Report on

Detection and Analysis of Cyber-bullying tweets based on Events

5891P :Text Mining Project

Vaishali Byloor
Matrikelnummer: **78613**

Maria Mamontova
Matrikelnummer: **82708**

July 2017

Under the Guidance of

Professor Dr. Handschuh
Chair of Digital Libraries and Web Information Systems

Jelena Mitrović
Research Associate at University Passau

CONTENTS :

1. MOTIVATION

The usage of social networks amongst children, adolescents and families is nowadays a common practice in our everyday lives. Social networking sites such as Twitter, Facebook, YouTube, blogs etc provide various ways for people to communicate with one another. With the rapid growth of social media, there is increasing number of users especially adolescents spending a significant amount of time on various social networking sites to connect with others, to share information, and to pursue common interests. While many beneficial effects definitely exist, the platform also has risk of exposing various offensive online contents against people which is termed as bullying.

Wikipedia defines cyberbullying as the use of information technology to repeatedly harm or harass other people in a deliberate manner. According to U.S. Legal Definitions, Cyberbullying could be limited to posting rumors or gossips about a person on the internet bringing about hatred in others minds; or it may go to the extent of personally identifying victims and publishing materials severely defaming and humiliating them.

A main reason individuals are targeted with bullying is perceived differences, i.e., any characteristic that makes an individual stand out differently from his or her peers. These include race, socioeconomic status, gender, sexuality, physical appearance, and behaviors.

Cyberbullying reflects a venue (other than face to face contact) through which verbal and relational forms can occur. This could have the negative impact on people, especially adolescents, which might lead to depression, anxiety, loneliness, aggression and also suicide. Recently, there are increasing cases of cyberbullying due to lack of preventing measures. Therefore, cyberbullying must be considered as a serious problem, which needs to be prevented from happening.

Existing detection techniques occur from the fields of natural language processing, computational linguistics, text mining, and a range from machine learning methods to rule-based methods. Machine learning methods involve training of models on specific collections of documents.

2. INTRODUCTION

In this work we tackle one of the problems identified above, that of cyberbullying, which is continuously increasing in the social Web and is becoming a major threat to teenagers and adolescents [3]. In fact, 55% of teens using social media have witnessed outright bullying via that medium.

Besides, there has been an increasing hate against immigrants and against Muslims in recent years. Our project focuses on these two issues to detect bullying against Islam and Immigrants in Twitter. We use a dictionary of weighted offensive words along with the presence of pronouns or collective nouns to calculate the Offensiveness percentage in each tweet. We also make use of a Sentiment Analysis API to detect sentiment of the tweets.

We use 30% manually labelled training set for Machine learning and then use trained model rest to collected dataset. The goal of our work was to analyze the gathered tweeter corpus, parse and preprocess it, build the model for bullies classification and analyze the geo location statistics after corpus

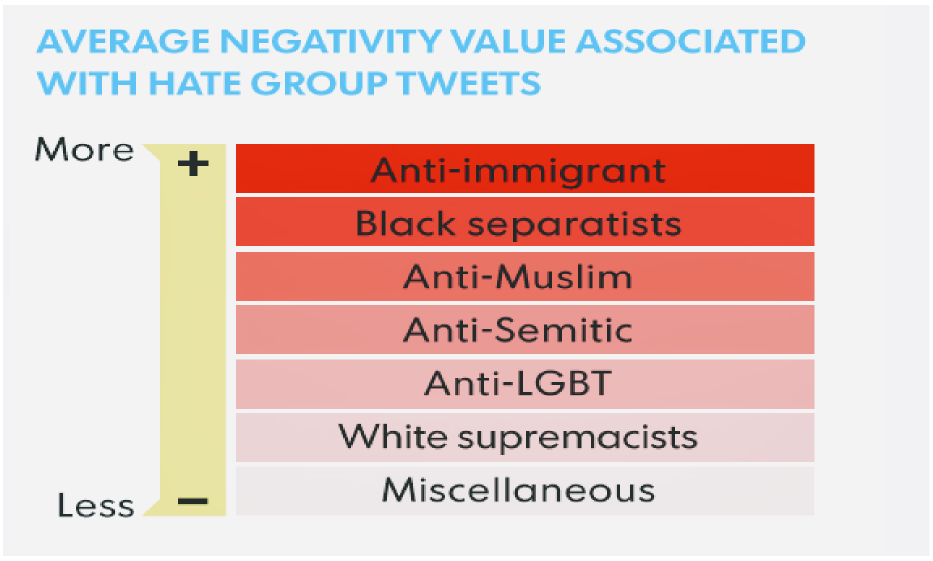
3. TWITTER

Twitter is an popular information network made up of 140-character messages called Tweets. It's an easy way to discover the latest news related to subjects you care about. The maximum length of a Twitter message is 140 characters. This means that we can practically consider a tweet to be a single sentence, void of complex grammatical constructs. Because of this and the 140-character limit, language used in Tweets tends be shorthand, and filled with slang and misspellings. Data availability makes twitter a preferred social platform. With the Twitter API, it is easy to collect millions of tweets for training and testing.

4. EVENTS

The two current world events selected are the refugee crisis in Europe and the Trump wall (hatred against Islam and immigrants). A study released in February 2017 by SafeHome.org[1], an organization of home security experts shows that social media engagement by groups labeled as hate organizations has been booming in the last two to three years and the largest shares of activity are focused on anti-immigrant and anti-Muslim sentiment.

A new study by security organization shows that likes and comments on hate group tweets rose more than 900% from 2014 to 2016.



Hashtags we used, to collect data from Twitter:

- 1 event - #refugees, #immigrants",#islam",#muslim",#assimilate
- 2nd event -#trumpcare, #Trumpwall,#MakeAmericaGreatAgain,#mexicanwall, #Americafirst

