ARISTOTLE UNIVERSITY OF THESSALONIKI

SCHOOL OF INFORMATICS
Data & Web Science MSc Program

WEB DATA MINING

Assignment 1, Part 2

Spring Semester 2019

TECHNICAL REPORT

Vrochidis Alexandros, AM: 6

Liapikos Theodore, AM: 11

CONTENTS

General Informations

Assignment was developed on a Linux Mint 18.0 system running the latest version of Cinnamon Desktop Enviroment. For code was used Python 3.6.7, using as IDE Spyder 3.3.2 and Jupyter Lab 0.35.4. Each part of the Assignment includes separate code files, that implement the required functionality. Finally, each part of the code has been thoroughly inspected and operating properly and in the manner described below.

1. Data collection from Twitter and Data storage

Over a period of 15 days (from April 25th to May 9th) we collected over 160.000 tweets, using Twitter API [2] on the subject of 'Climate Change' using the following HashTags:

#climateemergency, #climatechange, #ClimateChange, #ClimateEmergency, #GlobalWarming, #ClimateAction, #Climate

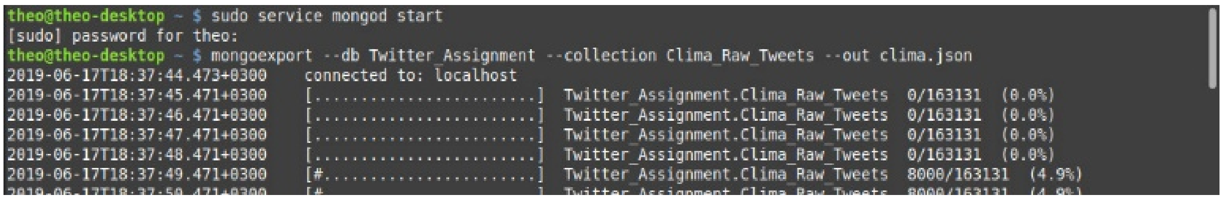
Tweets were temporally stored in a .json file ('*clima_tweets.json*') in '*tweets_data*' directory and then transferred in a local DataBase (DB), named '*Twitter_Assignment*', using MongoDB [3,4].

On this part of Assignment we used python code stored in the following files:

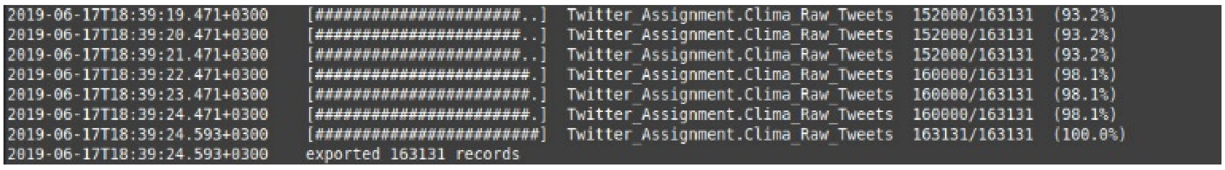
- '*files_folders_paths.py*', holds the paths to necessary directories and files.
- '*twitter_listener.py*', code responsible to establish connection with the Twitter API, download tweets and store them in the appropriate MongoDB DB.
- '*clima_twitter_credentials.py*', holds the necessary credentials needed to connect to the Twitter API.

1.1 BackUp Data Collection

The entire tweets' Collection downloaded and used for experimentation in Part 2 of the Assignment, was extracted from MongoDB and stored in a single .json file, so anyone to be able to reproduce the reported results.



```
theo@theo-desktop ~ $ sudo service mongod start
[sudo] password for theo:
theo@theo-desktop ~ $ mongoexport --db Twitter_Assignment --collection Clima_Raw_Tweets --out clima.json
2019-06-17T18:37:44.473+0300    connected to: localhost
2019-06-17T18:37:45.471+0300    [........................]  Twitter_Assignment.Clima_Raw_Tweets  0/163131  (0.0%)
2019-06-17T18:37:46.471+0300    [........................]  Twitter_Assignment.Clima_Raw_Tweets  0/163131  (0.0%)
2019-06-17T18:37:47.471+0300    [........................]  Twitter_Assignment.Clima_Raw_Tweets  0/163131  (0.0%)
2019-06-17T18:37:48.471+0300    [........................]  Twitter_Assignment.Clima_Raw_Tweets  0/163131  (0.0%)
2019-06-17T18:37:49.471+0300    [#.......................]  Twitter_Assignment.Clima_Raw_Tweets  8000/163131  (4.9%)
```



```
2019-06-17T18:39:19.471+0300    [######################..]  Twitter_Assignment.Clima_Raw_Tweets  152000/163131  (93.2%)
2019-06-17T18:39:20.471+0300    [######################..]  Twitter_Assignment.Clima_Raw_Tweets  152000/163131  (93.2%)
2019-06-17T18:39:21.471+0300    [######################..]  Twitter_Assignment.Clima_Raw_Tweets  152000/163131  (93.2%)
2019-06-17T18:39:22.471+0300    [#######################.]  Twitter_Assignment.Clima_Raw_Tweets  160000/163131  (98.1%)
2019-06-17T18:39:23.471+0300    [#######################.]  Twitter_Assignment.Clima_Raw_Tweets  160000/163131  (98.1%)
2019-06-17T18:39:24.471+0300    [#######################.]  Twitter_Assignment.Clima_Raw_Tweets  160000/163131  (98.1%)
2019-06-17T18:39:24.593+0300    [########################]  Twitter_Assignment.Clima_Raw_Tweets  163131/163131  (100.0%)
2019-06-17T18:39:24.593+0300    exported 163131 records
```

Extracted tweets' Collection was stored in '*clima.json*' file in '*0. Data Collection (json files)*' subdirectory.

2. Tweets Preprocessing and Modeling

2.1 Filtering Tweets

Next we extracted from DB the tweets to be used on the rest of the procedure [3,4]. Stored tweets were filtered to keep only the *unique* and *original* ones, so we removed all duplicates and retweets. We ended up with about 43.000 unique tweets.

From all information carried inside the tweet's structure [5], we decided to keep the following fields, in order to use them in the next steps of Assignment:

- Root level fields: 'created_at', 'coordinates', 'text', 'retweet_count', 'favorite_count'
- User level fields: 'id_str', 'screen_name', 'followers_count', 'favourites_count', 'listed_count', 'statuses_count', 'friends_count', 'location', 'time_zone', 'utc_offset'
- Entities level fields: 'hashtags', 'urls', 'user_mentions'
- Place level fields: 'country_code', 'full_name', 'coordinates'

Extracted unique and original tweets were stored in '*raw_tweets.pkl*' file in '*tweets_data*' directory.

On this part of Assignment we used python code stored in the following files:

- '*files_folders_paths.py*', holds the paths to necessary directories and files.
- '*get_original_unique_tweets.py*', holds the functionality to scan a specific MongoDB DB/Collection and return the indices of unique and original tweets.
- '*mongo_tweets_to_df.py*', uses the indices returned from previous file to extract tweets from MongoDB.

2.2 Extracting text Emoticons and Named Entities from tweets' text

Before we process and alter tweets' text, we extracted all text Emoticons and Named Entities included in it, to use them as features for the classification.

We used lists of categorized Positive and Negative Emoticons, found in literature [6,7], and then scanned each tweet's text for matching using Regular Expressions. In total found the following emoticons:

- 389 positive emoticons (~0.9% of total tweets)
- 246 negative emoticons (~0.6% of total tweets)

Next we extracted possible Named Entities included in each tweet's text using the appropriate module from NLTK library [17].

Both of these features were saved in separate columns along with the original content of the tweet. Modified tweets were stored in '*feat_extract1_tweets.pkl*' file in '*tweets_data*' directory.

On this part of Assignment we used python code stored in the following files:

- '*files_folders_paths.py*', holds the paths to necessary directories and files.
- '*emo_ne_extraction.py*', holds all functionality to execute the procedure.

2.3 Text Preprocessing and Modelling

After extracting Named Entities and Emoticons, we preprocessed tweets' text to use it in classification algorithms. The main procedure carried out by using appropriate Regular Expressions and modules from GenSim library [8]. Preprocessing contains following parts:

- Remove user mentions
- Remove URLs
- Remove HashTags

- Remove special words starting with ‘&’
- Replace repeated characters with only two of them, eg replace ‘looooooooool’ with ‘lool’
- Remove Stop-Words
- Remove punctuation
- Mark negations
- Replace each appearance of positive emoticon with the word ‘pos_emoticon’
- Replace each appearance of negative emoticon with the word ‘neg_emoticon’
- Lower characters
- Tokenization
- Mark Bigrams and Trigrams. These are not the known n-grams but groups of two or three words that must be treated as one word in order to keep their special meaning. Eg 'Aristotle University' carries different meaning than the two words separately, so it is marked as 'Aristotle_University' to be treated as a single word.
- Lemmatization and/or Stemming

Preprocessed tweets were stored in ‘*preproc_tweets.pkl*’ file in ‘*tweets_data*’ directory.

