Queen Mary University of London

Course: EECS784P

Data Analytics

Coursework: Tweets Classification for sentiment Analysis using NLP techniques

Students Full Name:

Abderrahim BOUTORH, student ID: 170676973

Ilyes Benmansour, student ID: 170723004

Sebastian Creighton, student ID: 130072030

***Programme:* Internet of Things (Data)**

2017/2018

Table of Content

1. Introduction

London has been struck with a knife and gang crime spread in the past weeks with the death toll exceeding 50 in the capital. Many have blamed the mayor of London Mr Khan for the devastating rise in violent gang crime and [1] some are not pleased with him such as LBC’s political editor Theo Usherwood as Mr. Khan admitted that he has not met families suffering from outcome of increased gang crime with London’s death rate overtaking New York for knife and gang crime, for the first time, with similar size populations and police forces. [2]

Meanwhile, the widespread and the adoption of social media has opened the door for numerous social media users to criticise him and heavily advised him to resign from the London mayor position [3]. Nowadays, social media uploaded content is widely used either by governments and data-driven company to extract insights and predict people’s orientation and opinions for both commercial and political usage.

In this work, we are interested in classifying people into two categories based on their opinion about the mayor of London especially after the lastly happening events in the capital and how Londoner’s opinion could possibly affect his election next year. For that purpose, we collected data from Londoner's tweet about Sadiq Khan, the mayor of London, from April 1st to April 10th, 2018. In fact, the reason for choosing this period of time is because it has witnessed a lot of knife crimes in the capital which makes the collected data strongly relevant to the context.

Starting from the hypothesis that, from his election, people were fully confident in his capacity in doing his job, but with the taken decisions and the recent happening events in London, his popularity has decreased dramatically and a lot of Londoners have changed their opinions about him and want him to resign. In fact, it is important to note that people’s opinion could be related to many other factors other than crimes, for instance, illegal immigration, London infrastructure, transportation, quality of life...etc, but we focus only on the factor crime because it is the hottest topic in the last weeks.

In order to confirm our hypothesis, we collected data of Londoners from Twitter. We have used Natural Language Processing NLP techniques and other powerful Machine Learning algorithms to analyse people tweets and their opinions in order to classify them into two categories: those who are with the mayor, and those who are not.

This report is structured into four sections. In the first section, we present the machine learning classification techniques used in this work and the theory behind. We test three different algorithms for classification: Logistic regression, Naive Bayes, and Support Vector Machine SVM. Next, we present the tools and libraries of python used for the purpose of this work and the metrics that allow us to assess and evaluate the models. In section three we discuss all the steps taken in this work starting from data extraction, data exploration, cleaning, model training and testing, selecting the best classification model and finally predicting the popularity of the mayor using the best performing model. Finally, we discuss the challenges that we faced and the successes that we achieved, and we present some possible extensions for this work and we finish by a conclusion.

2. Machine Learning classification techniques: Supervised learning and classification

2.1. Logistic Regression LR

Logistic regression is one of the simplest supervised learning and discriminative predictive analysis technique for binary or multi-class classification. It belongs to the family of classifier called exponential or log-linear classifiers [4] and widely used in text categorization and sentiment analysis. This technique of classification is based on finding the decision boundaries that separate the classes from each other, i.e. finding the best weights ***w*** for the decision boundaries that separate best the data set into multiple classes. Depending on the form of the selected features ***x***, the decision boundaries could take any form, for instance, linear, curved lines, or a hyperplane. However, it is important to understand that when talking about the complexity of the model, i.e. the complexity of the feature space, the learned decision boundaries are always linear with respect to the weights ***w***, but are not necessarily linear with respect to the features ***x***, i.e. features ***xi*** could have any complexity order (linear, quadratic, or any other complexity) [5].

Basically, logistic regression algorithms try to find the best weights ***w*** that minimizes the cost function (error) , i.e. maximizes the distance between samples of each class and the decision boundary as much as possible. This could be done either by computing the closed form solution analytically, i.e. compute the cost function, differentiate with respect to ***w***, and find the ***w*** that minimizes the cost function (***w*** is a vector). Or, using an iterative solution, such as gradient descent algorithm, to find the best weights iteratively.