5. METHODOLOGY

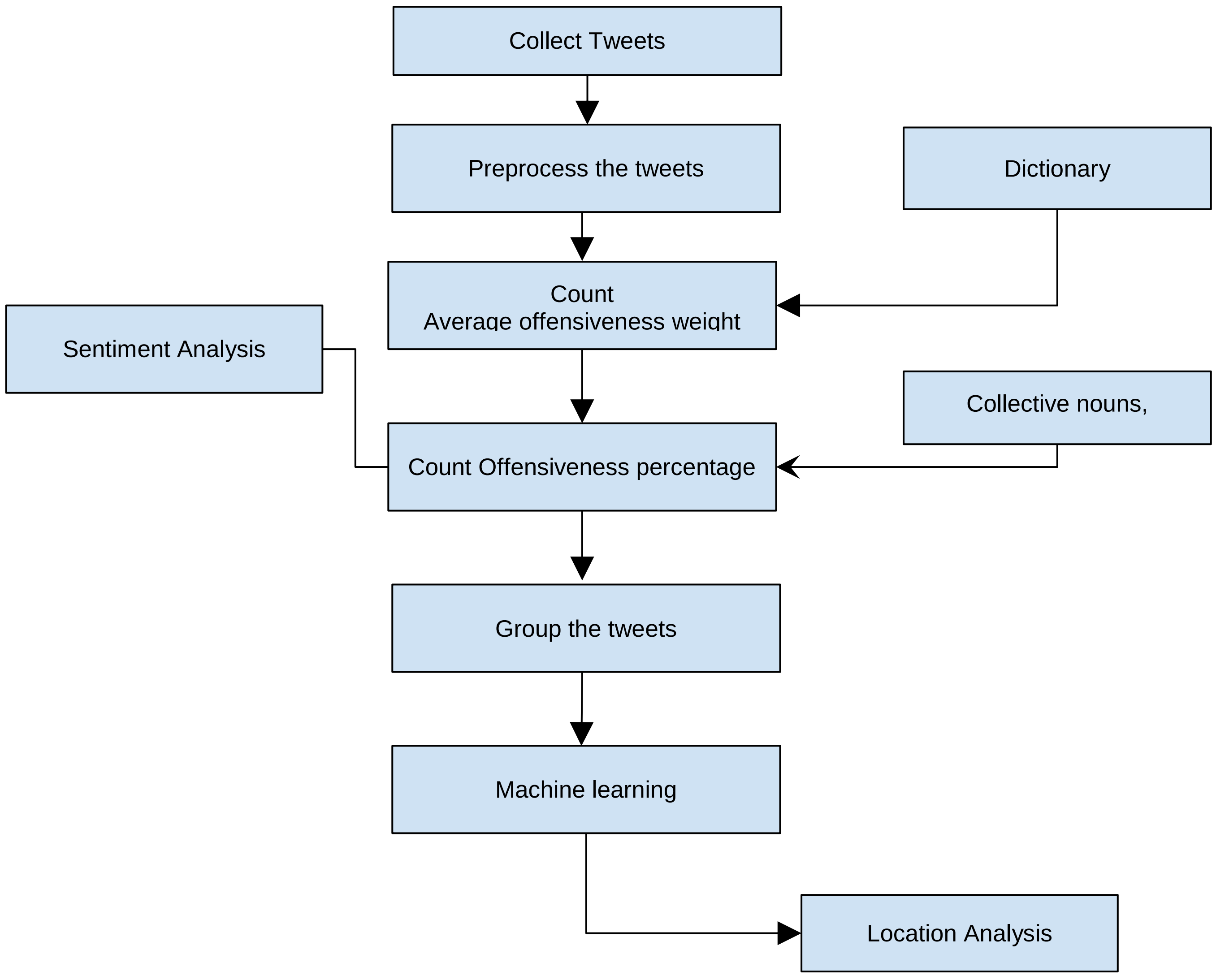


Figure: illustrates the methodology of the project

The overall methodology of our project is described in the above figure : we will look at each step in detail in the following sections

6. COLLECTING DATA

The first step is the registration of our app. In order to have access to Twitter data programmatically, we created an app that interacts with the Twitter API. In particular, we log-in to Twitter and register a new application →choose a name and a description for the app → we will receive a consumer key and a consumer secret and from the configuration page of your app, you can also get an access token and an access token secret. These four credentials must be kept private: they provide the application access to Twitter on behalf of our account.

Install Tweepy→ pip install tweepy==3.3.0

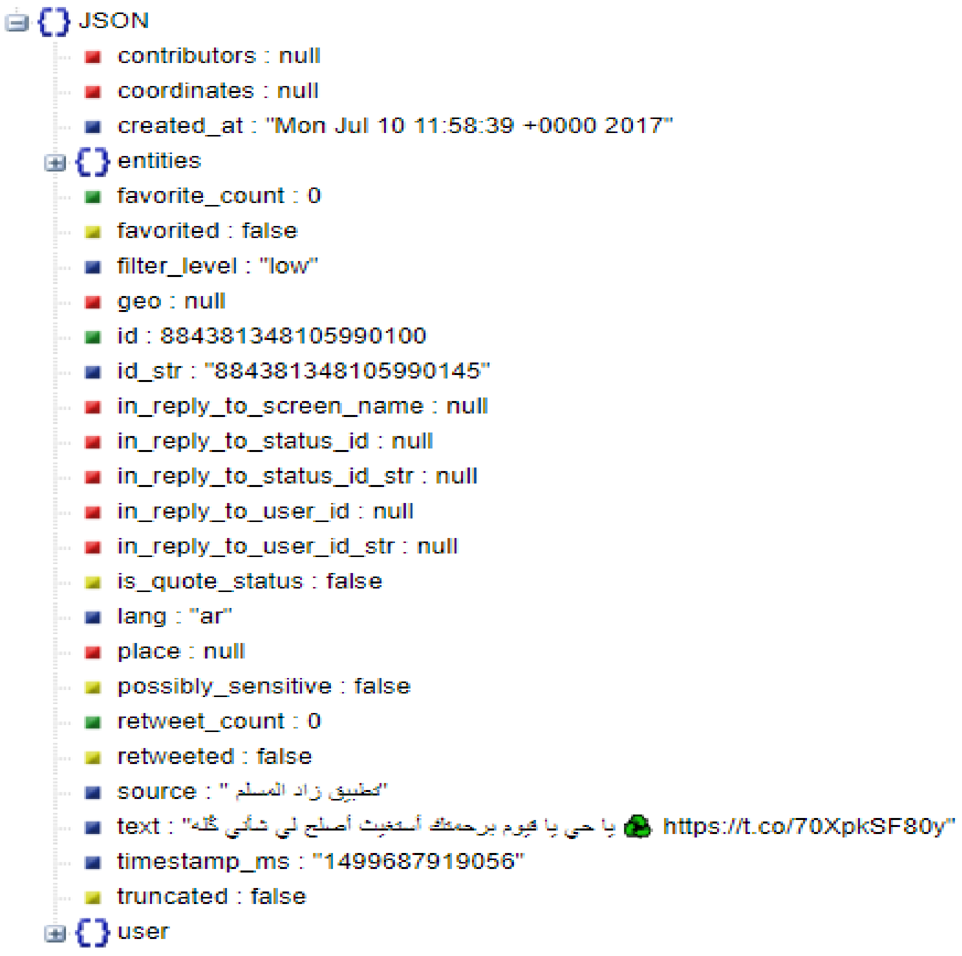
TweetStream.py - use the following python code to stream live tweets. We let this run for a period of time to have collected sufficient number of tweets. The tweets are stored in a json file.

```
import tweepy
from tweepy import OAuthHandler
from tweepy import Stream
from tweepy.streaming import StreamListener
consumer_key = 'YOUR-CONSUMER-KEY'
consumer_secret = 'YOUR-CONSUMER-SECRET'
access_token = 'YOUR-ACCESS-TOKEN'
access_secret = 'YOUR-ACCESS-SECRET'
auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)
api = tweepy.API(auth)
class MyListener(StreamListener):

    def on_data(self, data):
        try:
            with open('Corpus.json', 'a') as f:
                f.write(data)
                return True
        except BaseException as e:
            print("Error on_data: %s" % str(e))
        return True
    def on_error(self, status):
        print(status)
        return True
twitter_stream = Stream(auth, MyListener())
twitter_stream.filter(track=['#'])
```

