29: Spark Worker configurations (slaves).....	64
Figure 30: Starting Apache Spark Cluster	65
Figure 31: Twitter API Creation.....	66
Figure 32: Twitter Customer Key/Pair	67
Figure 33: Twitter Customer Keys	68
Figure 34: Scala OAuthUtilities Keys	68
Figure 35: OAuthUtilities.Scala Module	69
Figure 36: Twitter Account Schema.....	71
Figure 37: Tweet Cleaning Module	72
Figure 38: Tweet Sentiment Classifier Module	73
Figure 39: Labeled Tweets.....	74
Figure 40: Mongodb Save Function	77
Figure 41: Mongodb Saved Tweet JSON Document	78

Figure 42: Spark Streaming Dstreams	80
Figure 43: Spark Streaming Dstreams RDDs	80
Figure 44: SHA-256 Hash Key Generator	81
Figure 45: Tweet SHA-256 Hash keys JSON Document	82
Figure 46: Spark ML Pipeline.....	85
Figure 47: Spark ML pipeline Naive Bayes Classifier	86
Figure 48: SBT build.sbt.....	87
Figure 49: SBT JAR Package	88
Figure 50: Apache Spark GUI Monitor	88
Figure 51: Spark ML Model Cross Validation	90
Figure 52: Model Confusion Matrix	92
Figure 53: Categorized Hate Speech Tweets Pie Chart	93
Figure 54: Categorized Hate Speech Tweets Bar Chart	94
Figure 55: Tweet Sentiments Classification Pie Chart	95
Figure 56: Hate Speech Classified Tweets	96
Figure 57: Preserved Tweet Sample	97

LIST OF ABBREVIATIONS

API - Application Programming Interface

CRISP-DM - Cross Industry Standard Process for Data Mining

HDFS - Hadoop Distributed File System

HTML - Hyper Text Markup Language

IDS - Intrusion detection systems

IoT – Internet of Things

JSON - Javascript Object Notation

KDD - Knowledge Discovery in Databases

ML - Machine learning

NLP - Natural Language Processing

NLTK - Natural Language Tool Kit

POS - Part of Speech

RDD - Resilient Distributed Dataset

SEMMA - Sample Explore Modify Model and Assess

SHA-1 - Secure Hash Algorithm

SNA - Social Network Analysis

SVM - Support vector machine

CHAPTER ONE: INTRODUCTION

1.1 Background

In the past few years, the world has witnessed an exponential growth in volume of data generated by information systems that has being fueled further by the discovery of smart devices, social networks sites, and internet of things with many devices connected to the internet. This has also seen an increase in cyber threats caused by either individuals or organized criminal groups (OCG) with the intent to break security of information systems. Cybercrime have also increased in frequency and their degree of sophistication has advanced with advancement in technology. With this increasing volume of data, forensic analyst faces challenges when dealing with investigation involving large volumes of forensic data while at the same time constrained by computer processing power in terms of processor, storage and available memory space. Traditional digital forensic tools focus on transactional data commonly known as structured data for forensic analysis that is normally in relational or hierarchical database. Again most widely used traditional forensic tools have not undergone any major architectural change (Edwards 2011) hence they lack suitable features to handle big data forensic investigation.

Traditionally, digital forensic methodologies (identification, preservation, extraction, interpretation, and documentation) would include taking the suspect system offline and removing hard drives from suspected computer system containing source evidence(Press 2010), making bit copy of the original hard disk, calculating MD5/SHA-1 checksums, and performing physical collections that capture all metadata. The forensic analyst would then work from this copy, leaving the original hard disk unchanged. However, big data system limitations prevent investigators from applying these forensic methodologies and as such alternative methods for identifying, collecting, storing, analyzing and documenting such data are required. The discovery of social media sites and its subsequent adoption by people have rapidly created an enabling environment for connection among people, businesses, and organizations hence making people to keep in touch and interact with each another. These sites have enabled an increased content sharing while at the same time building and enhancing relationships as friends and families stay connected. Individuals and groups can share photos or videos and provide status updates that one feels are of interest to their friends.

With people, businesses and organization revealing more personal information and business activities online, social networking sites have often been targeted as a platform for committing crimes, including gang recruitment, identity theft, or online harassment and cyberbullying. The anonymity of social network sites makes its attractive for cyber criminals to mask their criminal activities online posing a challenge to law enforcers in tracking and uncovering the perpetrators. Cyber criminals leave electronic traces as part of their social networks activities and interactions which are contained within an enormous big datasets that are difficult to filter, analyze and correlate evidence using traditional forensic tools (Johnsen 2016). These evidences are not often visible but hidden within large dataset in the form of patterns and correlations. Forensic analysis of social networks and the associated metadata can help forensic investigators understand and solve various cybersecurity problems, including uncovering the online networks of extremists, organized criminal groups, hate speech and cyberbullying (Gupta & Brooks 2013). However, the huge stream of data generated from online social networks calls for research and design of new generation of forensics analytics methods and tools that can effectively process and correlate digital evidence found in big data more often in real-time or near real-time and within digital forensic standards.

Big Data is defined by the three attributes commonly known as 3V's i.e. Volume, Velocity and Variety in which for data to be categorized as Big data, the data must poses the three V's (Berman 2013). With cybercrime increasingly expanding from structured to unstructured data, forensic analysts need new tools and methods to get insights of large volume of data and correlate artifacts from multiples data sources. Forensic data analysis of network traffic, system log files, and financial transactions can be used to correlate evidence from several data sources into a visual representation and uncover suspicious activities or which deviates from the normal (Wang & Alexander 2015). For forensic data analysis, analyst can utilize wide range of data analytical techniques, including network analysis, social network analysis, graph mining and text mining among others depending on the cybercrime under investigation.

Intelligent forensics in conjunction with standard investigation procedures can be used in digital forensics to provide more intelligence of insights on big data and provide evidence correlation (Irons & Lallie 2014). The forensic analysis for network traffic (packets) and system events logs has traditionally faced big challenges as traditional forensic technologies fail to provide the toolset to

support forensics involving big data. This was contributed by the fact that storing and retaining a large log file data was not economically feasible with storage limitations and thus, most event log data and other recorded system activities were overwritten or deleted after a fixed retention period. In this study, we developed an apache spark based big data forensics tool for social network cybercrime detection and systematically analyzed big data from Twitter social network to explore artifacts which are relevant to Twitter social network site forensics. Apache Spark is a distributed computing framework which provides in-memory large data processing while at the same time enabling real times analytics and development of programs that can ran in parallel on different nodes in a cluster of computers (Pentreath 2015). To achieve this, the framework abstracts the tasks of system resource scheduling, job submission, job execution, tracking, and communication between cluster nodes. Again, Apache spark offers excellent large data streaming of live data which makes its suitable for streaming social network data and support big data analysis that can be distributed across nodes within a cluster.

Big Data analytic systems utilizes cluster based computing infrastructures that results to systems that are more reliable, fault tolerant and provide guarantees that queries on the systems are processed to completion. For digital evidence to be admissible before jury, the data must be properly identified, collected, preserved, documented, handled and analyzed using processes that should demonstrate that the data was not changed or altered in any way and adhered to the best practices accepted by a court of law and backed up by technical standards (Sremack 2015).

1.2 Problem Statement

With the proliferation of digital technology, Internet of Things and smart devices, large data now stream every day from social networks, mobile phones, credit cards and computers, city infrastructure and sensors networks among others. This data has grown exponentially resulting to what is commonly known as “Big Data”. With this ever-increasing volume of data, forensic analyst faces challenges in investigations involving huge data volumes from social networking sites while at the same time limited by computer processor speed, memory and storage resources of a single node.

The traditional digital forensic tools focus on transactional data commonly known as structured data for analysis in a relational or hierarchical database. Most widely used traditional forensic tools have not undergone any major architectural change (Edwards 2011) hence lacks suitable features to handle big data forensic investigation. Again, the use of traditional tool to analysis Big Data is time consuming, resources intensive and correlation of evidence from multiple source is not feasible. The ability to derive insights and correlate artifacts found in such big data become difficult using the traditional forensic tools. The range of data from social network sites for forensics increases considerably and increases further with numerous participants involved in social media resulting into challenges in carrying forensics investigation involving these large volumes of data. With this increased social network data, it has become difficult to collect, store and analyze such big data on a single computer node.

In order to collect, store and analyze such data fast and effectively, Apache Spark a leading distributed computing framework come in handy with features that can process voluminous amount of data that can range from terabytes to petabytes of data. Forensic analysis of social networks can help law enforcers understand and solve various cybersecurity problems, including uncovering the social media cybercrimes. The large data and the complex structure of social network sites calls for research and design of new generation of forensics analytics methods and tools that can effectively process and correlate digital evidence found in big data more often in real-time basis. The study developed an apache spark based big data forensics tool for social networks cybercrime detection and systematically analyzed big data from Twitter social network site to identify, collect, preserve and analysis artifacts that are relevant in Twitter social network forensics to supplement the shortcoming of traditional forensic tools in carrying out large data forensic investigations.

1.3 Objectives of the Study

The research project aims to achieve the following key objectives:

1.3.1 General Objectives

The main objective of this research study is to design and develop an apache spark based big data forensics tool for Twitter social network cybercrime forensics using big data analytics and data mining techniques.

1.3.2 Specific Objectives

- (i) To investigate and identify data sources (artifacts) of interest for forensic examiners on Twitter social network and how they can be collected in an automated mode.
- (ii) To investigate how social network forensics data can be collected and preserved to ensure it is authentic before court of law.
- (iii) To investigate techniques for social network analytics in the application for digital forensics.
- (iv) To investigate how big data analytic solutions can be used for social network forensics and more specifically application of apache spark.
- (v) To test and evaluate apache spark based big data forensics tool on apache spark based standalone cluster machine and display available circumstantial evidence found on Twitter social network.

1.4 Research Questions

This thesis provides an automatic spark streaming of Big Data social network forensics for data generated from online social networks. Identification of the research gaps in traditional forensics tools and Big Data forensic challenges in social networks has led to the formulation of the following research questions to help achieve the stated objectives:

- (i) How can big data analytic techniques be used on large volume of data to reveal hidden patterns, correlations and discover other useful forensic information for digital forensics from social networks sites.
- (ii) How can forensic analyst identify traces of criminal activity/misuse behavior by using correlation techniques against data stored on social network like Twitter or Facebook?
- (iii) How can link analysis techniques be used on big data to identify correlation of forensic artifacts on social networks like twitter and Facebook?
- (iv) How can social network traces (artifacts) be identified, collected and preserved to enable its authenticity, validity and admissibility in court of law.
- (v) How can apache spark be used for big data streaming to collect voluminous amounts of social media data and what techniques can be used to analysis and correlate artifacts?
- (vi) How can social network analytics be employed to derive insights and correlation on big data found on online social networks likes Twitter and Facebook.

1.5 Significance of the Study

The anonymity of social networks makes its attractive for cyber criminals to mask their criminal activity online leaving law enforcement agencies ill prepared for new threats from cybercrime. Cyber criminals leave electronic traces as part of their social networks activities and interactions that are contained within enormous big datasets that becomes difficult to filter, analyze and correlate artifacts using traditional forensic tools. Social networks forensics can help law enforcers understand and solve various cybersecurity problems, including uncovering cyberbullying, hate speech mongers, violent terrorists and fraudulent activities. The existing traditional forensics tools have architectural limitations regarding their efficiency and ability to handle increasing big data volumes. In particular, traditional forensic tools are becoming insufficient in handling big data investigation with mostly requiring the forensic examiner to manually review artefacts and relate events to come up with correlation to prove or disapprove commitment of a crime. Again, with increased social network data, it has become difficult to collect, store and analyze such big data on a standalone computer node.

The research study will contribute the following to the body of knowledge:

- a) The study will demonstrate how distributed computing frameworks like Apache Spark can be used in collecting, storing and analyzing big data from social network sites for digital forensics which has become difficult to collect, store and analyze on a standalone computer node.
- b) This research will increase awareness of application of big data analytic solutions and data science techniques within the digital forensic investigators and to show how they can be utilized in solving large data set challenges and supplement traditional forensics tools in investigations involving big data.
- c) The research will also help investigators during Big data forensics to find links between evidences that is hidden within big datasets and which can be easily be overlooked by a forensic investigator especially because of the large amount of data involved.
- d) The forensic tool will help law enforcers in investigation involving social media to uncover and correlated evidence found on suspected cyber criminals like cyberbullies and hate speech mongers.

1.6 Scope of the Study

The research study is intended for developing an apache spark based data streaming forensic tool for social network cybercrime forensics that will help in forensic investigations involving social network sites and help supplement traditional forensic tools in solving big data forensics challenges on social network sites. There are several social networks in existence today including Twitter, WhatsApp, Myspace, Facebook, LinkedIn and Instagram among others. It will not be possible to carry out the study on all social network sites due to time and resource constraints, and therefore big data from Twitter will be used.

1.7 Assumptions and Limitations of the Study

- a) The language used in Twitter sometimes consists of words and phrases that are not formal language (Sheng slang) which makes it difficult to classify sentiments, the study will be limited to phrases made in English.
- b) There exists a lot of cybercrime related to social network sites including spreading hate speech, cyberbullying, Identity theft, Harassment, terrorist recruitment and organized criminal groups among others, the study will be limited to forensics involving identification of hate speech and cyberbullying crimes in Twitter social network.

CHAPTER TWO: LITERATURE REVIEW

2 Introduction

In Chapter one, I introduced the forensic challenges faced by forensic investigators on Big Data forensics and forensic investigation on online social networks. It was noted that the traditional forensic tools face challenges while carrying out forensic analysis involving Big Data. This chapter involves reviewing literature related to the research problem of Big Data analytics, Digital Forensics; Apache Hadoop/Apache spark framework, sentiment and social network site analysis.

2.1 Digital Forensics

With the rapid advancement in information technology like Internet of Things (IoT), smart devices and online social networks, the world has witnessed an increase in cyber threats caused by either individuals or organized criminal groups (OCG) with the intent to break the security of information systems. Cybercrime have also increased in frequency and their degree of sophistication has advanced with advancement in technology. To uncover these cybercrimes and to bring the culprit to book, digital forensic investigation is carried out to extract evidence from seized computer systems. Digital forensics is defined as “the process of using scientifically derived and proven methods toward the identification, preservation, collection, validation, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal or which has led to a particular incident” (Altheide & Carvey 2011).

It seeks to recover data from digital devices like internet, online social networks, network devices, computers, file servers, web servers and smart devices so as to reconstruct events to confirm or deny allegations of commitment of cybercrime or obtain cyber security intelligence information (Beebe et al. 2011). While the goal of any given forensic investigation is to find facts and through chronology of events recreate the truth of events, the integrity of the original data should be persevered and maintained to ensure the evidence is authentic before a jury. The investigator reveals the truth of events by discovering and exposing the artifacts (remnants) of the events that were left behind on the compromised system when the cybercrime committed.

2.1.1 The Digital Forensics Process

Traditional digital forensics typically focuses on more common sources of data, which is mostly unstructured, such as servers, firewall, computers and network. It typically focuses on metadata and involves the calculation of an MD5 hash or SHA-1 hash checksum for checking the integrity of the hard disk image. Digital forensics process involves several steps whose main goals are to identify and locate all relevant data (Identification process), collect the data in a sound manner (collection and preservation process), analysis of the data so as accurately describes the events (analysis process) and present the findings (documentation and presentation).

Digital forensics revolves around evidence and such may be presented before a jury to prove or disprove a claim or issue by logically establishing facts (Sremack 2015). The digital evidence must be admissible before a jury and for the evidence to be admissible; the data should be correctly identified, preserved, collected, documented, handled, and analyzed in a manner that adheres to the established digital forensic standards and procedures. The process by which the forensic data was identified, collected, and handled is critical to demonstrate that the data was not changed or altered in any way during the exercise. The whole process should adhere to the best practices accepted by the court and backed by technical forensic standards.

Finally, documentation (chain of custody) of the entire forensic process should be properly maintained showing the chronology of events from seizure to the presentation and should readily available for presentation. This should clearly demonstrate all the steps performed as shown on figure one below.



Figure 1: Traditional Digital Forensics Processes

2.2 Big Data Forensics

Over the last few years, the data generated globally has increased exponentially with smart devices, social media sites, Internet of Things and sensors increasing it further. This has led to the phenomenon of “Big data” which describes a collection of large datasets that are complex such that it becomes difficult to collect, store and process using traditional data processing applications and management systems. Because of the voluminous nature and velocity of data generation, big data systems require an enabling distributed storage to collect, store and process heterogeneous data collected from multiple sources. Big data comes from several sources and in a variety of forms including structured and non-structured data such as social media data, sensor data, network logs and system logs.

As data volumes explode with increased velocity, Big Data solutions should be designed to handle these voluminous datasets through distributed storage and computing while at the same time capable of scaling up and down in respond to computational demand of the application. The solutions should provide methodologies for collecting, storing and analyzing large amount of data that is not possible to be stored on a standalone computer. To get actionable insights from such large data, data analytics is normally carried out using analytic software tools to discover crime or anomaly patterns and other supporting forensic information. Big data analytics has been employed in analyzing system log files, network traffic, and fraud involving financial transactions to help identify anomalies, suspicious and fraudulent activities. However, analysis involving big data creates several challenges including privacy violation, legal and technical issues regarding data collection, storage and analysis that data scientists/analysts need to handle. In cybersecurity perspective, big data has opened up new possibilities in terms of analytics and security solutions to protect information systems, data and prevent future attacks. Big Data has been applied in distributed analytics by (Wang & Alexander 2015) to analyze log files, network traffic and financial transactions. They used distributed analytics facilitated by Hadoop to correlate information from multiple sources into logical view to identify anomalies and suspicious activities. In the era of cloud computing, large data volume is generated which poses security problem with traditional forensic methodologies and tools becoming inefficient for cloud based system forensics. (Cho, Chin & Chung 2012) developed Hadoop based cloud forensic tool that supported live evidence collection and analysis that decreased the amount of time taken in identifying, collecting and analyzing evidence. The growing cyber threats against information systems has called for deployment of various security monitoring systems to protect the information

systems from cyber-attacks. Large datasets of logs and events are generated by security monitoring systems and intrusion detection systems (IDS) which need an efficient design for collecting, integrating and processing them at a faster rate to get an insights and correlate security threats. This calls for storage efficient system to store the large volumes of data like one designed by (Juturu 2015) which used HBase and Apache Hadoop to collect, stores and retrieve datasets related datasets.

In forensic investigation involving Big Data, terabytes or petabytes of storage data may be involved, and data may be lost when such Big Data systems are brought offline for taking forensic images. In such cases, data collections usually require logical or targeted data collection methods by way of capturing active files and selectively copying specific files. Traditional forensics processes heavily rely on making bit by bit images of original hard disk and calculating of MD5 hash and SHA-1 hash so as to be able validate the integrity of the digital evidence. While this method works well for small volume of data, it is not always feasible to take big data systems offline to take bit by bit disk image and using hashing algorithms to validate the integrity of the forensic data collected. This poses big challenges to forensics investigators that includes collecting, storing and analyzing of such huge volume of unstructured data, handling high velocity data streams and analyzing the data so as to finding out hidden insights and correlations.

The Traditional forensic tools and technologies are incapable of collecting, storing and processing such a huge amount of diverse data hence the need for alternative methods for collecting, storing and analyzing such voluminous data are required. This has been worsened by the adaption of cloud computing which has made computation ubiquitous resulting to challenges in carrying out forensics analysis using traditional ways. To address cloud and social networking forensic challenges, (Chen et al. 2015) proposed digital forensic model targeting cloud and network social network forensics. However, this model did not provide implementation for testing its applicability. (Zawoad & Hasan 2015) developed conceptual model of handling big data for digital forensics based on Hadoop Distributed File System (HDFS) and cloud for supporting reliable digital forensics involving big data. The only drawback with this model is that the model was not tested on ideal system to process this big dataset of spam emails from bounce.io, which could have helped to identify several issues, such as phishing and spam campaigns and the criminals who are behind the strongest phishing/spam campaigns.

2.2.1 Big Data Attributes

There are several specific attributes that define big data and which a dataset must possess for it to be categorized as big data. These attributes are also important in the design and architecture of a big data analytic solution. As shown in figure 2, most of big data solutions these attributes are known as the four V's: volume, variety, velocity, and veracity.

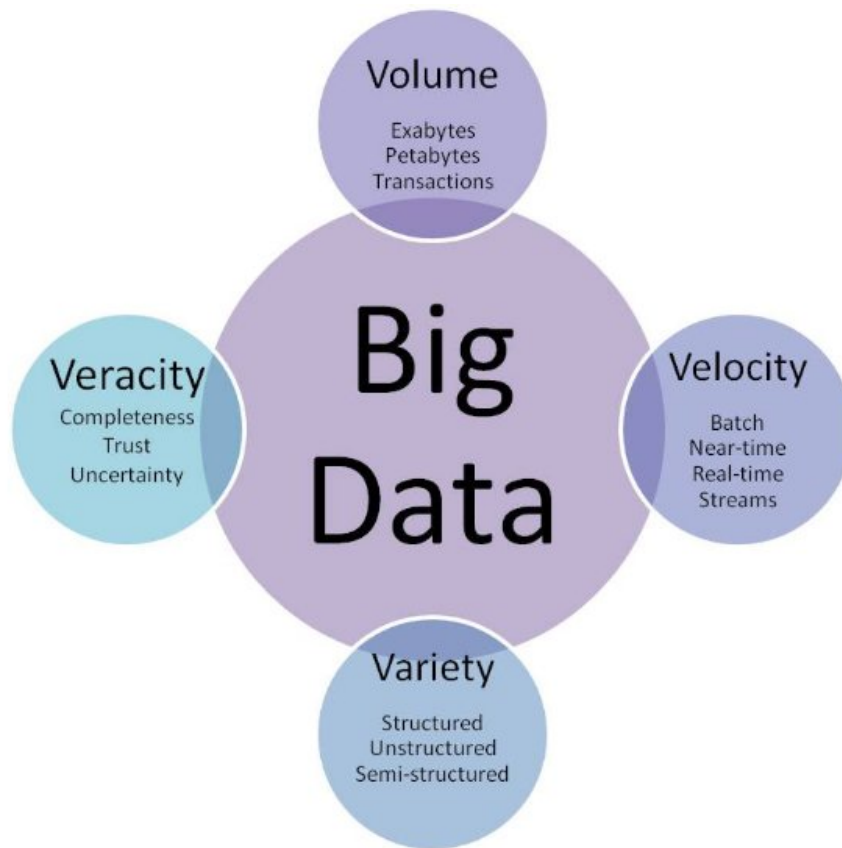


Figure 2: Attributes of Big Data (DAVE 2016)

Volume is defined as the amount of data. Big Data solutions require processing of large datasets such as data harvested from network traffic, twitter data feeds, clicks on a web page, sensor-enabled equipment's among other sources. The volume of data has grown drastically and Big Data solutions should be designed to handle the voluminous data sets through distributed storage and computing while capable of expanding to scale out as computing demand increases. The distributed nature of big data solutions provides means for collecting, storing and analyzing large volumes of data that could not be stored on a standalone computer system. **Velocity** describes the pace at which data arrives that

is usually in real-time stream of data like sensor data and social media data. Data normally streams direct into system memory while as the same time being written to disk. The speed at which the data are being created can outpace traditional analysis tools. Analyzing real-time data like social media data requires specialized tools and techniques for quickly retrieving, storing, transforming, and analyzing the information. **Variety** is the third V of Big Data that refers to a different type of data being produced from different sources which constitutes either to structured, semi-structured or unstructured data. This makes traditional analysis insufficient or unsuitable for analyzing such large data.

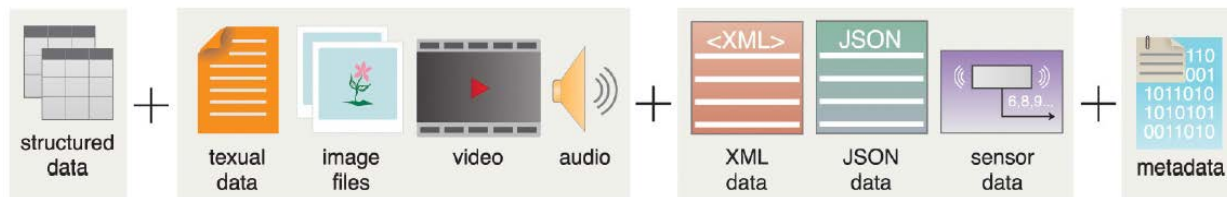


Figure 3: Big Data Variety (Erl, Khattak & Buhler 2016)

Veracity is the quality of data and indicates whether the informational content of data can be trusted. To ensure meaningful and trustful information, large data being processed by big data systems might contain abnormalities and noise hence need to be cleaned through data preprocessing activities to ensure the minded results are dependable to solve the problem at hand.

2.2.2 Big Data Architecture

Big data architecture is a conceptual model which shows how big data will be captured, stored, managed and accessed by the various user groups and applications. It defines how big data solutions will be integrated together in a unified manner, all components including hardware, storage, data sources and how applications are integrated and connected together. The design and deployment of big data solutions can vary greatly but several core concepts are common to design of most big data solutions (Sremack 2015).

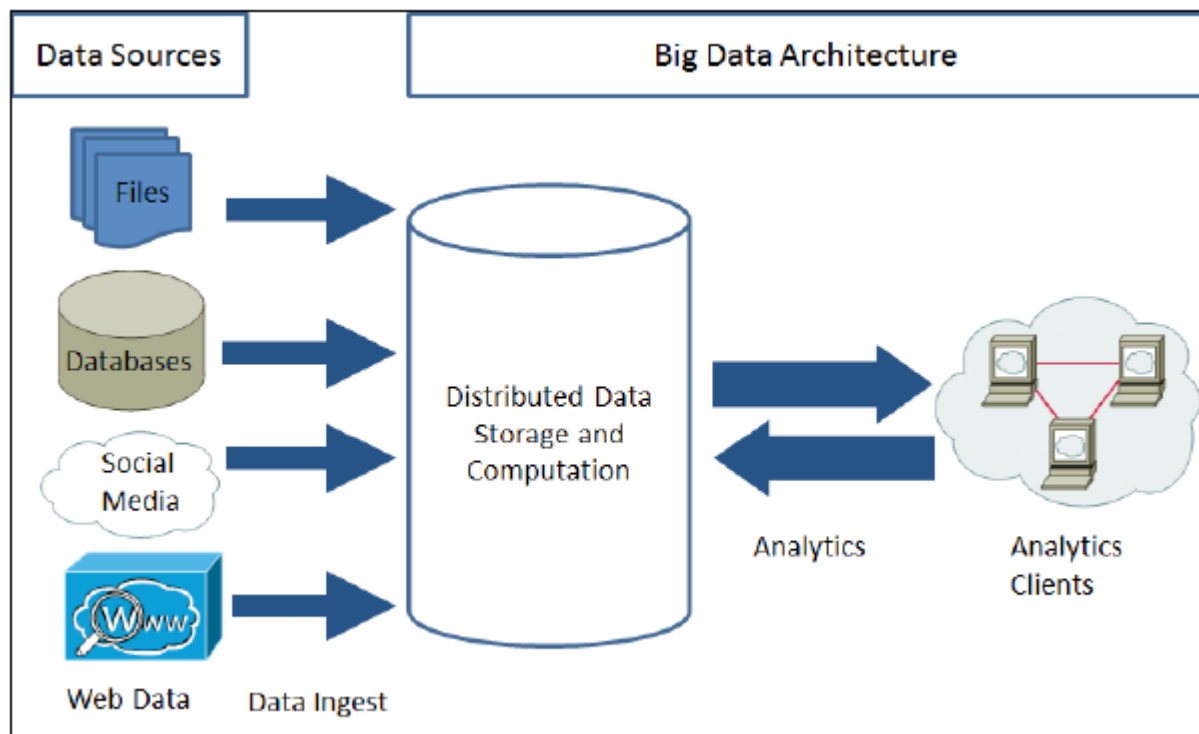


Figure 4: Big Data Architectural Overview (Sremack 2015).

In this architecture, data is collected and ingested in Big Data system from a multiple of sources like social media, system logs, web data e.tc. These data sources come in various types and formats which big data solutions should be designed to handle and allow the data to be ingested and stored together. The data ingestion module brings the data in for processing and analysis before the data is passed for storage or outputted in for of reports. The Big Data solution designers has to make decisions about what happens to the data ingested, how it is stored across a cluster of computer nodes, how access is managed internally, data transformation tools and eventually the manner in which applications are

granted access to the data. For storage of voluminous datasets involved in big data, the architecture should adopt distribution of storage across cluster nodes so as to cater for limitation of storage on a single node. To perform big data analysis, distribution of computation across nodes might be critical for performing the analysis within timely requirements. Big Data can include structured and unstructured data hence the solution should be capable of performing the analysis across various types of files and be able to utilize data from multiple data sources to carry out the analysis (e.g. relational and non-relational database). Big Data architecture may also contain text analytics (Sremack 2015) and machine learning component for analyzing unstructured sets of textual data like social media content and e-mail.

2.3 Data Mining and Machine Learning

The increasing volume of data for digital forensics raises issues, rendering traditional forensics tools inefficient for big data forensics like social networks. In the era of big data, the growing sizes of data makes it more difficult and challenging for law enforcers and intelligence agencies to analyze such large volumes of data involved in online cybercrime activities thus scientific method such as data mining and machine learning becomes suitable for discovering insightful, interesting, and hidden patterns within big data. Thus, (Dua & Du 2016) defines data mining as “an interdisciplinary field that employs the use of analysis tools from statistical models, mathematical algorithms, and machine learning methods to discover previously unknown, valid patterns and relationships in large data sets, which are useful for finding hackers and preserving privacy in cybersecurity”.

In digital forensics, data mining can be used in identifying correlations or association in big forensic data, discovering and sorting forensic data into groups (classification) based on similar features, discovering insightful patterns (forecasting) in big data that may lead to useful predictions (Kayarkar, Ricchariaya & Motwani 2014). In business intelligence, data mining has been used widely in making business decision and lately data mining techniques are being applied in the field of criminal forensics to discover insightful crime patterns from large data volume. By integrating data mining techniques with digital forensic science can help improve reliability of investigations especially in Analysis phase. (Quick & Choo 2014) in his study developed a framework that employed data reduction methods in data mining for minimizing the storage requirements for digital forensic evidence.

