**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**MONITORING SUSPICIOUS BEHAVIOUR ON SOCIAL MEDIA USING DATA MINING**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

****

**Supervisor:**

**Mr.**

**Akim Munthali**

**Author:**

**Terence Tachiona M147740**

**A RESEARCH PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE HONORS DEGREE IN COMPUTER SCIENCE**

**in the**

**Department of Mathematics and Computer Science**

**Gary Magadzire School of Agriculture and Natural Sciences**

**November 2017**

**Monitoring Suspicious Behaviour on social Media using Data Mining**

Project was done as a final project in a Bachelor of Science Honours Degree in Computer Science at Great Zimbabwe University in 2017.

A model built is a Twitter crawler that performs a sentiment analysis to determine whether tweets about something/someone are positive or negative with respect to crime behavior. The main goal of the project was to create a model that collects crime related comments and tweets from Twitter, perform a sentimental analysis of each tweet collected and to show the results by creating a webpage. The model was written with the Python programming language, uses the Twitter GET Search API to collect tweets, performs the analysis using Naive Bayes Classiﬁer and displays the results in a simple html webpage. An additional feature was to try to put collected data in a database. This is beneﬁcial as the model can easily fetch old data, for searches that have already been done before, and use it with the newly harvested tweets.

**Requirements**

The program uses the [data structure server Redis](http://redis.io/ "Redis Homepage"), to save the data collected. It therefore must be installed and running (by calling "redis-server") on the client's computer when using the program.

The program also uses [Mako Templates](http://www.makotemplates.org/download.html "Mako Homepage") for Python to create a webpage with the results.

**SECTION 1: DESIGN OF THE PROGRAM**

This section focuses on the program’s design and thoughts behind it. From the beginning the author knew that he wanted to accomplish three things:

• Harvest posts

• Analyze the harvested posts

• Display the results

So it naturally made sense to create separate classes for each of those tasks. The author also decided to create a main class for the program, whose task would be to call the other classes with appropriate parameters and display the program’s status during runtime. Then for the additional feature, to save the tweets collected in a database, it seemed only natural that the class responsible for the tweet harvesting should be the one that saves the data and oﬀers other classes access to the data to the database.

After looking at a few database solutions the author decided to use the Redis solution because of its simplicity and ease of use and also to minimize time required for the author to familiarize with the solution.

The program therefore consists of four separate Python classes and a Redis database. Each module servers a speciﬁc purpose in order to accomplish the program’s objective.

**Class CrimeAnalyzer**

The CrimeAnalyzer class is the main class in the program and gives other classes the parameters needed and receives the data from them. It also displays on the command line the progress of the search, analysis and html creation.

**Class TwitterAggregator**

The TwitterAggregator purpose is to search Twitter for tweets, get and save the relevant data to a database and oﬀer other classes the chance to retrieve the tweet data that has been harvested.

**Class SentimentAnalyzer**

The SentimentAnalyzer’s job is to create a Naive Bayes classiﬁer that uses manually analyzed data, created by the author, to train how to recognize positive, negative and neutral tweets. The class then takes a list of tweets and performs a sentimentanalysis to classify them into appropriate categories and returns a list with the classiﬁcation information appended to the list.

**Class HTMLCreator**

The HTMLCreator is the class responsible for creating the web page that displays the results from the football analyzer. It takes a dictionary of statistics gathered while harvesting and analyzing the tweets and a list of all tweets analyzed in the run of the program. The class then displays the statistics, creates a word cloud of the most popular words used in the tweets and lists every tweet sent to it, colored in a way so that it is easy to see how the classiﬁer classiﬁed each tweet that it was give.

**SECTION 2: IMPLEMENTATION OF THE PROGRAM**

This section focuses on how the classes and database were implemented in to the system.

**Class CrimeAnalyzer**

CrimeAnalyzer has the following functions:

Public

• run( self )

Private

• \_\_search( self )

• \_\_analyze( self, tweets )

• \_\_create\_webpage( self, analyzed\_tweets )

• \_\_start\_task( self )

• \_\_end\_task( self )

• \_\_print\_time( self, delay )

The CrimeAnalyzer creates an instance of the TwitterAggregator to get the tweets deﬁned in the search parameters. It then creates an instance of the SentimentAnalyzer and makes it analyze all the tweets that were harvested from Twitter to get the sentiment classiﬁcation of each tweet collected. Finally it creates an instance of the HTMLCreator, sends the data from the aggregator and analyzer and makes it create a webpage that shows the results from the program.