On this part of Assignment we used python code stored in the following files:

- ‘*files_folders_paths.py*’, holds the paths to necessary directories and files.
- ‘*tweet_preprocessing.py*’, holds all functionality to execute the procedure.

3. Emerging Topics and information evolution in time

Next we performed Emerging Topics analysis on preprocessed tweets. Based on literature [9] we chose to use latent Dirichlet allocation (LDA) algorithm which is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar [10]. In other words, given a corpus of documents, LDA detects number of specific topics inside the corpus, each one consisting of a group of specific words that contribute with a specific significance to the topic.

3.1 LDA Algorithm

We used the GenSim implementation of the LDA algorithm [11]. The whole procedure consists of the following steps:

1. Split tweets into 5 consecutive segments based on tweet's creation date
2. Decide which is the representative corpus for the tweets
3. Perform LDA analysis on each segment and detect individual topics
4. Calculate the similarity between all pairs of topics between any two consecutive segments
5. Detect Emerging Topics and topic's evolution in time
6. Visualization of detected Emerging Topics

In preliminary experiments we used various tweet's fields combinations as representative corpus:

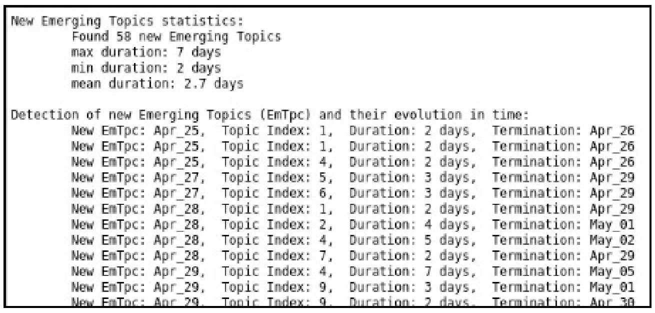
- Text field alone
- HashTags field alone
- Combination of Text and HashTags fields

We concluded that the best results delivered by using tweets' Text field alone. Corpus was normalized according to TF-IDF algorithm before use.

LDA algorithm was set to detect 10 topics on each segment. LDA model of each segment along with the related dictionary and corpus' bag of words were stored in '*LDA_data_dictionary*' file in '*emerging_topics_LDA*' subdirectory of '*tweets_data*' directory.

The calculation of similarity between two topics performed using Jaccard Distance metric. The metric threshold was set to an equivalent of 20% similarity. To be considered a topic as an emerging one it should be detected as similar to at least one other topic of a consecutive segment.

Topics' evolution in time was performed by merging similar topics of consecutive segments to one single topic. At this point system reports detailed data about new Emerging Topics detected (statistics, detection day, duration, termination day):

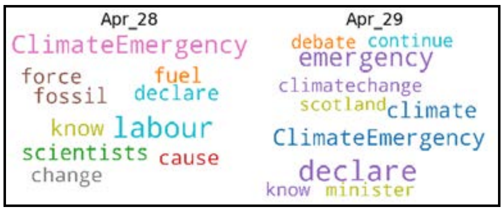


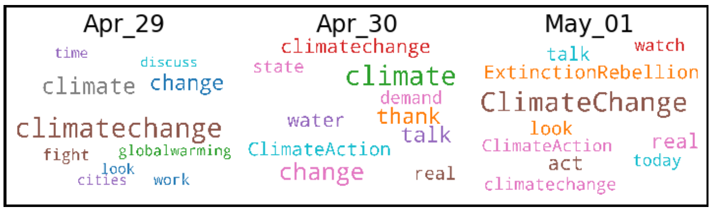
```
New Emerging Topics statistics:
        Found 58 new Emerging Topics
        max duration: 7 days
        min duration: 2 days
        mean duration: 2.7 days

Detection of new Emerging Topics (EmTpc) and their evolution in time:
        New EmTpc: Apr_25,  Topic Index: 1,  Duration: 2 days,  Termination: Apr_26
        New EmTpc: Apr_25,  Topic Index: 1,  Duration: 2 days,  Termination: Apr_26
        New EmTpc: Apr_25,  Topic Index: 4,  Duration: 2 days,  Termination: Apr_26
        New EmTpc: Apr_27,  Topic Index: 5,  Duration: 3 days,  Termination: Apr_29
        New EmTpc: Apr_27,  Topic Index: 6,  Duration: 3 days,  Termination: Apr_29
        New EmTpc: Apr_28,  Topic Index: 1,  Duration: 2 days,  Termination: Apr_29
        New EmTpc: Apr_28,  Topic Index: 2,  Duration: 4 days,  Termination: May_01
        New EmTpc: Apr_28,  Topic Index: 4,  Duration: 5 days,  Termination: May_02
        New EmTpc: Apr_28,  Topic Index: 7,  Duration: 2 days,  Termination: Apr_29
        New EmTpc: Apr_29,  Topic Index: 4,  Duration: 7 days,  Termination: May_05
        New EmTpc: Apr_29,  Topic Index: 9,  Duration: 3 days,  Termination: May_01
        New EmTpc: Apr_29,  Topic Index: 9,  Duration: 2 days,  Termination: Apr_30
```

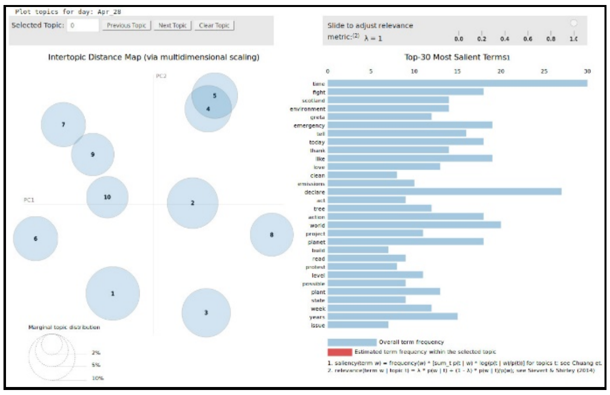
3.2 Visualization of Emerging Topics

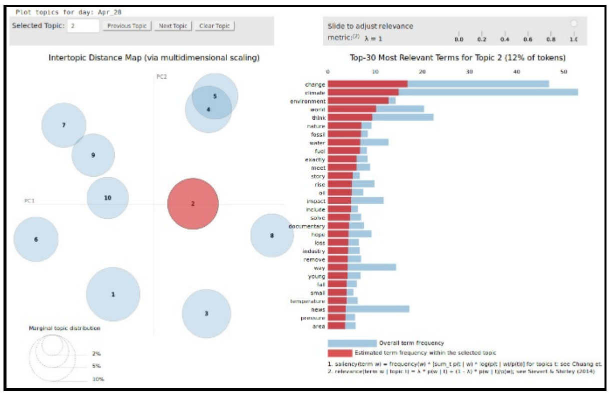
Visualization of detected new Emerging Topics was performed using WordCloud [12] representations of topics' evaluation in time, as show in representative examples below. The WordCloud representations of each Emerging Topic were stored in individual files in '*LDA_topics_wordclouds*' subdirectory of '*emerging_topics_LDA*' directory.





Another approach of visualizing topics of individual segments, was to use the pyLDAvis python library [13]. Each topic is plotted as a circle on a 2D plot, where diameter represents the significance of a specific topic against all others. Clicking one topics reveals the words that consist the particular topic and their significance as show in representative examples below.


Plot topics for day: Apr_28
Selected Topic:
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:
Intertopic Distance Map (via multidimensional scaling)
Top-30 Most Salient Terms
Marginal topic distribution
Overall term frequency
Estimated term frequency within the selected topic


Plot topics for day: Apr_28
Selected Topic: 2
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:
Intertopic Distance Map (via multidimensional scaling)
Top-30 Most Relevant Terms for Topic 2 (12% of tokens)
Marginal topic distribution
Overall term frequency
Estimated term frequency within the selected topic

On this part of Assignment we used python code stored in the following files:

- ‘*files_folders_paths.py*’, holds the paths to necessary directories and files.
- ‘*emerging_topics_LDA.py*’, holds all functionality to execute the procedure of detecting Emerging Topics, topics’s evaluation in time and visualization using WordClouds.
- ‘*pyLDAvis.ipynb*’, a Jupyter Lab file that holds all functionality to visualize segments’ topics using pyLDAvis python library.

4. Sentiment and emotion information extraction

4.1 Overall Sentiment Analysis

In this part we classify tweets as ‘Positive’ or ‘Negative’, according to Overall Sentiment they carry.