Clustering and textual analysis techniques of data mining have been used (Tsochataridou, Arampatzis & Katos) in digital forensics to extract electronic messages sent by employees in Enron scandal. Cybercriminals make use of fake email id for attempting many email cybercrimes and which makes it hard to identify the author of threatening email or other terrorist activities. (Nirkhi & Dharaskar) in his research developed a machine learning algorithm to identify the authorship of anonymous email when their identity is forged or hidden using proxy setting for online communication.

2.3.1 Data Mining Techniques

Data mining can be broadly categorized into supervised or unsupervised learning models. Supervised learning works by inferring relationship based on labeled data training and uses this function to map new unlabeled data (target variables). Supervised techniques predict the value of the target variables (output) based on a set of input variables. To do this, data analyst is required to develop a model from a training data set where the values of input and output variables are known. The model deduces the relationship between the predictor variables and target variables and uses it to predict for the data set where only predictor variables are known. For supervised learning, sufficient number of labeled data is required to train the model from the data. Unlike in supervised learning, there are no output variables to predict in unsupervised learning. The objectives of this class of data mining technique is to find patterns in the dataset based on the relationship between data points themselves. There are numerous types of data mining algorithms that can be used in forensic analysis among them Descriptive Modeling, Predictive Modeling, Classification, Regression, Combinatory algorithms, Multi-layer perceptron's (MLP) and Neural networks.

Predictive Analytics

Predictive analytics are supervised machine learning which aims to build an analytical model for predicting about unknown future events. By utilizing statistical methods and machine learning techniques, predictive analytics identifies the likelihood of unknown future events based on historical data. It makes use of Classification, Probabilistic rules, Markov models and regression techniques to predict a target variable based on input variables. In regression analysis, one tries to ascertain the causal relationship between target (dependent) variables and predictor (independent) variable where by the dependent variable is continuous and can vary along a predefined interval. Predictive analytics has been used to predict crime mapping and help authorities in investigation of crimes. (Chauhan &

Aluvalu) used Big Data Analytics with clustering and Predictive analysis to reduce the investigation time and help in retrieving the hidden information through correlation and categorization

Descriptive Analytics

Descriptive analytics are typically unsupervised machine learning that uses data aggregation and data mining techniques to forecast future trends, activity and behavior. Descriptive analytics uses many analysis techniques such as correlation analysis, Associative rules and clustering techniques to drill down into data and uncover details such. Clustering is the process of identifying natural groupings in the dataset with a set of data items in a group bearing similar characteristics. The data analyst investigates why these clusters are formed in the data and generalizes the uniqueness of each cluster. Data mining and correlation methods have been used for digital forensics (Flaglien 2010) to automatically identify malware traces across multiple computers.

The Association rules or Link Analysis technique are used to discover relationships between items and item features. It involves finding patterns that occur frequently in a dataset and represents the patterns in the form of association rules (Fei 2007). Association rules take the form of if/then statements to help find out interesting data items associations and correlations in a large set of data items. Associative rule mining has been employed in the identification of user ownership of files on windows hard disk (Kumar et al. 2012).

2.3.2 Data Mining Algorithms

Data mining algorithm is a set of well-defined procedures used to implement a specific data mining technique by taking data as input and creating data models or patterns as output. To create a model, the input data provided is first analyzed using the algorithm to look for specific types of trends or patterns of interest. To find the optimal parameters for creating the mining model, the results of the analysis are subjected to many iterations until optimal results are attained. The resulting model is then applied to testing data to prove the model before it is applied to a new dataset in order to extract actionable predictions or patterns.

2.4 Classification Algorithms

Classification algorithm is a family of supervised machine learning algorithms that classifies, categorizes, or labels data points into several pre-defined classes based on what has been observed in the past. Each classification algorithm requires training data. The training data consists of a set of data items where each data is a pair made up of an input data point (feature vector), and a corresponding output outcome for that input data. Classification algorithms involves three process (Sarkar 2016); Training process where the algorithm analyzes and tries to infer patterns out of training data such that it can identify which patterns lead to a specific outcome (class labels/class variables/response variables). The second process is the evaluation phase which involves testing the prediction performance of the model to see how well it has trained and learned on the training dataset. Finally, the model tuning process (hyper parameter tuning/optimization) which focus on trying to optimize a model to maximize its prediction power and reduce errors. There exist various data mining classification algorithms but for this research, I will focus on Naive Bayes classifier and Support vector machines (SVM) which recent studies have indicated that they perform well in problems involving semantic analysis and text mining/classification.

2.4.1 Naive Bayes (Multinomial) Classifier

The Naive Bayes classifier is a supervised learning algorithm that is based on the Bayes theorem that relates conditional and marginal probabilities by showing how conditional probability (posterior) of an outcome can be obtained based on its prior probability (marginal) and the inverse conditional probability (Cichosz 2014). As with supervised algorithms, Naive Bayes classifier builds a model based on labeled training dataset which is then used to categorize a predefined class label to new objects. Naive Bayes classifier assumes that the effect of particular feature or attribute in a particular category is independent or unrelated to the values of the other features (attributes) which is made to simplify the computations involved hence less computational power in terms of both CPU and memory requirements; and requires a small amount of training data. This assumption between the features or attributes is termed as conditional independence. It remains to be one of the most text classification algorithms in use for various applications such as document categorization, email spam detection and sentiment detection. In text analytics or document classification using Naive Bayes classifier, each word position within a text/document is defined as a feature and the value of that

feature to be the word found in that position. It assumes that each word in the document has nothing to do with the next word hence the naive assumption.

2.4.2 Support Vector Machines (SVM)

Support Vector Machine is a type of supervised learning algorithm that uses nonlinear classification to transform the training data into k-dimensional hyperplane (where k represents the number of features within the dataset) which separate the dataset into two exact categories with each feature being the value of a particular plane. Within the hyperplane are support vectors that are data points that are closest to the hyperplane and are therefore considered critical points of the dataset since if they are removed, they would alter the position of the dividing hyperplane. SVM trains a model that assigns new unseen objects into a particular class. This is achieved by creating a linear partition of the feature space into two categories. Based on the features in the new unseen objects like documents or emails, it places an object "above" or "below" the separation plane, leading to a categorization of either an email being a spam or not a spam.

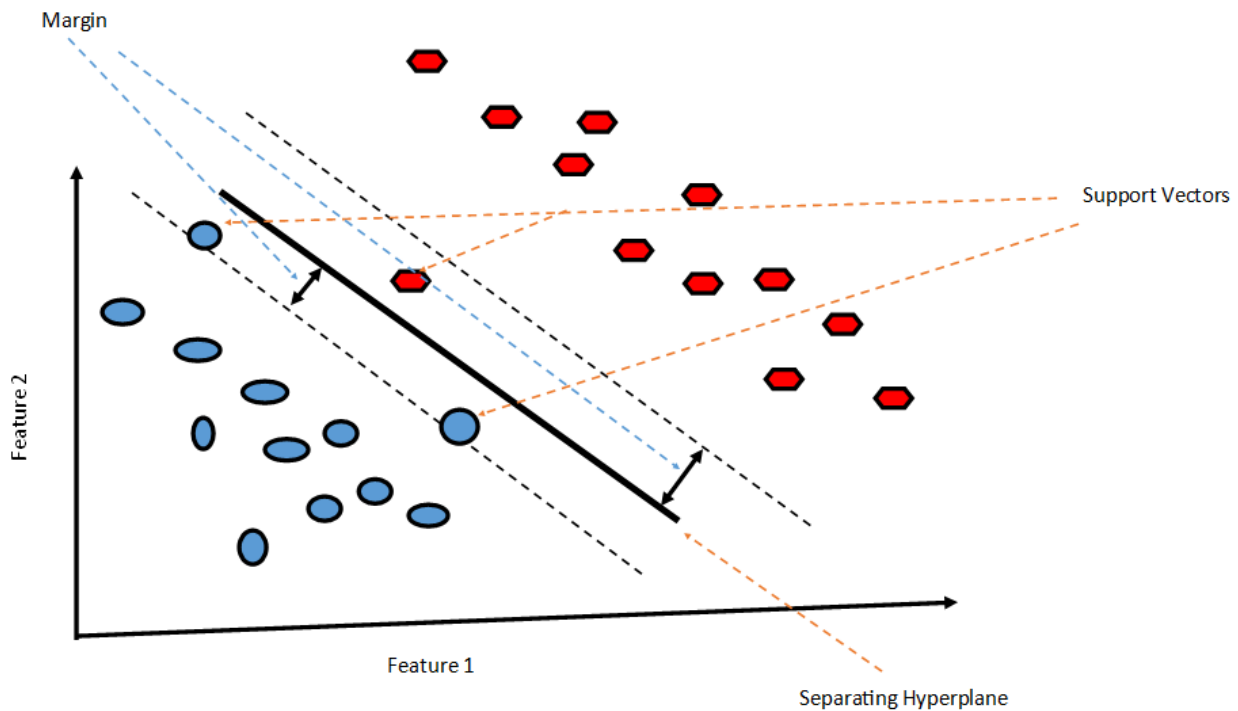


Figure 5: Support Vector Machines sample hyperplane

To achieve optimal data classification, the hyperplane that has the greatest possible margin between the hyperplane and any point within the training data points are considered as having achieved best classification.

2.5 Apache Hadoop

Apache Hadoop is defined by Apache Foundation as a “framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly available service on top of a cluster of computers, each of which may be prone to failures”. Apache Hadoop framework has seen a wide deployment by many companies involved in Big data analytics due to its open source nature and easy to scale up by adding cheap server nodes in a cluster which makes it used extensively for data intensive applications such as fraud analysis, network traffic analysis, social network analysis and machine learning applications among others.

For Big data analytics, Hadoop offers high scalable processing power across distributed cluster of computer while at the same time offering high availability and fault-tolerance computation with automatic code parallelization within computer nodes. By removing core data processing functions from those of distributed computing functions, Hadoop makes writing of distributed application easy (Guller 2015) by hiding the complexities of programming distributed and parallel applications.

With Facebook hitting billions of users and billion pages views every day globally, Facebook deployed multiple Hadoop clusters that are distributed across data centers. Apache Hadoop is used to support several services among them data warehouse which is used for web analytics, distributed database storage and backups for MySQL database (Borthakur 2010). Using Hadoop and Big data analytics, Facebook analyzes big data and inform its billion users about friend’s birthdays and recommends friends based on mutual friends. It has also been used by other big data analytics organization such as Yahoo, Google, Uber, Linkin, and YouTube to collect and analyze their massive data volumes. Google the leading web search engine uses Hadoop for indexing web pages and to provide suggestions of what the user is querying for; thereby providing highly personalized web search experience for internet users. Big Data analytics with distributed computing using Apache

Hadoop have been used in cybersecurity to analysis cybercrimes, detect and stop cyberattacks and facilitate post event digital forensic analysis (Wang & Alexander 2015). Big data analytic and Hadoop has been utilized by Uber to builds big data solutions on top of Hadoop and Spark systems to support its Operations (NATARAJAN 2016). Data analysis drives many functions within Uber like data science, machine learning, fraud detection with Uber data information including about trips, billing, and infrastructure health monitoring and services behind their apps.

2.5.1 Hadoop Core Components

The Hadoop software framework mainly includes three core modules namely MapReduce, Hadoop YARN and Distributed File System (HDFS).

2.5.1.1 Mapreduce

This is a highly scalable module designed to simplify the development of large-scale, distributed, fault-tolerant big data solutions for processing large datasets that are distributed across cluster of computer nodes. With the complexities and challenges involved in developing distributed systems, Mapreduce simplifies programming of distributed processing by abstracting cluster based computing and constructs for developing distributed applications to support data intensive processing (Guller 2015). For big data sets, MapReduce automatically schedules execution of applications across nodes within a cluster and manages load balancing, internode communication and node failures.

MapReduce application consists of map () and reduce () functions which are data processing module. Each of these functions accepts input data in form of key/value pairs and produces output as a set of key/value for the function with data types chosen by the application developer. Input dataset is usually split into key-value pair of multiple data items that are processed in parallel manner by the map function. The data items will be processed by the Map () function by sorting and grouping (mapping) the key/value and producing the mapped key and value pairs which are feed to reduce () function. Typically, the intermediate data is normally stored on Local file system while the Map () function key/pair are stored in HDFS file-system with MapReduce handling scheduling task.

The Map function sorts and shuffles out the intermediate data and passes the output as input data to reduce () function. The reduce () function combines the data values together and produces the combined data values with the key/values which becomes the results or answer to the large problem that needed solution. Figure 5 below demonstrate the MapReduce process of word count example.

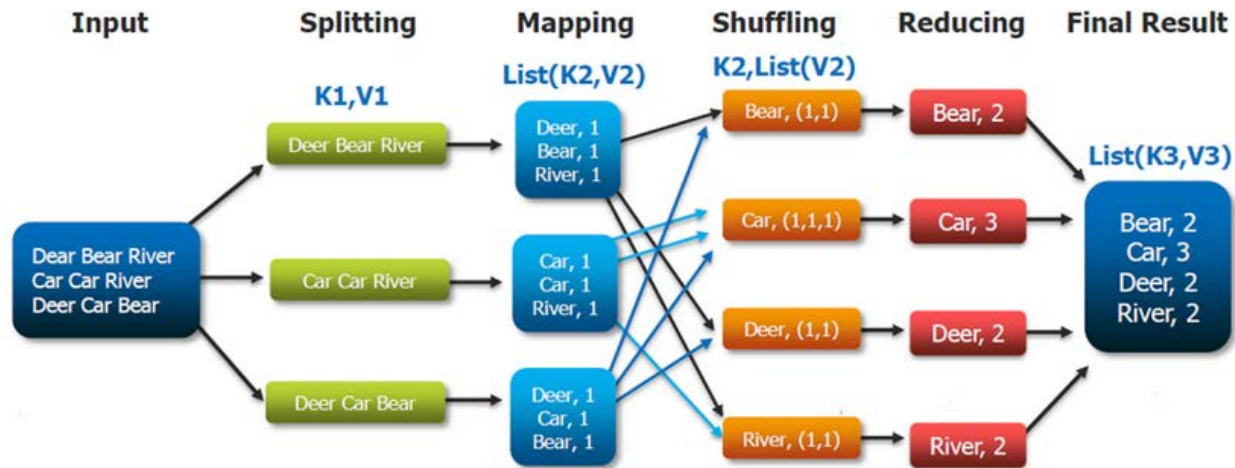


Figure 6: Example of Mapreduce Word Count Process.

The MapReduce framework is composed of two services which are key to the functioning of the cluster nodes. This first service JobTracker which is responsible for accepting requests from clients and scheduling them among cluster nodes. It is also responsible for resource management, monitoring of jobs and re-submitting failed tasks. The second service is the TaskTracker which is responsible for accepting from JobTracker, executes the tasks and alerts the Jocktracker upon completion.

2.5.1.2 Hadoop Distributed File System (HDFS)

This is a distributed file system that is designed on top of Hadoop framework to stores large data across computer nodes within a cluster in a network. It is designed to handle and store huge volume of data while at the same time providing fast and quick files access (Guller 2015). It provides high fault tolerant, scalable and reliable distributed storage across multiples cluster nodes. HDFS stores large files by partitioning them into fixed sized blocks usually of 64 or 128 MB and replicating the blocks on multiples cluster nodes. HDFS is designed following master-slave computing model with two types of nodes known as NameNode and DataNode. The NameNode (Master Node) manages DataNode and is responsible for managing the MapReduce filesystem namespace. It maintains

distributed systems files, directories trees and manages blocks that are present on the DataNodes. Also within HDFS file system are DataNodes (Slave Node) which are deployed and ran on each cluster node and manages storage on the nodes including reading and writing data requests from clients (Prajapati 2013). HDFS also includes a secondary NameNode whose main purpose is to carry out periodic checkpoints on filesystem and appends logs to Fimage such that when the NameNode starts up, I can load HDFS state from the image file.

2.6 Apache Spark

Developed at University of California, Apache Spark is a distributed and highly scalable in-memory data analytics framework which provides the ability to develop distributed computing applications using Java, Python and Scala programming languages (Frampton 2015). It provides clustered in-memory computing and implements an advanced execution engine leading to increased speed in data processing over Mapreduce. Spark enables in memory processing and allows applications to cache data hence minimizing disk I/O which improve significantly the overall job execution time. Unlike MapReduce which is suitable for batch processing (Shahrivari 2014), Spark comes handy with features for batch processing, stream processing, interactive data analysis and machine learning. With high demands for interactive queries in forensic analysis and big data streams, in-memory computing stands out as a notable solution that can handle forensic analysis for social media sites in both real-time or near real-time.

The core of Spark framework is Resilient Distributed Dataset (RDD) which is a logical collection of items or objects of same type that is distributed or partitioned across many nodes in a cluster (Ramamonjison 2015). Sparks framework extends MapReduce framework and access Hadoop data store (HDFS), which makes Spark's Core API analytics jobs easier to write. On top of Spark Core API is set of integrated API libraries that are required for specialized tasks such as data streaming, machine learning and graph analysis. Spark Core components forms the foundation for parallel and distributed processing of big datasets and its responsible for all the basic I/O functionalities, job scheduling and monitoring on spark clusters, dispatching of tasks across nodes, storage systems networking, failure recovery and ensuring efficient memory utilization.

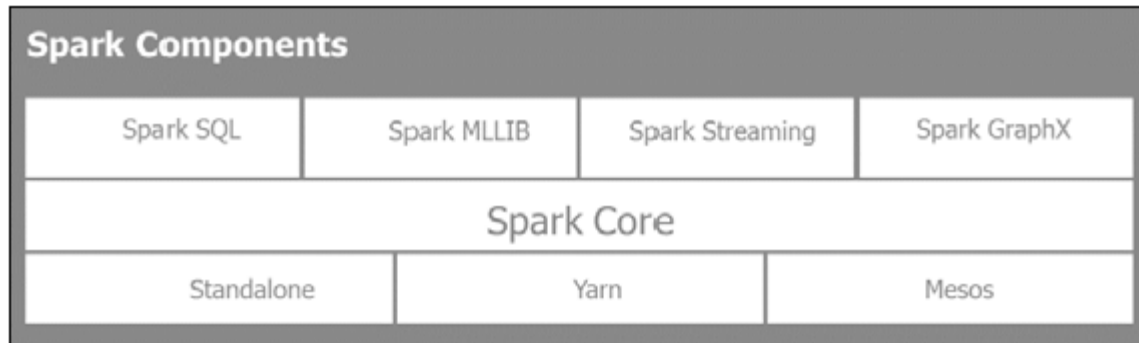


Figure 7: Spark Components (Nandi 2015)

As shown in figure 5, Spark comes with several libraries such (1) SparkSQL which provides SQL-like ability to query structured data while bringing support for native SQL query to spark programs and stream data processing. (2) Spark Streaming provides high degree of data streams with fault tolerance of live data stream like social media data. (3) SparkMLlib provides most commonly machine learning algorithms such as classification, clustering, regression and association algorithms which are suitable for Big Data Analysis. (4) Spark GraphX provides distributed graph processing for graph based data while enabling fault tolerant parallel computation.

2.6.1 Spark Streaming

This is a stream based processing module that enables live analysis of data streams and real-time data with continuous data streams divided into a discrete stream (DStream) or batches which are then passed to the spark engine for processing to produce final stream of results still in batches. Being an extension of Spark core APIs, spark streaming module provides high-throughput, scalable and fault-tolerant stream processing of real time data with data coming from sources like HDFS directories, TCP sockets, Flume, Kafka, Twitter. As shown in figure 8 below, streams of data can be processed with Spark's core APIs, DataFrames SQL, MLlib or GraphX APIs and the results stored to a file system, HDFS, database or presented on user dashboards.



Figure 8: Data sources for Apache Streaming (Frampton 2015)

Live input data streams are received by spark streaming module that divides the data into discrete batches then feed into the spark engine for processing to produce the final results in stream batches.



Figure 9: Spark Streaming Data flow

2.6.2 Use Cases of Spark/Spark Streaming

Many companies which had earlier used MapReduce applications and libraries as core distributed computing framework have either switched to spark or are implementing support for apache spark. Uber Taxi Company uses Spark Streaming in their continuous Streaming ETL pipeline to collect terabytes of daily event data from drivers smartphones for real-time telemetry analytics. (Nair & Shetty 2015) on his research study implemented real time analyzing and filtering system using apache spark to stream millions of twitter job advertisements and classified the jobs into various categories to enable effective and easy job search.

At Ericsson, a world-leader in communications technology (Koutsoumpakis 2014) designed Abnormal Log Detection application using spark and machine learning to analysis large log files produced during automated test loops and the testing process of communication equipment's. Apache spark have also been used for distributed real-time anomaly detection from multisource data stream to monitor VMware performance, stream CPU load and memory usage. The framework collected data simultaneously from all the VMwares which were connected to the network and notified the resource manager to reschedule its resources dynamically when it detected anomaly behavior on the

data collected (Solaimani et al. 2014). With the discovery of Machine to machine (M2M) and internet of things (IOT), millions of devices are interconnected together in what is known as machine-to-machine (M2M) communications with demand for real-time traffic analytics solutions required to help manage and monitor those devices deployed in the M2M application. In his work (Privitera et al. 2014) developed a real time GPRS traffic analytics solution based on Apache Spark which captured GPRS traffic, processed the data, and computed an array of statistics that were presented as charts and maps on a web based application dashboard.

2.7 MongoDB

The design and implementation of successful big data analytics solution depends entirely on successful implementation and choice of different computing components that integrates together. This revolves around choosing key architectural components among them scalable and reliable storage, data management and parallel computing. It is important to pay close attention on the ways the different computing components integrate, from the perspective of big data analytics, the interdependence between the underlying data management and analytic algorithms requires an in-depth consider big data storage. With big data involved in such analytics, traditional relational database management becomes insufficient to cope with scaling performance demands. Again, big data algorithms handling such huge data requires scalable, high-performance, elastic, and distributed data environment to cater for such high demand, new model of big data databases has been developed referred as NoSQL databases. this includes mongodb, couchdb, cassandra, and hbase

Developed by MongoDB inc under the banner of NoSQL, mongodb is a schema free cross-platform open database that is implemented using document-oriented data model instead of using tables and rows. Unlike relational databases that uses tables and rows, the architecture of mongodb comprises of collections and documents that are the basic building entities. MongoDB documents are basic building unit of data that is equivalent to rows in relational databases. Documents are represented using JSON format and comprises of a sets of key-value pairs and stored in binary JSON documents (BSON)(Copeland 2013). Similar to relational database table are “collections” that are groups of documents. Each document within the collection contains a unique identifier (known as “_ID” field) which identifies that document. This primary key can either be given explicitly by user upon document creation or generated automatically by the mongodb. MongoDB offers flexible dynamic

schema in that documents in the same mongodb collections do not need to have the same structure of fields and a document can hold different data types. By integrating apache spark applications with mongodb database can significantly improve performance for big data analytic applications requiring real-time analysis such as social media data and internet of things data (IoT) among others. For optimized performance, Mongodb includes document indexes which offers improved query performance. MongoDB supports advanced text search by incorporating specialized document indexes that uses advanced language specific linguistic rules for tokenization, stemming, and stop words removal.

2.7.1 MongoDB Document Structure

This represents how data models are used to reflect objects that applications will handle together with the relationship between those objects.

a) Embedded documents

While MongoDB is schemaless, Mongodb models data relationships by embedding data within a document by storing related data items within a single document structure. Like with traditional RDBMS, embedded documents enables modelling of one to many relations of a document where the top level (parent) document can have many low level (child) documents and the child documents can only have one parent document (Copeland 2013). This is known as demoralized data model as shown in Figure 10 below.

```
{
  "_id": ObjectId("52ffc33cd85242f436000001"),
  "contact": "987654321",
  "dob": "01-01-1991",
  "name": "Tom Benzamin",
  "address": [
    {
      "building": "22 A, Indiana Apt",
      "pincode": 123456,
      "city": "Los Angeles",
      "state": "California"
    },
    {
      "building": "170 A, Acropolis Apt",
      "pincode": 456789,
      "city": "Chicago",
      "state": "Illinois"
    }
  ]
}
```

Figure 10: Embedded document data model.

b) References Document Structure

MongoDB document reference creates data relationships by maintaining a separate child document on its own standalone document but creates relationship between the data item by including references or links from one document to another (O'higgins 2011). This is known as normalized data models as shown in Figure 11 below.

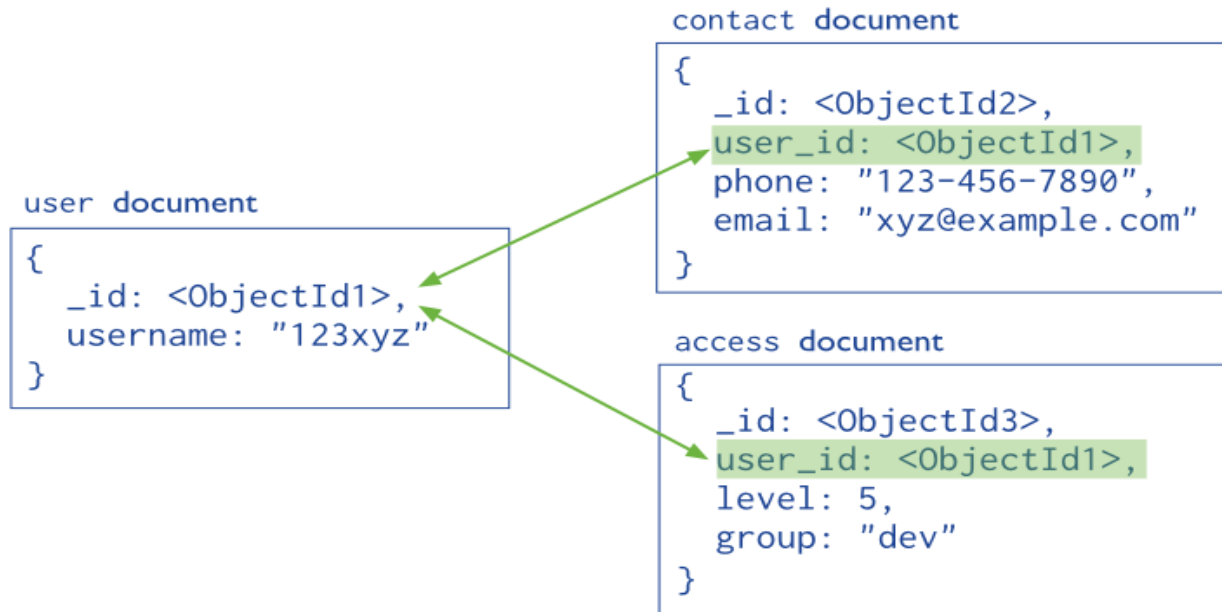


Figure 11: Reference Document Structure.

```
// Books
{
  "_id": ObjectId("500c680c1fe9193b67b898a3"),
  "publisher": "O'Reilly Media",
  "isbn": "978-1-4493-8156-1",
  "description": "How does MongoDB help you...",
  "title": "MongoDB: The Definitive Guide",
  "formats": ["Print", "Ebook", "Safari Books Online"],
  "authors": [{
    "lastName": "Chodorow",
    "firstName": "Kristina"
  }, {
    "lastName": "Dirolf",
    "firstName": "Michael"
  }],
  "pages": "210"
}

// Reviews
{
  "_id": ObjectId("500c680c1fe9193b67b898a4"),
  "rating": 5,
  "description": "The Authors made an excellent work...",
  "title": "One of O'Reilly excellent books",
  "created": ISODate("2012-07-04T09:48:17Z"),
  "book_id": {
    "$ref": "books",
    "$id": ObjectId("500c680c1fe9193b67b898a3")
  },
  "reviewer": "Giuseppe"
}
```

Figure 12: Reference Document example.

2.7.2 MongoDB Connector for Spark

Developed by MongoDB Inc. the connector provides integration between Apache spark analytic application and MongoDB database that enables the development of real time or near real time analytics applications on live or stream data. The connector exposes several Spark libraries that makes MongoDB data to be converted into DataFrames and Datasets that can be used for analysis using machine learning (MLlib), graph (GraphX), streaming and SparkSQL APIs. By utilizing Mongoddb secondary indexes, the Spark connector can filter, extract and process data that is only within the range of data it needs rather than relying on primary key only.

2.8 Social Networks

The proliferation of online social networks such as Facebook, Twitter and LinkedIn has attracted billions of users across the world to share information online. It provides a platform for interacting and sharing information with friends, families and organizations across the globe. Social network is defined as a social structure usually a website which is composed of a set of social actors normally individuals, businesses and organizations that allows interaction and sharing of information among the actors. Just like with messaging/instant messaging systems and email systems, social network sites have become an excellent platform for organizations and companies to interact and share information with their customers and the public. With the increased popularity of social networking sites, many people are willingly publishing a lot of personal information on social sites like status updates, personal email addresses, location, phone numbers, individual photos and friends which informs the public there whereabouts and what they are doing. This has raised security concern as this information can be used by cyber criminals in committing several online cybercrimes.

The large number of social media sites activities and their anonymous nature makes social network attractive for committing cybercrimes. Currently there is increased number of cybercrime cases reported which are related to online social sites or the use of online social sites in order to execute cybercrimes. However, Big Data generated by social networks sites and its anonymous nature make cybercrime investigation using traditional forensic tools extremely difficult to apply in social network sites, IoT and other cloud based systems. This calls for design of new tools to supplement the traditional forensic tools like the use of social network analysis (SNA) and visualization.

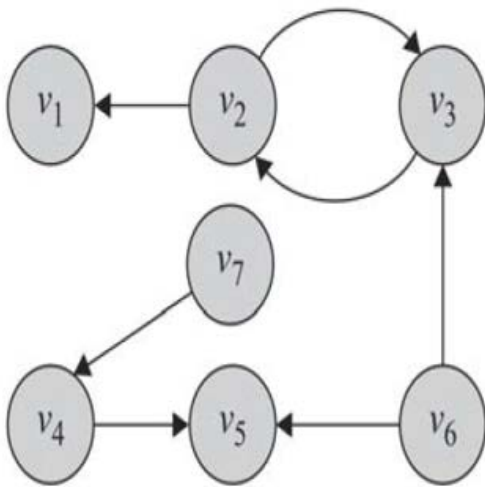
The application of social network analysis (SNA) and data visualization techniques in social network forensics can significantly help discover, collect and preserve social media forensic artifacts by identifying and understanding relationships and data flow between individuals and events within the social network interactions (Karran et al. 2011).