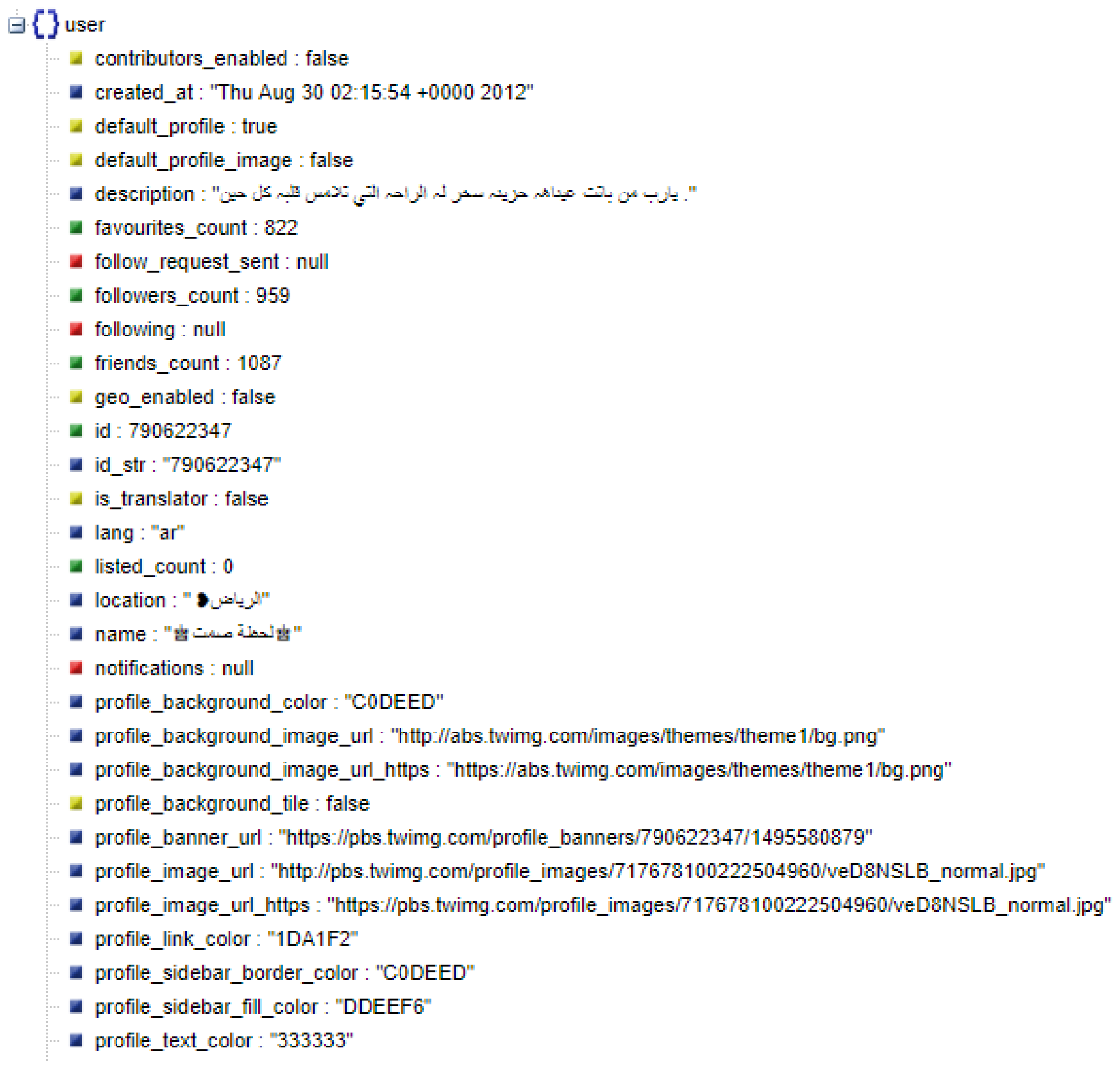
6.1. CORPUS STRUCTURE ANALYSIS

After the twitter corpus was gathered, the work in Project file could be started.The program read the files with json objects, casts the json to python json type and adds to the array. In order to use the gathered information, we have studied the json object structure.The example of the json object can be seen on the picture below :



```
{ } JSON
    contributors : null
    coordinates : null
    created_at : "Mon Jul 10 11:58:39 +0000 2017"
  { } entities
    favorite_count : 0
    favorited : false
    filter_level : "low"
    geo : null
    id : 884381348105990100
    id_str : "884381348105990145"
    in_reply_to_screen_name : null
    in_reply_to_status_id : null
    in_reply_to_status_id_str : null
    in_reply_to_user_id : null
    in_reply_to_user_id_str : null
    is_quote_status : false
    lang : "ar"
    place : null
    possibly_sensitive : false
    retweet_count : 0
    retweeted : false
    source : "تطبيق زاد المسلم"
    text : "يا حي يا قيوم برحمتك أستغيث أصلح لي شأني كله ♻️ https://t.co/70XpkSF80y"
    timestamp_ms : "1499687919056"
    truncated : false
  { } user
```