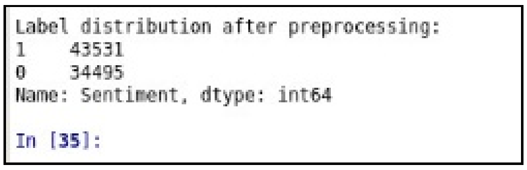
4.1.1 Preparing train dataset

For the Machine Learning models training we used a dataset obtained from Kaggle [14]. After extraction the train dataset file (‘train.csv’) was stored in ‘*Overall_Sentiment_Analysis / Kaggle_train_data*’ subdirectory. It consists of about 100.000 tweets’ text labeled as ‘0’ for negative sentiment and as ‘1’ for positive sentiment.

For training tweets’ text we followed exactly the same preprocessing procedure as with other tweets. In total found the following emoticons:

- 1850 positive emoticons (~2.4% of total tweets)
- 500 negative emoticons (~0.6% of total tweets)

Preprocessed tweets were stored at the same subdirectory. The final label distribution on preprocessed train set is the following:



```
Label distribution after preprocessing:
1    43531
0    34495
Name: Sentiment, dtype: int64

In [35]:
```

There are enough samples on both labels for the training purpose.

4.1.2 Feature selection

As tweets’ features we select just their preprocessed text which also contains reference about the appearance of Emoticons in the original text. Furthermore, preliminary experiments, using Grid Search technique [15], showed that best results delivered by using just *unigram* representation of the text and when just the *TF* part of the TF-IDF algorithm was applied.

4.1.3 Training Machine Learning models

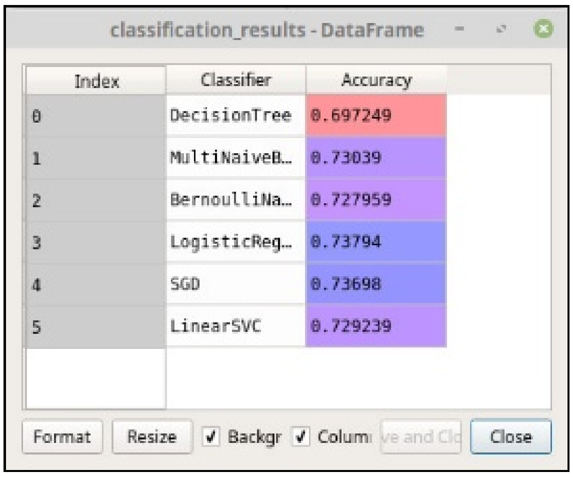
For prediction of Overall Sentiment we used the following Machine Learning algorithms:

1. Decision Tree
2. Multinomial Naive Bayes
3. Bernoulli Naive Bayes
4. Logistic Regression
5. SGD (Stochastic Gradient Descent)
6. LinearSVC
7. An ensemble technique that combines the results from all above algorithms and returns the majority decision (voting technique)

To implement the algorithms, we used the python sklearn library [16].

At preliminary experiments, using Grid Search technique [15], we set the optimal values for the various hyperparameters used by the algorithms.

Prior to training, dataset is divided into two parts, one to use for training and the other to measure the accuracy of the trained model. Trained models were stored in ‘*Overall_Sentiment_Analysis / trained_classifiers*’ subdirectory. Their accuracy data were stored in ‘*Overall_Sentiment_Analysis’* subdirectory and is shown below.



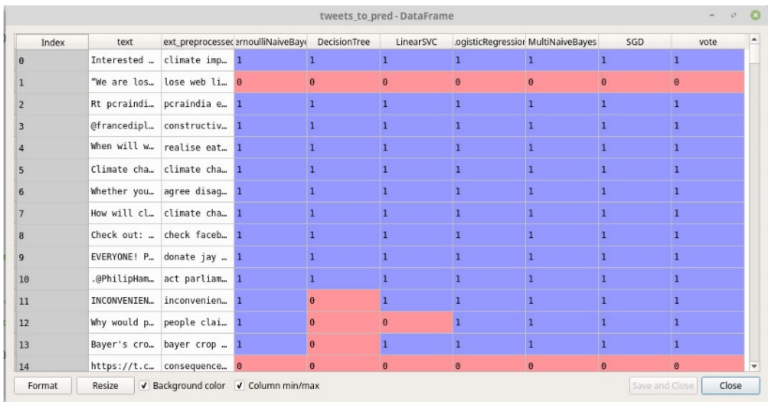
classification_results - DataFrame

Index	Classifier	Accuracy
0	DecisionTree	0.697249
1	MultiNaiveB…	0.73039
2	BernoulliNa…	0.727959
3	LogisticReg…	0.73794
4	SGD	0.73698
5	LinearSVC	0.729239

The mean accuracy achieved by the trained models was 72.77±1.38%. Preliminary experiments, run with preprocessed tweets' text without references about the appearance of Emoticons in the original text, resulted in mean accuracy of 72.12±1.41%. So the usage of emoticons as a feature only slightly improved performance of our models.

4.1.4 Overall Sentiment predictions on tweets

Next we used the trained models to perform predictions on our tweets, using tweets' preprocessed text. The prediction results were stored in a dataframe in '*Overall_Sentiment_Analysis'* subdirectory. Below is a sample of how these results look like:



tweets_to_pred - DataFrame

Index	text	ext_preprocessec	ernoulliNaiveBay	DecisionTree	LinearSVC	ogisticRegressio	MultiNaiveBayes	SGD	vote
0	Interested …	climate imp…	1	1	1	1	1	1	1
1	"We are los…	lose web li…	0	0	0	0	0	0	0
2	Rt pcraindi…	pcraindia e…	1	1	1	1	1	1	1
3	@francedipl…	constructiv…	1	1	1	1	1	1	1
4	When will w…	realise eat…	1	1	1	1	1	1	1
5	Climate cha…	climate cha…	1	1	1	1	1	1	1
6	Whether you…	agree disag…	1	1	1	1	1	1	1
7	How will cl…	climate cha…	1	1	1	1	1	1	1
8	Check out: …	check faceb…	1	1	1	1	1	1	1
9	EVERYONE! P…	donate jay …	1	1	1	1	1	1	1
10	.@PhilipHam…	act parliam…	1	1	1	1	1	1	1
11	INCONVENIEN…	inconvenien…	1	0	1	1	1	1	1
12	Why would p…	people clai…	1	0	0	1	1	1	1
13	Bayer's cro…	bayer crop …	1	0	1	1	1	1	1
14	https://t.c…	consequence…	0	0	0	0	0	0	0

On this part of Assignment we used python code stored in the following files:

- '*files_folders_paths.py*', holds the paths to necessary directories and files.
- '*Kaggle_train_set.py*', holds all functionality to preprocess and store training dataset.
- '*OS_train_algos.py*', holds all functionality to set models' hyperparameters, to train the models and to perform the predictions.

4.2 Specific Emotions Analysis

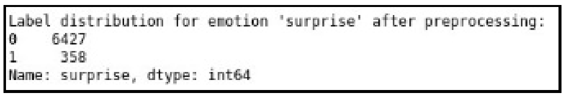
In this part we classify tweets in Specific Emotions, namely 'anger', 'disgust', 'fear', 'joy', 'sadness' and 'surprise'.

4.2.1 Preparing train dataset

For the Machine Learning models training we used the dataset provided by the Assignment. After extraction the train dataset files were stored in 'Specific_Emotions_Analysis / *SpEm_train_data*' subdirectory. There is a file for all above Specific Emotions and an extra file for each one of 'anger', 'fear', 'joy' and 'sadness'.

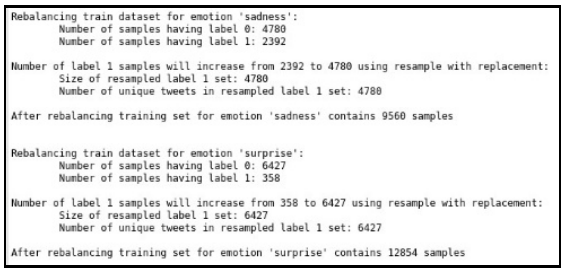
We separated the samples of each emotion into separate files and saved these files on disc. Then we preprocessed the tweets' text following exactly the same procedure as with other tweets. Preprocessed tweets was stored at the same subdirectory.