2.8.1 Social Network Structure

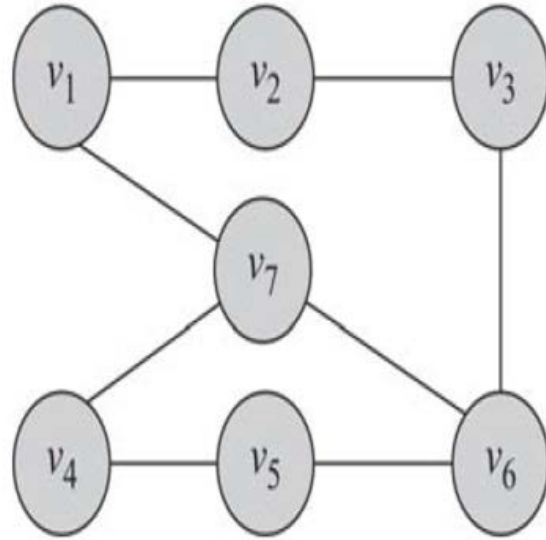
Social networks can be described as a set of connected entities, which are modelled as a collection of nodes (vertices) and links (edges) connected together. Each individual within the network are represented as a node (actor) and individuals who are acquainted to one another are connected using an edge (Zafarani, Abbasi & Liu 2014).

2.8.1.1 Graph

Social networks can easily be modelled using graphs where by a graph is composed of both sets of objects called nodes (actors, vertices) and links (edges) connecting nodes between each other. In social media, the nodes represent individuals, organizations, companies and links between nodes represents friendship among these nodes. Nodes are linked or connected together by edges (links) and indicates relationships among nodes and are known as relationships or (social) ties representing social entities such as people. Edges to a node can have directions in which one node is connected to another node and not vice versa. This is known as directed graph. Again, a node can be connected on both ways forming undirected graph as shown in figure 8 below.



Directed Graph



Undirected Graph

Figure 13: A Directed Graph and an Undirected Graph (Zafarani, Abbasi & Liu 2014)

Edges can include other additional features such as labels, which can be used in analysis. The label gives more information about the relationship between the nodes (people) which could be their relationship (e.g. sister, brother mother, cousin), or other information relating to their relationship.

2.8.1.2 Node Degree

A node degree within a social network defines the number of links to other nodes, which represents the number of friends a given actor (person) has. For example, in Facebook social media, node degree represents that person's number of friends, while in Twitter social media, in-degree represent the number of followers one has and out-degree represents the number of people followed by the person(followees).

2.8.1.3 Degree Distribution

In analysis and study of graphs, the degree of a node is the measure of the number of connections or edges the node has to other nodes within the large network while the degree distribution (neighbor distribution) describes the probability distribution of these degrees over the whole network i.e. the probability that a randomly chosen node has certain connections (or neighbors)

2.8.2 Social Network Analysis (SNA)

In social media analysis, measuring different structural properties of a social network can help better understand individuals and their roles within the large network. For example, in determining which nodes are most important or influential in the social network, one needs to define measures for quantifying centrality, level of interactions, and similarity, among other qualities. To compute these measures, a graph representation of a social interaction is taken in as input, such as nodes friendships (adjacency matrix), from which the measure value is computed. By using graph measures of centrality we can identify the most prominent actors commonly known as the key players in the network. In modeling and mining social media several centrality measures are defined among them *degree centrality* which describes nodes degree which is the number of edges the node has whereby if a given node has high degree, the more central the node is. By using these measures, one can identify various types of central nodes in a network and answer questions like “Who are the influential individuals in the network?”. The second measure is *closeness centrality* which indicates how close a node is to all other nodes in the network whereby rather than considering the neighbors node, all nodes are taken into consideration (Magnusson 2012). It indicates nodes as more central if they are closer to most of the nodes in the graph and it has measured as the average distance from the source vertex to any other vertex within the graph. This metric allows us answers questions like “What interaction patterns are common in within friends?”. The third metric *between-ness centrality* indicates how important a node is to the shortest paths through the network and measures to what degree an actor has to traverse through a specific node in so as to reach other nodes within the network. The last measure is the *Eigenvector centrality* which unlike centrality measures, it tries to generalize degree centrality by incorporating the importance of the neighbors and the influence an actor has in the social network (Golbeck 2013).

2.8.3 Social Network Sites Forensics

The identification and collection of digital evidence from big data systems has become challenging for forensic investigators and especially investigation involving cloud based systems and social networks sites. This have been made difficult by the fact that most forensic artifacts are not stored on hard disk rather the data shared on social media sites is largely volatile with no guarantee of later retrieval as it can be deleted or updated. Social network analysis and data visualization techniques can significantly help in the discovery of social media evidence and collection by identifying and understanding relationships and data flow between individuals and events within social networks like Facebook, Twitter. SNA is defined as “a multidisciplinary area involving social, graph theory, statistical and computer science”. It uses analytical techniques to discover social relationships that are formed from individuals and groups, the structure of those relationships, and how relationship and their structure influence (or are influenced by) social behavior, attitudes, beliefs and knowledge.

SNA have been used in a wide range of interdisciplinary studies. For example, this approach has been used to discover and analysis individuals in organized criminal groups (Johnsen 2016). In his study, graph based algorithms and methods were used to analyze network structures in identifying interesting and central individuals within a criminal network. An automatic analysis tool for (Wijeratne et al. 2015) social media posts was proposed to understand the functions, structure, operations of gangs within streets of Chicago, IL region. It involved using Twitter as a source of data to captures tweets posted by gangs and used an automated analysis to discover gang structures, functions, and operations.

Intelligent social media analysis and other types of media data can help in understanding and solving various cybersecurity problems including uncovering online terrorist networks and radicalization. (Agarwal & Sureka 2015) in his study, applied social media analysis and machine learning in discovering and predicting civil unrest and online radicalization detection. Structural analysis of social networks sites like Facebook can provide forensic insight about how people relate to one another and where they fit within the larger social network. The social network sites can be exploited by criminals to commit several cybercrimes among them identity theft, cyberbullying, sexual harassment to children and spreading hate speech.

These cybercrimes require forensics analysis tools that can effectively be used in identifying the perpetrators and collect the evidence needed for prosecution. SNA has previously been used to uncover such cybercrimes for example a study by (Chen et al. 2012) who applied text analysis methods in detecting offensive social media contents in protecting the safety of adolescent. By using Lexical and Syntactical Feature he was able to identify content which is offensive in social network sites, and also predict user's potential to send out contents that are offensive. Social networks analysis can also been used for analysis of fraud as more often fraud is committed through illegal set-ups with many accomplices hence social network analysis might give new insights by investigating how people influence each other in what is called guilt-by-associations, where it is assumed that fraudulent influences run through the network (Baesens, Van Vlasselaer & Verbeke 2015).

2.8.4 Legal Challenges to Social Media Evidence Authentication

With the increased use of social networking sites and its target by cybercriminals, social media evidence is becoming highly relevant in cybercrimes investigations, legal disputes and broadly discoverable, but challenges lies in evidence authentication as there is lack of best practices, technology and processes. Social media status updates, posts and photographs on Social networking sites are increasingly denied admission as evidence in criminal litigation with courts citing issues with the evidence authentication. An article by (Patzakis 2012) states that “Given the transient and cloud-based nature of social media data, it generally cannot be collected and preserved by traditional computer forensics tools and processes. Full disk images of computers in the cloud is effectively impossible and the industry has lacked tools designed to collect social media items in a scalable manner while supporting litigation requirements such as the capture and preservation of all key metadata, read only access, and the generation of hash values and chain of custody.”

With these challenges, social media evidential data must be properly identified, collected and preserved in a manner that is consistent with digital forensics best practices so at to ensure all available circumstantial evidences are collected, including account metadata and a proper chain of custody established through the evidence collection. With this in place and associated account metadata preserved, it become easier to establish or reveal authenticity of the evidence.

For example, metadata fields for individual Facebook account posts such as status updates, photographs among others can provide important information to reveal the authenticity of the Facebook posts when collected and preserved using best digital forensic standards.

Metadata Field	Description
Uri	Unified resource identifier of the subject item
fb_item_type	Identifies item as Wallitem, Newsitem, Photo, etc.
parent_itemnum	Parent item number-sub item are tracked to parent
thread_id	Unique identifier of a message thread
recipients	All recipients of a message listed by name
recipients_id	All recipients of a message listed by user id
album_id	Unique id number of a photo or video item
post_id	Unique id number of a wall post
application	Application used to post to Facebook (i.e, from an iPhone or social media client)
user_img	URL where user profile image is located
user_id	Unique id of the poster/author of a Facebook item
account_id	Unique id of a user's account
user_name	Display name of poster/author of a Facebook item
created_time	When a post or message was created
updated_time	When a post or message was revised/updated
To	Name of user whom a wall post is directed to
to_id	Unique id of user whom a wall post is directed to
Link	URL of any included links
comments_num	Number of comments to a post
picture_url	URL where picture is located

Figure 14: Sample Key metadata fields for individual Facebook posts (Patzakis 2012)

In the evidence authentication process, actor accounts metadata can be examined to establish authenticity of the content whereby hidden metadata fields that are not visible on the face of a social media site (including dates, URLs, IDs, usernames among others) can be used to reveal authenticity and hence crucial for proper preservation and production of social media evidence.

2.8.5 Sentiment Analysis

With the increased use of social media, sentiment analysis has become a popular research area together with social media analysis particularly in assessing users posts and tweets. It is a special form of text mining mainly focused on analyzing individual's opinions, attitudes and emotions and classify the polarity (i.e. if a document or text expresses a negative, positive, or a neutral sentiment.) of a given text as either negative sentiments, Positive sentiments or neutral sentiments. Sentiment analysis involves determining sentence subjectivity, which includes distinguishing factual sentences (objective sentence) about the word and sentence subjective, which expresses one's feelings, attitudes, beliefs or views. Subjectivity classification is described as the process that involves determining whether a sentence is subjective or objective(Madhavan 2015). Sentiment analysis is categorized under two tasks, which includes both subjectivity classification and sentiment classification. The term sentiment classification, is defined as "the task of classifying texts whose objective is to classify a text according to the sentimental polarities of opinions it contains" (Li et al. 2010). This sentiment classification is also divided into two categories: *binary categorization* and *multi-class (multinomial) categorization*. Binary sentiment categorization classifies sentiments as either positive or negative while multi-class categorization classifies sentiments into one of three categories as either positive, negative or neutral.

The most commonly used machine learning classification algorithms for sentiment analysis includes naive Bayes, maximum entropy, and support vector machine. Digital forensics analysts can use these machine-learning techniques to uncover social media crime activities by analyzing factors such as users posts/tweets, time, location, address, physical characteristics and metadata from user account/wall and extra and correlate evidence. To perform a sentiment analysis on a social media text, one need to consider the various features of the text that imply its sentiment which involves identifying terms used in the text such as phrases or words. Sentiment words (opinion words) forms the basis for text mining that helps determine how positive, neutral and negative a sentiment is e.g. good, bad, hate among others (Madhavan 2015).

2.9 Proposed Solution

Based on forensic challenges identified regarding Big Data forensics on social network sites and limitation of traditional forensic tools, the research proposed an apache spark based forensic tool to support forensics investigation involving big data. Using the traditional forensic tools and techniques of digital forensics, it is not feasible to collect such large volume of evidential data, store and analyze it using the traditional forensic tools. In order to collect, store and analyze such data fast and effectively, Apache Spark a leading distributed computing framework come in handy with features that can process voluminous amount of data ranging from terabytes to petabytes of data. Again, Apache spark offers excellent large data streaming of live data which makes its suitable for streaming social network data and support big data analysis which can be distributed across nodes within a cluster.

2.10 Proposed Apache Spark Forensic Tool Conceptual Model

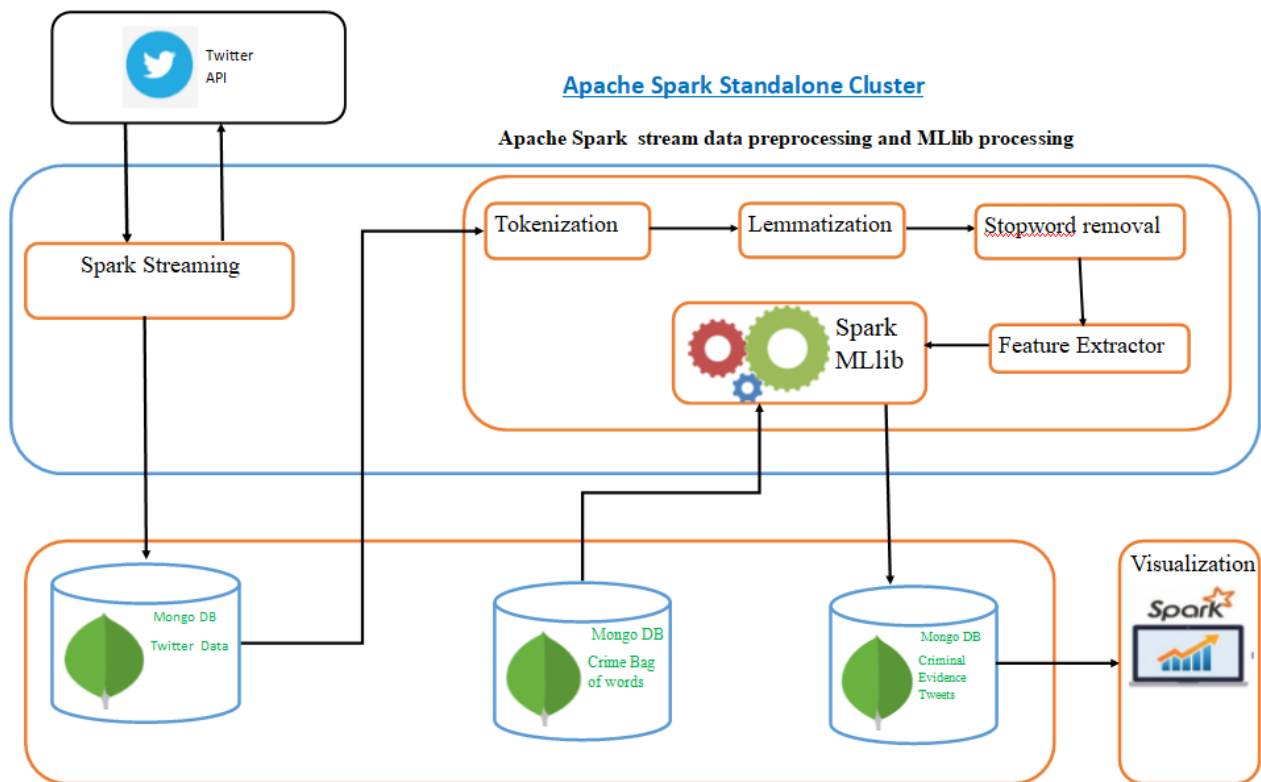


Figure 15: Proposed Apache Spark Forensic Tool Conceptual Model

The figure above shows the conceptual model which depicts the acquisition and processing of forensic data from Twitter social network. The forensic tool utilized public available REST API to stream data from the Twitter social media site. Apache Spark Streaming Module connected to the REST API and extracted streaming data which was saved to MongoDB database for preservation and later used for training Twitter forensic classification model. Two MongoDB documents was maintained within a single collection. The forensic data went through tokenization, lemmatization and stopword removal using Apache Spark feature transformers. The resulting dataset was subjected through Spark MLlib library for feature extraction. The feature vector was classified using Naïve Bayes algorithm with Spark MLlib module to identify and classify tweet/post as either hate speech or cyberbullying. The evidence was then stored within MongoDB document as evidence which was used for evidence visualization on web based interface.

2.11 Literature Summary

As shown in the literature, the era of big data has presented a big challenge to the forensic analyst because of large volume of data be generated in the modern technology like Internet of Things, social networking sites, smart devices and the cloud. There are many digital forensic tools currently available in the market like network forensic tools, mobile forensic tools, and computer forensics tools that could be utilized in conducting forensic investigation, acquiring and analyzing admissible evidence. However, online social networks forensics tools are still in their infancy and there is need for development of standardized software tools which will be able to help in digital forensics involving big data in accepted digital forensic standards and acceptable before a court of law. With the era of big data, the traditional forensic methodologies which includes taking the suspect system offline, removing the original hard disk containing the source evidence, making a bit copy of the original hard disk and calculating MD5/SHA-1 checksum is not possible with Big Data System like the Internet of things (IoT), the cloud based systems and social network sites.

A lot of social network analysis has been applied in various cybercrime investigation involving social network sites using various data mining algorithms but still none has explored big data analytics solutions offered by Apache spark framework. This research study was made to address the research gap identified and demonstrate how Big Data Analytics, data science and Apache Spark can be utilized in Twitter social network cybercrime forensics to supplement traditional forensic tools in big

data forensics. The concept undertaken utilized Apache Spark for Big Data processing and Spark Streaming module to capture on live Twitter updates so as to collect the digital evidence near live. Mining social network data for forensic Investigations can be complex and provides online evidence that is different from the conventionally accepted evidence. Digital forensic investigators and law enforcers can start to consider a new approach for effective ways to bring data from SNSs into investigations and develop standards to enhance its authenticity before a court of law. Even though social network investigators can apply and learn much from digital forensics disciplines, investigation requires different tools and techniques for social network forensics to ensure the evidence (artefacts) are authentic and the process is forensically sound.

CHAPTER THREE: RESEARCH METHODOLOGY

3 Introduction

This chapter describes in detail the strategy that we adopted in carrying out the study. It includes the following, research Design, Target population, the sampling frame, Sampling techniques, Sample size, Data collection methods, research procedures and Data analysis methods.

3.1 Research Design

To help achieve the objectives of the research, the study used both quantitative and Exploratory research design to collect, store and analyze Twitter stream data. There are few research studies regarding the application of Big data and Apache spark in social network sites forensics hence these research design was useful in exploring the application of big data solutions and distributed computing frameworks in the field of digital forensics to collect, store and analyze big data. The study employed data mining methodologies to get insightful information regarding cybercrime from big data collected from Twitter social network site. As in software development methodologies, there are various data mining methodologies applied in projects involving data mining. The most popular methodology used in data mining is Cross Industry Standard Process for Data Mining(Ncr et al. 1999). CRISP-DM methodology describes step by step approaches that can be used in tackling projects involving data mining. In this methodology, the data mining process are broken into six major phases where by the phases do not strictly follow the sequence but allows for back and forth movement between the project phases (Ncr et al. 1999).

The main objectives of this methodology include ensuring big data is of quality and dependable so that data mining results can be relied upon in solving problems, reducing skills required for data mining, capturing experience for reuse, general purpose. For projects involving data mining, CRISP-DM remains to be the most commonly used methodology. This methodology was an excellent fit to this project because it is robust and well-proven methodology in which data mining tasks can be carried out in a different order while allowing one to return to previous phases and rerun certain tasks. The problem which was being handled in this study involved understanding the problem space and through this building a forensic tool which required several iterations to understand big data forensic issues and social network crimes and build a forensic tool with the ability to extract, preserve evidence and correlate crime evidence from big data from Twitter social network.

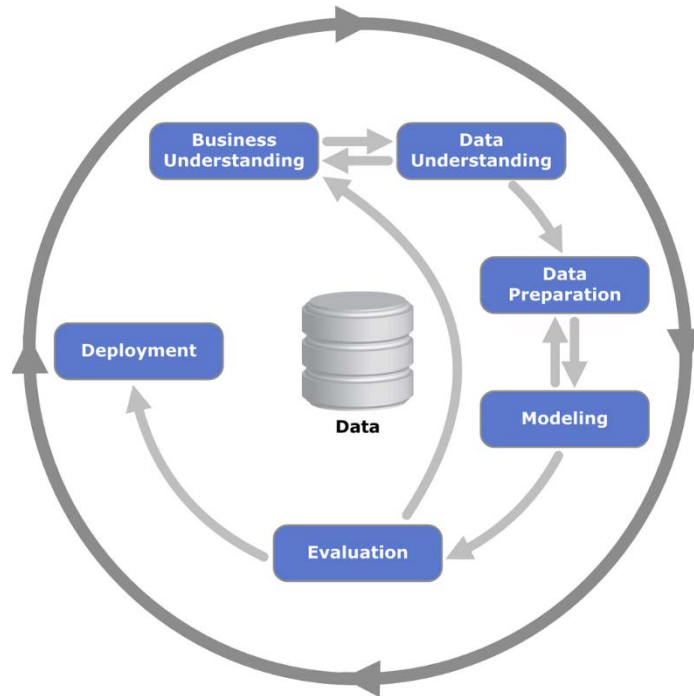


Figure 16: CRISP-DM Methodology (Ncr et al. 1999)

3.1.1 CRISP-DM Overview

a) Business understanding (Understanding Case)

This involved understanding the case or investigation requirements and objectives from the digital forensics perspective that helped in transforming this information into a big data analysis and evidence mining problem definition and a preliminary project plan designed to achieve the objectives.

b) Data Understanding

Data understanding was concerned with initial evidence gathering and proceeded with activities to familiarize with social media evidences; to identify evidence quality problems with might affects its authenticity, to discover initial insights into evidence data or to uncover interesting patterns hidden within the large datasets.

c) Data Preparation

Data preparation involved data preprocessing activities to convert the final evidence dataset from the initial unstructured social media raw data which was fed into the spark based forensic tool model. Several tasks were performed on evidence data in multiple times to prepare the data for classification using classifier algorithms.

d) Modelling

This is the phase in which different evidence extraction modeling tools and social media event reconstruction methods were identified, selected and their parameters of interest fine-tuned to get optimal values. Typically, there are several methods for the same evidence mining problem type.

e) Evaluation

Before proceeding to final reporting of the forensic evidence, it was important to more thoroughly evaluate the scenarios/event lines and review the steps executed to construct and extract the relevant ones to be certain it properly achieves the case objectives. A key objective was to determine if there was some important case aspect that has not been sufficiently considered.

f) Deployment (Evidence Presentation)

This was the final phase in which the evidence gained was organized and presented in a way that the forensic investigator can use it for litigation process. It involved subjecting the model into live or real-time Twitter streaming to detect and extract circumstantial evidence. Depending on the forensic requirements, the deployment phase can be as simple as generating evidential report or presenting the evidence before a jury in support of cybercrime committed.

3.2 Sources of Data and Sample Population

Primary data was used to get forensic data for evidence retrieval and data received from social networking sites Twitter, Facebook, LinkIn, Myspace would have been appropriate sample population for the study. However, because of the limited resources and time, the study targeted one of popular social network site, Twitter. Primary data included data collected from actual Twitter pages including tweets/retweets and metadata using Twitter API which enabled us to pull data in real time using Spark Stream module and saving the data into MongoDB for evidence preservation and later text mining/classification using Naive Bayes classifier algorithm and sentimental analysis. In order to have full representation of the entire population, data streaming was carried out using spark stream module to collect real-time tweets and was repeated several times to ensure relatively large volume dataset (3,138,367 million tweets) of tweets and posts are fetched on various cybercrime topics. These data was used to extract features for training and modeling the forensic tool which then was used to carry out real time sentiment analysis on Twitter social network site.

3.3 Data Collection and Data Collection tools

With the data sources identified in section 3.2 above, an Apache spark tool for data collection, mining, and cleaning was implemented using Scala programming languages in Apache Spark. The study focused on social media data and metadata from social network site Twitter. Harvesting of social networking forensic data was facilitated through the use of Twitter API which are specific and available to individual social network sites. The study utilized Streaming API by Twitter API to collect datum from all the data sources to ensure complete data was collected for the research. Integration of the forensic tool with Twitter API ensured that key metadata unique to individual account and which is only available through the publisher's API were captured. Scala programming languages together with Spark Stream were used to perform web crawling and scraping.

The harnessed data collected was stored in Mongodb before text preprocessing was applied and hashed to ensure its authenticity. The use of Python, Scala in spark and Spark stream module was chosen due to its versatility, agility and previous studies have shown it to be a viable solution for web crawling, indexing and scraping with wider support from data science and development communities.

For streaming of data from Twitter, keywords were used particularly the ones oriented to hate speech crimes, bullying like “gun”, “kill”, “murder”, “rape”, “assault”, “kidnap”, “shot”, "gun," "crime," "sinister”, “bitch among others. The web crawled data and metadata were stored in the MongoDB database before data preprocessing and transformation was applied using Spark Mlib library. MongoDB was ideal for storing social media API responses since they are designed to efficiently store JSON data while providing powerful query operators and indexing capabilities (Russell 2013). Digital forensics standards dictate that forensic data to be collected in forensically sound manner and to enforce these standards, key information such as SHA-256 hash keys of individual items and logs were maintained to ensure integrity of the evidence collected is verifiable before a jury.

Individual item SHA-256 hash values were calculated upon capture and before storage to database and maintained through to analysis. Social media Account metadata unique to individual account and tweets were harvested through integration with REST API's provided by Twitter. The social media account metadata in forensic analysis plays very important role in proofing the authenticity of the evidence collected and help in establishing chain of custody.

3.4 Data Preparation

The collected data from social media is usually unstructured and contains unwanted characters such as html tags, xml markups, links, exclamation marks, question marks and other irrelevant characters thus required to be prepared, processed and transformed for data evaluation and validation before it can be ingested into spark classification module. Thus, for this study, data collected from Twitter social network site underwent preprocessing with the intent to reduce some noises, incomplete and inconsistent data. The preprocessing included the following tasks:

a) Text Preprocessing

The collected social media data is usually not only unstructured but also contains other irrelevant and non-textual characters. Text preparation involved cleaning before analysis is performed. The preprocessing is broken down into the following steps:

(i) Tokenization.

Tokenization involves the process of splitting a text into its desired smaller parts (tokens) seeking to isolate as much sentiment information as possible. Tokenization helps in keeping the vocabulary small as possible. Common with social media, emoticons and abbreviations are identified through the process and treated as individual tokens. This was carried out using Apache Spark machine learning Tokenizer and regex pattern matching. Apache Spark ML come packaged with word tokenization feature (`spark.ml.feature.Tokenizer`) that was used in tokenizing.

(ii) Text Normalization.

One challenge involving social media data is the abbreviation (e.g. think's, r, u) of texts hence requires text normalization which will involve replacing abbreviated word by the meaning they represent (e.g. thnks > thanks, u>you). This involved text case conversion from caps to lower case and character repetitions are reduced using the Apache spark Normalizer feature transformer to normalize each vector.

(iii) Stop Word Removal

The data harnessed from Twitter contains words which occurs very frequently but are not useful as they are used to structure words together in a sentence (e.g. the, at, and, etc.) and they don't contribute to the context or content of textual documents. In text analytics, their high frequency occurrence presents an obstacle in understanding the content of the documents. For this research "StopWordsRemover transformers" built-in module in Spark ML feature package was used for stop word removal.

(iv) Part-of-speech (POS) tagging.

Part-of-Speech tagging which is also referred to as grammatical tagging involves assigning words within a sentence their respective part of speech to (such as a verb, noun, conjunctions or adjective) understand its role within the sentence. POS helps to determine what are important keywords within a document or to assist in searching for specific usages of a word in a text document. This involves marking and classification of words in a text sentence based on definition, context, its adjacent relationship with related words in a sentence, or paragraph. Apache Spark includes most popular libraries for NLP in Python among them NLTK, OpenNLP, CoreNLP, WordNet which can be used for text Stemming, lemmatization and POS-tagging.

b) Bag of Words (Space Vector) Model

Bag of words approach is the process of classifying documents where by each word occurrence within the document is used as feature for model training and developing text classifier. A text is represented as a bag of words without paying attention to grammar and even words order but keeping its multiplicity. For document classification, space vector remains the commonly used in method where the frequency or occurrence of each word is used as a feature for training a classifier. This forms part of the text classifiers by taking individual words into account and giving them a specific subjectivity score. Keywords such as "gun", "kill", "murder", "rape", "assault", "fuck", "shot", "nigga," "crime," "sinister", "bitch among others were used to identify crime-related on Twitter posts by matching the word in the tweets/posts in the Bag of word dictionary.

3.5 Data Mining Algorithm and Sentiment Classification

The study employed the use of supervised machine learning approach and specifically Naive Bayes classifier algorithm. Naive Bayes classifier algorithm is simple yet very efficient a kind of classifier which from previous studies has shown to perform well particularly for text classification and sentiment analysis. After the text in Tweet dataset has been segmented into words, the words have been tokenized and normalized, a bag-of-words was modeled by taking individual words and assigning each word a specific subjectivity score whereby if the total score is negative the text was classified as negative and if it's positive the text was classified as positive. This bag-of-words formed a dictionary of words and the training dataset for sentiment analysis/text classification. Sentiment analysis was used to determine an author's attitude with respect to either a particular topic or a document's overall contextual polarity.

3.6 Data Analysis

Twitter streamed data was stored in MongoDB database which formed raw data. The data underwent filtering by using the keywords as search tools which were ran as queries on the database. The filtered data collected was mapped against different crime categories as either cyberbullying, violence based, ethnic based and Sexual based language together with the total counts of each occurrence and exact phrase of crime. Data was analyzed using both Scala and Python libraries and was presented using of graphs and in tabular format. Python was chosen for data analysis because is one of the most popular languages for data analysis and data mining which includes a broad range of libraries suitable for data analysis problems and visualization. It is also an open source software and enjoys a wide support from the data science community.

Scala forms one of the mostly used language for programming distributed computing systems because of its scalability and was one of the languages supported by Spark framework. For this study quantitative methods were used for analysis of the data. Quantitative analysis was used to graph social media crime incidences categories (e.g. sexual, ethnic, religious, and violence, bullying hate speech) against total tweets.

3.7 System Implementation

To achieve the objectives of the study, an Apache Spark Standalone cluster was setup and a web based forensic tool was implemented using Apache Spark which offered a high scalable data intensive processing which is suitable for big data processing like Twitter data. In addition, Spark offers scalable real live streaming module (Spark Streaming) for data, making it suitable for use with Twitter API. Python and Scala programming language was used for both development of logic applications and interfacing. Python and Scala were chosen because of its versatility and mature package libraries around. MongoDB was used as the back end for storing stream data and the data fetched. MongoDB is ideal for storing social media API responses since they are designed to efficiently store JSON data while providing powerful query operators and indexing capabilities.