‘User’ attribute structure:



```
{} user
   contributors_enabled : false
   created_at : "Thu Aug 30 02:15:54 +0000 2012"
   default_profile : true
   default_profile_image : false
   description : "يارب من باتت عيناه حزينه سخر له الراحه التي تلامس قلبه كل حين ."
   favourites_count : 822
   follow_request_sent : null
   followers_count : 959
   following : null
   friends_count : 1087
   geo_enabled : false
   id : 790622347
   id_str : "790622347"
   is_translator : false
   lang : "ar"
   listed_count : 0
   location : " ◗الرياض"
   name : "لحظة صمت"
   notifications : null
   profile_background_color : "C0DEED"
   profile_background_image_url : "http://abs.twimg.com/images/themes/theme1/bg.png"
   profile_background_image_url_https : "https://abs.twimg.com/images/themes/theme1/bg.png"
   profile_background_tile : false
   profile_banner_url : "https://pbs.twimg.com/profile_banners/790622347/1495580879"
   profile_image_url : "http://pbs.twimg.com/profile_images/717678100222504960/veD8NSLB_normal.jpg"
   profile_image_url_https : "https://pbs.twimg.com/profile_images/717678100222504960/veD8NSLB_normal.jpg"
   profile_link_color : "1DA1F2"
   profile_sidebar_border_color : "C0DEED"
   profile_sidebar_fill_color : "DDEEF6"
   profile_text_color : "333333"
```

For our analysis we need the text of the tweets. After observing some tweets we noticed, that user’s description significantly helps in making decision whether the tweet is bullying or not.

Next thing we needed it location. According to the twitter’s policy, users can disable their location. However, in user attribute, the attribute location often stores the textual description of the location. We are using this textual description and with the help on geopy [1] package identify the coordinates of the location. We are observing tweets only in English, so “lang” attribute is also important for us.

First, we were using users’ names and id’s, but for the current state this information is not crucial and for the purposes of speed increasing we decided for now to drop this attribute.

So finally, we are using:

- tweet[‘text’]
- tweet[‘lang’]
- tweet[‘user’][‘description’]
- tweet[‘user’][‘location’]

7. DATA PRE-PROCESSING

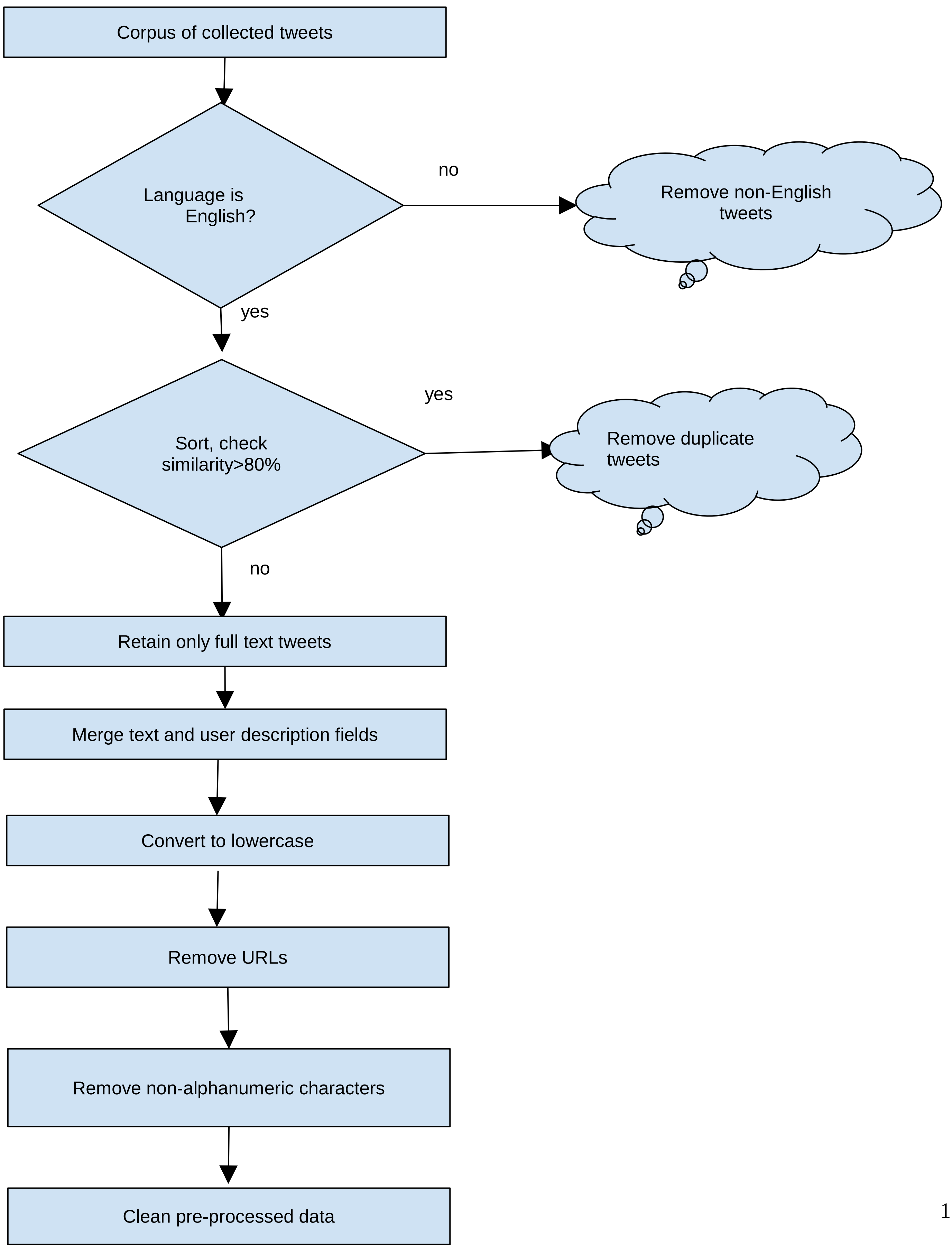
Data preprocessing describes any type of processing performed on raw data to prepare it for another processing procedure. It is a data mining technique, which involves transforming raw data into an understandable format.

Requirement of Data Preprocessing:

Real-world data are often

- Incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data;
- Noisy: containing errors or outliers;
- Inconsistent: containing discrepancies in codes or names.

Data preprocessing is a proven method of resolving such issues. Moreover, user-generated content on the web is rarely present in a form usable for learning. It becomes important to normalize the text by applying a series of pre-processing steps. We have applied an extensive set of pre-processing steps to decrease the size of the feature set to make it suitable for learning algorithms.