We checked the label distribution on preprocessed train sets and found that in many cases classes were extremely imbalanced, as shown in the following example:



```
Label distribution for emotion 'surprise' after preprocessing:
0    6427
1     358
Name: surprise, dtype: int64
```

As a final step we decided to rebalance the training sets, using random resampling with replacement technique, so in each set the two classes will have exactly the same number of samples, as show in the following two examples:



```
Rebalancing train dataset for emotion 'sadness':
        Number of samples having label 0: 4780
        Number of samples having label 1: 2392

Number of label 1 samples will increase from 2392 to 4780 using resample with replacement:
        Size of resampled label 1 set: 4780
        Number of unique tweets in resampled label 1 set: 4780

After rebalancing training set for emotion 'sadness' contains 9560 samples


Rebalancing train dataset for emotion 'surprise':
        Number of samples having label 0: 6427
        Number of samples having label 1: 358

Number of label 1 samples will increase from 358 to 6427 using resample with replacement:
        Size of resampled label 1 set: 6427
        Number of unique tweets in resampled label 1 set: 6427

After rebalancing training set for emotion 'surprise' contains 12854 samples
```

Rebalanced training sets were saved in the same subdirectory as the rest training files.

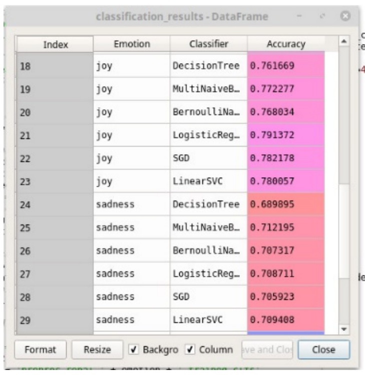
4.2.2 Feature selection

We used exactly the same features as with Overall Sentiment analysis.

4.2.3 Training Machine Learning models

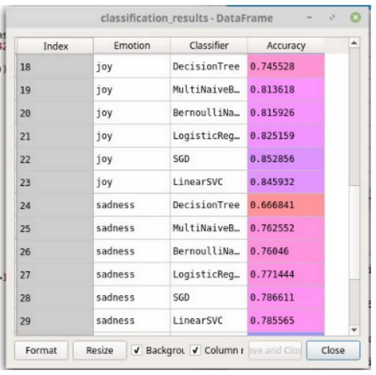
For prediction of Specific Emotions we used exactly the same Machine Learning algorithms as with Overall Sentiment analysis. To implement the algorithms, we used the python sklearn library [16]. We also used exactly the same optimal values for the various algorithms' hyperparameters.

Prior to training, each dataset is divided into two parts, one to use for training and the other to measure the accuracy of the trained model. In preliminary experiments we compared the accuracy of the models trained both on imbalanced and rebalanced training sets and found that rebalancing technique improved overall accuracy in every case, as shown in the examples below.



classification_results - DataFrame

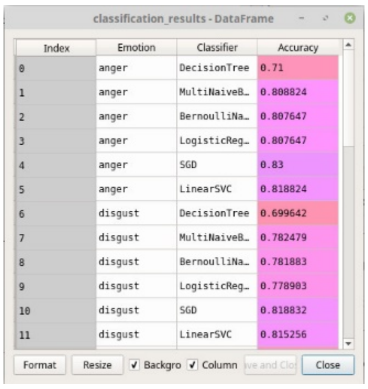
Index	Emotion	Classifier	Accuracy
18	joy	DecisionTree	0.761669
19	joy	MultiNaiveB…	0.772277
20	joy	BernoulliNa…	0.768034
21	joy	LogisticReg…	0.791372
22	joy	SGD	0.782178
23	joy	LinearSVC	0.780057
24	sadness	DecisionTree	0.689895
25	sadness	MultiNaiveB…	0.712195
26	sadness	BernoulliNa…	0.707317
27	sadness	LogisticReg…	0.708711
28	sadness	SGD	0.705923
29	sadness	LinearSVC	0.709408



classification_results - DataFrame

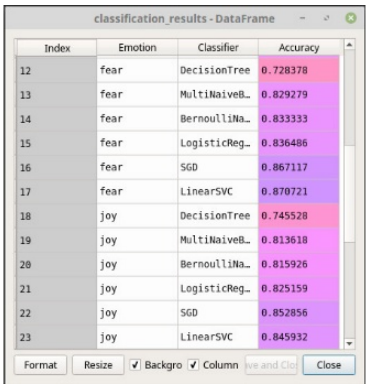
Index	Emotion	Classifier	Accuracy
18	joy	DecisionTree	0.745528
19	joy	MultiNaiveB…	0.813618
20	joy	BernoulliNa…	0.815926
21	joy	LogisticReg…	0.825159
22	joy	SGD	0.852856
23	joy	LinearSVC	0.845932
24	sadness	DecisionTree	0.666841
25	sadness	MultiNaiveB…	0.762552
26	sadness	BernoulliNa…	0.76046
27	sadness	LogisticReg…	0.771444
28	sadness	SGD	0.786611
29	sadness	LinearSVC	0.785565

So for the rest of the training procedure we used the rebalanced datasets. Trained models were stored in '*Specific_Emotions_Analysis / spem_trained_classifiers*' subdirectory. Their accuracy data were stored in '*Specific_Emotions_Analysis'* subdirectory and is shown below.



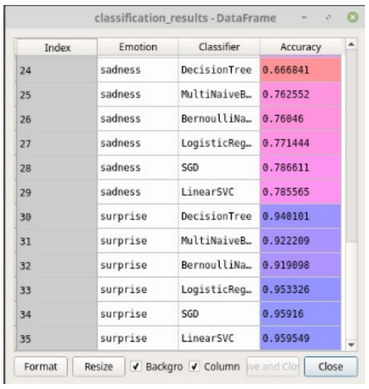
classification_results - DataFrame

Index	Emotion	Classifier	Accuracy
0	anger	DecisionTree	0.71
1	anger	MultiNaiveB…	0.808824
2	anger	BernoulliNa…	0.807647
3	anger	LogisticReg…	0.807647
4	anger	SGD	0.83
5	anger	LinearSVC	0.818824
6	disgust	DecisionTree	0.699642
7	disgust	MultiNaiveB…	0.782479
8	disgust	BernoulliNa…	0.781883
9	disgust	LogisticReg…	0.778903
10	disgust	SGD	0.818832
11	disgust	LinearSVC	0.815256



classification_results - DataFrame

Index	Emotion	Classifier	Accuracy
12	fear	DecisionTree	0.728378
13	fear	MultiNaiveB…	0.829279
14	fear	BernoulliNa…	0.833333
15	fear	LogisticReg…	0.836486
16	fear	SGD	0.867117
17	fear	LinearSVC	0.870721
18	joy	DecisionTree	0.745528
19	joy	MultiNaiveB…	0.813618
20	joy	BernoulliNa…	0.815926
21	joy	LogisticReg…	0.825159
22	joy	SGD	0.852856
23	joy	LinearSVC	0.845932



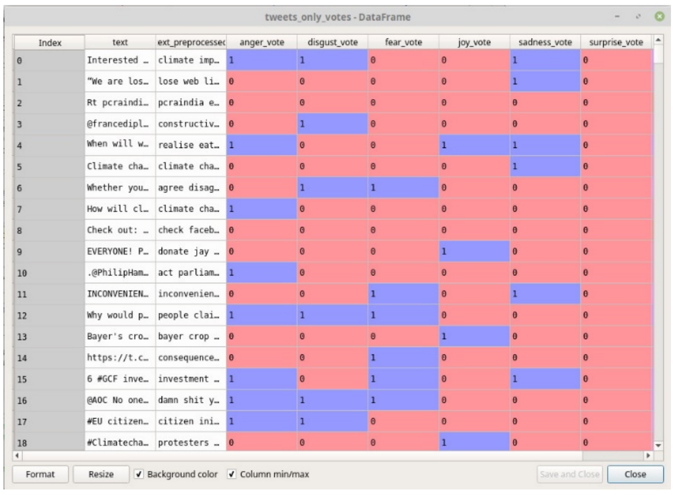
classification_results - DataFrame

Index	Emotion	Classifier	Accuracy
24	sadness	DecisionTree	0.666841
25	sadness	MultiNaiveB…	0.762552
26	sadness	BernoulliNa…	0.76046
27	sadness	LogisticReg…	0.771444
28	sadness	SGD	0.786611
29	sadness	LinearSVC	0.785565
30	surprise	DecisionTree	0.940101
31	surprise	MultiNaiveB…	0.922209
32	surprise	BernoulliNa…	0.919098
33	surprise	LogisticReg…	0.953326
34	surprise	SGD	0.95916
35	surprise	LinearSVC	0.959549

The mean accuracy achieved by the trained models was 82.23±7.26%. Preliminary experiments, run with preprocessed tweets' text without references about the appearance of Emoticons in the original text, resulted in mean accuracy of 82.16±7.04%. So the usage of emoticons as a feature only slightly improved performance of our models.