The implementation comprised of a forensics web search engine, MongoDB database and apache spark classifier. The forensic tool was designed using a dictionary of words and highlighted the crime words together with the full text of the information. It also provided additional information of Twitter account metadata and location where the crime was committed or uttered. The dictionary of words was mainly a dataset of potential crime feature words such as “kill”, “murder”, “rape”, “kidnap”, “shot”, "gun," "crime," "sinister, “bitch” and others which were used as the baseline for assessing the crime forensic data collected.

3.8 Architectural Design

The apache spark forensic system was implemented as a three-tier application using apache spark, MongoDB, Scala, Python programming languages. The Figure 17 below shows an overview of the system components and the interconnections between them that enable it to perform Extract, store, transform and carry forensic analysis of tweets and posts.

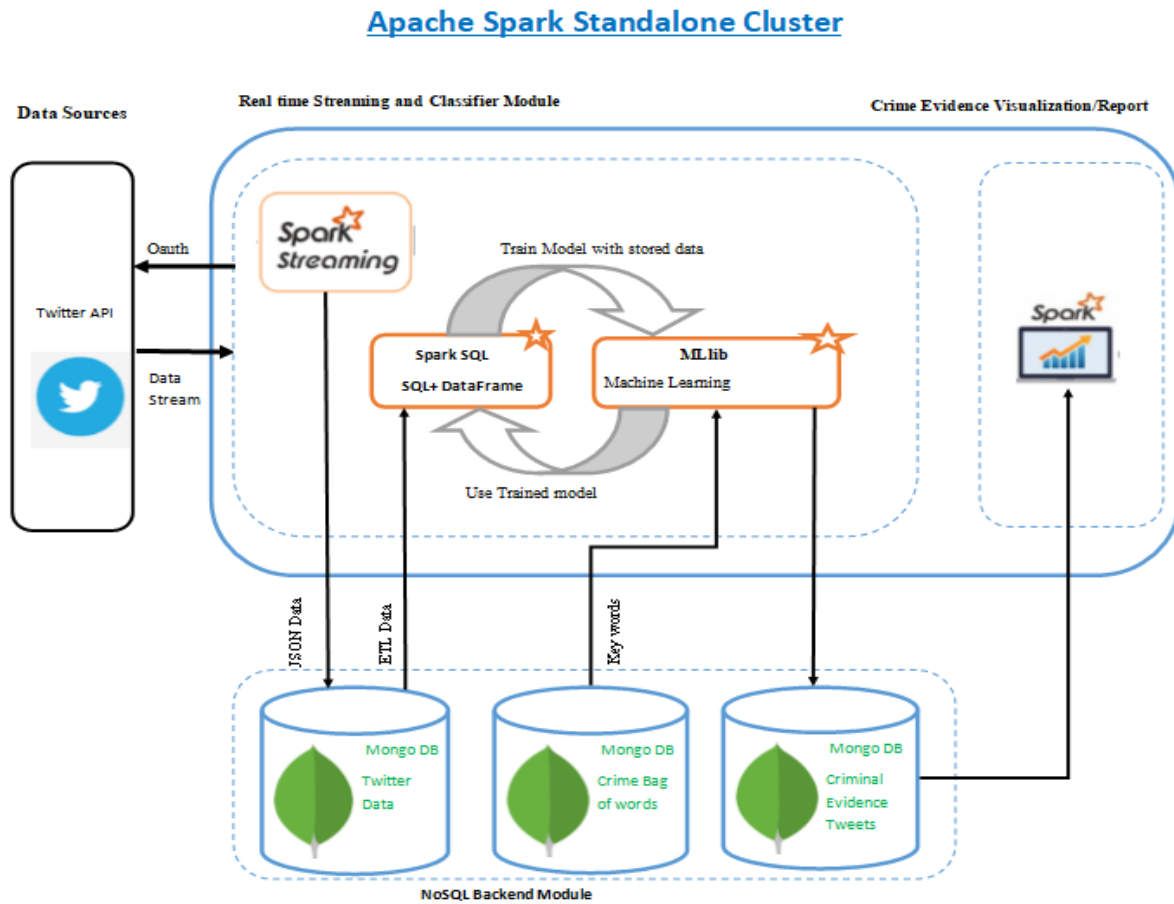


Figure 17: Forensic Tool Architectural Design

a) Spark Streaming Module

Implemented using Spark Streaming module of Apache Spark, this module captures live streams from Twitter API, parses the data in JSON format to the desired format and stores the data in MongoDB database. The data contains all the information about the original tweets, time it was streamed, Analyzer collecting the data and metadata required for authenticity of evidence.

b) Apache Classifier Module

This module was implemented using apache spark using Scala programming languages. The module incorporated classification algorithms specifically Naive Bayes Classifier that is available in Spark ML. Spark ML pipeline was used in providing a set of tokenization, stemming, tagging and stop words removal. Spark ML is used for sentiment analysis and classification of data streams before they are stored on MongoDB and based on the analysis, each tweet is classified as positive (crime) or negative (not crime) sentiment and the result is then persisted into MongoDB as crime evidence which is used later by Report Module.

c) MongoDB Backend Module

This was implemented using MongoDB and it was responsible for storing data streamed from Twitter social network site. It stores raw data from Twitter in the form of JSON document format. MongoDB also stores twitter data which has been classified as of criminal in nature which was later used by reporting module. Bag of words which contains hate/bully words which are used for feature extraction and tweet classification was also stored in MongoDB database.

d) Report Module

This is reporting module which provides visualizations of data classification results as forensic report showing Tweets which were identified as of criminal nature in form of percentages and Charts. It also shows the account user, Tweets and account metadata pertaining the user.

3.9 Model Evaluation

The developed model needed to be evaluated for performance and correctness with test dataset before it is deployed for use on live data stream. To evaluate the performance of the forensic model in terms of quality or predictive effectiveness, different metrics can be used. F-measure (F1-score) is a statistical measure of model's test accuracy which is the weighted harmonic mean of precision and recall of the test where recall is the fraction of all samples classified correctly as positive by the model and Precision describes the ratio of all positives samples classified as true positives by the model. Apache Spark comes with `spark.ml.evaluation` which provides a suite of metrics that are suitable for evaluating the performance of data mining models. For problems involving application of supervised classification like one in this study, there are two outputs which are a true output and output predicted by the model for every data point and therefore the following categorization was used to measure the accuracy and effectiveness of the forensic tool in classifying social media cybercrime. These categories will include

- (i) True Positive (TP) which indicates that the outcome is positive and the model prediction is also positive.
- (ii) True Negative (TN) which indicates that the outcome is negative and the model prediction is also negative.
- (iii) False Positive (FP) which occurs that the outcome is negative (yes) but the model incorrectly predicts the outcome as positive.
- (iv) False Negative (FN) which indicates that the outcome expected is positive but the model incorrectly predicts the outcome as negative.

This is summarized in the following table

	Predicted Negative	Predicted Positive
Negative Outcomes	TN	FP
Positive Outcomes	FN	TP

For this study, F-measure which is provided in `spark.ml.evaluation` was used to evaluate the model performance. F-measure was chosen because it includes metrics like precision and recall which are used in order to take into account errors which might occur if dataset is highly unbalanced.

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

The overall model should have the capacity to:

- a) Collect posts, user account metadata from Twitter social media site and pipeline them appropriately into a MongoDB repository.
- b) Identify, collect and preserve relevant circumstantial evidence (i.e. indirect evidence that relies on an inference to link events to a conclusion of commitment of crime) features and train the forensic model to classify post as either of criminal nature e.g. cyberbullying or hate speech.
- c) Carry out sentiment analysis in terms of Positive or Negative comment regarding social media cybercrime among them cyberbullying and hate speech.
- d) Provide visualization interface for various cybercrime and related account metadata.

3.10 Ethical Issues

The study adhered to privacy of individuals and accounts within the social media and information obtained was only used for this research study. Where necessary the data streamed was anonymized to hide individual account names. Data gathered was secured against use or disclosure beyond the research study.

3.11 Summary

This research study was targeted to investigate the application of Big Data Analytics and data science in Big Data to supplement traditional forensic tools in countering big data forensic challenges. The research was to help in uncovering cybercrimes in social media by extracting and preserving evidence and help bring the culprits into book. Social networking site twitter was used as source of Big Data for analysis to uncover hidden correlation and anomalies hidden within Big Data.

Additionally, the study showed the application of Apache Spark in Digital Forensics involving Big Data such as Twitter. The study was intended to collect social networking sites data/metadata and from it show how big data analytics can be used to analysis large volume of data, uncover hidden correlation and show forensic artifacts which are important in social media forensics. The results from the inputs would pave the way for the development of apache spark big data forensic tool for collecting and analyzing Big Data found on social networks.

CHAPTER FOUR: DESIGN AND IMPLEMENTATION

4 Introduction

This chapter introduces the core instruments which were used to design, implement and test the proposed forensic tool. It included setting up apache spark standalone cluster, configuration of programming environment and apache spark twitter streaming to collect forensic data which was subjected to sentimental analysis to identify hate speech and cyberbullying using Spark ML Module to generate forensic model. The model was trained using 3,138,367 million tweets and used to correctly classify twitter data according to the three categories namely positive, neutral, negative (hate speech/cyberbullying). The chapter concludes with the evaluation of the model and analysis of the forensic tool results for this study.

4.1 Modeling Tools and Techniques

We used Scala and Python which are leading powerful tool used in data mining, machine learning due to its productive user interface. The following table shows a list of system development tools which were used in the development of the forensic tool.

TOOL	DESCRIPTION
VMware Workstation 12 Pro	The Apache Spark cluster for the project was setup in virtualized environment, to achieve this, VMware Workstation was used as the virtualization environment.
Ubuntu 14.04 LTS	This formed the operating system upon which Apache spark cluster was setup and web server for front-end reporting interface.
MongoDB 3.4.2	To store tweet data and classified tweets, the project utilized open source NOSQL database MongoDB for easy storage and retrieval.
Apache Spark 2.1.0 Framework	This was used to setup big data distributed computing cluster which was used to stream Twitter data and Big data analysis.
Apache Server	This was used to setup webserver to serve front end reporting module.
Flask 0.12 Framework	Flask framework was used to design the front-end reporting web interface.

PyCharm 2017.1 IDE	This formed the IDE for developing the front-end reporting website in python.
IntelliJ IDEA 2017.1 IDE	To develop the back end big data program, the project adopted IntelliJ IDEA 2017.1 as Scala IDE.
SCALA	Scala formed the core language for developing distributed computing program which were executed on the apache cluster.
PYTHON	To develop the front-end reporting module, the project utilized python language.
Bootstrap v3.3.7	To apply styling on the front-end reporting module, the project made use of Bootstrap v3.3.7.
PyMongo 3.4.0	This was used to connect python scripts to MongoDB and retrieve data.
Chart.js	Chart.js was used for visualization of project reports and analysis graphs.

Table 1: Forensic design software and Tools

4.2 Spark Forensic Model Analysis

Functional Requirements

- a) Stream Twitter JSON data via Twitter API.
- b) Store the Twitter on Local Hard disk for Model training and Testing.
- c) Train and Create Naïve Bayes model to classify tweets as either Hate speech or bullying.
- d) Stream Live tweets using the trained model and classify the tweets.
- e) Store streamed tweets in MongoDB for evidence preservation.
- f) Store Twitter data stream and date for Evidence preservation and authentication.
- g) Generate SHA-256 Hash key for each tweet post and store the key in MongoDB.
- h) Categorize classified tweets as either Ethnicity, Religious, Violence, Bully and Others.

Reporting Front End Module

- a) Connect to MongoDB and retrieve classified tweets with Account Metadata.
- b) Connect to MongoDB and retrieve Raw Tweets with Stream data and Hash Keys.
- c) Provide Search and filter capabilities.

4.3 Forensic Tool Module Analysis

The forensic tool was designed and programmed using both Scala and Python programming languages and is composed of mainly nine main components:

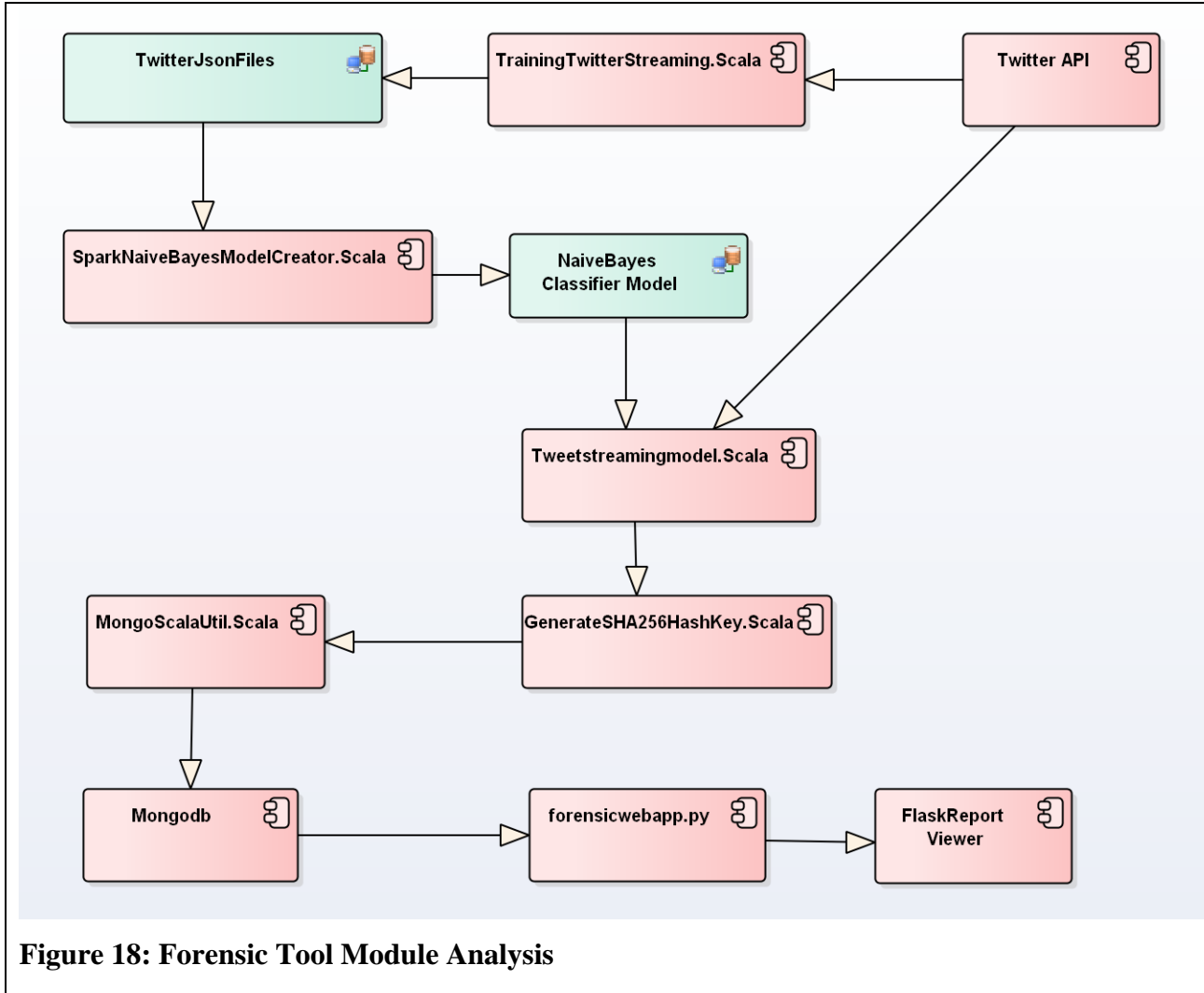


Figure 18: Forensic Tool Module Analysis

a) Training TwitterStreaming.Scala

This was Scala based module implemented Spark Streaming API and was responsible for streaming live tweets and storing them on local hard disk under Twitter Json files. This tweets were used for training Naïve Bayes model.

b) SparkNaiveBayesModelCreator.Scala

This was Scala based module which implemented Spark ML API and Naïve Bayes classifier pipelines. The module worked by loading locally stored JSON tweets and training Naïve Bayes classifier model which was saved on local disk as “NaiveBayes Classifier Model”.

c) Tweetstreamingmodel.Scala

This was live tweet streaming module which was implemented using Spark ML API and utilized earlier saved trained model to stream live tweets and classifying them as hate speech or bullying. It was also responsible for categorizing the tweets in different categories as either bully, ethnicity, sexual, religious and others for tweets which didn't fall under the defined categories.

d) MongoScalaUtil.Scala

This module implemented Mongoddb connector for spark and utilized MongoDB Scala Driver for storing of live classified tweets in forensicdb database within Mongoddb. It was also responsible to persisting raw tweets to Mongoddb database for evidence preservation and authenticity.

e) Mongoddb

This module formed the backend storage for preservation of evidence and classified tweets. The study utilized Mongoddb 3.4 version for tweet storage and preservation.

f) Forensicwebapp.Py

This is a Flask based web application which gets data which connects to MongoDB using PyMongo module to retrieve classified tweets for presentation. The module was also used for tweet analysis using graphs and charts.

g) FlaskReport Viewer

Implemented using flask, HTML and JavaScript, this module was responsible for presenting forensic hate speech reports and analysis graphs.

4.4 Cluster Setup and Configurations

To achieve the objectives of the project as stated in chapter one, the project implemented Apache Spark cluster as the backend for tweet streaming and data processing. The cluster was implemented on Ubuntu 14.4 operating system. Apache Spark can be deployed in three ways depending on individual needs i.e. Standalone, YARN, and Apache Mesos.

Standalone deployment: With the standalone deployment, one can statically allocate resources on all or a subset of machines in a Hadoop cluster and run Spark side by side with Hadoop. The user can then run arbitrary Spark jobs on her HDFS data.

Hadoop Yarn deployment: Hadoop users who have already deployed or are planning to deploy Hadoop Yarn can simply run Spark on YARN without any pre-installation or administrative access required. This allows users to easily integrate Spark in their Hadoop stack and take advantage of the full power of Spark, as well as of other components running on top of Spark.

4.4.1 Hadoop Yarn Configuration

The following shows the configuration for Hadoop yarn configuration which was used by Apache spark for resource allocation and management.

Core-site.xml

```
<configuration>
<property>
  <name>fs.default.name</name>
  <value>hdfs://KENBO-SPK08.forensics.net:54310</value>
</property>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/usr/local/hadoop/tmp</value>
</property>
</configuration>
```

Figure 19: Core-site.xml Configurations

YARN configuration properties (**yarn-site.xml**)

```
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>KENBO-SPK08.forensics.net</value>
</property>
<property>
  <name>yarn.nodemanager.local-dirs</name>
  <value>/usr/local/hadoop/yarn/data</value>
</property>
<property>
  <name>yarn.nodemanager.logs-dirs</name>
  <value>/usr/local/hadoop/logs</value>
</property>
</configuration>
```

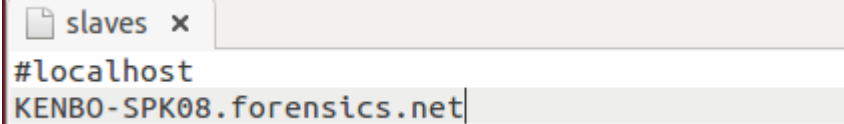
Figure 20: Yarn-site.xml Configurations

HDFS node configuration (**hdfs-site.xml**)

```
<configuration>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
</property>
</configuration>
```

Figure 21: hdfs-site.xml Configurations

Slaves.xml



```
slaves x
#localhost
KENBO-SPK08.forensics.net
```

Figure 22: Slaves.xml Configurations

4.4.2 Starting Hadoop Cluster Manger

Running Apache Spark on Hadoop requires both HDFS file system and Hadoop resource manager to be up and running. The Hadoop cluster manager comes with scripts which are used for starting and stopping the cluster. These scripts are stored under bin folder on Hadoop home folder. The scripts are start-dfs.sh and start-yarn.sh which are used for starting the HDFS/Namenode and ResourceManager and NodeManager daemon respectively.

```
smulwa@KENBO-SPK08:/usr/local/hadoop/sbin$ ./start-dfs.sh
```

```
smulwa@KENBO-SPK08:/usr/local/hadoop/sbin$ ./start-yarn.sh
```

```
smulwa@KENBO-SPK08:~$ cd /usr/local/hadoop/sbin
smulwa@KENBO-SPK08:/usr/local/hadoop/sbin$ ./start-dfs.sh
Starting namenodes on [KENBO-SPK08.forensics.net]
KENBO-SPK08.forensics.net: starting namenode, logging to /usr/local/hadoop/logs/hadoop-smulwa-namenode-KENBO-SPK08.forensics.net.out
KENBO-SPK08.forensics.net: starting datanode, logging to /usr/local/hadoop/logs/hadoop-smulwa-datanode-KENBO-SPK08.forensics.net.out
Starting secondary namenodes [account.jetbrains.com]
account.jetbrains.com: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-smulwa-secondarynamenode-KENBO-SPK08.forensics.net.out
smulwa@KENBO-SPK08:/usr/local/hadoop/sbin$ ./start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-smulwa-resourcemanager-KENBO-SPK08.forensics.net.out
KENBO-SPK08.forensics.net: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-smulwa-nodemanager-KENBO-SPK08.forensics.net.out
smulwa@KENBO-SPK08:/usr/local/hadoop/sbin$ jps
3585 SecondaryNameNode
3761 ResourceManager
3926 NodeManager
3160 NameNode
3323 DataNode
4238 Jps
```

Figure 23: Starting Hadoop Cluster Manger

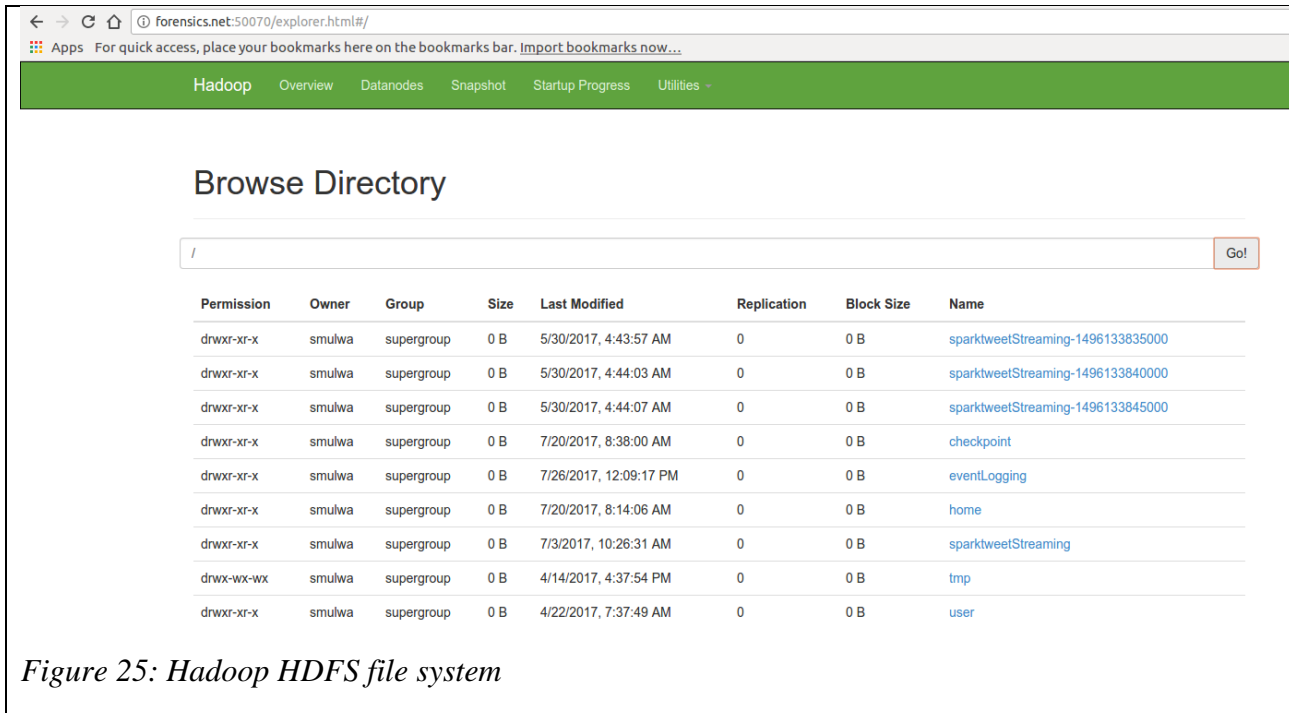
After starting the Hadoop/HDFS cluster resource manager through the URL <http://forensics.net:8088/cluster/nodes>, the following page is opened.

The screenshot shows the Hadoop cluster resource manager web interface. The page title is "Nodes of the cluster". On the left, there is a navigation menu with options like "Cluster", "About", "Nodes", "Node Labels", "Applications", "NEW", "NEW SAVING", "SUBMITTED", "ACCEPTED", "RUNNING", "FINISHED", "FAILED", "KILLED", and "Scheduler". The main content area displays "Cluster Metrics" and "Scheduler Metrics". Below these, there is a table showing the status of nodes in the cluster.

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	VCores Used	VCores Avail	Version
/default-rack		RUNNING	KENBO-SPK08.forensics.net:33836	KENBO-SPK08.forensics.net:8042	Sat Aug 26 07:29:25 -0400 2017		0	0 B	8 GB	0	8	2.7.3

Figure 24: Hadoop/HDFS cluster resource manager

The following screen shot shows Hadoop HDFS file system showing browser directory.



The screenshot shows a web browser window with the URL `forensics.net:50070/explorer.html#/`. The page title is "Browse Directory" and the address bar shows `/`. A navigation menu at the top includes "Hadoop", "Overview", "Datanodes", "Snapshot", "Startup Progress", and "Utilities". Below the navigation menu is a search bar with a "Go!" button. The main content area displays a table of files and directories in the HDFS file system.

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	smulwa	supergroup	0 B	5/30/2017, 4:43:57 AM	0	0 B	sparktweetStreaming-1496133835000
drwxr-xr-x	smulwa	supergroup	0 B	5/30/2017, 4:44:03 AM	0	0 B	sparktweetStreaming-1496133840000
drwxr-xr-x	smulwa	supergroup	0 B	5/30/2017, 4:44:07 AM	0	0 B	sparktweetStreaming-1496133845000
drwxr-xr-x	smulwa	supergroup	0 B	7/20/2017, 8:38:00 AM	0	0 B	checkpoint
drwxr-xr-x	smulwa	supergroup	0 B	7/26/2017, 12:09:17 PM	0	0 B	eventLogging
drwxr-xr-x	smulwa	supergroup	0 B	7/20/2017, 8:14:06 AM	0	0 B	home
drwxr-xr-x	smulwa	supergroup	0 B	7/3/2017, 10:26:31 AM	0	0 B	sparktweetStreaming
drwx-wx-wx	smulwa	supergroup	0 B	4/14/2017, 4:37:54 PM	0	0 B	tmp
drwxr-xr-x	smulwa	supergroup	0 B	4/22/2017, 7:37:49 AM	0	0 B	user

Figure 25: Hadoop HDFS file system

4.4.3 Apache Spark Configuration

The following show the configuration for the Apache Spark which were implemented. This involved downloading the precompiled version from <https://spark.apache.org> site.

spark-defaults.config

```
spark.master                spark://forensics.net:7077
spark.serializer             org.apache.spark.serializer.KryoSerializer
spark.eventLog.enabled true
spark.history.kerberos.enabled false
spark.history.ui.port 18080
spark.eventLog.dir hdfs://forensics.net:54310/eventLogging
spark.history.fs.logDirectory hdfs://forensics.net:54310/eventLogging
spark.history.kerberos.keytab none
spark.history.kerberos.principal none
spark.yarn.historyServer.address forensics.net:18080
spark.yarn.queue default
```

Figure 26: Spark-defaults configuration

Spark-env.sh

```
export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
export YARN_CONF_DIR=/usr/local/hadoop/etc/hadoop
SCALA_HOME=/usr/share/scala
SPARK_MASTER_HOST=forensics.net
SPARK_CONF_DIR=/usr/local/spark/conf
SPARK_WORKER_DIR=/usr/local/spark/data
#SPARK_WORKER_INSTANCES=4
#SPARK_EXECUTOR_INSTANCES=4
#SPARK_WORKER_MEMORY=4
#SPARK_WORKER_CORES=4
#SPARK_EXECUTOR_CORES=4
#SPARK_EXECUTOR_MEMORY=4
```

Figure 27: Spark-env.sh Configuration

log4j.properties

```
# Set everything to be logged to the console
log4j.rootCategory=WARN, console
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n

# Set the default spark-shell log level to WARN. When running the spark-shell, the
# log level for this class is used to overwrite the root logger's log level, so that
# the user can have different defaults for the shell and regular Spark apps.
log4j.logger.org.apache.spark.repl.Main=WARN

# Settings to quiet third party logs that are too verbose
log4j.logger.org.spark_project.jetty=WARN
log4j.logger.org.spark_project.jetty.util.component.AbstractLifeCycle=ERROR
log4j.logger.org.apache.spark.repl.SparkIMain$ExprTyper=INFO
log4j.logger.org.apache.spark.repl.SparkILoop$SparkILoopInterpreter=INFO
log4j.logger.org.apache.parquet=ERROR
log4j.logger.parquet=ERROR

# SPARK-9183: Settings to avoid annoying messages when looking up nonexistent UDFs in SparkSQL with Hive support
log4j.logger.org.apache.hadoop.hive.metastore.RetryingHMSHandler=FATAL
log4j.logger.org.apache.hadoop.hive.ql.exec.FunctionRegistry=ERROR
```

Figure 28: Spark log4j.properties Configuration

Spark Worker configurations (slaves)

```
# A Spark Worker will be started on each of the machines listed below.
#localhost
forensics.net
```

Figure 29: Spark Worker configurations (slaves)

4.4.4 Starting Apache Spark Cluster

Like with Hadoop resource manager, Apache Spark comes with scripts which automates the process of starting and stopping of cluster. Among the scripts are *start-master.sh* and *start-slave.sh* which are used for starting spark master node and slave node respectively. The following shows how to start the forensic spark cluster. These scripts are found in the sbin folder with the Spark home folder/

```
smulwa@KENBO-SPK08:/usr/local/spark/sbin$ ./start-master.sh
```

```
smulwa@KENBO-SPK08:/usr/local/spark/sbin$ ./start-slave.sh spark://forensics.net:7077
```

```
smulwa@KENBO-SPK08:/usr/local/hadoop/sbin$ cd /usr/local/spark/sbin
smulwa@KENBO-SPK08:/usr/local/spark/sbin$ ./start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /usr/local/spark/logs/spark-smulwa-org.apache.spark.deploy.master.Master-1-KENBO-SPK08.forensics.net.out
smulwa@KENBO-SPK08:/usr/local/spark/sbin$ ./start-slave.sh spark://forensics.net:7077
starting org.apache.spark.deploy.worker.Worker, logging to /usr/local/spark/logs/spark-smulwa-org.apache.spark.deploy.worker.Worker-1-KENBO-SPK08.forensics.net.out
smulwa@KENBO-SPK08:/usr/local/spark/sbin$ ./start-history-server.sh
starting org.apache.spark.deploy.history.HistoryServer, logging to /usr/local/spark/logs/spark-smulwa-org.apache.spark.deploy.history.HistoryServer-1-KENBO-SPK08.forensics.net.out
smulwa@KENBO-SPK08:/usr/local/hadoop/sbin$
```

Figure 30: Starting Apache Spark Cluster

4.5 Twitter API Connection

Data for the project was collected from Twitter using Twitter API and Scala custom codes. To stream data from twitter, the application was required to have OAuth authentication with a Twitter account. To do this, we had to setup a consumer key/secret pair and an access token/secret pair using a Twitter account.