To use the CrimeAnalyzer you have deﬁne search terms and specify how many pages of tweets you want the class to search for and how many tweets should be on each page. With those parameters deﬁned you can create an instance of the class and make it run by calling the run() function.

**Class TwitterAggregator**

CrimeAnalyzer has the following functions:

Public

• twitter\_search( self, search\_terms, pages, results\_per\_page )

• get\_tweets( self, search\_terms, return\_all\_tweets )

Private

• \_\_get\_tweet\_ids( self, search\_results )

• \_\_is\_english\_tweet( self, tweet )

The TwitterAggregator performs a Twitter GET Search and harvests tweets using the search parameters given to it. It saves the twitter data, from the search, to a redis database and allows the user to get the data by calling the function "get\_tweets()".

Redis is a Key-Value type of database. The aggregator saves four diﬀerent keys in the system. First it saves the search parameter, with spaces replaced by underscores, with the value True so it is easy to see if a search has been performed with those parameters. An example of this would be the key "Destroy Presidential statutes" and value "True". Next is saves a key with the search parameter and "$TweetIds" appended to it and the value is a list of all tweet ids found in the search. Example of this is the key "Tajamuka$TweetIds" and value [u’143863607367700480’,

u’143863033024876544’...].

To keep track of how many times a search has been performed in the program, the aggregator saves a key with the search parameter and "$SearchCount" appended to it. Example is the key "Tajamuka$SearchCount" and value 5. Finally, to get the data from each tweet id, the aggregator saves each tweet id with the name "ID$" and the id appended to the name. The value is a list which contains the tweet text, username and a URL to the proﬁle picture of the user who created the tweet. An example would be the key "ID$143863033024876544" and value ["The minister was injured today Great!", ’http://a1.twimg.com/sticky\_images/normal.png’, ’velvetdismality’].

To use the TwitterAggregator you create an instance of the aggregator and call the twitter\_search() function with the search parameters, how many pages and number of tweets requested as parameters. Then to get the tweets collected, you call the get\_tweets() function with the search terms you want data from and True or False depending on whether you want all tweets with that search term or just the tweets harvested in the last run.

**Class SentimentAnalyzer**

CrimeAnalyzer has the following functions:

Public

• analyze( self, data )

• get\_analysis\_result( self, data\_to\_get )

• show\_most\_informative\_features( self, amount )

Private

• \_\_init\_naive\_bayes( self )

• \_\_check\_word( self, word )

• \_\_analyze\_tweet( self, tweet )

• \_\_analyse\_using\_naive\_bayes( self, data )

The SentimentAnalyzer uses the Naive Bayes classiﬁer, that is included in the Natural Language toolkit, to classify tweets. It trains the classiﬁer so that it can determine whether a tweet is positive, negative or neutral. The class opens up three diﬀerent ﬁles, titled "tweets\_positive", "tweets\_negative", "tweets\_neutral", gets the text and places it in the classiﬁer. The training data used that was manually categorized by the author and contains over 700 lines of words and sentences.

The data was taken from Tweets harvested while creating the program and from Facebook and Twitter. When doing an analysis the SentimentAnalyzer removes known stop-words, all links found and words that have less than 3 letters. It does so by calling the check\_word() function, for each word, to see if it should include the word or not.

To use the SentimentAnalyzer one has to simply create an instance of it and send a list of tweets to the analyze() function which returns the list with each tweet classiﬁed.

**Class HTMLCreator**

CrimeAnalyzer has the following functions:

Public

• create\_html( self )

Private

• \_\_create\_stats\_info( self )

• \_\_create\_tweet\_list( self )

• \_\_create\_word\_cloud( self )

The HTMLCreator creates a HTML webpage that displays statistics, word cloud and a list of all tweets harvested. The class opens a html template which has all the markup for the webpage and stores it in a string. It takes the statistics given and puts them in a div element with the id "stats".

The HTMLCreator then creates a word cloud of the 30 most frequent words in the tweets found by the aggregator, each word within a <li></li> element with an appropriate css class and places them in the unordered list that has the class "word-cloud". The class for each word is determined by calculating how many times the word has appeared and scaled so that it is between 1 and 25.

The class then creates a list of tweets on the webpage. The webpage shows three tweets per line and assigns each tweet with the class "left-tweets" or "right-tweets" depending on where it should be shown. It also creates an extra div element to color the background of each tweet, green, red or white depending on the results from the classiﬁcation from the SentimentAnalyzer.