4.2.4 Specific Emotions predictions on tweets

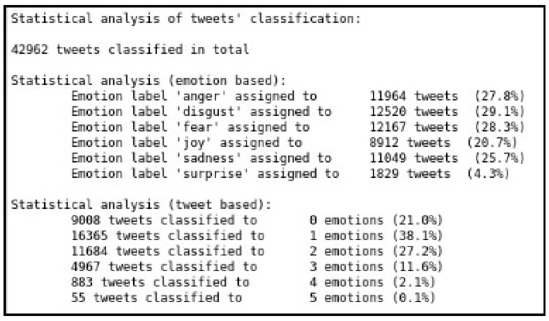
Next we used the trained models to perform predictions on our tweets, using tweets' preprocessed text. The prediction results were stored in a dataframe in '*Specific_Emotions_Analysis'* subdirectory. Below is a sample of how these results look like only for the ensemble (voting) method:



tweets_only_votes - DataFrame

Index	text	ext_preprocessec	anger_vote	disgust_vote	fear_vote	joy_vote	sadness_vote	surprise_vote
0	Interested …	climate imp…	1	1	0	0	1	0
1	"We are los…	lose web li…	0	0	0	0	1	0
2	Rt pcraindi…	pcraindia e…	0	0	0	0	0	0
3	@francedipl…	constructiv…	0	1	0	0	0	0
4	When will w…	realise eat…	1	0	0	1	1	0
5	Climate cha…	climate cha…	0	0	0	0	1	0
6	Whether you…	agree disag…	0	1	1	0	0	0
7	How will cl…	climate cha…	1	0	0	0	0	0
8	Check out: …	check faceb…	0	0	0	0	0	0
9	EVERYONE! P…	donate jay …	0	0	0	1	0	0
10	.@PhilipHam…	act parliam…	1	0	0	0	0	0
11	INCONVENIEN…	inconvenien…	0	0	1	0	1	0
12	Why would p…	people clai…	1	1	1	0	0	0
13	Bayer's cro…	bayer crop …	0	0	0	1	0	0
14	https://t.c…	consequence…	0	0	1	0	0	0
15	6 #GCF inve…	investment …	1	0	1	0	1	0
16	@AOC No one…	damn shit y…	1	1	1	0	0	0
17	#EU citizen…	citizen ini…	1	1	0	0	0	0
18	#Climatecha…	protesters …	0	0	0	1	0	0

Finally below are some statistical data about the classification of tweets on the various Specific Emotions:



```
Statistical analysis of tweets' classification:

42962 tweets classified in total

Statistical analysis (emotion based):
        Emotion label 'anger' assigned to       11964 tweets  (27.8%)
        Emotion label 'disgust' assigned to     12520 tweets  (29.1%)
        Emotion label 'fear' assigned to        12167 tweets  (28.3%)
        Emotion label 'joy' assigned to         8912 tweets  (20.7%)
        Emotion label 'sadness' assigned to     11049 tweets  (25.7%)
        Emotion label 'surprise' assigned to    1829 tweets  (4.3%)

Statistical analysis (tweet based):
        9008 tweets classified to       0 emotions (21.0%)
        16365 tweets classified to      1 emotions (38.1%)
        11684 tweets classified to      2 emotions (27.2%)
        4967 tweets classified to       3 emotions (11.6%)
        883 tweets classified to        4 emotions (2.1%)
        55 tweets classified to         5 emotions (0.1%)
```

On this part of Assignment we used python code stored in the following files:

- '*files_folders_paths.py*', holds the paths to necessary directories and files.
- '*SpEm_train_set.py*', holds all functionality to separate, preprocess, rebalance and store training dataset.
- '*SpEm_train_algos.py*', holds all functionality to train the models and to perform the predictions.

5. Statistical Analysis and in-depth data exploration

In this part we performed statistical analysis on data, gathered in previous parts of Assignment, in order to emphasize and visualize hidden aspects of the subject.

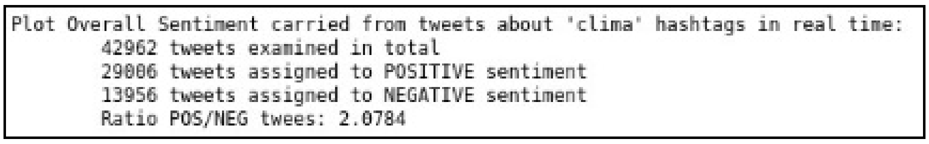
On this part of Assignment we used python code stored in the following files:

- '*files_folders_paths.py*', holds the paths to necessary directories and files.
- '*stat_graphs.py*', holds all functionality for the various calculations and for the graphs' production.

All graphs produced, were stored in '*Statistical_Analysis_Graphs*' subdirectory.

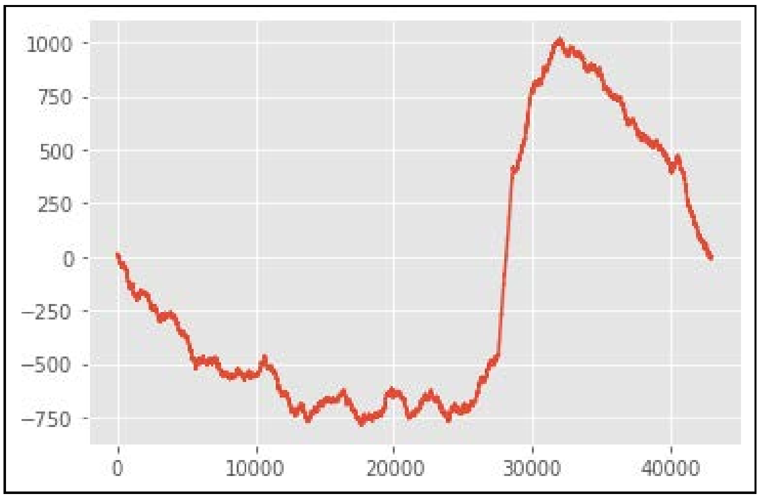
5.1 Display real time Overall Sentiment

At this graph tweets are displayed on the basis of their creation time. Distribution of Sentiments in total tweets collected in the period of 15 days is the following:



```
Plot Overall Sentiment carried from tweets about 'clima' hashtags in real time:
        42962 tweets examined in total
        29006 tweets assigned to POSITIVE sentiment
        13956 tweets assigned to NEGATIVE sentiment
        Ratio POS/NEG twees: 2.0784
```

At the graph, displayed variable has initial value 0. The value is increased by 1 for each positive tweet arrived and decreased by 2.0784 (equal to pos/neg ratio) for each negative tweet arrived. So if the above calculated ratio remains constant throughout the 15 days one should expect that the variable value should remain near 0. Otherwise the slope of the curve will be an indication that one Sentiment is superior (exceeds its average rate) to the other.



Another way of seeing the graph is that it plots the equation:

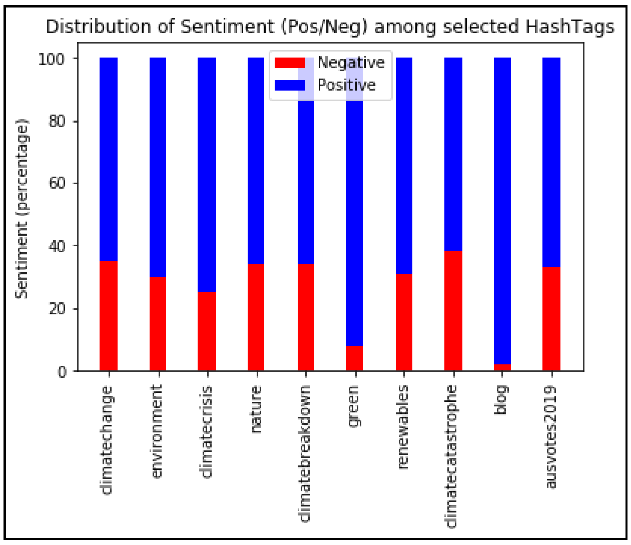
$$y = pos - 2.0784 \cdot neg$$

where pos and neg are the positive and negative tweets up to this moment.

The graph results show that in the first approximately 28,000 tweets the negative Sentiment dominates, for the next around 2000 tweets the positive Sentiment overwhelms, and then for the rest tweets the negative Sentiment dominates once again.

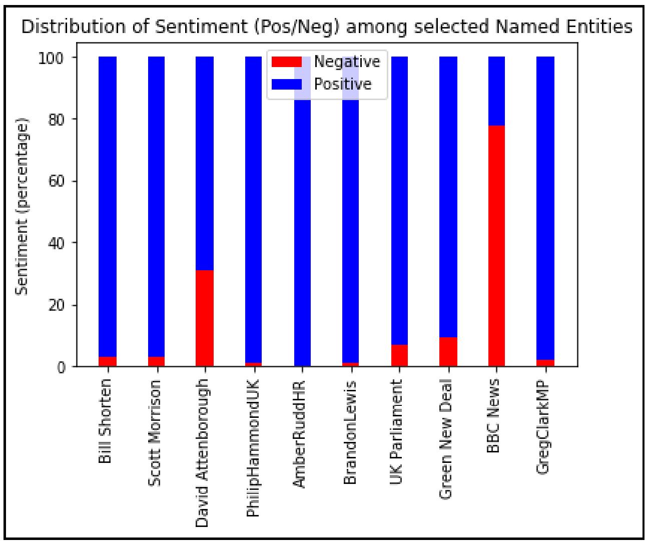
5.2 Overall Sentiment distribution among various HashTags

Positive/Negative Sentiment distribution among various (random or specific) HashTags used in tweets.