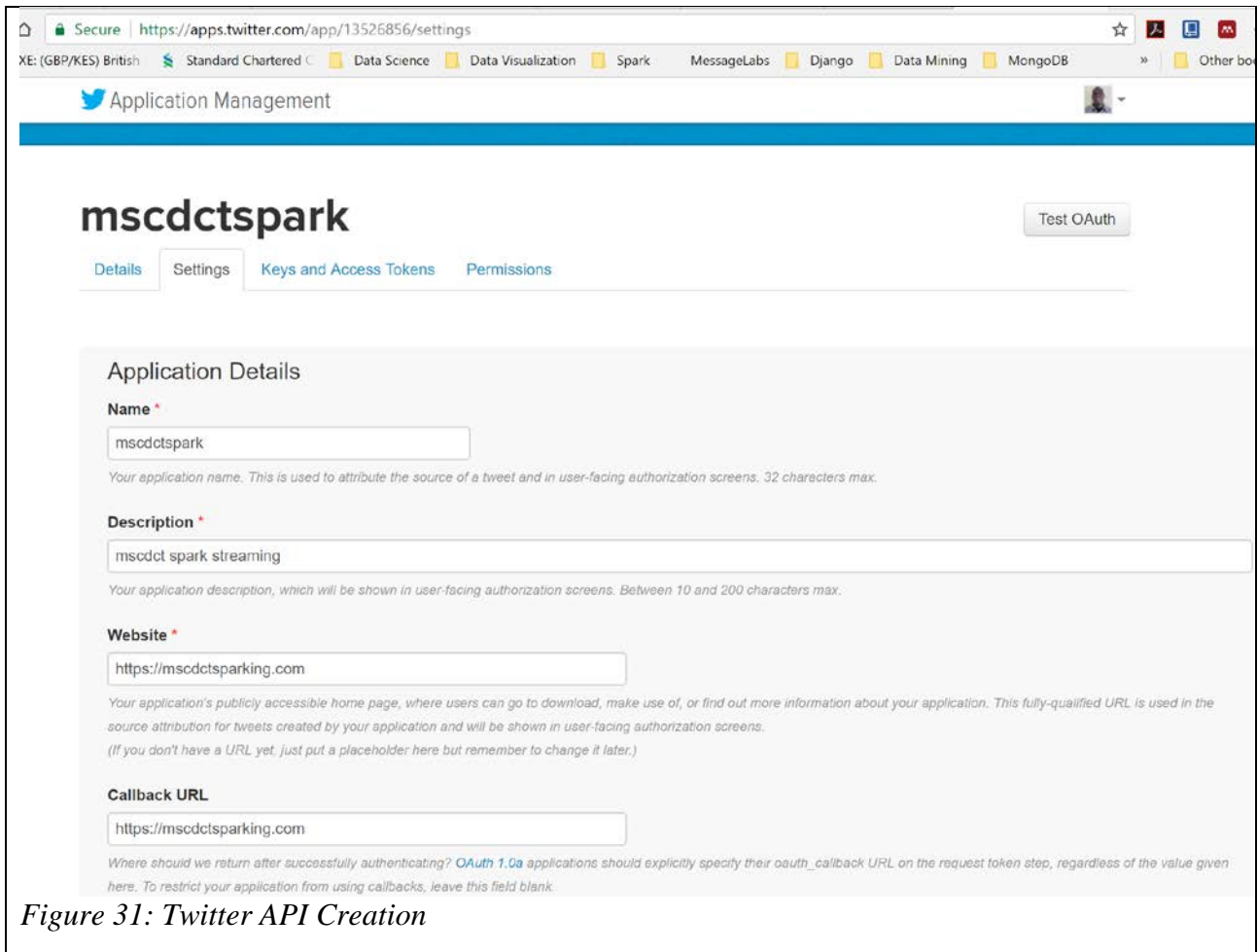


Figure 31: Twitter API Creation

This allowed us to get twitter account metadata, which is not publicly available on twitter page but can be retrieved upon authentication. The Twitter consumer key/pair allows the forensic application to authenticate with twitter before it can stream data.

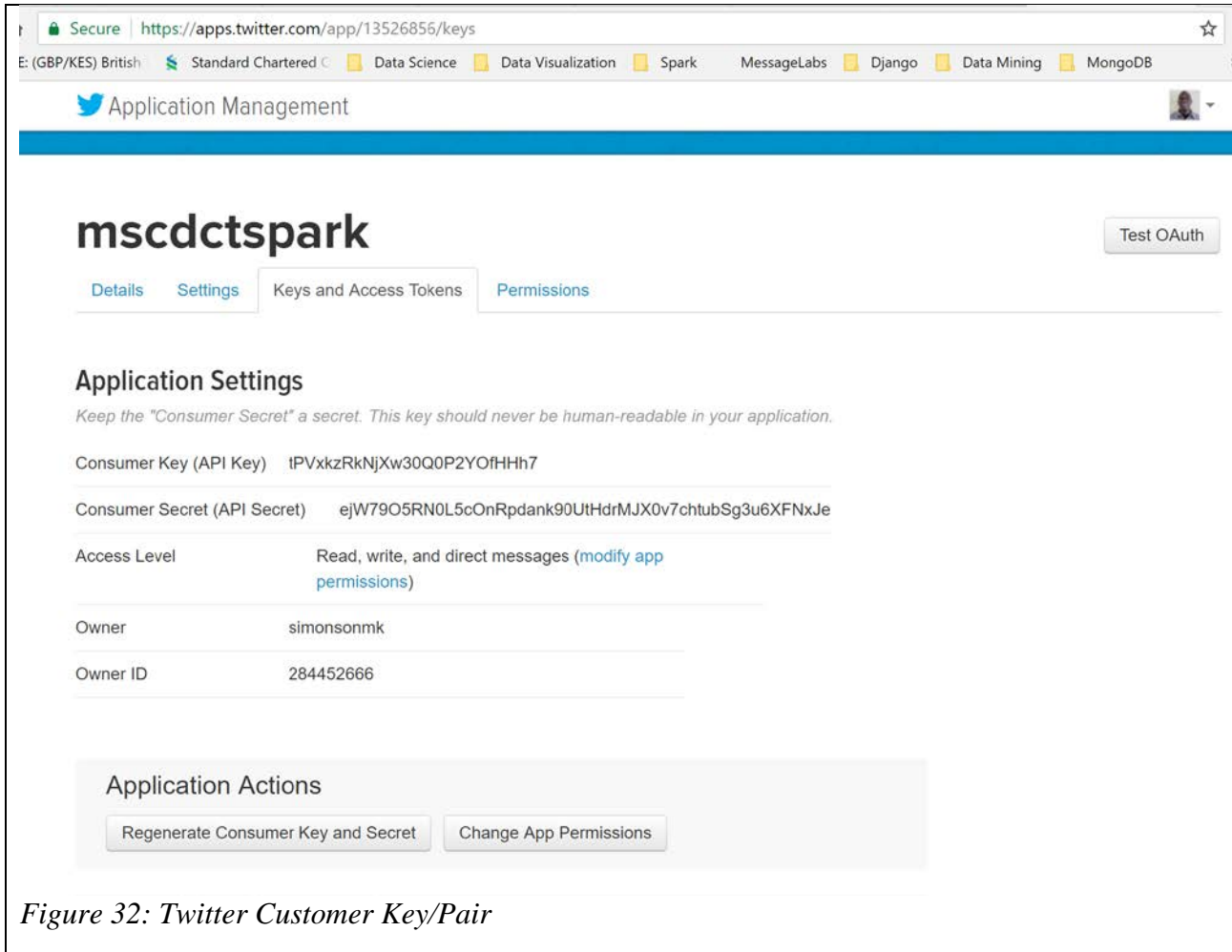


Figure 32: Twitter Customer Key/Pair

Below screenshot shows part of the key/pairs which were used by the application to get tweet data and account metadata

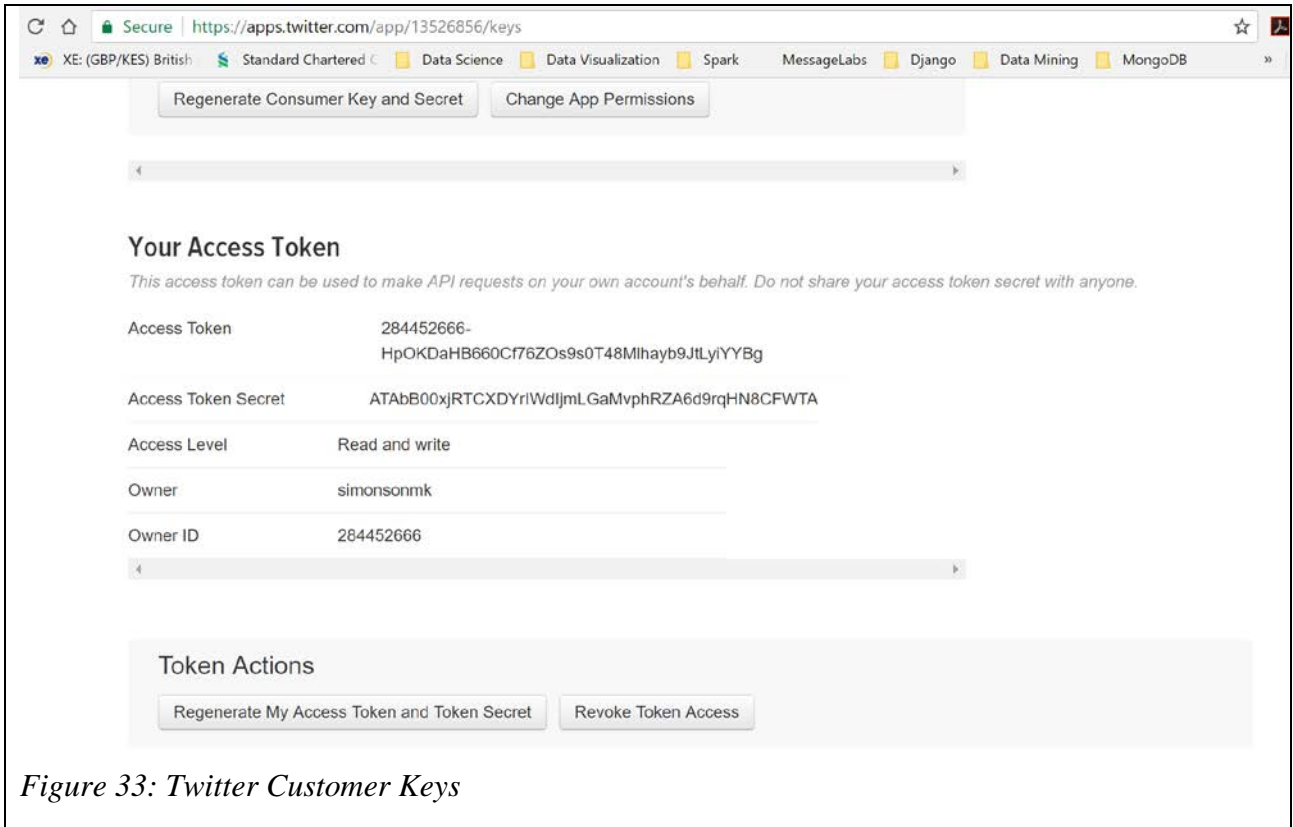


Figure 33: Twitter Customer Keys

These key pairs were used to authenticate and allowed us get data streaming using spark streaming API. These customer authentication key/pair code were stored within the application.conf file with the spark application project and were referenced by the OAuthUtilities.Scala module.

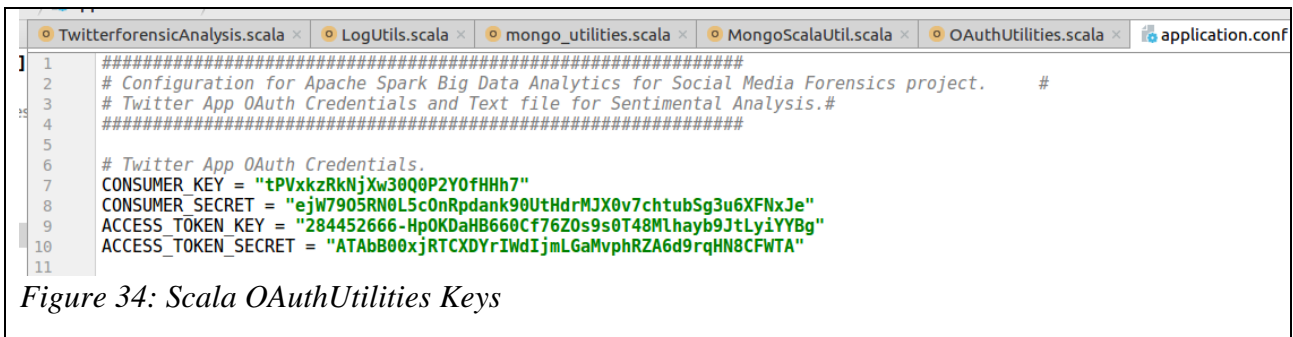


Figure 34: Scala OAuthUtilities Keys

The following Scala code (OAuthUtilities.Scala module) shows how the key/pair were used in the application.

```
import twitter4j.auth.OAuthAuthorization
import twitter4j.conf.ConfigurationBuilder

object OAuthUtilities {

  def getTwitterOAuth(): Some[OAuthAuthorization] = {

    val twitterconfigbuilder = new ConfigurationBuilder
    twitterconfigbuilder.setDebugEnabled(true)
    .setOAuthConsumerKey(loadconfigProperties.consumerKey) //Load Authentication credentials from application config
    .setOAuthConsumerSecret(loadconfigProperties.consumerSecret)
    .setOAuthAccessToken(loadconfigProperties.accessToken)
    .setOAuthAccessTokenSecret(loadconfigProperties.accessTokenSecret)
    .setIncludeEmailEnabled(true)
    .setTweetModeExtended(true)

    val twitteroAuth = Some(new OAuthAuthorization(twitterconfigbuilder.build()))

    twitteroAuth //Return Twitter Authentication details
  }
}
```

Figure 35: OAuthUtilities.Scala Module

4.6 Data Collections

Data for the project was streamed from Twitter social network site and we utilized publicly accessible Twitter API with Spark streaming API. We streamed 3,138,367 tweets which were stored locally on local hard disk. The tweets were filtered using a set of keywords that are viewed as offending, insulting, and intimidating to people or of inflammatory in nature or bullying. For this study, several keywords were used among them- hate you, nigga, stupid, idiot, fuck you, faggot, kill, bitch, dyke, gay, black nigger, white people, black people, ugly, terrorists. These keywords were also categorized into different groups based on their biasness i.e. ethnicity, religious, sexual, violence, bully. The collected Tweets formed our tweets dataset which were later used to train Naïve Bayes Model to analyze and classify tweets as either positive sentiments, hate (negative) sentiments, and neutral sentiments.

4.7 Feature Selection

This involved selecting a subset of relevant features that would help in identifying inflammatory or offensive tweets and can be used in the modeling of the classification problems using Naïve Bayes model. We did stream the whole Twitter profile account and retrieved all the properties or features making a Twitter Account. This was presented in JSON. The study focused more on Twitter status update which is represented as text. The text field formed the main feature of interest for the study as its Twitter's status update for users. For Twitter, forensic analysis we grouped the feature set into two categories i.e. comment based features and metadata based features. Comment based features involved Twitter comments and replies to the comments and metadata based features involve Account features such as created_at, tweet id, account id, name, screenname, coordinates among others. The Account metadata can be used for account authentication of forensic data and was also focus of the study for forensic evidence preservation and authentication. The figure 36 below shows part of Twitter Account Structure and data types.

```

root
|-- accessLevel: long (nullable = true)
|-- contributors: array (nullable = true)
|   |-- element: string (containsNull = true)
|-- createdAt: long (nullable = true)
|-- currentUserRetweetId: long (nullable = true)
|-- displayTextRangeEnd: long (nullable = true)
|-- displayTextRangeStart: long (nullable = true)
|-- favoriteCount: long (nullable = true)
|-- favorited: boolean (nullable = true)
|-- geolocation: struct (nullable = true)
|   |-- latitude: double (nullable = true)
|   |-- longitude: double (nullable = true)
|-- hashtagEntities: array (nullable = true)
|   |-- element: struct (containsNull = true)
|       |-- end: long (nullable = true)
|       |-- start: long (nullable = true)
|       |-- text: string (nullable = true)
|-- id: long (nullable = true)
|-- inReplyToScreenName: string (nullable = true)
|-- inReplyToStatusId: long (nullable = true)
|-- inReplyToUserId: long (nullable = true)
|-- lang: string (nullable = true)
|-- mediaEntities: array (nullable = true)
|   |-- element: struct (containsNull = true)
|       |-- displayURL: string (nullable = true)
|       |-- end: long (nullable = true)
|       |-- expandedURL: string (nullable = true)
|       |-- id: long (nullable = true)
|       |-- mediaURL: string (nullable = true)
|       |-- mediaURLHttps: string (nullable = true)
|       |-- sizes: struct (nullable = true)
|           |-- 0: struct (nullable = true)
|               |-- height: long (nullable = true)
|               |-- resize: long (nullable = true)
|               |-- width: long (nullable = true)
|           |-- 1: struct (nullable = true)
|               |-- height: long (nullable = true)
|               |-- resize: long (nullable = true)
|               |-- width: long (nullable = true)
|           |-- 2: struct (nullable = true)
|               |-- height: long (nullable = true)
|               |-- resize: long (nullable = true)
|               |-- width: long (nullable = true)
|           |-- 3: struct (nullable = true)
|               |-- height: long (nullable = true)
|               |-- resize: long (nullable = true)
|               |-- width: long (nullable = true)
|       |-- start: long (nullable = true)
|       |-- text: string (nullable = true)
|       |-- type: string (nullable = true)

```

Figure 36: Twitter Account Schema

4.8 Data Preprocessing

The collected data from Twitter is usually unstructured and contained a lot of unwanted characters such as html tags, xml markups, links, exclamation marks, question marks and other irrelevant characters thus the data required to be prepared, processed and transformed for data evaluation and validation before it can be ingested into spark classification module. This preprocessing involved removal of special symbols from text features, stop words, and text tokenization. Tokenization involved breaking down a sentence into meaningful words/elements called tokens. This was followed by removal of words which are very common within a text but that do not contribute significantly to the relevant content. To remove unnecessary special characters/symbols within the tweets, we used pattern matching regular expression to capture special symbols to be eliminated from the tweets. The following figure shows the code scripts in Scala programming language which we utilized for tweet text cleaning and stop-word removal. We used a corpus dictionary of words to remove words which don't contribute much to the structure of the tweet.

```
def getTweetTextcleaner(tweetText: String, stopWordsList: List[String]): Seq[String] = {  
  //Clean tweet text by removing URLs, RT and other redundant chars / strings.  
  tweetText.toLowerCase()  
  .replaceAll("\\n", "")  
  .replaceAll("rt\\s+", "")  
  .replaceAll("\\s+@\\w+", "")  
  .replaceAll("@\\w+", "")  
  .replaceAll("\\s+#\\w+", "")  
  .replaceAll("#\\w+", "")  
  .replaceAll("(?:https?|http?)://[\\w/%.-]+", "")  
  .replaceAll("(?:https?|http?)://[\\w/%.-]+\\s+", "")  
  .replaceAll("(?:https?|http?)//[\\w/%.-]+\\s+", "")  
  .replaceAll("(?:https?|http?)//[\\w/%.-]+", "")  
  .split("\\W+")  
  .filter(_.matches("[a-zA-Z]+"))  
  .filter(!stopWordsList.contains(_))  
}
```

Figure 37: Tweet Cleaning Module

The above function code receives twitter text, stop word list as arguments and returns plain tweet text which is subjected to labeling to classify the tweet as either negative (hate speech in nature), positive or of neutral sentiment.

4.9 Training Tweet Labelling

Naïve Bayes machine learning classifier requires that training dataset to be labeled as either positive, negative (hate) or neutral. This labeling can be done manually by going through each tweet and labeling it as positive, negative or neutral. Although this will create a highly reliable corpus lexicon, it can be a very tedious exercise for large volumes of tweets. For this study, we adopted an automatic labeling approach by collecting 3,138,367 of tweets and running a Scala function through those tweets which compares each individual tweet word with a list of positive words and a dictionary of hate speech words. The figure below shows Scala function which we adopted for tweet labeling.

```
def CalculateTweetSentiment(tweet: String): Double= {
  var sentimentscore = 0
  for (tword <- tweet.split(" ")) {
    for (positiveWord <- positiveWordsArr) {
      if (tword != "" && positiveWord.toString.toLowerCase() == tword) {
        sentimentscore = sentimentscore + 1
      }
    }
    for (negativeword <- negativeWordsArr) {
      if (tword != "" && negativeword.toString.toLowerCase() == tword) {
        sentimentscore = sentimentscore - 1
      }
    }
  }
  if (sentimentscore > 0) {
    //return 1 for positive sentiment
    return 1;
  }
  else if (sentimentscore < 0) {
    //return 0 negative/hate sentiment
    return 0
  }
  else
    //return 2 for neutral sentiment
    return 2
}
```

Figure 38: Tweet Sentiment Classifier Module

The labeled tweets were later fed into Spark ML Pipeline to generate Naïve Bayes Classifier model to automatically classify the tweet as either hate speech, bullying in nature are positive/neutral. The following figure shows sample of labeled tweets.

```
+-----+-----+
|          text|label|
+-----+-----+
|stop texting firs...| 0.0|
|you not toys but ...| 0.0|
|me right now to a...| 2.0|
|      trump is an| 1.0|
|sexual jokes and...| 1.0|
|      next tt rr| 2.0|
|have you booked y...| 0.0|
|a best friend is ...| 1.0|
|order was repeal ...| 2.0|
|woman holding phd...| 0.0|
|nigga must have b...| 0.0|
|so shut ur fuckin...| 0.0|
|farms and its not...| 0.0|
|anybody else seen...| 2.0|
|also me on judgem...| 2.0|
|made fresh bids o...| 2.0|
|gorakhpur is our ...| 2.0|
|police police vis...| 0.0|
|lok what song is ...| 0.0|
|      discuss| 2.0|
+-----+-----+
only showing top 20 rows
```

Figure 39: Labeled Tweets

4.10 Social Media Evidence Identification

Social media users create massive amounts of data which becomes challenging when trying to extract evidence as it's hidden with enormous big data. In this study, 3,138,367 tweets were streamed and analyzed to identify hate speech and bullying tweets. This involved identifying which attributes might be used as evidence for commitment of cybercrime in Twitter Social Network. The study went ahead to identify also attributes which might be used to supporting the evidence and whether such tweet account was used to commit the said cybercrime or hate speech. In this study, it was noted that user status updates what is commonly referred as tweets are used to express hate speech or bully comments. The status updates (tweet) are represented as text field in twitter account structure as shown in table below. Twitter evidence can also include photographs which might carry out inflammatory messages or contents which might be of hate speech or bullying in nature. With the development of GPS smartphones and location-based services, Twitter enables user to tag or provide location information which might be used during search warrant of a culprits in case of cybercrime commitment on Twitter.

Twitter Field Name		Evidence/Comments
Twitter Status Update (Twitter Text field)		Tweet updates which indicate user's/Account status updates or posts.
originalProfileImageURL		This shows users profile picture as uploaded by the user.
created_at		This can be used to show when such tweet was created and can be used to authenticate when tweet which has be categorized as hate speech/bullying was created or posted.
id		This is unique identifier for the tweet in question and can be used to uniquely identify each individual tweet.
Users	created_at	The time when the account in question was opened with twitter.
	id	A unique identifier which represents twitter user account.
	location	This defines the user location for this account's profile and might be used to identify the location of the user or where user might have created the account in.
	Name	The name of the user, as they've defined it.
	profile_image_url	The account user's profile image which can be used to identify the user physically.
	screen_name	An alias which the user identifies himself with.

Table 2: Twitter Account Metadata of interest in forensics

4.10.1 Evidence Retrieval

Social media users create massive amounts of data which becomes challenging when trying to extract evidence as it's hidden with enormous big data. As highlighted in the above table, Twitter Account profile encompasses many fields which are not possible to be retrieved by snapshotting or printing Twitter web pages. This invalidates such evidence collected by screen shots or web page printouts as it lacks the supporting metadata. To improve on evidence validity, this study focused on an automatic retrieval of Tweet user status updates together with the Account metadata as supporting evidence which might be relevant in proofing authenticity before a court of law. To achieve this, we designed the forensic tool to retrieve Twitter status updates together with account metadata.

The above table shows Twitter schema attributes which were automatically retrieved and which we felt can be used to authenticate the evidence received. The Forensic tool streams live tweets using Twitter API together with Spark Stream API. SHA-256 hash key for each individual tweet was generated and stored in MongoDB together with each individual user tweet both accompanied by Twitter Account unique identifier. The following figure shows the Scala Code used for saving the tweet to MongoDB.

```
object MongoScalaUtil {
  def SaveRawTweetsToMongodb(tweet:String, twitter_account_id:Long , tweet_id:Long,cTime:String):Unit={
    try {
      val mongoClient = MongoClient()
      val database: MongoDB = mongoClient.getDatabase("forensicdb")
      val collection: MongoCollection[Document] = database.getCollection("Tweets")

      val tweethashkey = SentUtilities.generateSHA256HashKey(tweet)
      val doc: Document = Document("_id" -> tweet_id, "twitter_account_id" -> twitter_account_id, "cTime" -> cTime, "usertweets" -> tweet, "tweethashkey" -> tweethashkey)

      val observable: Observable[Completed] = collection.insertOne(doc)
      observable.subscribe(new Observer[Completed] {
        override def onNext(result: Completed): Unit = println("Inserted")

        override def onError(e: Throwable): Unit = println(e.toString)

        override def onComplete(): Unit = println("Completed")
      })
      // mongoClient.close()
    }
    catch {
      case e: Exception => println("Error Saving tweets to Mongodb: ", e)
    }
  }

  def saveRawTweetsToMongoDB(rdd: RDD[Status]): Unit = {
    try{
      val sqlContext = spark_SparkSession.sqlContext
      val tweet = rdd.map(status => jacksonObjectMapper.writeValueAsString(status))

      val rawTweetsDF = sqlContext.read.json(tweet)

      val readConfig: ReadConfig = ReadConfig(Map("uri" -> "mongodb://127.0.0.1:27017/forensicdb.RAWTweets?readPreference=primaryPreferred"))
      val writeConfig: WriteConfig = WriteConfig(Map("uri" -> "mongodb://127.0.0.1:27017/forensicdb.RAWTweets"))
      MongoSpark.save(rawTweetsDF.coalesce(1).write.format("org.apache.spark.sql.json").option("forensicdb", "RAWTweets").mode("append"), writeConfig)
    }
    catch {
      case e: Exception => println("Error Saving tweets to Mongodb:", e)
    }
  }
}
```

Figure 40: Mongodb Save Function

The following figures shows a sample tweets which has been labeled and stored within the MongoDB together with SHA-256 hash key for full tweet and also SHA-256 hash key for the tweet status update (text).

```

1 db.getCollection('LiveclassifiedTweets').find({"category": "Ethnicity"})
LiveclassifiedTweets 0.018 sec.
1 /* 1 */
2 {
3   "_id" : ObjectId("59be806dbcc9834a88131c7d"),
4   "tweet_id" : NumberLong(909417241334226944),
5   "Name" : "Kibet0",
6   "ScreenName" : "LampardMutai",
7   "originalProfileImageURL" : "http://pbs.twimg.com/profile_images/878577413030019073/5DwL3xE6.jpg",
8   "source" : "<a href='\"http://twitter.com/download/android\"' rel='\"nofollow\"'>Twitter for Android</a>",
9   "useraccount_id" : NumberLong(936639992),
10  "location" : "Koinin Elburgon",
11  "geoLocation" : "0.0, 0.0",
12  "Hashtag" : "",
13  "Accountcreationdate" : "09-11-2012",
14  "Accountcreationtime" : "02:06:18 PM",
15  "originaltext" : "But i hear Kalenjins warn Kikuyus of unspecified consequences if they will not vote for you in 2022 https://t.co/17vx4f9KZ",
16  "tweetCreationdate" : "17-09-2017",
17  "tweetCreationtime" : "05:02:21 PM",
18  "text" : "but i hear kalenjins warn kikuyus of unspecified consequences if they will not vote for you in",
19  "prediction" : 0.0,
20  "cDate" : "17-09-2017",
21  "cTime" : "05:02:20 PM",
22  "userStatusUpdate" : "StatusJSONImpl(createdAt=Sun Sep 17 17:02:21 EAT 2017, id=909417241334226944, text='But i hear Kalenjins warn Kikuyus of unspecified cons",
23  "originaltextHashkey" : "78a506a904dd2d00ac5753a1177d52da40e9a0532dc4b223ad97655572133016",
24  "usertweetmd5hash" : "93c2ed85f514016dd9a2079d0eb30e9883f09db2c02a8804097444adff912708",
25  "category" : "Ethnicity"
26 }
27
28 /* 2 */
29 {
30   "_id" : ObjectId("59be81d2bcc9834a88131d46"),
31   "tweet_id" : NumberLong(909418735357890566),
32   "Name" : "Josekid Krystal Vybz",
33   "ScreenName" : "VybzPalmer",
34   "originalProfileImageURL" : "http://pbs.twimg.com/profile_images/901497149292240896/BdzqIns9.jpg",
35   "source" : "<a href='\"http://twitter.com/download/android\"' rel='\"nofollow\"'>Twitter for Android</a>",
36   "useraccount_id" : NumberLong(823627958),
37   "location" : "Nairobi, Kenya",
38   "geoLocation" : "0.0, 0.0",
39   "Hashtag" : "",
40   "Accountcreationdate" : "14-09-2012",
41   "Accountcreationtime" : "08:03:33 PM",
42   "originaltext" : "RT @harrisonmumia: The Ngunjiri petition against Maraga has definitely deepened the feeling amongst other tribes that Kikuyus are selfish/...",
43   "tweetCreationdate" : "17-09-2017",
44   "tweetCreationtime" : "05:08:17 PM",
45   "text" : "the ngunjiri petition against maraga has definitely deepened the feeling amongst other tribes that kikuyus are",
46   "prediction" : 0.0,
47   "cDate" : "17-09-2017",
48   "cTime" : "05:08:17 PM",
49   "userStatusUpdate" : "StatusJSONImpl(createdAt=Sun Sep 17 17:08:17 EAT 2017, id=909418735357890566, text='RT @harrisonmumia: The Ngunjiri petition agaInst Mara",
50   "originaltextHashkey" : "52dd948d222b9636fab3d0d83fcff54e1f7b0e69ace36a8a0dba18022e25c303",
51   "usertweetmd5hash" : "ecdc7140847ef4a04d93f2ff8cb269ad5c3887b763c63c3389dc8007a2ffc85e",
52   "category" : "Ethnicity"
53 }
54