Corpus of collected tweets
Language is English?
no
Remove non-English tweets
yes
Sort, check similarity>80%
yes
Remove duplicate tweets
no
Retain only full text tweets
Merge text and user description fields
Convert to lowercase
Remove URLs
Remove non-alphanumeric characters
Clean pre-processed data

Figure 2 : workflow of pre-processing of tweets corpus

The tweets were gathered in Json format, each line of the file is a json object of the following format:
The program is reading file line by line and casts each non empty line to the Json format. The json objects then are stored in the array.
After retrieving the jsons, the program starts preprocessing the data(figure 2):

1) Take only lines with language =”English”
2) Sort the tweets , Check similarity between consecutive tweets , if similarity is greater than 80%, remove the duplicate tweets
3) Search for the full_text of the tweet (In cases of retweets or too big tweets, additional attributes are added to the tweet
4) Next, we Merge text and Description fields of every tweet line as retaining description of the user would be a beneficial addition to classification of tweets as bullying or not
5) Deleting all non-alphanumeric characters
6) Retrieving the location (latitude, longitude) of the tweet. The location in twitter is a plain text, so in order to retrieve location geopy [1] is used.

Text	Description	Offensiveness Weight	Offensiveness Level	Offensive Words Amount	Sentimental Analysis	Class	Geo_Location
Tweet_txt	Description	2,5	58,33333	2	0	1	44.933143; 7.540121
Tweet_txt2		2	50	1	0	1	27.7567667 ;- 81.4639835

Table 1: The view of the data after first step of preprocessing : Empty columns are created for future calculation

8. MODEL

After manual examination of the certain amount of tweets we decided to build the model for assessing the tweet as bullying or not based on three features:

- Mean Offensiveness Weight;
- Offensiveness Percentage;
- Sentimental analysis of the tweet.

In the Data preprocessing step, we merge text and description fields of all the tweets, as in many scenarios description of user account provide an insight and is a beneficial addition for Offensiveness percentage calculation. Hence to decide, whether the tweet/user is bullying or not, we analyze not only the text, but the description of the user as well.

8.1. MEAN OFFENSIVENESS WEIGHT

We have a list of Offensive/Profane Words collected list from [2] . Each offensive word is manually weighted according to its severity 1 to 5 .1 being negligible and 5 being extremely offensive.
We check all the tweets for presence of words from this list, if present it gets the corresponding weights of all those words and calculated mean of those weights

The following is done to each tweet and added to the column Offensiveness weight

```
def offensiveness_weight(TweetText):
        templist = TweetText.split()
        offensiveWordsList=list(offensiveWords.keys())
        temp = set(offensiveWordsList).intersection(templist)
        my_list = list(temp)
        final = 0
        result=0

        if len(my_list)!=0:
        for word in my_list:
                final = float(final) + float(offensiveWords[word])
        result = final / len(my_list)
        return result
```

8.2. OFFENSIVENESS PERCENTAGE

Presence of offensive word alone cannot necessarily mean the text is bullying [4] ,for example : this book is stupid , cannot be classified as a Bullying tweet.

The text/tweet can be considered more offensive if the offensive word is directed at a particular individual, group of people, race, country, religion etc. (which is definition of bullying) for instance: tweet - “His country is Stupid” gives higher offensiveness percentage due to presence of country(collective noun) and his (pronoun). For this reason we also created list of pronouns and collective nouns

Hence we define a function to calculate Offensiveness percentage based on presence of Offensive word along with pronouns or collective nouns.

The function checks if the offensiveness weight is greater than zero and if true , increases the value by 1 and then calculates a percentage of offensiveness in the tweet

The following is done to each tweet and added to the column Offensiveness level (i,e percentage)

```
def offensive_level(TweetText):

        nounsExistance = wordpresence(TweetText,collectivenouns)
        pronounsExistance = wordpresence(TweetText,pronouns)

        if offensiveness_weight(TweetText) > 0 and (wordpresence or pronounsExistance) :
        return (offensiveness_weight(TweetText) + 1)/6 * 100
        else:
        return (offensiveness_weight(TweetText))/6 * 100
```

8.3. SENTIMENTAL ANALYSIS

Sentiment analysis (sometimes known as opinion mining or emotion AI) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.
To analyze the sentiment of each tweet we make use of Sentimental Analysis API provided by Jacob Perkins [5] and label each tweet as: 0 : Neutral , 1: Positive 2: Neutral.

The post request to the API could be found below:

```
import requests
def SentimentalAnalysis(tweet):
        r = requests.post("http://text-processing.com/api/sentiment/", data={'text': tweet})
        label= r.json()['label']
        if label =='neutral':
        return 0
        elif label=='positive':
        return 1
        else :
        return 2
```

9. LABELLING AND CLEANING THE DATA

After adding above three features for each tweet in the file now, we now proceed to create training set. We took 30% of the file and manually labelled each tweet as Bullying or not.
(0 - False , 1- True).
We also removed the columns of the file which is not required for further training of data.
We remove the rows with Nan character.
Below is sample of how our labelled training set looks:

Offensiveness Weight	Offensiveness Level	Sentimental Analysis	Class
0	0	2	0
1.5	41.6666666666667	0	1
2.5	58.3333333333333	0	1
1	33.3333333333333	2	0

10. MACHINE LEARNING

Machine learning is a method of data analysis that automates analytical model building. It is a type of artificial intelligence (AI) that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis to predict an output value within an acceptable range.

1. Load the labelled data: We are using pandas to load our data into dataset variable.
2. Create a Validation Dataset: That is, we are going to hold back some data that the algorithms will not get to see and we will use this data to get a second and independent idea of how accurate the best model might actually be. We split the loaded dataset into two, 80% of which we will use to train our models and 20% that we will hold back as a validation dataset.We now have training data in the *X_train* and *Y_train* for preparing models and a *X_validation* and *Y_validation* sets that we can use later.

```
array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y,
test_size=validation_size, random_state=seed)
```

3. Cross-validation: Cross validation is a testing method, not a model development method. **It** is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it. In **k-fold cross-validation**, the original sample is randomly partitioned into **k** equal size subsamples. We will use 10-fold cross validation to estimate accuracy. This will split our dataset into 10 parts, train on 9 and test on 1 and repeat for all combinations of train-test splits. We are using the metric of '*accuracy*' to evaluate models. This is a ratio of the number of correctly predicted instances in divided by the total number of instances in the dataset multiplied by 100 to give a percentage.

4. Building Models :We evaluate 6 different algorithms:
   - -Logistic Regression (LR)
   - -Linear Discriminant Analysis (LDA)
   - -K-Nearest Neighbors (KNN).
   - -Classification and Regression Trees (CART).
   - -Gaussian Naive Bayes (NB).
   - -Support Vector Machines (SVM).