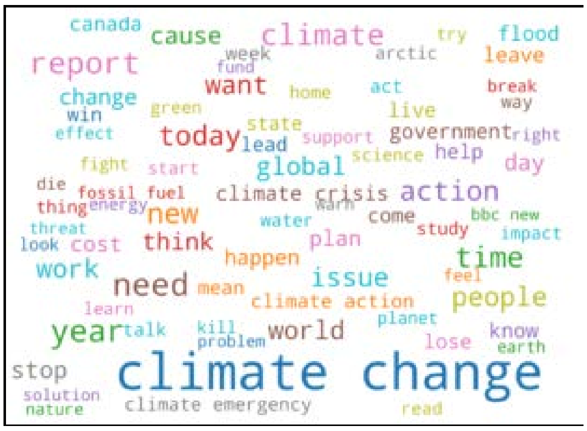
5.3 Overall Sentiment distribution among various Named Entities

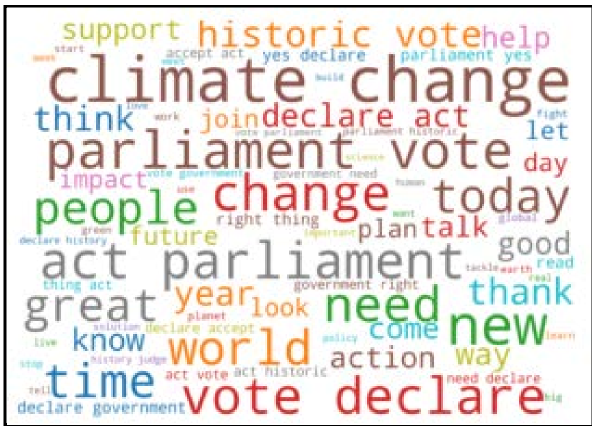
Positive/Negative Sentiment distribution among various (random or specific) Named Entities found in tweets.



5.4 WordClouds of Top Keywords in each Sentiment

Which are the words found most commonly in tweets with specific Sentiment label?

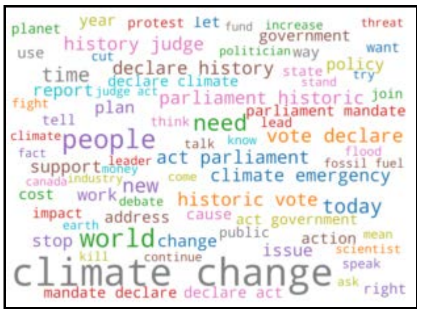


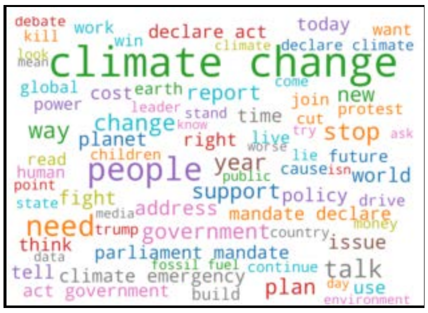


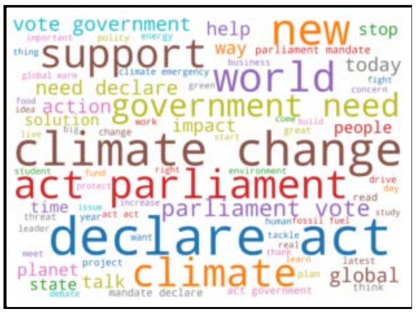
5.5 WordClouds of Top Keywords in each Sentiment

Which are the words found most commonly in tweets with specific Sentiment label?

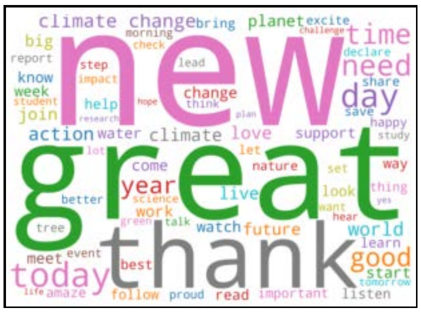
Anger. Disgust and Fear Emotions:

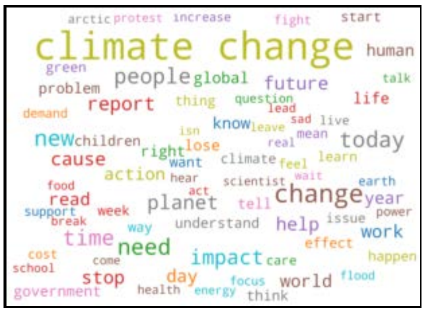


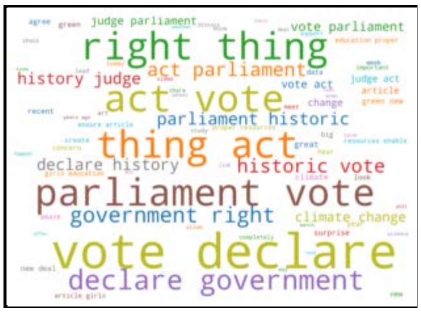




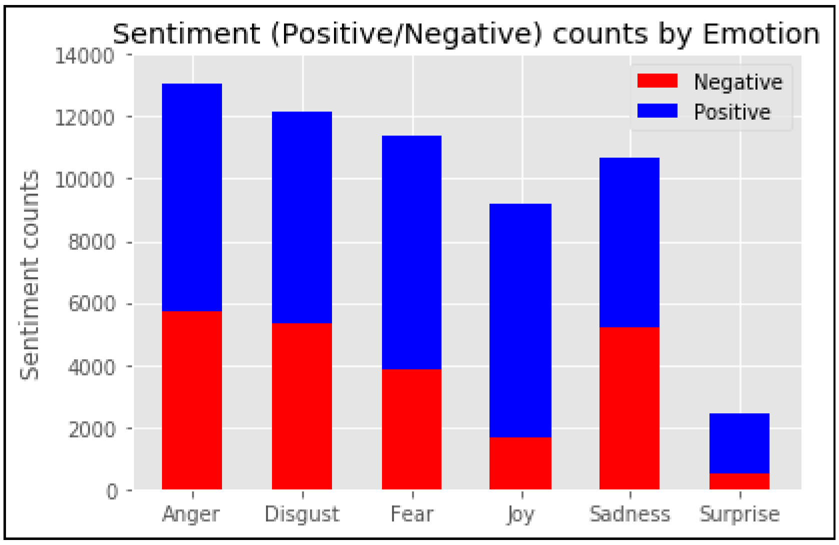
Joy, Sadness and Surprise Emotions:



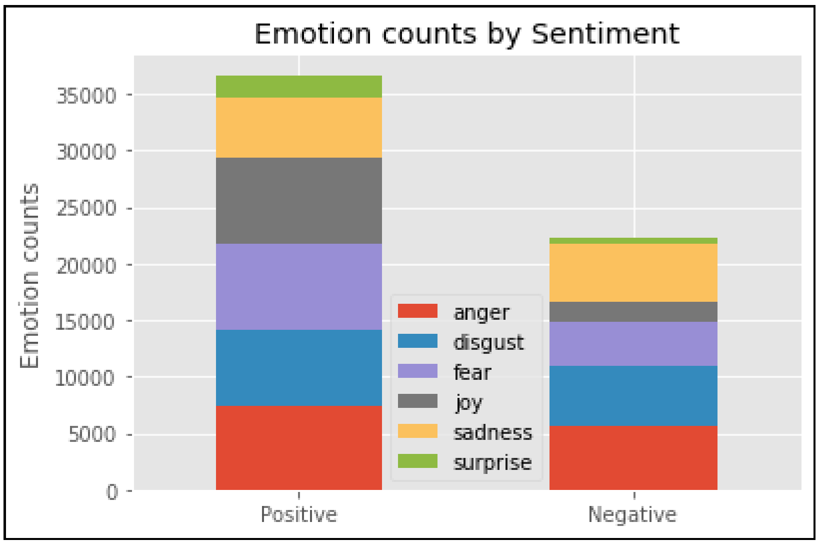




5.6 Sentiment distribution in each Emotion label

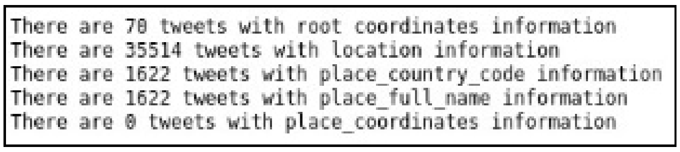


5.7 Emotion distribution in each Sentiment label


Emotion counts by Sentiment
Emotion counts
35000
30000
25000
20000
15000
10000
5000
0
Positive
Negative
anger
disgust
fear
joy
sadness
surprise

6. Geo – information extraction

In this part of Assignment we set up a Machine Learning procedure in order to extract geographic information about twitter users who don't share their exact location coordinates, based on users that they do share their location coordinates. Unfortunately in our case only a tiny portion of collected tweets had location GPS coordinates (70 tweets, <0.2%).