```

Figure 41: Mongoddb Saved Tweet JSON Document

The following figure shows a sample of a full user tweet together with the Twitter unique account Id, tweet unique id, date and time of tweet capture and the SHA-256 hash key which is used for evidence preservation and authenticity.

```

{
  "_id" : NumberLong(896812261679005697),
  "twitter_account_id" : NumberLong(893358895875452928),
  "cTime" : "2017-08-13T15:13:49.529-04:00",
  "usertweets" : "StatusJSONImpl{createdAt=Sun Aug 13 15:14:39 EDT 2017,
id=896812261679005697, text='Big Fat Slut Lets Her Man Open And Explore Her Pink Hole
https://t.co/kD3vkRIebM', source='<a href=\"http://twitter.com\" rel=\"nofollow\">Twitter Web
Client</a>', isTruncated=false, inReplyToStatusId=-1, inReplyToUserId=-1, isFavorited=false,
isRetweeted=false, favoriteCount=0, inReplyToScreenName='null', geoLocation=null, place=null,
retweetCount=0, isPossiblySensitive=false, lang='en', contributorsIDs=[], retweetedStatus=null,
userMentionEntities=[], urlEntities=[URLEntityJSONImpl{url='https://t.co/kD3vkRIebM',
expandedURL='http://bit.ly/2wFIA1n', displayURL='bit.ly/2wFIA1n'}], hashtagEntities=[],
mediaEntities=[], symbolEntities=[], currentUserRetweetId=-1,
user=UserJSONImpl{id=893358895875452928, name='Athena Ellerson', email='null',
screenName='GuboninGleb', location='null', description='null', isContributorsEnabled=false,
profileImageUrl='http://pbs.twimg.com/profile_images/894557199254638594/ncwnoqH3_normal.jpg',
profileImageUrlHttps='https://pbs.twimg.com/profile_images/894557199254638594/ncwnoqH3_normal.j
pg', isDefaultProfileImage=false, url='null', isProtected=false, followersCount=0, status=null,
profileBackgroundColor='F5F8FA', profileTextColor='333333', profileLinkColor='1DA1F2',
profileSidebarFillColor='DDEEF6', profileSidebarBorderColor='C0DEED',
profileUseBackgroundImage=true, isDefaultProfile=true, showAllInlineMedia=false, friendsCount=0,
createdAt=Fri Aug 04 02:32:13 EDT 2017, favouritesCount=0, utcOffset=-1, timeZone='null',
profileBackgroundImageUrl="", profileBackgroundImageUrlHttps="", profileBackgroundTiled=false,
lang='ru', statusesCount=256, isGeoEnabled=false, isVerified=false, translator=false, listedCount=0,
isFollowRequestSent=false, withheldInCountries=null}, withHeldInCountries=null, quotedStatusId=-1,
quotedStatus=null}",
  "tweethashkey" : "567722fd56cbe27dd830ceef29f0907f9ce22e706da20f44b1ad249f5803b737"
}

```

Table 3: Streamed Twitter JSON Sample Data

4.11 Evidence Preservation

Spark Framework provides Streaming API which divides data in stream of batches in every predefined time interval normally in seconds called Discretized Stream (DStream). In Spark, this are sequence of data which represents RDDs. In this study, we utilized stream interval of five seconds and in every batch, an SHA-256 hash key for the tweet update status was calculated and each tweet post (text) hash key was also generated. The Spark forensic application processes the received RDDs using Spark APIs, and the processed results of the RDD operations are returned in batches. Figure 42 and figure 43 below shows stream batches arriving in time interval.

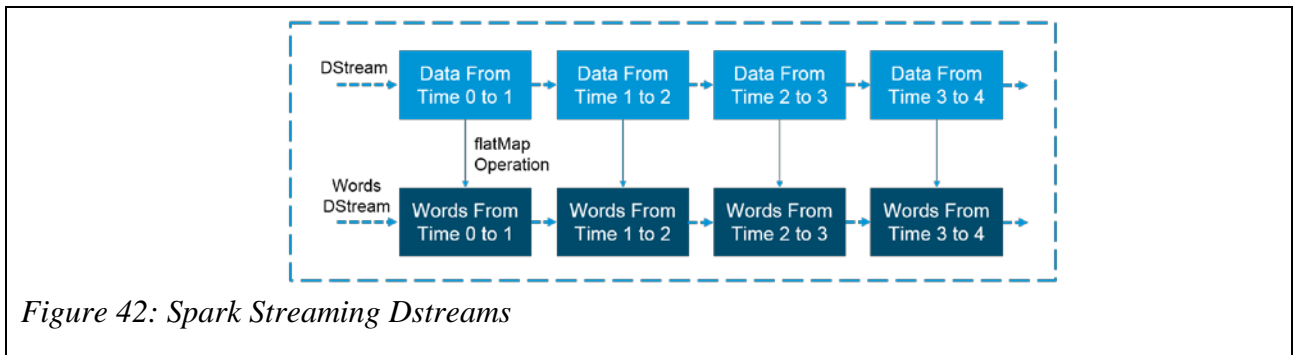


Figure 42: Spark Streaming Dstreams

Discretized Stream (DStream) forms the basic abstraction provided by Spark Streaming and represents a continuous stream of data which is received from a data source or a processed data stream generated by transforming the input stream.

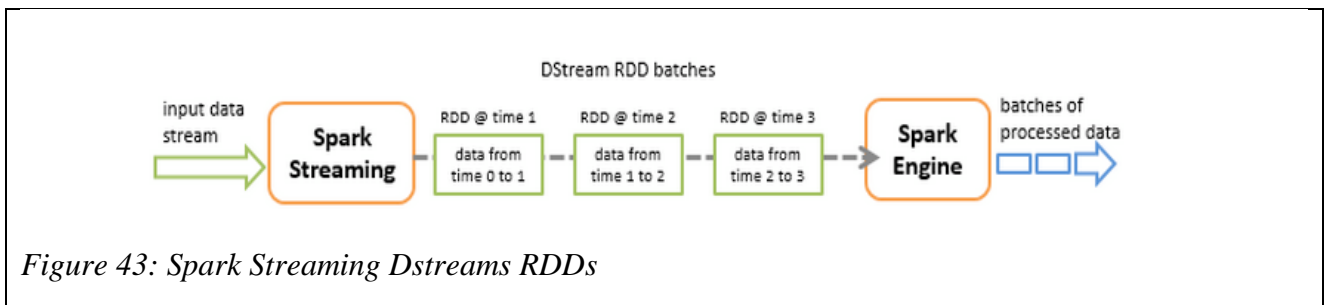


Figure 43: Spark Streaming Dstreams RDDs

To enable repeatability and reproducibility of the captured data and evidence preservation, the system utilized spark streaming Dstreams (RDD) which are generated in batches at time interval as indicated in figure 42 and figure 43 above. The system generates SHA-256 hash key for each batch and tweet item before it is stored to MongoDB database. This ensures repeatability and reproducibility whereby data gathered can be used to reproduce the same results when using the same method on identical test algorithms or different algorithm on different labs and by different forensic analyst. This can also ensure evidence authentication before court of law to proof that captured data haven't modified after

capture. Below Scala code was used to generate SHA-256 Hash key for both tweet text and tweet batches as they are received through spark streaming API.

```
//Generate MD5 Hash key for each tweet text
val forensicsmd5hashkey = SentUtilities.generateMD5Hashing(getTweetTextcleaner(status.getText))
//Generate MD5 Hash key for each batch of tweets received
val usertweetmd5hash = SentUtilities.generateMD5Hashing(status.toString)

object forensicsHashGeneratorUtils {
  def generateMD5(message: String): String = hashString(message, "MD5")

  def generateSHA1(message: String): String = hashString(message, "SHA-1")

  def generateSHA256(message: String): String = hashString(message, "SHA-256")

  private def hashString(message: String, algorithm: String): String = {
    val digest: MessageDigest = MessageDigest.getInstance(algorithm)
    val hashedBytes: Array[Byte] = digest.digest(message.getBytes("UTF-8"))
    convertByteArrayToHexString(hashedBytes)
  }

  private def convertByteArrayToHexString(arrayBytes: Array[Byte]): String = {
    val stringBuffer: StringBuffer = new StringBuffer()
    for (i <- 0 until arrayBytes.length) {
      stringBuffer.append(
        java.lang.Integer
          .toString((arrayBytes(i) & 0xff) + 0x100, 16)
          .substring(1))
    }
    stringBuffer.toString
  }
}
```

Figure 44: SHA-256 Hash Key Generator

The generated SHA-256 hash key is stored in MongoDB with the corresponding tweet as shown in figure below.

```

MongoConnection localhost:27017 forensicdb
1 db.getCollection('LiveclassifiedTweets').find({"category": "Ethnicity"})
LiveclassifiedTweets 0.019 sec.
1 /* 1 */
2 {
3   "_id": ObjectId("59be806dbcc9834a88131c7d"),
4   "tweet_id": NumberLong(909417241334226944),
5   "Name": "Kibet@",
6   "ScreenName": "LampardMutai",
7   "originalProfileImageUrl": "http://pbs.twimg.com/profile_images/878577413030019073/50wL3xE6.jpg",
8   "source": "<a href='\"http://twitter.com/download/android\"' rel='\"nofollow\">Twitter for Android</a>",
9   "useraccount_id": NumberLong(936639992),
10  "location": "Kono in Elburgon",
11  "geoLocation": "0.0, 0.0",
12  "Hashtag": "",
13  "Accountcreationdate": "09-11-2012",
14  "Accountcreationtime": "02:06:18 PM",
15  "originaltext": "But i hear Kalenjins warn Kikuyus of unspecified consequences if they will not vote for you in 2022 https://t.co/17vxf4f0kZ",
16  "tweetCreationdate": "17-09-2017",
17  "tweetCreationtime": "05:02:21 PM",
18  "text": "but i hear kalenjins warn kikuyus of unspecified consequences if they will not vote for you in",
19  "prediction": 0.0,
20  "cDate": "17-09-2017",
21  "cTime": "05:02:20 PM",
22  "userStatusUpdate": "StatusJSONImpl{createdAt=Sun Sep 17 17:02:21 EAT 2017, id=909417241334226944, text='But i hear Kalenjins warn Kikuyus of unspecifi",
23  "originaltextHashkey": "78a506a904dd2d00ac5753a1177d52da40e9a06532dc4b223ad9765572133016",
24  "usertweetmd5hash": "03c2ed85f514016dd9a2079d0eb3ee9883f09db2c02a8804097444adf912708",
25  "category": "Ethnicity"
26 }
27

```

Figure 45: Tweet SHA-256 Hash keys JSON Document

4.11.1 SHA-256 Hash Key Verification

To ensure that the Hash key generated during the tweet streaming and saved in MongoDB is valid and can be used to verify the preservation of the streamed evidence, detect any changes on the streamed data, we used third party online website MD5 & SHA1 Hash Generator <http://onlinemd5.com/>. We copied sample tweet text from MongoDB and generated SHA-256 Hash Key on the <http://onlinemd5.com/> web site which we compared with already generated hash key for that tweet stored in MongoDB. The result indicated that the two generated Hash key matched and were the same. Table 4 below shows our comparison.

Tweet Sample Stored In MongoDB:

But i hear Kalenjins warn Kikuyus of unspecified consequences if they will not vote for you in 2022 <https://t.co/i7vxf4f0kZ>

A MD5 & SHA1 Hash Generator For Text

Generate the hash of the string you input.

But i hear Kalenjins warn Kikuyus of unspecified consequences if they will not vote for you in 2022 <https://t.co/i7vxf4f0kZ>

Checksum type: MD5 SHA1 SHA-256

String hash:

Mongodb Stored Tweet SHA-256 Hash Key

78a506a904dd2d00ac5753a1177d52da40e9a0532dc4b223ad97655572133016

Sample Raw JSON Tweet Stored In MongoDB

```
StatusJSONImpl{createdAt=Sun Sep 17 16:05:24 EAT 2017, id=909402911666577408, text='Independent of the international election observers is in doubt for legitimizing recent electoral fraud in Kenya.', source='<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>', isTruncated=false, inReplyToStatusId=-1, inReplyToUserId=-1, isFavorited=false, isRetweeted=false, favoriteCount=0, inReplyToScreenName='null', geoLocation=null, place=null, retweetCount=0, isPossiblySensitive=false, lang='en', contributorsIDs=[], retweetedStatus=null, userMentionEntities=[], urlEntities=[], hashtagEntities=[], mediaEntities=[], symbolEntities=[], currentUserRetweetId=-1, user=UserJSONImpl{id=1894715149, name='Samwel Orwa', email='null', screenName='OrwaSamwel', location='Switzerland', description='null', isContributorsEnabled=false, profileImageUrl='http://pbs.twimg.com/profile_images/695151844729950209/TV_jmsn2_normal.jpg', profileImageUrlHttps='https://pbs.twimg.com/profile_images/695151844729950209/TV_jmsn2_normal.jpg', isDefaultProfileImage=false, url='null', isProtected=false, followersCount=23, status=null, profileBackgroundColor='C0DEED', profileTextColor='333333', profileLinkColor='1DA1F2', profileSidebarFillColor='DDEEF6', profileSidebarBorderColor='C0DEED', profileUseBackgroundImage=true, isDefaultProfile=true, showAllInlineMedia=false, friendsCount=68, createdAt=Sun Sep 22 21:39:42 EAT 2013, favouritesCount=2, utcOffset=-25200, timeZone='Pacific Time (US & Canada)', profileBackgroundImageUrl='http://abs.twimg.com/images/themes/theme1/bg.png', profileBackgroundImageUrlHttps='https://abs.twimg.com/images/themes/theme1/bg.png', profileBackgroundTiled=false, lang='en', statusesCount=17, isGeoEnabled=false, isVerified=false, translator=false, listedCount=1, isFollowRequestSent=false, withheldInCountries=null}, withHeldInCountries=null, quotedStatusId=-1, quotedStatus=null}
```

MD5 & SHA1 Hash Generator For Text

Generate the hash of the string you input.

```
StatusJSONImpl{createdAt=Sun Sep 17 16:05:24 EAT 2017, id=909402911666577408, text='Independent of the international election observers is in doubt for legitimizing recent electoral fraud in Kenya.', source='<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>', isTruncated=false, inReplyToStatusId=-1, inReplyToUserId=-1, isFavorited=false, isRetweeted=false, favoriteCount=0, inReplyToScreenName='null', geoLocation=null, place=null, retweetCount=0, isPossiblySensitive=false, lang='en', contributorsIDs=[], retweetedStatus=null, userMentionEntities=[], urlEntities=[], hashtagEntities=[], mediaEntities=[], symbolEntities=[], currentUserRetweetId=-1, user=UserJSONImpl{id=1894715149, name='Samwel Orwa', email='null', screenName='OrwaSamwel', location='Switzerland', description='null', isContributorsEnabled=false, profileImageUrl='http://pbs.twimg.com/profile_images/695151844729950209/TV_jmsn2_normal.jpg', profileImageUrlHttps='https://pbs.twimg.com/profile_images/695151844729950209/TV_jmsn2_normal.jpg', isDefaultProfileImage=false, url='null', isProtected=false, followersCount=23, status=null, profileBackgroundColor='C0DEED', profileTextColor='333333', profileLinkColor='1DA1F2', profileSidebarFillColor='DDEEF6', profileSidebarBorderColor='C0DEED', profileUseBackgroundImage=true, isDefaultProfile=true, showAllInlineMedia=false, friendsCount=68, createdAt=Sun Sep 22 21:39:42 EAT 2013, favouritesCount=2, utcOffset=-25200, timeZone='Pacific Time (US & Canada)', profileBackgroundImageUrl='http://abs.twimg.com/images/themes/theme1/bg.png', profileBackgroundImageUrlHttps='https://abs.twimg.com/images/themes/theme1/bg.png', profileBackgroundTiled=false, lang='en', statusesCount=17, isGeoEnabled=false, isVerified=false, translator=false, listedCount=1, isFollowRequestSent=false, withheldInCountries=null}, withHeldInCountries=null, quotedStatusId=-1, quotedStatus=null}
```

Checksum type: MD5 SHA1 SHA-256

String hash:

21678920068FE0AFEC8B92F1EDA21873751698DA42B6C07A31C9352D1ED43C12

Calculate

Mongodb Stored Tweet SHA-256 Hash Key

21678920068fe0afec8b92f1eda21873751698da42b6c07a31c9352d1ed43c12

Table 4: Tweet SHA-256 Verification

4.12 Model Design and Classification

We collected 3,138,367 tweets (24.2GB) which were used for training the Naïve Bayes classifier. The tweets were used to train Naïve Bayes classifier model which was saved on the Spark cluster and later used for streaming live tweets and classify them for hate speech and cyberbullying. To design the model, we utilized Spark ML API (spark.ml) which provides ML pipelines (workflow) for creating, tuning, and evaluating of machine learning model. In Spark ML, a pipeline is defined as a sequence of stages, and each stage is either a Transformer or an Estimator. These stages are run in order, and the input DataFrame is transformed as it passes through each stage. Below figure shows the Spark ML Pipeline stages we adopted for the model design.

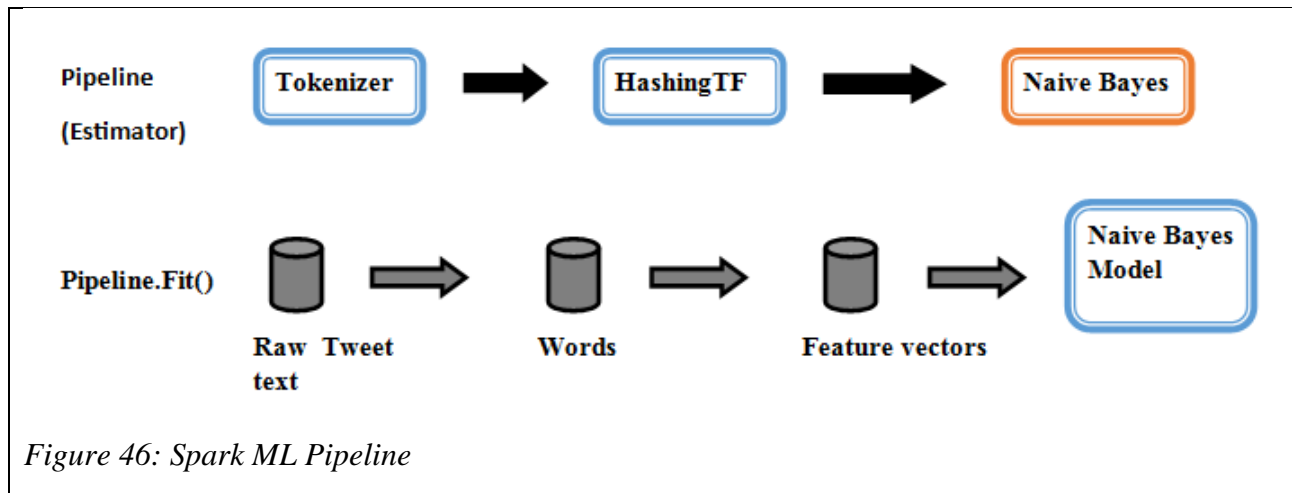


Figure 46: Spark ML Pipeline

For this study, training raw tweets were read from local disk, cleaned by passing through Scala function which removed unnecessary characters. The cleaned tweets were ingested into Spark ML. Tokenizer were the tweet text were broken down into their constituent words. The tokenized tweets were again passed through Spark ML StopWordsRemover with a dictionary of stop words. This removed commonly appearing words which does not contribute to the structure of the tweets. We split the tweet data into two datasets, 70% (2,197,498, tweets) as training dataset and 30% (940,869, tweets) as testing dataset. The training dataset was used to train Naïve Bayes model, and test dataset was used to evaluate the model accuracy.

For accuracy of the model, we used cross validation using Spark evaluation tool namely MulticlassClassificationEvaluator within the spark.ml.evaluation.MulticlassClassificationEvaluator and apache.spark.ml.tuning.{CrossValidator, ParamGridBuilder} packages. After the model was trained, evaluated and tested with training dataset, the model was saved on local disk within the Apache Spark Cluster. The model was later reloaded for live tweet streaming and tweet hate speech classification and categorization. The following figure show the Spark ML pipeline as used in this study.

```
// register as cleanedTweetTable
labeledtweetsDF.createOrReplaceTempView("cleanedTweetTable")

// Split data into training (60%) and test (40%).
val splits = labeledtweetsDF.randomSplit(Array(0.6, 0.4), seed = 12345L)
val trainingdata = splits(0).cache()
val testingdata = splits(1)

// Naive Bayes Approach
println("\nTotal tweet Count = " + labeledtweetsDF.count() + "\n")
println("\nTraining tweet Count = " + trainingdata.count() + ", " + trainingdata.count * 100 / (labeledtweetsDF.count().toDouble + "%") + "\n")
println("\nTest tweet Count = " + testingdata.count() + ", " + testingdata.count * 100 / (labeledtweetsDF.count().toDouble + "%") + "\n")

// Configure an ML pipeline, which consists of SIX stages: tokenizer, stopwordsremover, hashingTF, IDF normalizer and Naive Bayes.
val doctokenizer = new Tokenizer()
  .setInputCol("text")
  .setOutputCol("words")

val stopwords = Array[String] = SentUtilities.getstopwords(loadconfigProperties.nlTKStopWords).toArray
val stopwordsremover = new StopWordsRemover()
  .setStopWords(stopwords)
  .setInputCol(doctokenizer.getOutputCol)
  .setOutputCol("filtered")
  .setCaseSensitive(false)

val hashingTF = new HashingTF()
  .setNumFeatures(20000)
  .setInputCol(doctokenizer.getOutputCol)
  .setOutputCol("rawFeatures")

val idf = new IDF()
  .setInputCol(hashingTF.getOutputCol)
  .setOutputCol("features")
  .setMinDocFreq(0)

// A normalizer is a common operation for text classification.
// It simply gets all of the data on the same scale... for example, if one article is much longer and another, it'll normalize the scales for the different features.
// If we don't normalize, an article with more words would be weighted differently
val normalizer = new Normalizer()
  .setInputCol(idf.getOutputCol)
  .setOutputCol("#features")

val NaiveBayesModel = new NaiveBayes()

// Define ML Pipeline with the Six stages defined above
val pipeline = new Pipeline().setStages(Array(doctokenizer, stopwordsremover, hashingTF, idf, normalizer, NaiveBayesModel))
// Fit the pipeline to training documents.
val model = pipeline.fit(trainingdata)

// Select example rows to display.
val predictions = model.transform(testingdata)
// Now we can optionally save the fitted pipeline to disk
model.write.overwrite().save("/home/smulwa/data/naiveBayesModel")
// Print the model prediction Variables
val predictionDF: DataFrame = predictions.select("tweet_id", "name", "screenName", "originalProfileImageURL", "source", "useraccount_id", "location", "AccountcreatedAt",
  "originaltweets", "text", "tweetCreatedAt", "label", "prediction").toDF()
```

Figure 47: Spark ML pipeline Naive Bayes Classifier

4.13 Model Deployment

To submit the spark forensic model to Spark, we needed to compile, package it into jar file and submit it to Spark. Since our forensic tool depended on other several libraries to run, we had to package our code with these dependencies into one package (jar) that can be submitted to Spark cluster. To do this we used Scala dependency manager called SBT, which we installed to our cluster. With SBT installed, we added our code dependencies with SBT build.sbt file as shown in figure below.

```
name := "SocialNetworkForensics"

version := "1.0"

scalaVersion := "2.11.8"

// Repositories
resolvers += "Maven Central" at "https://repol.maven.org/maven2/"

// https://mvnrepository.com/artifact/org.scala-lang/scala-library
libraryDependencies += "org.scala-lang" % "scala-library" % "2.11.8"

// https://mvnrepository.com/artifact/org.apache.spark/spark-core_2.11
libraryDependencies += "org.apache.spark" % "spark-core_2.11" % "2.1.0"

// https://mvnrepository.com/artifact/org.apache.spark/spark-streaming_2.11
libraryDependencies += "org.apache.spark" % "spark-streaming_2.11" % "2.1.0"

// https://mvnrepository.com/artifact/org.apache.spark/spark-sql_2.11
libraryDependencies += "org.apache.spark" % "spark-sql_2.11" % "2.1.0"

// https://mvnrepository.com/artifact/org.apache.spark/spark-mllib_2.11
libraryDependencies += "org.apache.spark" % "spark-mllib_2.11" % "2.1.0"

// https://mvnrepository.com/artifact/org.apache.bahir/spark-streaming-twitter_2.11
libraryDependencies += "org.apache.bahir" % "spark-streaming-twitter_2.11" % "2.1.0"

// https://mvnrepository.com/artifact/com.typesafe/config
libraryDependencies += "com.typesafe" % "config" % "1.3.1"

// https://mvnrepository.com/artifact/log4j/log4j
libraryDependencies += "log4j" % "log4j" % "1.2.17"

// https://mvnrepository.com/artifact/org.twitter4j/twitter4j-core
libraryDependencies += "org.twitter4j" % "twitter4j-core" % "4.0.6"

// https://mvnrepository.com/artifact/org.twitter4j/twitter4j-stream
libraryDependencies += "org.twitter4j" % "twitter4j-stream" % "4.0.6"
//#####
// https://mvnrepository.com/artifact/org.mongodb/mongodb-driver
//libraryDependencies += "org.mongodb" % "mongodb-driver" % "3.4.2"
```

Figure 48: SBT build.sbt

Once we installed SBT, we packaged our project and dependencies into a single jar using the command package as shown in figure below.

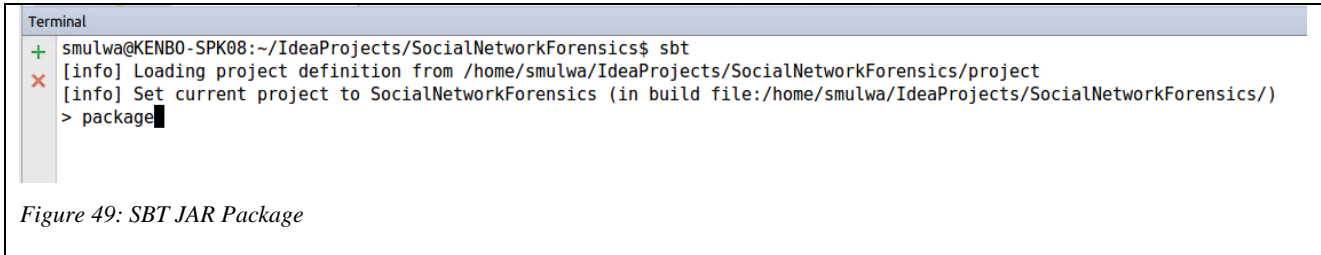


Figure 49: SBT JAR Package

This will create a single jar file inside the target folder which we copied to the jar folder within Spark home folder in our Spark Standalone cluster. Once we copied the spark forensic jar to the cluster, we used spark-submit command which is bundled with spark to deploy the jar to Spark cluster. The following code shows the command we used to submit our forensic tool job to our cluster.

```
smulwa@KENBO-SPK08:~$ spark-submit \
    --class TwitterforensicAnalysis\
    --master spark://KENBO-SPK08.forensics.net:7077 \
    --executor-memory 6G \
    --total-executor-cores 2 \
    /usr/local/spark/jars/socialnetworkforensics_2.11-1.0.jar
```

Once the job has been submitted to the Spark cluster, it can be monitored through the spark GUI on the browser. The following figure shows Spark GUI showing currently executing job.