   We discuss the above models in next section.
5. We reset the random number seed before each run to ensure that the evaluation of each algorithm is performed using exactly the same data splits. It ensures the results are directly comparable. We reset the random number seed before each run to ensure that the evaluation of each algorithm is performed using exactly the same data splits.
6. We now get 6 models and accuracy estimations for each. This will make it easy to compare the models to each other and select the most accurate.

```
LR: 0.844167 (0.055853)
LDA: 0.844167 (0.055853)
CART: 0.844167 (0.055853)
NB: 0.832167 (0.054248)
SVM: 0.844167 (0.055853)
```

Table : Accuracy estimations of each model

10.1. CLASSIFIERS USED

Logistic Regression : It is a statistical method for analyzing a dataset and predictive modeling in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). In our case the outcomes are bullying or non-bullying. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limit.

Linear Discriminant Analysis (LDA):.Linear Discriminant Analysis is the go-to linear method for multi-class classification problems. Even with binary-classification problems, both logistic regression and linear discriminant analysis works good .LDA model estimates the mean and variance from our data for each class. LDA makes predictions by estimating the probability that a new set of inputs belongs to each class. The class that gets the highest probability is the output class and a prediction is made. The model uses Bayes Theorem to estimate the probabilities.

Classification and Regression Trees (CART): CART is a term introduced by Leo Breiman to refer to Decision Tree algorithms that can used for classification or regression predictive modeling problems.

The representation for the CART model is a binary tree. Each root node represents a single input variable (x) and a split point on that variable (numeric).The leaf nodes of the tree contain an output variable (y) which is used to make a prediction.

NaiveBayes(NB): Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems. In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. It is used mostly as a baseline method in text classification. It

is called *naive Bayes* or *idiot Bayes* because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable, assumption is made that the attributes do not interact. Two probabilities are calculated.
Class Probabilities: The probabilities of each class in the training dataset.
Conditional Probabilities: The conditional probabilities of each input value given each class value.

Support Vector Machines (SVM): Support Vector Machines are one of the most popular machine learning algorithms. The numeric input variables (x) in our data (the columns) form an n-dimensional space. For example, if you had two input variables, this would form a two-dimensional space. A hyperplane is a line that splits the input variable space. In SVM, a hyperplane is selected to best separate the points in the input variable space by their class, either class 0 or class 1. The algorithm attempts to find an optimal vector that separates them. Indeed, the algorithm would like to separate the instances by a maximum margin. This is so that unseen examples are more likely to be classified correctly.

10.2. PREDICT REMAINING DATA

From the accuracy estimations above one of the method that yields an acceptable amount of accuracy is Logical Regression model: 84.41% accurate. To get an idea of the accuracy of the model on our validation set, we run the LR model directly on the validation set and summarize the results as a final accuracy score, a confusion matrix and a classification report.

```
0.758064516129 #accuracy
[[27  7]
 [ 8 20]]         #Confusion matrix

             precision    recall  f1-score   support

        0.0       0.77      0.79      0.78        34
        1.0       0.74      0.71      0.73        28        #Classification report

avg / total       0.76      0.76      0.76        62
```

Next Step is to classify the rest of the tweets as bullying or not (bully-1 or not-0) and write the results into the file for the rest of the remaining tweets.

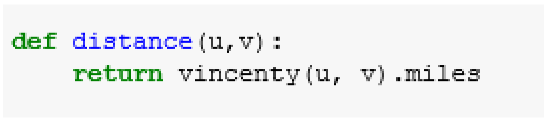
11. GEO LOCATION ANALYSIS

After building the model and training the classifier the geo location analysis could be started.
Main steps:

1) Preprocess and asses metrics for the rest of the Corpus ;
2) Classify the rest of the corpus (whether the tweet is bullying or not);
3) Read all gathered locations of the bullying tweets into the array;
4) Make hierarchical agglomerative clusterization by means of scipy package;
5) Analyze the received diagram.

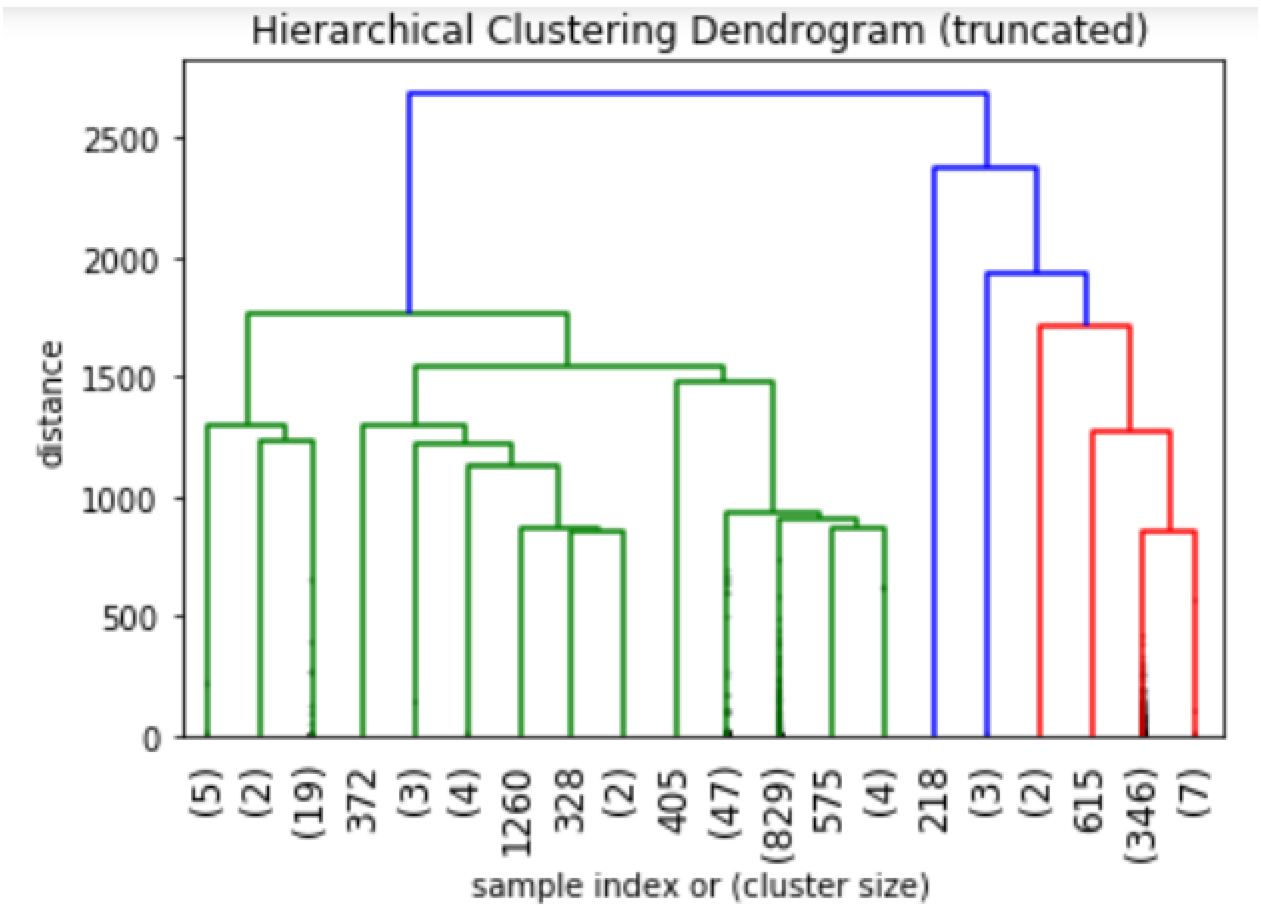
I. We run the program for the rest of the twitter corpus, which is circa 19 th. tweets.
After the first round of preprocessing there is 2400 tweets. After merging retweets we have 1077 tweets. As it was mentioned before, location could be disabled in tweeter. So on the last step we take only the tweets which were classified as bullying tweets and with existing geolocation(class=1 and Geo_Location !=nan and Geo_Location !=''). Each tweet could have more than one coordinate, so we parse the gathered coordinates and store them into one file.