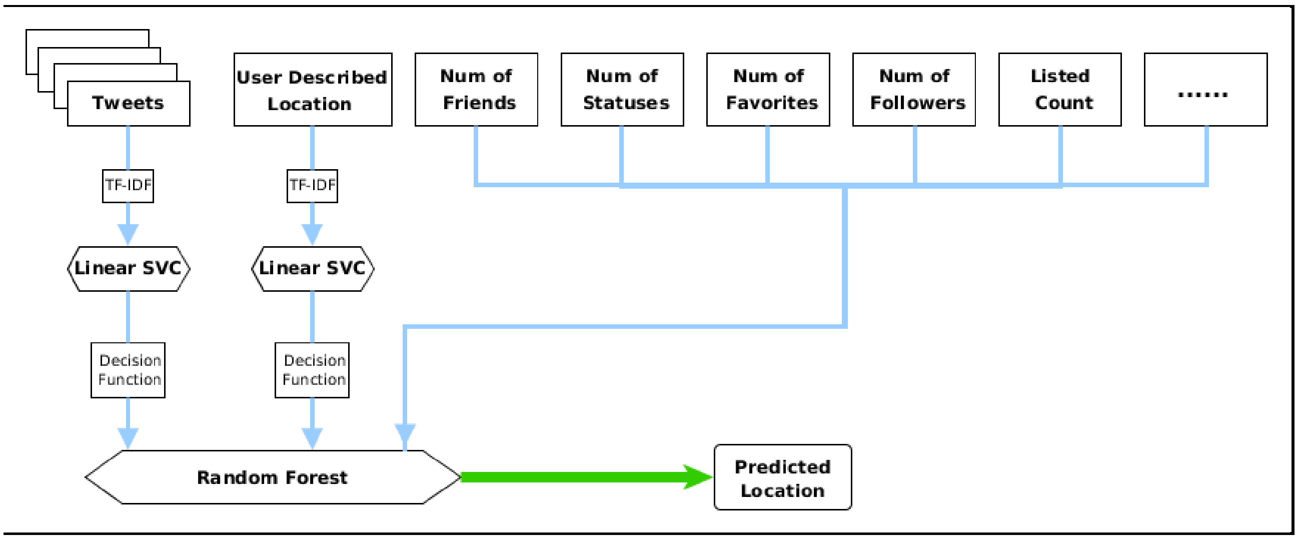


```
There are 70 tweets with root coordinates information
There are 35514 tweets with location information
There are 1622 tweets with place_country_code information
There are 1622 tweets with place_full_name information
There are 0 tweets with place_coordinates information
```

6.1 Estimate user location using Machine Learning algorithms

We set up a 2 level Machine Learning procedure. In first level two linear SVC models [18] were trained using as features the users' location information and their tweets' text respectively. In the second level the results from the first level along with other data such as follower, friends, statuses count, full name, country code etc, were used as features to train a Random Forest model [19]. This final Random Forest model could then be used to predict the geographical information for other users.

Following graph illustrates the whole procedure.



Trained models will be stored in '*Geolocation_Analysis*' subdirectory. Unfortunately, due to lack of sufficient number of samples (only 70 samples) it was impossible to train our model and to obtain reliable predictions.

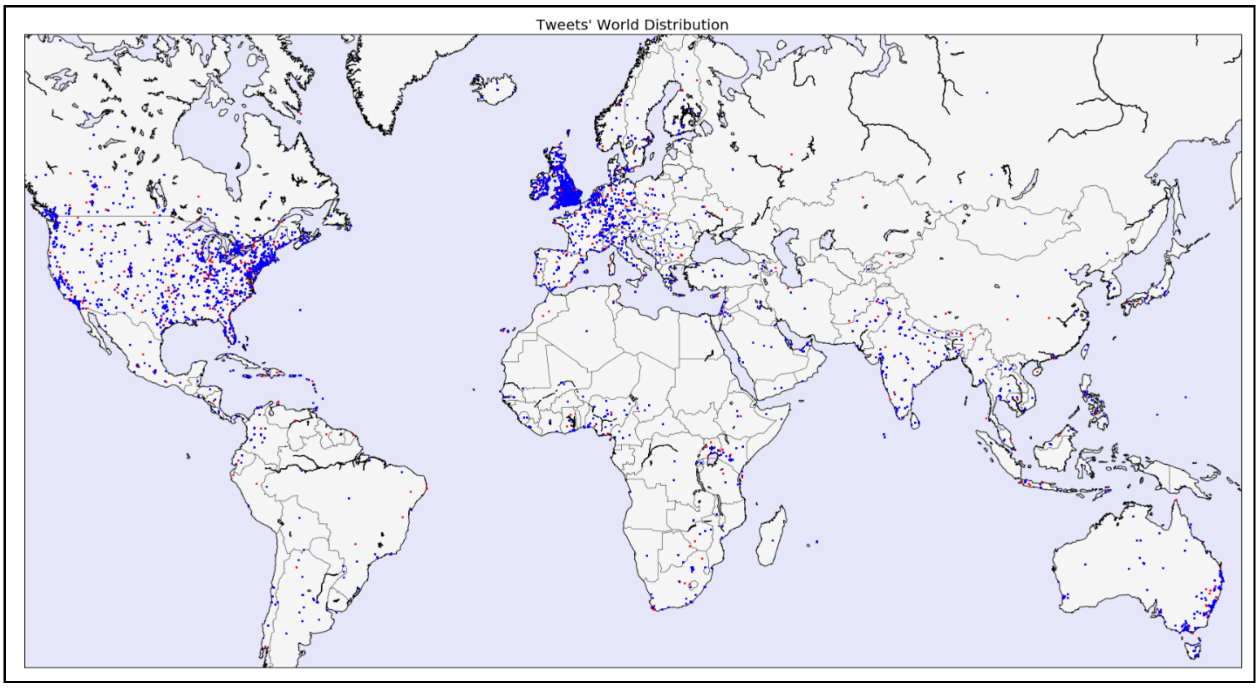
On this part of Assignment we used python code stored in the following files:

- '*files_folders_paths.py*', holds the paths to necessary directories and files.
- '*geolocate_tweets_ML.py*', holds all functionality needed for training the Machine Learning algorithms used and performing predictions.

6.2 Display users' location on a map

Here we will display each tweet as a dot on a world map, according to user location coordinates. As mentioned above only a few users shares their location coordinates. In order to obtain more tweets to display on map, we took advantage of user reported locations, although they may not be their real locations.

So in a first step we combined information from fields user_location, place_full_name and place_country_code, trying to extract information about a reported location (village, city, country etc.). In a second step we used the API from the on-line Geocoder service OpenCage Geocoder [20], which turned the locations into coordinates. Then we displayed the reported location coordinates on a world map. In addition, we separated displayed tweets according to their Overall Sentiment (blue color for Positive and red color for Negative Sentiment), using data obtained from part 4 of the Assignment.