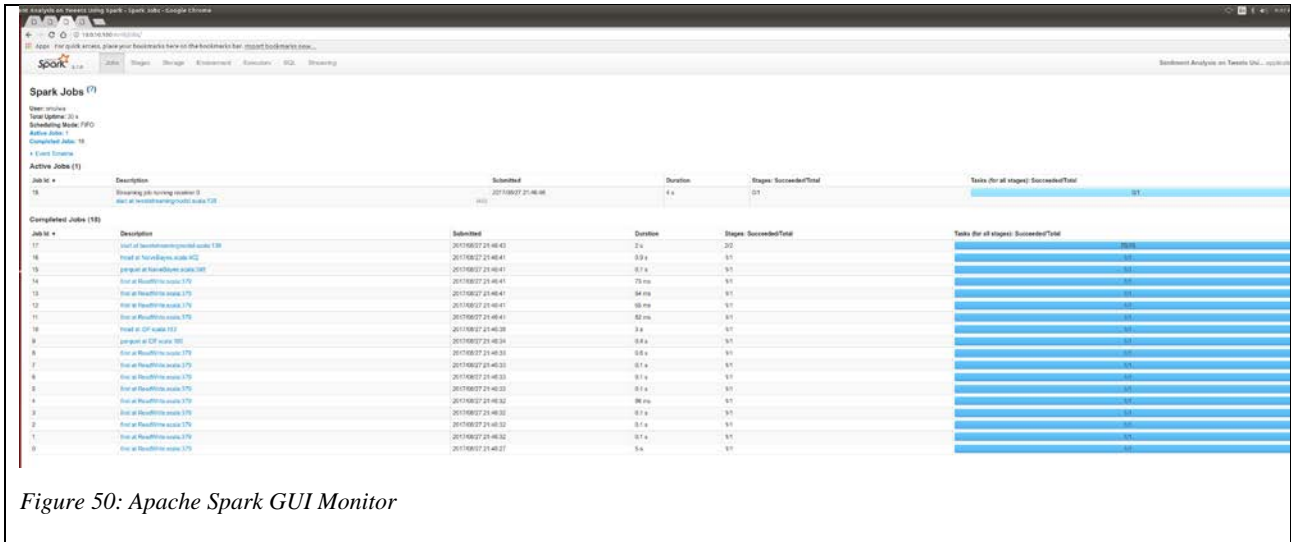


Figure 50: Apache Spark GUI Monitor

4.14 Model Evaluation

The forensic model needed to be evaluated for performance and correctness with test dataset before it is deployed for use on live data stream. To evaluate the performance of the forensic model in terms of quality or predictive effectiveness, different metrics were used. F-measure (F1-score) is a statistical measure of model's test accuracy that is the weighted harmonic mean of precision and recall of the test where recall is the fraction of all samples classified correctly as positive by the model and Precision describes the ratio of all positives samples classified as true positives by the model. For evaluating our model, we used spark.ml.evaluation packages which provides a suite of metrics that are suitable for evaluating the performance of spark data mining models.

For this study, F-measure which is provided in spark.ml.evaluation was used to evaluate the model performance. F-measure was chosen because it includes metrics like precision and recall that are used in order to take into account of errors that might occur if dataset is highly unbalanced.

Accuracy	$ACC = \frac{TP}{TP+FP} = \frac{1}{N} \sum_{i=0}^{N-1} \hat{\delta}(\hat{\mathbf{y}}_i - \mathbf{y}_i)$
----------	---

Precision by label	$PPV(\ell) = \frac{TP}{TP+FP} = \frac{\sum_{i=0}^{N-1} \hat{\delta}(\hat{\mathbf{y}}_i - \ell) \cdot \hat{\delta}(\mathbf{y}_i - \ell)}{\sum_{i=0}^{N-1} \hat{\delta}(\hat{\mathbf{y}}_i - \ell)}$
--------------------	--

Recall by label	$TPR(\ell) = \frac{TP}{P} = \frac{\sum_{i=0}^{N-1} \hat{\delta}(\hat{\mathbf{y}}_i - \ell) \cdot \hat{\delta}(\mathbf{y}_i - \ell)}{\sum_{i=0}^{N-1} \hat{\delta}(\mathbf{y}_i - \ell)}$
-----------------	--

F-measure by label	$F(\beta, \ell) = (1 + \beta^2) \cdot \left(\frac{PPV(\ell) \cdot TPR(\ell)}{\beta^2 \cdot PPV(\ell) + TPR(\ell)} \right)$
--------------------	---

Weighted precision	$PPV_w = \frac{1}{N} \sum_{\ell \in L} PPV(\ell) \cdot \sum_{i=0}^{N-1} \hat{\delta}(\mathbf{y}_i - \ell)$
--------------------	--

Weighted recall	$TPR_w = \frac{1}{N} \sum_{\ell \in L} TPR(\ell) \cdot \sum_{i=0}^{N-1} \hat{\delta}(\mathbf{y}_i - \ell)$
-----------------	--

Weighted F-measure	$F_w(\beta) = \frac{1}{N} \sum_{\ell \in L} F(\beta, \ell) \cdot \sum_{i=0}^{N-1} \hat{\delta}(\mathbf{y}_i - \ell)$
--------------------	--

The accuracy, precision by label, recall by label, and F-measure by label of the model was used to evaluate the performance of the model. Recall metric measures the overall classification correctness, which represents the percent of tweet posts that were correctly identified as hate speech. The false positive (FP) rate represents the percent of tweet posts that are not truly offensive but classified as offensive. The false negative (FN) rate represents the percent of tweets which are offensive but classified as positive tweets. Precision presents the percent of identified tweets that are truly offensive messages, and f-score represents the weighted harmonic mean of precision and recall.

For estimating the performance of classification model, we used Cross-validation (CV) which is a method for evaluating the performance of a model classifier for unseen data. Cross-validation (CV) works by randomly splitting the whole labeled data set K (K-folds) equal partitions. For each data partition, the classifier is trained on the remaining K-1 partitions and is tested on data from that partition and the final accuracy of the model is calculated as the average of all K accuracies. Below figure shows the cross validation codes which we used for validation.

```

// select (prediction), true label, and compute cost error
val modevaluator = new MulticlassClassificationEvaluator()
  .setLabelCol("label")
  .setPredictionCol("prediction")
  .setMetricName("weightedPrecision")

//Evaluate the model for accuracy
val modelaccuracy = modevaluator.evaluate(predictions)
println("Prediction Model Accuracy: " + modelaccuracy)
println(modevaluator.isLargerBetter)

// Tune hyperparameters
val paramGrid = new ParamGridBuilder().addGrid(hashingTF.numFeatures, Array(1000, 10000, 100000))
  .addGrid(idf.minDocFreq, Array(0, 10, 100))
  .build()
//*****
// Cross validation
val cv = new CrossValidator()
  .setEstimator(pipeline)
  .setEvaluator(modevaluator)
  .setEstimatorParamMaps(paramGrid).setNumFolds(10)

// cross-evaluate
val cvModel = cv.fit(trainingdata)
//Predict on test data (testdata) (DO NOT

```

Figure 51: Spark ML Model Cross Validation

The following table outlines the model performance as evaluated using spark.ml.evaluation library and upon cross validation against 10 folds (K-folds).

Multiclass Metrics	Fraction
Model Accuracy	0.7706885868277092
Weighted precision	0.7776074500404562
Weighted recall	0.7705571409937038
Weighted F1 score	0.770139251942318
Weighted false positive rate	0.11993472033775189

Table 5: Forensic Model Performance Metrics

This study involved Multilabel classification problem which involves mapping each sample in a dataset to a set of class labels. In this type of classification problem, the labels are not mutually exclusive hence the predictions and true labels are now vectors of label sets, rather than vectors of labels. Multilabel metrics, therefore, extend the fundamental ideas of precision, recall to operations on sets. The following table shows individual Multilabel metrics.

Multiclass Label	Measure
Class 0.0 precision	0.77933239509749
Class 1.0 precision	0.6818904322661206
Class 2.0 precision	0.82054164908874
Class 0.0 recall	0.8380440691745968
Class 1.0 recall	0.8213156784958556
Class 2.0 recall	0.6997099653163271
Class 0.0 F1-score	0.8076225987633927
Class 1.0 F1-score	0.7451370760495283
Class 2.0 F1-score	0.755323873211827

Table 6: Multiclass Label Metrics

The study also employed use of confusion matrix which is a matrix where rows represent actual classes and columns represent predicted classes to see the classifier effectiveness. The following table show confusion matrix which was generated using Spark ML API.

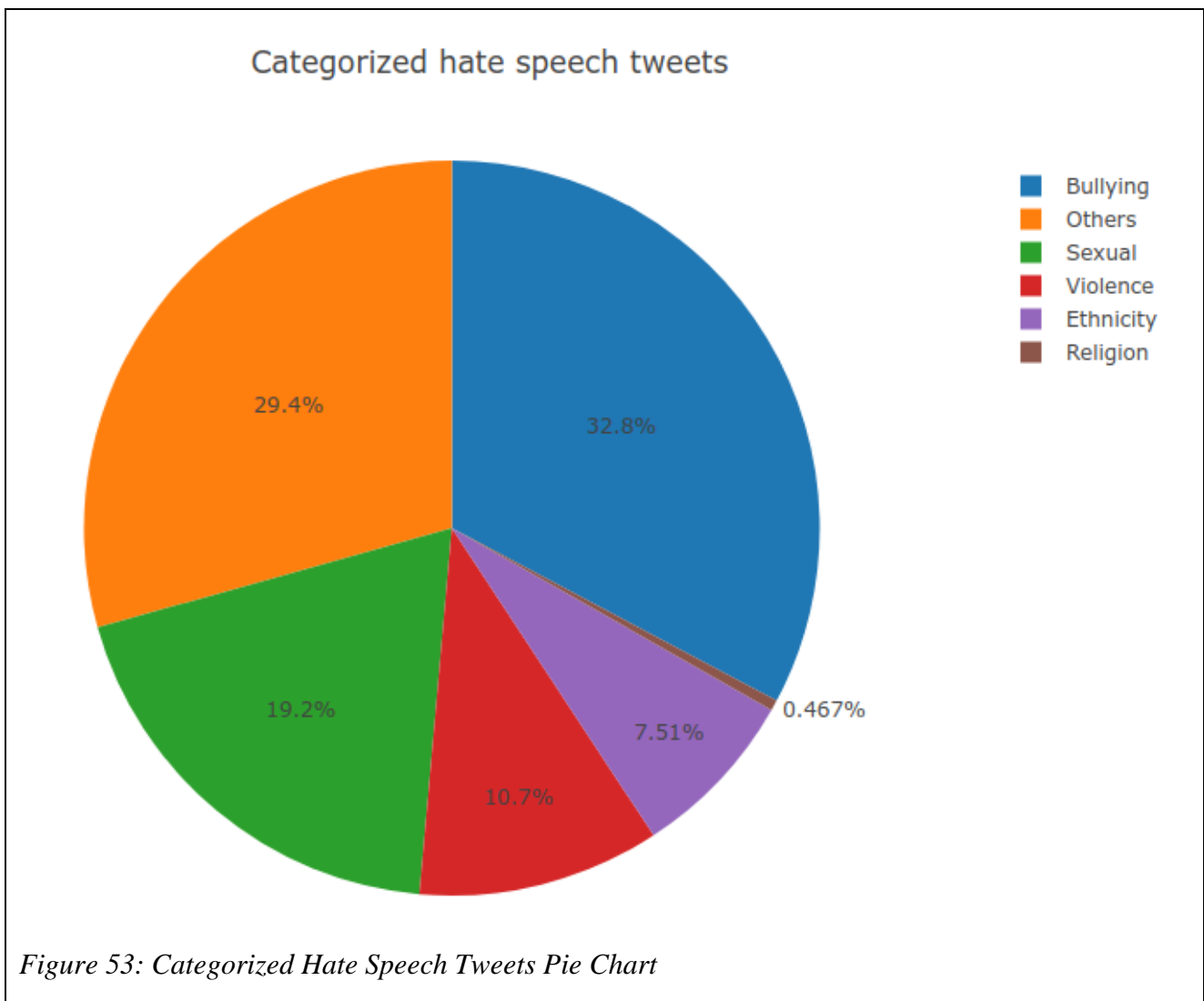
		Predicted		
		Hate Speech Sentiments	Positive Sentiments	Neutral Sentiments
Actual	Hate Speech Sentiments	256,050	11,511	38,328
	Positive Sentiments	71,75	164,693	28,334
	Neutral Sentiments	65,268	65,136	304,374

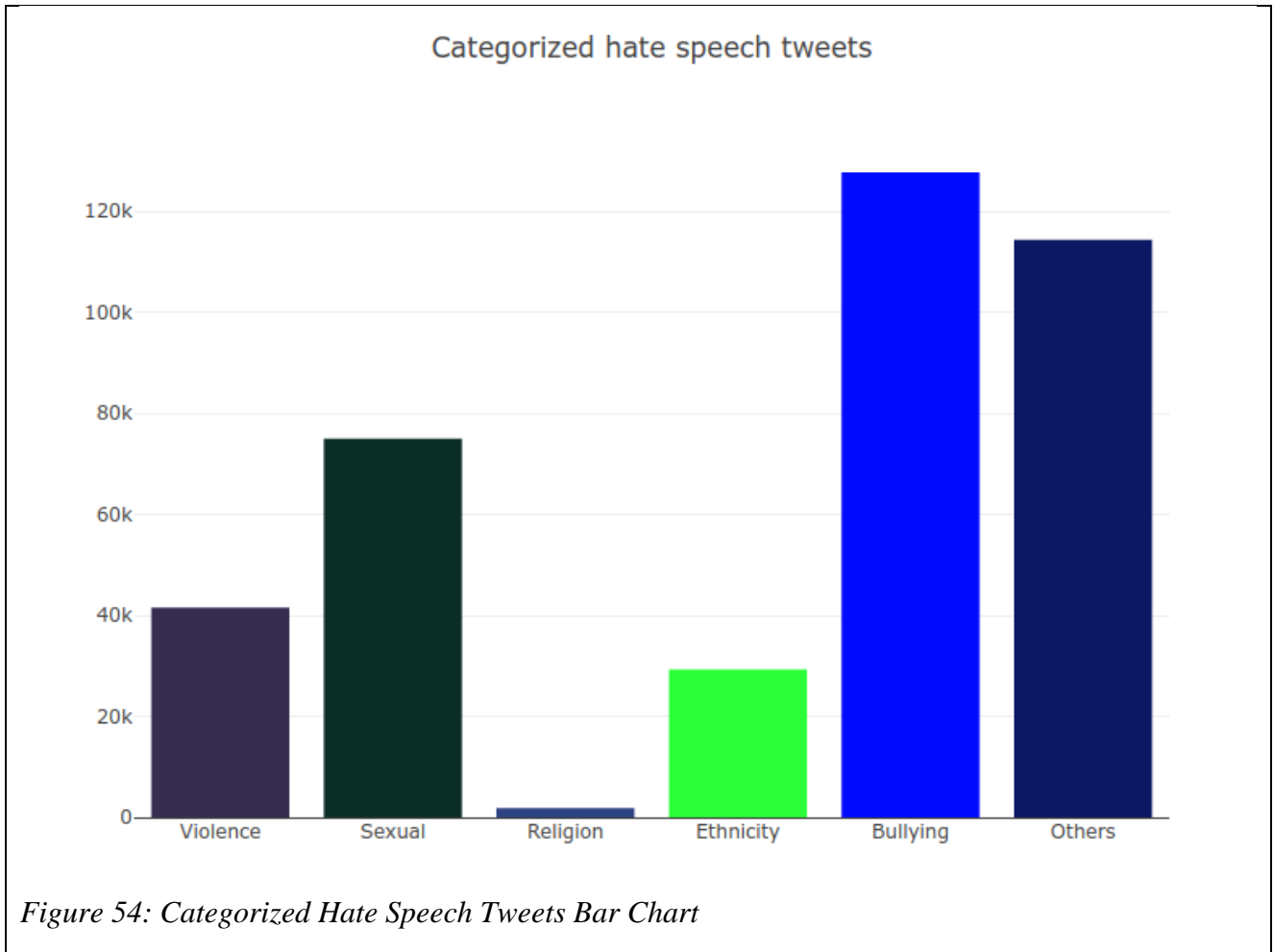
Figure 52: Model Confusion Matrix

This formed 940,869 (30%) of the testing tweets and as per the above table, 256,050 Tweets were correctly classified as containing words which are offending, insulting, intimidating, inflammatory or bullying in nature while 164,693 Tweets were correctly classified as Positive Sentiments and 304,374 Tweets were classified as of Neutral Sentiments. Out of 940,869 Testing Tweets, 215,752 Tweets were wrongly classified as either hate speech Sentiments, Positive sentiments or Neutral Sentiments which formed 23%.

4.15 Model Results and Analysis

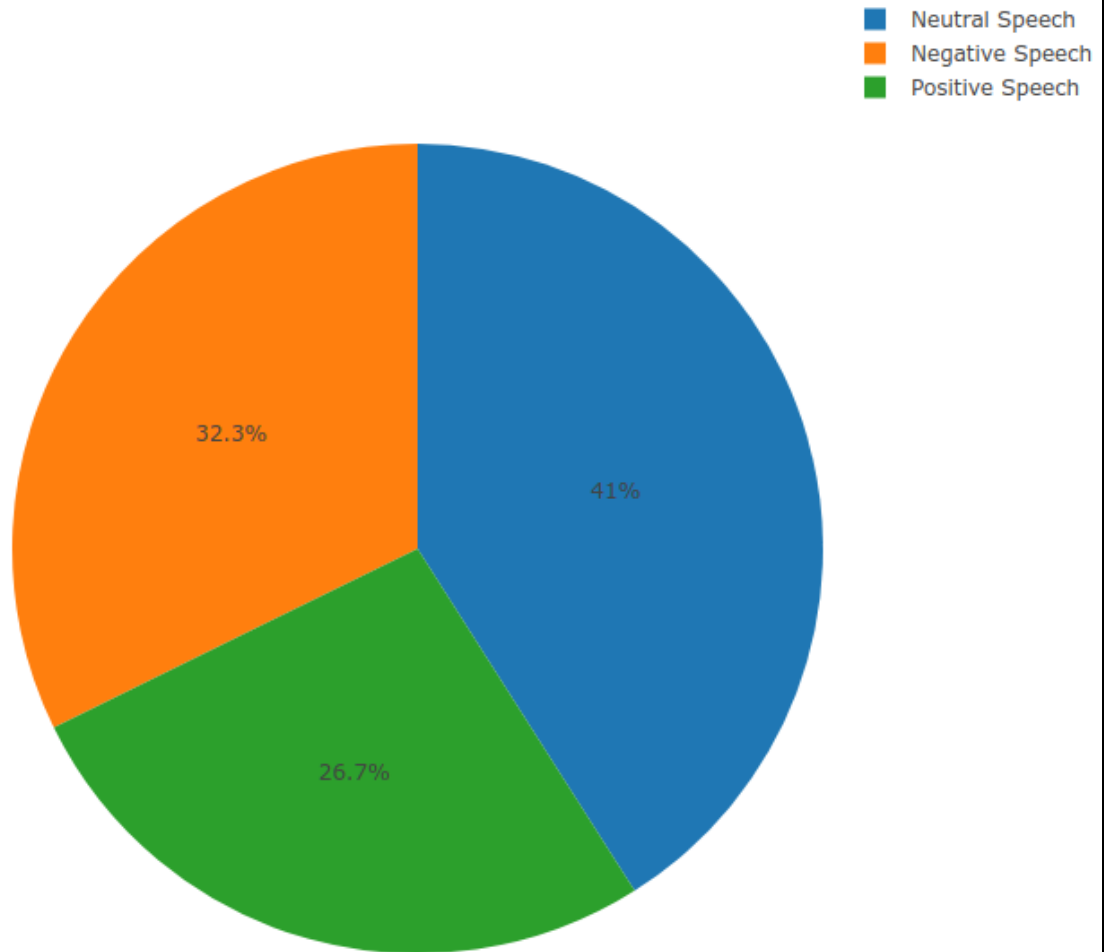
From the Model reports that we presented in bar charts and pie charts, it was evident that bullying and sexual related offensive language/hate speech was rampant within Twitter social network with 32.8% and 19.2% respectively. They were followed by violence related hate speech that formed 10.7%. Ethnicity and Religious related hate speech formed the lowest with ethnicity representing 7.51% and 0.467% respectively. At the same time, we had tweets that did not fall with the range of buying, sexual, ethnicity, religious or violence and they were categorized as others and formed 29.4% of the tweets streamed. The following two charts shows sample of the forensic report represented using pie chart and a bar chart.





The following pie chart shows the overall classification of the tweets streaming using the forensics models classified as either hate speech (Negative), Positive and Neutral tweet statement.

Tweet Sentimental Classification



© Copyright 2017 forensic.net

Figure 55: Tweet Sentiments Classification Pie Chart

Again, the chart shows that 32.3% of the tweets formed Negative statements, 41% tweets were Neutral and 26.7% tweets were Positive statements.

For reporting, we designed web based front end interface which allowed investigators to view classified tweets with account metadata, stream date and time which can be used to authenticate each tweet post. The following shows sample of hate speech classified tweets.

Twitter for Windows

Filter hate speech by category: Neutral/Positive Search

Tweet keyword search: Enter Search Keywords Search

Search tweets by stream date: MM/DD/YYYY Search

Tweets Classified as Hate Speech With Account Metadata

Tweet Entry No	Twitter Account Id	User Name	ScreenName	Account Creation Date	Account Creation Time	Account Creation Location	originalProfileImageURL	source	Tweet Id	Tweet Creation Date	Tweet Creation Time	Status Update	Category	label	prediction	Stream Date	Stream Time	Tweet Item SHA256 Hash Key	User Tweet SHA256 Hash Key
1	93863692	Kibet9	LampardMuta	09-11-2012	02:06:18 PM	Kenoin Elburgon		Twitter for Android	909417241334226944	17-09-2017	05:02:21 PM	But I hear Kalerjins warn Kikuyas of unspecified consequences if they will not vote for you in 2022 https://t.co/7xWf9Z	Ethnicity	Neutral speech	Hate speech	17-09-2017	05:02:20 PM	7ba506a904462d09ac5f53a1177652ba3b9a3532c6c223a878656572132016	89c2e489514016cda2079a0eb3ee98830c8c2c2a880405f744ad912708
2	823627958	Joselid Kyalal Yitz	VyhuPamer	14-09-2012	08:03:33 PM	Nairobi, Kenya		Twitter for Android	90941873635780566	17-09-2017	05:08:17 PM	RT @hanscommunia: The Ngunjiri petition against Mariga has definitely deepened the feeling amongst other tribes that Kikuyas are selfish...	Ethnicity	Neutral speech	Hate speech	17-09-2017	05:08:17 PM	52c0948422299638c6c3c63c3f54e179c6f89c9a8a0d9a16822e25c303	e0dc7140847e4d32098a269a5c3887f703c3c3398a20728f85e
3	184757330	AlroGamma	AlroGamma	30-09-2010	03:13:11 PM	Rio de Janeiro, Brazil		Twitter for iPhone	90942192345670416	17-09-2017	05:20:57 PM	RT @CPKItax: @Aamob_ @Kumano_ Jr My opinion. Luo and kikuyas will never fight. Fight all ways betn kikuyas and kalen. This c...	Ethnicity	Neutral speech	Hate speech	17-09-2017	05:20:58 PM	872254581284c0eef8a95c45f05b68aa0b0c13d8e5877e2c039049f85	76a564120f6c2c67c22a127b05aa2e512a049366548c93c0cc44f9c5
4	217257048	Voice Of Despora	FrankKiriko	02-11-2013	05:30:03 PM	Ongata Rongai		Twitter for Android	90942902157800257	17-09-2017	05:52:45 PM	RT @hansSANDKENNEDY: There will be no elections on oct 17 whether Kikuyas and githin media like it or not #NoReformNoElections #NoOct...	Ethnicity	Neutral speech	Hate speech	17-09-2017	05:52:48 PM	60f6a269982923da7a47c48e1428482a2f5944572a9f7568ba377	5c9de4e988f0c012828a3ad97aa8752ee9660c7f0da08fa020bead8099
5	53872366	NOM	n_muyale	28-03-2012	07:41:39 AM	Kenya		Feris 2 Preview	909431270706468816	17-09-2017	05:58:06 PM	@WilliamsRuto Don't count me in the 45 millions. Count the Kikuyas and Kalenjins affiliated to Jubilee leaders.	Ethnicity	Neutral speech	Hate speech	17-09-2017	05:58:05 PM	12bb42324787aeb6030772ea0cbb105f1488e4ab5a8d84e16c0778bc3	22864aa534cbed71b4efbc2b2720c4553bc23f1ee933bed4beb0465c6e7
6	171523823	JungleBook	Arolic10	27-07-2010	05:06:08 PM	Nairobi, Kenya		Cloudhopper	90943412717820362	17-09-2017	06:09:27 PM	There Will Be No Election With Uhamia! Mafu A! EBC AKA KEBIC OR JEBIC (KIKUYULEBIC)	Ethnicity	Neutral speech	Hate speech	17-09-2017	06:09:26 PM	60e979c86877f63e15e63f167b3b450f1c0ab667b1ac0809a1931701c	6180a32c1a88b9e9fb788a33a693295600c125a0a07beb18924ec971
7	807510236078117888	Agogo Mwaroh	hedororc236	10-12-2016	12:00:17 PM	Nairobi, Kenya		Twitter for Android	909437143884145153	17-09-2017	06:21:26 PM	RT @odgingr_ctuna: @WilliamsRuto you remember 2007 'you were in charge on inciting Kalenjins against Kikuyas And many died and now you behave...	Ethnicity	Neutral speech	Hate speech	17-09-2017	06:21:25 PM	8209988a2be75945d6c0e15a80328453bba154702e999590500c712f	b79f75e0a0400876c2240404028244ab98a12c5fa0e4517ac35851
8	3321870961	[]kayee[]	MwaDaudh	12-06-2015	11:43:11 PM	Nairobi, Kenya		Mobile Web (iG)	90943757677213666	17-09-2017	06:23:05 PM	RT @goreocouta: Jubilee is a national party, Nasa is a tribal party for the luos backed by ihyias and currently worshipped by kambas #Jub...	Ethnicity	Neutral speech	Hate speech	17-09-2017	06:23:11 PM	634080b468c2a964ad7c5e85c2818f178322018942888488ad478c0a615	cb80208995f6917c33a01c5e6ebaf38157c0a9549888aa3121c3199a18
9	598387553	Yvonne Mutig	mutig_yvonne	01-06-2012	02:55:52 PM	Kenya		Twitter Lite	90943844532865794	17-09-2017	06:26:36 PM	RT @goreocouta: Jubilee is a national party, Nasa is a tribal party for the luos backed by ihyias and currently worshipped by kambas #Jub...	Ethnicity	Neutral speech	Hate speech	17-09-2017	06:26:36 PM	634080b468c2a964ad7c5e85c2818f178322018942888488ad478c0a615	1316e420e17840705640909045da0851675c08b6b2189463326c35ad55
10	82233221999494144	George Muthi	GeorgeM27467622	20-01-2017	09:37:32 AM			Twitter for Android	909438579100852225	17-09-2017	06:27:08 PM	RT @goreocouta: Jubilee is a national party, Nasa is a tribal party for the luos backed by ihyias and currently worshipped by kambas #Jub...	Ethnicity	Neutral speech	Hate speech	17-09-2017	06:27:11 PM	634080b468c2a964ad7c5e85c2818f178322018942888488ad478c0a615	6993460c01178a741acdf42488625a6671e2a07a2313ceab80a0c0451c5
11	219395918	Sir_Rawlings	Sir_Rawlings	15-11-2010	09:25:55 PM	Rongai, Kenya		Twitter for Android	909438787033060528	17-09-2017	06:27:58 PM	RT @goreocouta: Jubilee is a national party, Nasa is a tribal party for the luos backed by ihyias and currently worshipped by kambas	Ethnicity	Neutral speech	Hate speech	17-09-2017	06:27:55 PM	634080b468c2a964ad7c5e85c2818f178322018942888488ad478c0a615	43acada3e0a244a71a3a7d5401a2940e186e1b16b216786c3c2b3d3302b

Figure 56: Hate Speech Classified Tweets

Again, the following shows some of tweets classified as hate speech

Status Update	Category	label	prediction	Stream Date	Stream Time	Tweet item SHA256 Hash Key	User Tweet SHA256 Hash Key
But I hear Kalenjins warn Kikuyus of unspecified consequences if they will not vote for you in 2022 https://t.co/7vx4f0kZ	Ethnicity	Neutral speech	Hate speech	17-09-2017	05:02:20 PM	78a506a904dd2d00ac5753a1177d52da40e9a0532dc4b223ad97655572133016	93c2e85f514016dd9a2079d0eb3ee9883f09db2c02a8804097444adff91270
RT @harrisonmumia: The Ngunjiri petition against Maraga has definitely deepened the feeling amongst other tribes that Kikuyus are selfish...	Ethnicity	Neutral speech	Hate speech	17-09-2017	05:08:17 PM	52dd948d222b9636fab3d0d83cf54e1f7b0e69ace36a8a0db18822e25c303	ecdc7140847e4a04d93f2f8c269ad5c3887b763c63c3389dc8007a2ffc85e
RT @CPATius: @Aaamoh_@Kamranja_jnr My opinion: luos and kikuyus will never fight.Fight will always btwn kikuyus and kales.This c...	Ethnicity	Neutral speech	Hate speech	17-09-2017	05:20:58 PM	b7325549812bb4c6eefafaf9d450d9b98aa0dbc13df8e59677ed293f90d9f65	7de56de128f6e2c8c8e7a92a4127b05aa2d5112a049368548f0d6b3cdda49b
Transgender student shot, killed by police on Georgia Tech campus https://t.co/DgbGZDwLuN	Sexual	Neutral speech	Hate speech	17-09-2017	07:00:32 PM	0a29ca2e6d39a02e30d3ea34130a171aeeef38d3650707dc0c845785362211e	eeae2be99b0805267e121edce40774ba869434506082c145d962e1d512a16d
@WilliamsRulo Plan A violence failed. Mass killings in Kisumu, Kibra, Mathare + burning of schs. Loading round 2. We are alert! #NoReformsNoElections	Violence	Neutral speech	Hate speech	17-09-2017	05:25:25 PM	a50d607024a2988e2c13602c2fcd7dd3c7a325351c0fb1ba815843a63d8a8c	70a3921bcd4a148708d4c7788b793504dccb1d821a9b1a3e511388c1ef857990
RT @Kommonsense7: @WilliamsRulo Plan A violence failed. Mass killings in Kisumu, Kibra, Mathare + burning of schs. Loading round 2. W...	Violence	Neutral speech	Hate speech	17-09-2017	05:26:32 PM	74db4c01334b735635d5fba3ed9cd165e65ae93552f00da4bc06b0374f19cd8	0645b584abc164192fb3fed6d712f2ce05718af047ea76722bab7c74e304

Table 7: Sample Hate Speech Tweets

CHAPTER FIVE: CONCLUSIONS AND RECOMMENDATIONS

5 Conclusions

In our study, we designed a forensic tool that analyzes tweets sentiments for hate speech and cyberbullying using spark machine learning techniques. The classification algorithm was implemented in Apache Spark cluster using the Apache Spark's Machine Learning library, namely Spark ML API. The study relied on distributed contributing framework Apache Spark and made use of Spark streaming API to stream Twitter data and Spark ML API for tweet analysis and classification. We designed Naïve Bayes model by utilizing a dataset of 2,197,498 tweets to train the model and 940,869 tweets for testing. The model was used to stream and classify hate speech and detect cyberbullying for Kenyan based hate speech during 2017 general election and following the nullification of presidential election. Data mining techniques namely text analysis (sentimental analysis) proved effective in detecting hate speech and cyberbullying in Twitter Social network.