Finally we receive the list of 1347 coordinates. In order to analyze the coordinates we use clusterization. The amount of clusters is not clean, we have relatively a low of data and the data itself is presented as a list of coordinates, so we decided to use agglomerative hierarchical clustering (and use Scipy [1] for that).
As a **distance metric** we define the custom function which counts the distance between two points(lat,long). The function is provided by geopy package.
The function call is provided in the picture below (). Parameters are the coordinates, between which the distance should be count.



```
def distance(u,v):
    return vincenty(u, v).miles
```

As a result of clusterization we receive the array Z, which stores the iterations of all agglomerations, made during the clusterization.
The structure of formed clusters could be seen on the dendrogram below:



So we can see that about 4 big clusters were retrieved. In order to analyze the content of the clusters we are write an algorithm for showing which exactly locations were grouped into one cluster.

The algorithm:

```
X2 = X.tolist()
X3=list(map(lambda x:[x], X2))
j=0
for c in Z:
    X3.append(X3[int(c[0])])
    ml= copy.copy(X3[len(X3)-1])
    ml.extend(X3[int(c[1])])
    X3[len(X3)-1]=copy.copy(ml)
```

Then we store information about each agglomeration operation and initial samples being connected into the file.

The file structure is the following:

first cluster	second cluster	distance	Number of samples	coordinates	iteration
547	2039	86,13829459	516	[44.933143, 7.540121][44.933143, 7.540121]	1161
1347	2038	0	515	[44.933143, 7.540121][44.933143, 7.540121]	692

The iterations with the biggest amount Number of samples and smallest distance are the most interesting for us, because they represent the areas from where the biggest amount of potential bullies came from.

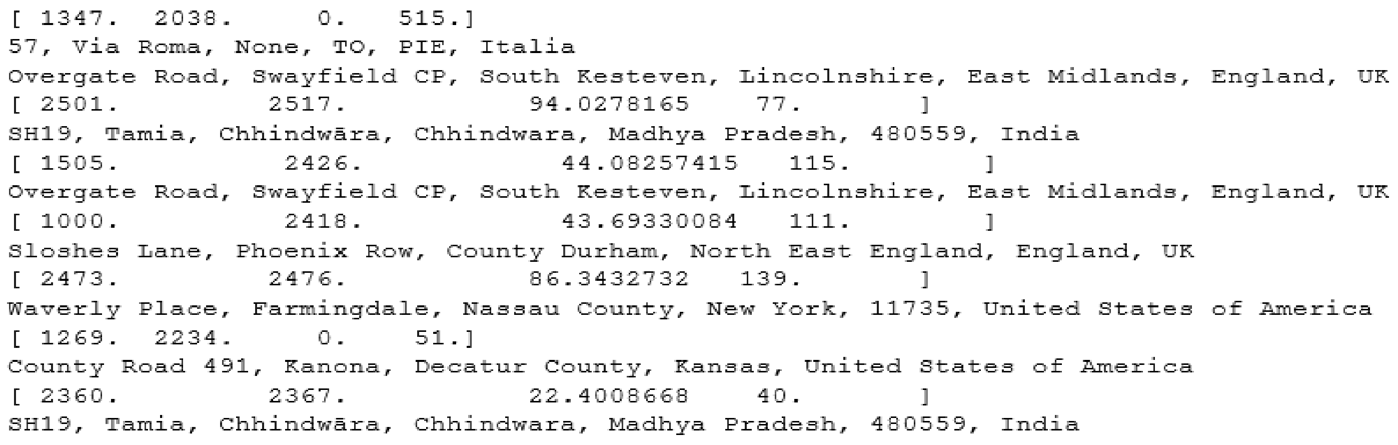
The information about the iteration could be retrieved from array Z, which is the result of the agglomeration clustering.

For example, calling Z[691] gives us:

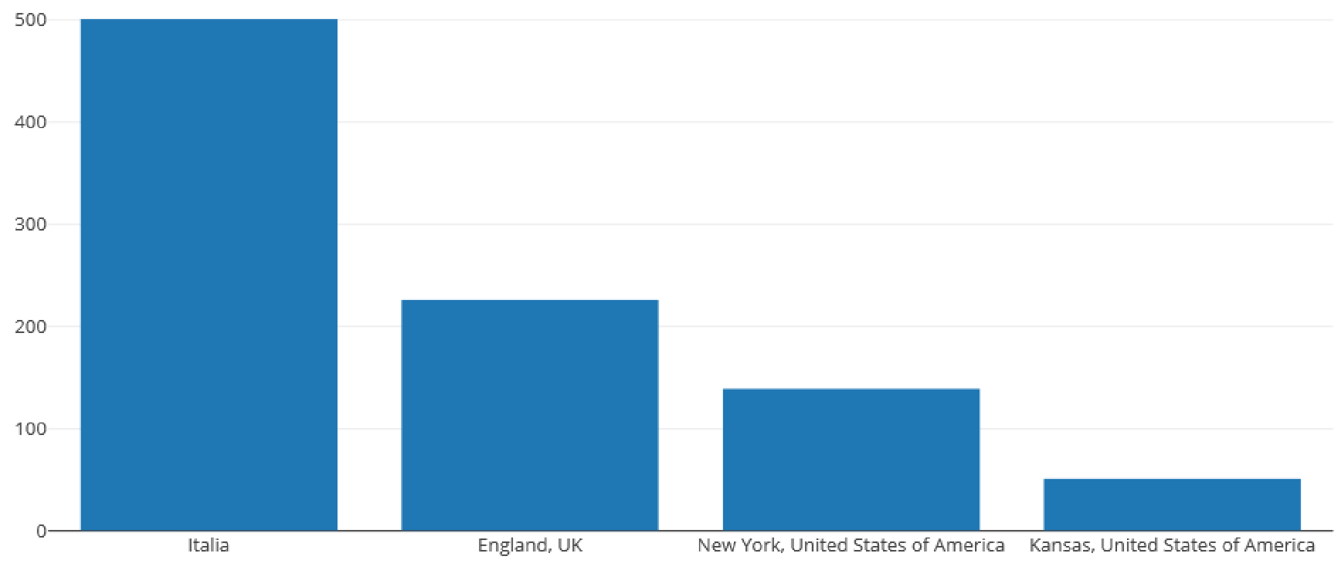
[1347. 2038.   0.   515.]