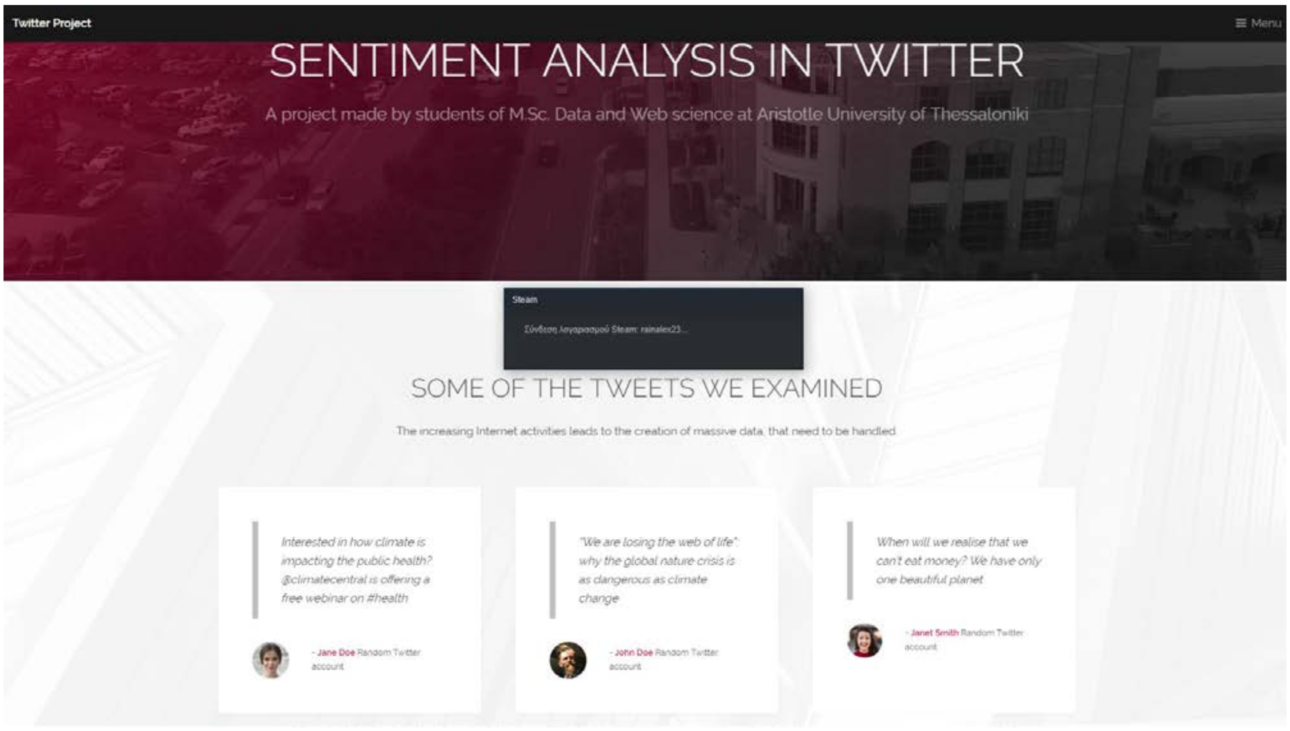


All data from this part were stored files in '*Geolocation_Analysis*' subdirectory. On this part of Assignment we used python code stored in the following files:

- '*files_folders_paths.py*', holds the paths to necessary directories and files.
- '*geolocate_tweets_API.py*', holds all functionality for to connect and use the on-line Geocoder service to turn locations into coordinates.
- '*geolocate_tweets.py*', holds all functionality to separate tweets according to their Sentiment and to display their coordinates on a world map.

7. Creating a web application to display the analysis results

In order to present the analysis results we created a web site. The website contains two pages. The first page is showing what we did in the project and the results presented by images. The second pages show some information about us. The web page is written in html and is presented below.



SENTIMENT ANALYSIS IN TWITTER

A project made by students of M.Sc. Data and Web science at Aristotle University of Thessaloniki

Steam

Σύνδεση λογαριασμού Steam: rainalex23...

SOME OF THE TWEETS WE EXAMINED

The increasing Internet activities leads to the creation of massive data, that need to be handled

Interested in how climate is impacting the public health? @climatecentral is offering a free webinar on #health

- Jane Doe Random Twitter account

"We are losing the web of life": why the global nature crisis is as dangerous as climate change

- John Doe Random Twitter account

When will we realise that we can't eat money? We have only one beautiful planet

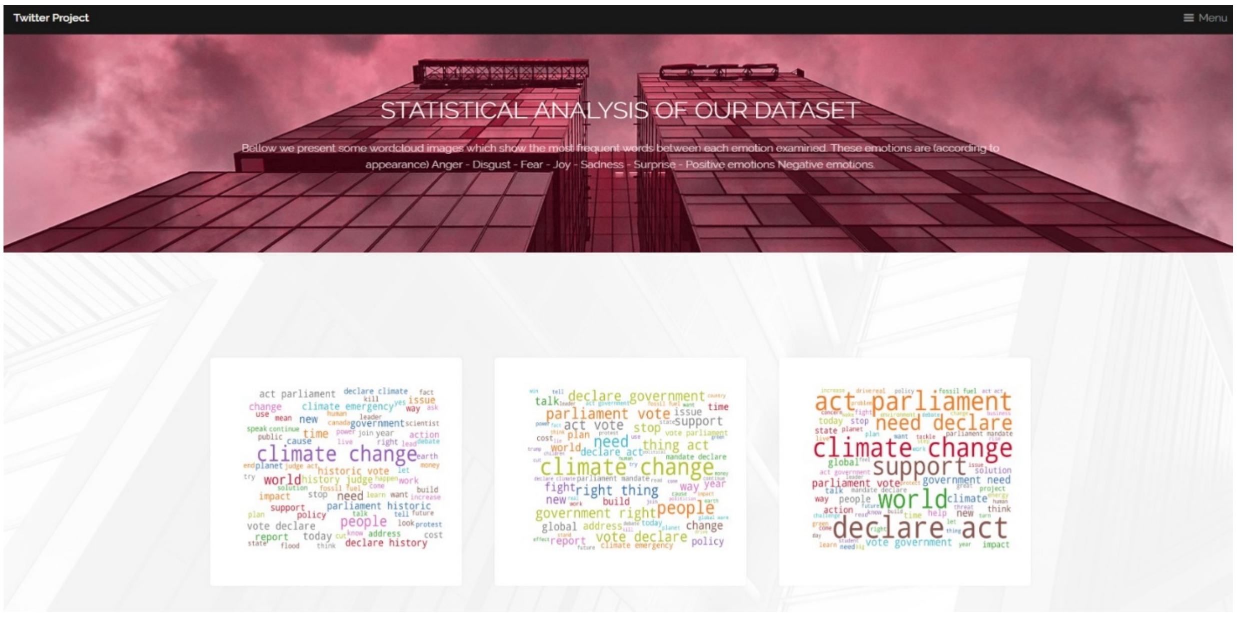
- Janet Smith Random Twitter account

At first, we show 3 of the tweets we examined. The photos of people are unreal and it is used just for better appearance of the results. After showing them, we present some WordCloud images which show the most frequent words between the tweets we examined. This section is about emerging topics and information evolution in time. After that we show the Top 30 relevant terms for topic ClimateChange and bellow how many times we see some terms that contribute to some interesting conclusions and then some statistics.

These statistics show the distribution of the sentiments that some specific terms had in percentage. Two stacked bars we present, show the emotion counts by sentiment and the opposite. These stacked bars are followed by a diagram which shows the tweets in horizontal axis and the sentiment score they had (normalized) in the period of their extraction. At last section of the web site we present some geolocation information shown previously on this paper. It shows the locations where these tweets were written and they are divided in positive (blue dots) and negative (red dots) sentiment features.

Files holding the code responsible for the implementation of the site shown above were stored in '*Site Implementation / Website'* subdirectory.

We also used flask in order to build a web app that is accessible by just run the app.py file inside the flask folder. Please make sure that you have installed flask in your computer. If you don't have it, you can install it by typing pip install flask in your command prompt. In order to do that we had to build the python code and then make some changes our html file that had to do with the images (we had to relocate them according to flask rules). We also change the cascade stylesheet path, because it was relocated. A representative image that shows how our website is structured is showed below.



STATISTICAL ANALYSIS OF OUR DATASET

Bellow we present some wordcloud images which show the most frequent words between each emotion examined. These emotions are (according to appearance) Anger - Disgust - Fear - Joy - Sadness - Surprise - Positive emotions Negative emotions.

Files holding the code responsible for the implementation of the site shown above, using Flask, were stored in ‘*Site Implementation / Flask’* subdirectory.

References

1. Course materials

2. https://developer.twitter.com

3. https://docs.mongodb.com/manual/tutorial/getting-started/

4. https://www.tutorialspoint.com/mongodb/

5. https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json.html

6. Hao Wang, Jorge A. Castanon - Sentiment Expression via Emoticons on Social Media, Silicon Valley Lab, IBM, USA. 2015

7. Amritkumar Tupsoundarya, Padma S. Dandannavar - Sentiment Expression via Emoticons on Social Media: Twitter, International Journal for Research in Applied Science & Engineering Technology, 6, 2018

8. https://radimrehurek.com/gensim/parsing/preprocessing.html

9. Atefeh, F., & Khreich, W. - A survey of techniques for event detection in twitter, Computational Intelligence, 31(1), 132-164, 2015

10. https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

11. https://radimrehurek.com/gensim/models/ldamulticore.html

12. https://www.datacamp.com/community/tutorials/wordcloud-python

13. https://github.com/bmabey/pyLDAvis

14. https://www.kaggle.com/c/twitter-sentiment-analysis2/data

15. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

16. https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

17. https://www.nltk.org/book/ch05.html

18. https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

19. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

20. https://opencagedata.com