The model was able to successfully detect and classify hate speech which mostly were ethnic based. The study has demonstrated how twitter social network data can be collected and preserved within MongoDB database for forensic analysis to ensure its authenticity before court of law and ensure forensic reproducibly. The study has shown that by generating SHA-256 hash key for each twitter item within DStreams and saving the hash key with each individual tweet item in database can be used to detect changes to the data stream during analysis or different forensic analyst can verify the twitter data and thus repeatability/reproducibility of the forensic data can be done. This feature can be used for forensic evidence preservation and ensure changes to the streamed evidence data can be detected by regenerating the SHA-256 Hash key and comparing it with already stored tweet item key in MongoDB database. The study also has shown that by preserving each tweet stream date and time can be used to document the acquisition of the evidence hence improving the chain of custody. The forensic tool was able to ensure chain of custody by maintaining "When" the evidence was captured (Date/Time), when each tweet was created/posted, "Where" the evidence was posted from (source) and tweet ID.

Twitter page printouts and screenshot may not be authenticated or allowed as evidence before a court of law because they lack indication or proof of its creator, source, or custodian. The study has demonstrated which twitter account metadata might be relevant in forensic analysis of twitter posts and how it can be captured. It was evident that a lot of cyberbullying and hate speech is rampant in twitter social media and when the data is well retrieved and preserved, it can form basis for forensic investigation. However, the issue of the dynamic nature of the updates makes it a challenge which calls for real live streaming of social media data which might demand large storage space. The research has shown that Apache Spark Streaming API can utilized to supplement traditional forensic tools in solving challenges involved in handling big data volumes, velocity of data generation, storage and processing of big forensic tweet data.

5.1 Limitations

The study was restricted to English tweets and was not able to analysis and classify tweets posted on other languages like native or Sheng languages which is commonly used on most social networking sites. In addition to tweet text updates, Twitter social network also makes use emotions and Symbols to express one's feelings. The study was not able to analysis and classify this emoji and symbols to determine if they are of hate speech or cyberbullying in nature.

5.2 Recommendations

The spark forensic tool is recommended for use in forensics involving twitter social network site in detecting hate speech and cyberbullying related crimes. In order to improve the performance and classification accuracy, the study recommends the following:

- a) The study didn't exhaustively utilize all hate and bully related words hence more research should be carried out to boost the hate dictionary.
- b) Social networking sites forensics tools are still in their infancy and more research is required to improve on the preservation and authentication of evidence obtained from such sites.
- c) The tool can be extended to cover forensics on other social media and also handle emotion (emoji) and symbol classification and categorization.
- d) Out of 940,869 Testing Tweets, 215,752 Tweets were wrongly classified hence we recommend incorporating others classification algorithms so as to improve the accuracy to near 100%.

5.3 Future Plan

As future work, we plan to extend the forensic tool to include all other social media with capability to provide hot maps which indicates the specific region where such hate speech was posted on social media. It will also involve using other machine learning algorithms to try and increase the effectiveness of the tool in identifying and categorizing hate speech and cyber bullying on social network sites.

REFERENCES

- Agarwal, S. & Sureka, A. 2015, 'Applying social media intelligence for predicting and identifying on-line radicalization and civil unrest oriented threats', arXiv preprint arXiv:1511.06858.
- Altheide, C. & Carvey, H. 2011, *Digital forensics with open source tools*, Elsevier.
- Baesens, B., Van Vlasselaer, V. & Verbeke, W. 2015, *Fraud Analytics Using Descriptive, Predictive, and Social Network Techniques: A Guide to Data Science for Fraud Detection*, John Wiley & Sons.
- Beebe, N.L., Clark, J.G., Dietrich, G.B., Ko, M.S. & Ko, D. 2011, 'Post-retrieval search hit clustering to improve information retrieval effectiveness: Two digital forensics case studies', *Decision Support Systems*, vol. 51, no. 4, pp. 732-44.
- Berman, J.J. 2013, *Principles of big data: preparing, sharing, and analyzing complex information*, Newnes.
- Borthakur, D. 2010, *Looking at the code behind our three uses of Apache Hadoop*, <https://web.facebook.com/notes/facebook-engineering/looking-at-the-code-behind-our-three-uses-of-apache-hadoop/468211193919/?_rdr>.
- Chauhan, T. & Aluvalu, R., 'Using Big Data Analytics for developing Crime Predictive Model'.
- Chen, L., Xu, L., Yuan, X. & Shashidhar, N. 2015, 'Digital forensics in social networks and the cloud: Process, approaches, methods, tools, and challenges', *Computing, Networking and Communications (ICNC), 2015 International Conference on*, pp. 1132-6.
- Chen, Y., Zhou, Y., Zhu, S. & Xu, H. 2012, 'Detecting offensive language in social media to protect adolescent online safety', *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Conference on Social Computing (SocialCom), IEEE*, pp. 71-80.
- Cho, C., Chin, S. & Chung, K.S. 2012, 'Cyber forensic for hadoop based cloud system', *International Journal of Security and its Applications*, vol. 6, no. 3, pp. 83-90.
- Cichosz, P. 2014, *Data Mining Algorithms: Explained Using R*, John Wiley & Sons.
- Copeland, R. 2013, *MongoDB applied design patterns*, " O'Reilly Media, Inc."
- Dua, S. & Du, X. 2016, *Data mining and machine learning in cybersecurity*, CRC press.
- Edwards, D. 2011, 'Computer Forensic Timeline Analysis with Tapestry', *SANS Gold Paper* accepted November.
- Fei, B.K.L. 2007, 'Data visualisation in digital forensics', Citeseer.
- Flaglien, A.O. 2010, 'Cross-computer malware detection in digital forensics'.
- Frampton, M. 2015, *Mastering Apache Spark*, Packt Publishing Ltd.
- Golbeck, J. 2013, *Analyzing the social web*, Newnes.
- Guller, M. 2015, *Big Data Analytics with Spark*, Springer.
- Gupta, R. & Brooks, H. 2013, *Using Social Media for Global Security*, John Wiley & Sons.
- Irons, A. & Lallie, H.S. 2014, 'Digital forensics to intelligent forensics', *Future Internet*, vol. 6, no. 3, pp. 584-96.
- Johnsen, J.W. 2016, 'Algorithms and Methods for Organised Cybercrime Analysis'.
- Juturu, L.S. 2015, 'Applying big data analytics on integrated cybersecurity datasets', Texas Tech University.
- Karran, A.J., Haggerty, J., Lamb, D.J., Taylor, M.J. & Llewellyn-Jones, D. 2011, 'A Social Network Discovery Model for Digital Forensics Investigations', *WDFIA*, pp. 160-70.

- Kayarkar, P.V., Ricchhariaya, P. & Motwani, A. 2014, 'Mining Frequent Sequences for Emails in Cyber Forensics Investigation', *International Journal of Computer Applications*, vol. 85, no. 17.
- Koutsoumpakis, G. 2014, 'Spark-based Application for Abnormal Log Detection'.
- Kumar, K., Sofat, S., Aggarwal, N. & Jain, S. 2012, 'Identification of User Ownership in Digital Forensic using Data Mining Technique', *International Journal of Computer Applications*, vol. 50, no. 4.
- Li, S., Lee, S.Y.M., Chen, Y., Huang, C.-R. & Zhou, G. 2010, 'Sentiment classification and polarity shifting', *Proceedings of the 23rd International Conference on Computational Linguistics*, Association for Computational Linguistics, pp. 635-43.
- Madhavan, S. 2015, *Mastering Python for Data Science*, Packt Publishing Ltd.
- Magnusson, J. 2012, 'Social Network Analysis Utilizing Big Data Technology'.
- Nair, L.R. & Shetty, S.D. 2015, 'STREAMING TWITTER DATA ANALYSIS USING SPARK FOR EFFECTIVE JOB SEARCH', *Journal of Theoretical and Applied Information Technology*, vol. 80, no. 2, p. 349.
- Nandi, A. 2015, *Spark for Python Developers*, Packt Publishing Ltd.
- NATARAJAN, M. 2016, *STREAMIFIC, THE INGESTION SERVICE FOR HADOOP BIG DATA AT UBER ENGINEERING*, <<https://eng.uber.com/streamific/>>.
- Ncr, P.C., Clinton, J., Ncr, R.K., Khabaza, T., Reinartz, T., Shearer, C. & Wirth, R. 1999, 'CRISP-DM 1.0'.
- Nirkhi, S. & Dharaskar, R., 'Authorship Identification in Digital Forensics using Machine Learning Approach'.
- O'higgins, N. 2011, *MongoDB and Python: Patterns and processes for the popular document-oriented database*, " O'Reilly Media, Inc."
- Patzakis, J. 2012, *Overcoming Potential Legal Challenges to the Authentication of Social Media Evidence*, viewed 17/12/2016 2016, <<https://articles.forensicfocus.com/2012/04/02/overcoming-potential-legal-challenges-to-the-authentication-of-social-media-evidence/>>.
- Pentreath, N. 2015, *Machine Learning with Spark*, Packt Publishing Ltd.
- Prajapati, V. 2013, *Big data analytics with R and Hadoop*, Packt Publishing Ltd.
- Press, E.-C. 2010, 'Computer Forensics: Investigation procedures and response', *Course Technology Cengage learning*, USA.
- Privitera, G., Ghidini, G., Emmons, S.P., Levine, D., Bellavista, P. & Smith, J.O. 2014, 'Soft real-time GPRS traffic analytics for commercial M2M communications using spark', *Smart Computing (SMARTCOMP)*, 2014 International Conference on, IEEE, pp. 13-20.
- Quick, D. & Choo, K.-K.R. 2014, 'Data reduction and data mining framework for digital forensic evidence: storage, intelligence, review and archive', *Trends & Issues in Crime and Criminal Justice*, vol. 480, pp. 1-11.
- Ramamonjison, R. 2015, *Apache Spark Graph Processing*, Packt Publishing Ltd.
- Russell, M.A. 2013, *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More*, " O'Reilly Media, Inc."
- Sarkar, D. 2016, *Text Analytics with Python: A Practical Real-World Approach to Gaining Actionable Insights from your Data*, Apress.
- Shahrivari, S. 2014, 'Beyond batch processing: towards real-time and streaming big data', *Computers*, vol. 3, no. 4, pp. 117-29.

- Solaimani, M., Iftekhhar, M., Khan, L., Thuraisingham, B. & Ingram, J.B. 2014, 'Spark-based anomaly detection over multi-source VMware performance data in real-time', Computational Intelligence in Cyber Security (CICS), 2014 IEEE Symposium on, IEEE, pp. 1-8.
- Sremack, J. 2015, *Big Data Forensics—Learning Hadoop Investigations*, Packt Publishing Ltd.
- Tsochataridou, C., Arampatzis, A. & Katos, V., 'Improving Digital Forensics Through Data Mining'.
- Wang, L. & Alexander, C.A. 2015, 'Big Data in Distributed Analytics, Cybersecurity, Cyber Warfare and Digital Forensics', Digital Technologies, vol. 1, no. 1, pp. 22-7.
- Wijeratne, S., Doran, D., Sheth, A. & Dustin, J.L. 2015, 'Analyzing the social media footprint of street gangs', Intelligence and Security Informatics (ISI), 2015 IEEE International Conference on, IEEE, pp. 91-6.
- Zafarani, R., Abbasi, M.A. & Liu, H. 2014, *Social media mining: an introduction*, Cambridge University Press.
- Zawoad, S. & Hasan, R. 2015, 'Digital Forensics in the Age of Big Data: Challenges, Approaches, and Opportunities', High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conference on Embedded Software and Systems (ICCESS), 2015 IEEE 17th International Conference on, IEEE, pp. 1320-5.

APPENDICES

Sample Project Code Twitter Steaming Module

```
import com.fasterxml.jackson.databind.ObjectMapper
import com.mongodb.spark._
import com.mongodb.spark.config.{ReadConfig, WriteConfig}
import org.apache.spark.ml.PipelineModel
import org.apache.spark.rdd.RDD
import org.apache.spark.sql.{DataFrame, SparkSession}
import org.apache.spark.storage.StorageLevel
import org.apache.spark.streaming.dstream.DStream
import org.apache.spark.streaming.twitter.TwitterUtils
import twitter4j.Status
import twitter4j.auth.OAuthAuthorization
import scala.io.Source

object tweetstreamingmodel {
  /*******
  @transient
  @volatile private var spark_SparkSession: SparkSession = _ //Equivalent of SQLContext
  val naivemodelpth="/home/smulwa/data/naiveBayesModel"
  case class SummaryStats(Recall: Double, Precision: Double, F1measure: Double,Accuracy:Double)
  var tweetcategory:String=_
  /*******
def main(args: Array[String]) {
  try{
    var totalTweets: Long = 0

    if (spark_SparkSession == null) {
      spark_SparkSession = SentUtilities.getSparkSession() //Get Spark Session Object
    }
    val spark_streamcontext = SentUtilities.getSparkStreamingContext(spark_SparkSession.sparkContext)
    spark_streamcontext.checkpoint("hdfs://KENBO-SPK08.forensics.net:54310/checkpoint/")
    // Load Naive Bayes Model from local drive.
    val naiveBayesModel = PipelineModel.load(naivemodelpth)
    val sqlcontext = spark_SparkSession.sqlContext //Create SQLContext from SparkSession Object
    import sqlcontext.implicits._
    val twitteroAuth: Some[OAuthAuthorization] = OAuthUtilities.getTwitterOAuth()
    val tweetfilters=MongoScalaUtil.getTweetFilters(spark_SparkSession)

    val Twitterstream: DStream[Status] = TwitterUtils.createStream(spark_streamcontext, twitteroAuth, tweetfilters,
    StorageLevel.MEMORY_AND_DISK_SER).filter(_._getLang() == "en")

    Twitterstream.foreachRDD { rdd =>
      if (rdd != null && !rdd.isEmpty() && !rdd.partitions.isEmpty) {
        saveRawTweetsToMongoDB(rdd)
        rdd.foreachPartition { partitionOfRecords =>
          if (!partitionOfRecords.isEmpty) {
            partitionOfRecords.foreach(record =>
            MongoScalaUtil.SaveRawtweetstoMongodb(record.toString,record.getUser.getId,record.getId,SentUtilities.getStreamDate(),SentUtilities.getStreamTime()))//mongo_utilities.save(record.toString,spark_SparkSession.sparkContext))
          }
        }
      }
    }
  }
}
```

```

    }
  }
}
}
//*****
//Get sentiments and tweet text
val data = Twitterstream.map { status =>
  //clean tweets for use on saved model.
  val cleanedTweetttext=getTweetTextcleaner(status.getText)
  //Generate SHA-256 Hash key for each tweet text
  val originaltextHashkey = SentUtilities.computeSHA256HashKey(status.getText.toString)
  //Generate MD5 Hash key for earch batch of tweets received
  val usertweetSHA256hash = SentUtilities.computeSHA256HashKey(status.toString)
  //Generate tweets schema to be saved later to mongodb
  var latitude:Double=0.0
  var longitude:Double=0.0
  if(status.getGeoLocation != null) {
    latitude = status.getGeoLocation.getLatitude().toDouble
    longitude = status.getGeoLocation.getLongitude().toDouble
  }
  (status.getId,
  status.getUser().getName(),
  status.getUser.getScreenName,
  status.getUser.getOriginalProfileImageURL,
  status.getSource,
  status.getUser.getId,
  status.getUser.getLocation,
  latitude.toString + ", "+longitude.toString ,//Tweet longitude
  status.getText.split(" ").filter(_ startsWith("#")).mkString(" "),
  SentUtilities.getDateformat(status.getUser.getCreatedAt),//Account Creation Date
  SentUtilities.getTimeformat(status.getUser.getCreatedAt.getTime),//Account Creation Time
  status.getText,
  SentUtilities.getDateformat(status.getCreatedAt),//Tweet Creation Date
  SentUtilities.getTimeformat(status.getCreatedAt.getTime),//Tweet Creation Time
  cleanedTweetttext,
  SentUtilities.getStreamDate(),
  SentUtilities.getStreamTime(),
  status.toString,
  originaltextHashkey,
  usertweetSHA256hash)
}
data.cache()
//*****
data.foreachRDD { rdd =>
  if (rdd != null && !rdd.isEmpty() && !rdd.partitions.isEmpty) {
    // convert RDD into DataFrame
    val tweetdf = rdd.toDF("tweet_id", "Name", "ScreenName", "originalProfileImageURL",
    "source", "useraccount_id", "location", "geoLocation", "Hashtag", "Accountcreationdate", "Accountcreationtime",
    "originaltext", "tweetCreationdate", "tweetCreationtime", "text", "cDate", "cTime", "userStatusUpdate",
    "originaltextHashkey", "usertweetmd5hash").dropDuplicates()
    //using earlied loaded model, pass the tweetdf dataframe for classification.
    val predictions = naiveBayesModel.transform(tweetdf)
  }
}

```

```

val predictionDF: DataFrame = predictions.select("tweet_id", "Name", "ScreenName", "originalProfileImageURL",
"source","useraccount_id", "location","geoLocation","Hashtag","Accountcreationdate",
"Accountcreationtime", "originaltext","tweetCreationdate", "tweetCreationtime","text", "prediction",
"cDate","cTime","userStatusUpdate","originaltextHashkey", "usertweetmd5hash").toDF()
//Categorize the classified tweets into either sexual, ethnicity, bully,religious etc.
val categorizedDF = predictionDF.map { status =>
  if (status.getAs[Double]("prediction") == 0.0) {
    tweetcategory = Classify_Util.CategorizeTweets(status.getAs[String]("text"))
  }
  else {
    tweetcategory = "Neutral/Positive"
  }
}
(status.getAs[Long]("tweet_id"), status.getAs[String]("Name"), status.getAs[String]("ScreenName"),
status.getAs[String]("originalProfileImageURL"),status.getAs[String]("source"),
status.getAs[Long]("useraccount_id"),status.getAs[String]("location"),status.getAs[String]("geoLocation"),status.get
As[String]("Hashtag"),status.getAs[String]("Accountcreationdate"),
status.getAs[String]("Accountcreationtime"),status.getAs[String]("originaltext"),status.getAs[String]("tweetCreation
date"),status.getAs[String]("tweetCreationtime"),status.getAs[String]("text"), status.getAs[Double]("prediction"),
status.getAs[String]("cDate"),status.getAs[String]("cTime"),status.getAs[String]("userStatusUpdate"),status.getAs[St
ring]("originaltextHashkey"),status.getAs[String]("usertweetmd5hash"),tweetcategory)

}.toDF("tweet_id", "Name", "ScreenName", "originalProfileImageURL", "source","useraccount_id",
"location","geoLocation","Hashtag", "Accountcreationdate","Accountcreationtime",
"originaltext", "tweetCreationdate", "tweetCreationtime", "text", "prediction", "cDate","cTime",
"userStatusUpdate","originaltextHashkey", "usertweetmd5hash","category")

categorizedDF.cache().dropDuplicates()
//Save classified tweets to mongodb.
val writeConfig: WriteConfig = WriteConfig(Map("uri" ->
"mongodb://10.0.10.100:27017/forensicdb.LiveclassifiedTweets"))
MongoSpark.save(categorizedDF.write.option("forensicdb", "LiveclassifiedTweets").mode("append"), writeConfig)
}
}
//*****
data.foreachRDD((rdd, time) => {
  // Ignore empty Rdd batches
  if (rdd.count() > 0) {
    // Combine each partition's results into a single RDD:
    val repartitionedRDD = rdd.repartition(1).cache()
    totalTweets += repartitionedRDD.count()
    if (totalTweets > 500000) {
      System.exit(0)
    }
  }
})
spark_streamcontext.start()
spark_streamcontext.awaitTermination()
}
catch {
  case e: Exception => println("Error has occurred on main forensic module :", e)
}
}
}

```

```

#####
def getTweetTextcleaner(tweetText: String): String = {
  //Remove URLs, RT, MT and other redundant chars / strings from the tweets.
  tweetText.toLowerCase()
    .replaceAll("\n", "")
    .replaceAll("rt\s+", "")
    .replaceAll("\s+@\w+", "")
    .replaceAll("@\w+", "")
    .replaceAll("\s+#\w+", "")
    .replaceAll("#\w+", "")
    .replaceAll("(?:https?|http?):[/\[\w/%.-]+", "")
    .replaceAll("(?:https?|http?):[/\[\w/%.-]+\s+", "")
    .replaceAll("(?:https?|http?):[/\[\w/%.-]+\s+", "")
    .replaceAll("(?:https?|http?):[/\[\w/%.-]+", "")
    .split(" ")
    .filter(_.matches("^[a-zA-Z]+$"))
    .fold("")(a, b) => a.trim + " " + b.trim.trim
}

#####
//Jackson Object Mapper for mapping twitter4j.Status object to a String for saving raw tweet.
val jacksonObjectMapper: ObjectMapper = new ObjectMapper()
// @param rdd -- RDD of Status objects to save.
def saveRawTweetsToMongoDB(rdd: RDD[Status]): Unit = {
  try{
    val sqlContext = spark_SparkSession.sqlContext
    val tweet = rdd.map(status => jacksonObjectMapper.writeValueAsString(status))

    val rawTweetsDF = sqlContext.read.json(tweet)

    val readConfig: ReadConfig = ReadConfig(Map("uri" ->
"mongodb://10.0.10.100:27017/forensicdb.LiveRawTweets?readPreference=primaryPreferred"))
    val writeConfig: WriteConfig = WriteConfig(Map("uri" ->
"mongodb://10.0.10.100:27017/forensicdb.LiveRawTweets"))
    MongoSpark.save(rawTweetsDF.coalesce(1).write.format("org.apache.spark.sql.json").option("forensicdb",
"LiveRawTweets").mode("append"), writeConfig)
  }
  catch {
    case e: Exception => println("Error Saving tweets to Mongoddb:", e)
  }
}
}

```

MongoDB Save Module

```
import com.mongodb.async.client.{MongoClient => JMongoClient}
import com.mongodb.spark.MongoSpark
import com.mongodb.spark.config.{ReadConfig, WriteConfig}
import org.apache.spark.sql.{DataFrame, SparkSession}
import org.mongodb.scala.bson.collection.mutable.Document
import org.mongodb.scala.{MongoClient, MongoCollection, MongoDBase, _}

object MongoScalaUtil {
  #####
  def SaveRawtweetstoMongodb(tweet:String, twitter_account_id:Long , tweet_id:Long, cDate:String,
  cTime:String):Unit={
    try {
      //val mongoClient: MongoClient = MongoClient("mongodb://10.0.10.100:27017")
      val mongoClient: MongoClient = MongoClient()
      val database: MongoDBase = mongoClient.getDatabase("forensicdb")
      val collection: MongoCollection[Document] = database.getCollection("LiveTweets")

      val tweethashkey = SentUtilities.generateSHA256HashKey(tweet)
      val doc: Document = Document("_id" -> tweet_id, "twitter_account_id" -> twitter_account_id,"cDate" ->
      cDate,"cTime" -> cTime, "usertweets" -> tweet, "tweethashkey" -> tweethashkey)

      val observable: Observable[Completed] = collection.insertOne(doc)
      observable.subscribe(new Observer[Completed] {
        override def onNext(result: Completed): Unit = println("Inserted")

        override def onError(e: Throwable): Unit = println(e.toString)

        override def onComplete(): Unit = println("Completed")
      })
      // mongoClient.close()
    }
    catch {
      case e: Exception => println("Error Saving tweets to Mongodb: ", e)
    }
  }
  #####
  def SaveLabeledTweets(LabeledtweetDF:DataFrame):Unit={
    try {
      val writeConfig: WriteConfig = WriteConfig(Map("uri" ->
      "mongodb://10.0.10.100:27017/forensicdb.LivelabeledTweets"))

      MongoSpark.save(LabeledtweetDF.coalesce(1).write.format("org.apache.spark.sql.json").option("forensicdb",
      "LivelabeledTweets").mode("append"), writeConfig)
    }
    catch {
      case e: Exception => println("Error Saving tweets to Mongodb:", e)
    }
  }
}
```



```

#####
var tweetfilters:List[String]=_
def getTweetFilters(sparksession :SparkSession):List[String]={
  try {
    val sqlcontext = sparksession.sqlContext //Create SQLContext from SparkSession Object
    import sqlcontext.implicits._
    val readConfig = ReadConfig(Map("uri" ->
"mongodb://10.0.10.100:27017/forensicdb.Tweetfilters?readPreference=primaryPreferred"))
    // retrieve twitter streaming filters
    val tweetfiltersDF = MongoSpark.load(sparksession.sparkContext, readConfig).toDF()

    tweetfilters = tweetfiltersDF.select("tweetfilters").map(r => r.getString(0)).collect.toList
  }
  catch {
    case e: Exception => println("Error reading tweets from Mongodb: ", e)
  }
  tweetfilters
}
#####
var hatebasedict:List[String]=_
def gethatebase_dict(sparksession :SparkSession):List[String]={
  try {
    val sqlcontext = sparksession.sqlContext //Create SQLContext from SparkSession Object
    import sqlcontext.implicits._
    val readConfig = ReadConfig(Map("uri" ->
"mongodb://10.0.10.100:27017/forensicdb.Hatebase_dict?readPreference=primaryPreferred"))
    // retrieve filters
    val tweetfiltersDF = MongoSpark.load(sparksession.sparkContext, readConfig).toDF()

    hatebasedict = tweetfiltersDF.select("hatebasedict").map(r => r.getString(0)).collect.toList
  }
  catch {
    case e: Exception => println("Error reading tweets from Mongodb: ", e)
  }
  hatebasedict
}
}

```

Front End Flask(Python) Module

```
from flask import Flask, render_template,request,redirect,url_for, g, current_app
from pymongo import MongoClient
import click
from flask_paginate import Pagination, get_page_args

MONGODB_HOST = '10.0.10.100'
MONGODB_PORT = 27017
DBS_NAME = 'forensicdb'
clientConn = MongoClient(MONGODB_HOST, MONGODB_PORT)

app = Flask(__name__)
app.config.from_object('settings')
app.config.from_pyfile('app.cfg')

@app.route('/tweetclassifier' , methods=['GET', 'POST'])
def getclassifiedtweets():
    COLLECTION_NAME='LiveclassifiedTweets'
    collection = clientConn[DBS_NAME][COLLECTION_NAME]

    page, per_page, offset = get_page_args(page_parameter='page', per_page_parameter='per_page')
    total = 0
    tweets = []
    if request.method == "POST":
        if request.form.get('btnsubmit',None)=='bycategory':
            hatecategory = request.form.get('hatecategory')
            total=collection.find({"category":hatecategory}).count()
            tweets = collection.find({"category":hatecategory}).skip(offset).limit(per_page)
        elif request.form.get('btnsubmit',None)=='searchbydate':#Search tweets by capture/stream date
            tweetbydate = request.form.get('searchbydate')
            total=collection.find({"cDate":tweetbydate}).count()
            tweets = collection.find({"cDate": tweetbydate}).skip(offset).limit(per_page)
        else: #Tweet keyword search
            tweetwordsearch = request.form.get('tweetwordsearch')
            total=collection.find({"originaltext": {'$regex':tweetwordsearch}}).count()
            tweets = collection.find({"originaltext": {'$regex':tweetwordsearch}}).skip(offset).limit(per_page)
    else:
        total = collection.find().count()
        tweets = collection.find().skip(offset).limit(per_page)

    pagination = get_pagination(page=page, per_page=per_page, total=total, record_name='Tweets',
    format_total=True,format_number=True)

    return render_template('tweetclassifier.html', tweets=tweets, page=page, per_page=per_page, pagination=pagination)

@app.route('/categorizedtweetchart')
def getcategorizedtweets():
    COLLECTION_NAME='LiveclassifiedTweets'
    collection = clientConn[DBS_NAME][COLLECTION_NAME]
    tweets=collection.find({"prediction":0.0})
    categorizedtweetsArr=[]

    violence = 0
```

```

Sexual = 0
Religion = 0
ethnicity = 0
bully = 0
others = 0
labels = ["Violence", "Sexual", "Religion", "Ethnicity", "Bullying", "Others"]
for tweet in tweets:
    if tweet["category"]=="Violence":
        violence +=1
    elif tweet["category"]=="Sexual":
        Sexual += 1
    elif tweet["category"]=="Religion":
        Religion +=1
    elif tweet["category"]=="Ethnicity":
        ethnicity += 1
    elif tweet["category"]=="Bullying":
        bully += 1
    else: #Others
        others+=1

categorizedtweetsArr = [violence, Sexual, Religion,ethnicity,bully,others]

return render_template('categorizedtweetchart.html', labels=labels, categorizedtweetsArr=categorizedtweetsArr)

@app.route('/index')
def getindex():
    COLLECTION_NAME='LiveTweets'
    collection = clientConn[DBS_NAME][COLLECTION_NAME]
    page, per_page, offset = get_page_args(page_parameter='page',per_page_parameter='per_page')
    total=collection.find().count()

    tweets = collection.find().skip(offset).limit(per_page).sort('cDate',1)

    pagination = get_pagination(page=page,per_page=per_page,total=total,
    record_name='Tweets',format_total=True,format_number=True)

    return render_template('viewtweets.html', tweets=tweets, page=page, per_page=per_page, pagination=pagination)

def get_page_items():
    page = int(request.args.get('page', 1))
    per_page = request.args.get('per_page')
    if not per_page:
        per_page = 12
    else:
        per_page = int(per_page)
    offset = (page - 1) * per_page
    return page, per_page, offset

def get_css_framework():
    return current_app.config.get('CSS_FRAMEWORK', 'bootstrap3')

def get_link_size():
    return current_app.config.get('LINK_SIZE', 'sm')

```

```
def show_single_page_or_not():
    return current_app.config.get('SHOW_SINGLE_PAGE', False)

def get_pagination(**kwargs):
    kwargs.setdefault('record_name', 'records')
    return
    Pagination(css_framework=get_css_framework(),link_size=get_link_size(),show_single_page=show_single_page_or_not(), **kwargs)

@click.command()
@click.option('--port', '-p', default=5000, help='listening port')
def run(port):
    app.run(debug=True, port=port)

#####
if __name__ == '__main__':
    app.run(host='forensics.net', port=5000)
```