where first parameter 1347 is the first cluster being agglomerated, 2038 - the second cluster; 0 - the biggest distance between cluster, 515- the number of samples from the initial array (namely, the number of bully-users from this location).

By calling the function *getLocationByCoordinates*('22.3511148, 78.6677428') on the location from the cluster we receive the address by the coordinates:



```
[ 1347.  2038.     0.   515.]
57, Via Roma, None, TO, PIE, Italia
Overgate Road, Swayfield CP, South Kesteven, Lincolnshire, East Midlands, England, UK
[ 2501.        2517.          94.0278165    77.       ]
SH19, Tamia, Chhindwāra, Chhindwara, Madhya Pradesh, 480559, India
[ 1505.         2426.           44.08257415   115.        ]
Overgate Road, Swayfield CP, South Kesteven, Lincolnshire, East Midlands, England, UK
[ 1000.         2418.           43.69330084   111.        ]
Sloshes Lane, Phoenix Row, County Durham, North East England, England, UK
[ 2473.        2476.          86.3432732   139.       ]
Waverly Place, Farmingdale, Nassau County, New York, 11735, United States of America
[ 1269.  2234.     0.    51.]
County Road 491, Kanona, Decatur County, Kansas, United States of America
[ 2360.        2367.          22.4008668    40.       ]
SH19, Tamia, Chhindwāra, Chhindwara, Madhya Pradesh, 480559, India
```



As we can see from the picture, the biggest amount of bullying tweets were detected in Italy, which is not expected result, which could lead to the investigation of the anti-Islam activities in this region. The fact that England is on the second place is less surprising, considering Brexit and separation policy.

12. JUPYTER NOTEBOOK STRUCTURE:main.ipynb

№	**Section Name**	**Description**
1	Imports	The imports of the required
2	Program Execution	The sequence of function calls
3	Loading Dictionaries	Loads prepared dictionaries of Offensive Words, Personal Pronouns and Collective Nouns
4	Functions	Contains the service functions, used across the project
5	Third Party Libraries	Contains post request to Sentimental Analysis Library
6	Preprocessing	Preprocessing of the text and description of the tweet, deleting repeating tweets , sentiment analysis, offensiveness level calculation
7	Machine learning	Train the classifier and classify the tweets
8	Location analysis	Retrieve the location, clustering location, locations analysis

13. <u>FUTURE WORK:</u>

The results of our work have shown the potential for future development. For the simplification we were using the standard classifiers on the preprocessed data, our next step would be building the classifier which processes not the preprocessed numerical data, but the raw text, based on the model we build and tested in current work.

During the work we made attempts for applying the sarcasm detector to the text. This was out of the work scope, so we postpone the sarcasm detection to the later phases of the anti-cyberbullying project. We are planning to increase the accuracy of the model by retrieving additional features of the text.

On the long term, giving the excellent accuracy, the cyberbullying detector can be used in social researches.

14. CONCLUSION

In the current work we have built the corpus of the tweets, gathered by streaming the tweet's online. We chose two trending events as a source of potential bullying behavior. First event is refugees crisis and the second is Trump's elections. We used neutral hashtags in order to have not only haters, but also the supporters.

Closer look to the tweets content, proved the possibility to retrieve the features for cyberbullying detection. As these features we have chosen Offensiveness Level, which we count on the base offensive words mean weight and existence of Collective nouns and personal pronouns. By identifying the bullying by existence of offensive words in the same context with personal pronouns and, which is more important, collective nouns, we provide a novel approach of calculating offensiveness level. Another feature of the text, which we use, is Sentiment Analysis of the tweet.

Besides, we try to use twitter user information (namely, description) to fulfill the gap in tweet's context. Very often the user's description helps to make decision about the character of the tweet.

After manual identification of the bullying tweets and training the classifier, we were label to label the rest of the gathered tweeter corpus. Then, we were able to retrieve the location of potential bullies and find out the most aggressive regions. Based on this information, social and political researches can be conducted.

15. REFERENCES

[1] Study: 'Hate' groups explode on social media by Melanie Eversley, USA TODAY- https://www.usatoday.com/story/news/2017/02/23/hate-groups-explode-social-media/98284662/

[2]Offensive/Profane wordlist - http://www.cs.cmu.edu/~biglou/resources/ --

[3] Analysis of Cyberbullying Tweets in Trending World Events -Keith Cortis , Prof Dr.Siegfried Handschuh

[4] Detecting Offensive Language in Social Media to Protect Adolescent Online Safety-Ying Chen Yilu Zhou et.al- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.404.520&rep=rep1&type=pdf

[5] - Sentimental Analysis API . Available at : http://text-processing.com/

[6] http://whatis.techtarget.com/definition/machine-learning

[7] http://machinelearningmastery.com - Your First Machine Learning Project in Python Step-By-Step by Jason Brownlee

[8] http://machinelearningmastery.com - Classification And Regression Trees, Naive Bayes Support Vector Machines Logistic Regression and Linear Discriminant Analysis for Machine Learning, by Jason Brownlee

[9] J. Natividad, «Geopy,» [Internet Resource]. Available: https://github.com/geopy/geopy

[10] An Introduction To Cross-Validation -https://www.salford-systems.com/videos/tutorials/how-to/an-introduction-to-cross-validation

[11] Learning from Bullying Traces in Social Media - Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, Amy Bellmore