Logistic regression can be used in either binary classification or multi-class classification [6]. For binary classification, LR estimates ***P(y|x)*** by extracting some set of features from the input, combining them linearly (multiplying each feature by a weight and adding them up), and then applying a function to this combination. The most used function to predict the class is the sigmoid function showed in equation (1).

$$f(x, w) = y = \frac{1}{1 + e^{-w^T x}} \textbf{...(1)}$$

where ***w*** is the weights vector and ***x*** is the features vector (inputs).

Due to the expected complexity of the feature space, the iterative gradient descent is the mostly used algorithm to calculate the weights **w** because of the difficulty or even the incapacity to find the closed form solution when the complexity of the feature space is very high. The gradient descent algorithm is shown in equation (2)

$$\mathbf{w}' := \mathbf{w} - \alpha \sum_i (\sigma(\mathbf{w}^T \mathbf{x}_i) - y_i)\mathbf{x}_i \ldots(2)$$

where ***w*** is the weights vector, **α** the learning rate, ***xi*** the features (input) vector, ***yi*** is the target value, and ***σ()*** the sigmoid function.

One of the issues that could appear while using this technique is overfitting. It means that the model learns perfectly how to separate the classes of the trained data set, but generalize badly on the test data set. To avoid this issue, a regularization term is added to the cost function and multiple parameters are used, for example, regularizer value and type, for cross-validation. This term penalizes large weights and makes the model predict more accurately. These regularization parameters are chosen based on the type of data and are chosen based on multiple experiments. In the rest of this coursework, we will highlight the parameters that we played with in order to maximize the accuracy.

2.2. Naive Bayes classifier

Naive Bayes classifier is also widely used classifier in text categorization and sentiment analysis [7]. It is a special case of logistic regression that is based on conditional probabilities and Naive Bayes theorem to find the weights ***w***, where each feature is assigned with a probability. In general, this classifier relies on a straightforward technique based on text representation as *"Bag of words"*, i.e. each sentence in a document (tweet in our case) is represented by a group of words. The most well-known technique for this purpose is called *tokenization* that we will discuss in the pre-processing section.

The Naive Bayes classifier receives as input a document (tweet) ***d***, a number of classes ***C*** (positive class and negative class in our work), and a training set of labeled documents (tweets) in the form of tuples ***(di, ci)*** where ***di*** is the tweet and the ***ci*** is the class of this tweet.

Naive Bayes classifier gives as output the class of the tweet by computing the probability of how much likely this tweet is of class ***ci*** given the tweet by performing Bayes rule (equation 3), i.e. it tries to maximize the posterior probability ***p(c | d)***, which means maximizing ***p(d|c) . p(c)*** in order to give accurate prediction.

$$P(c \vee d) = \frac{P(d \vee c) * P(c)}{P(d)} \textbf{...(3)}$$

The Naive Bayes learning works as follow. It first represents each document (tweet) ***d*** as a set of features ***{x1, x2, …, xn}***, where ***xi*** is a word, assign each word a probability (weight) using one of the well-known techniques that we will discuss later, for instance, **term frequency-inverse document frequency tf-idf** or word count frequencies, and then tries to maximize the posterior probaility ***p(d | c)*** with respect to the class ***c***, i.e. computes

$$\text{MAPc} = \text{argmax } p(\, c \mid d) = \text{argmax } P(d \mid c)*P(c) = \text{argmax } P(x\ 1\ , x\ 2\ ,..., x\ n \mid c)*P(c)$$

where, for word count frequecny technique, the ***p(cj)*** is computed by dividing the probability of picking a document of class ***cj*** by all the documents of class ***cj*** (for example, if we have 5000 documents in class ***c1***, then ***p(c1) = 1/5000***) and the likelihood ***p(xi|cj)*** is equal to the number of words of class ***cj*** that a document of class ***cj*** contains divided by the number of all words that the same document (of class cj) contains. [8]

2.2.1. Limitations and assumptions of Naive Bayes classifier in text classification:

The Naive Bayes classifier assumes that the order of appearance of words in a tweet does not matter [8]. That means each word is assigned a weight independently of the context and the other words of the tweet. It can happen that It gives a positive word a high weight and this positive word is preceded by a negation, then the Naive Bayes classifier doesn't take into account this negation while training the model. In this case, the meaning of the sentence and the context cannot be fully understood by the machine and the classification is only done based on the polarity and weights of the words separately which can be misleading in many cases. In fact, this limitation is not only unique to naive Bayes classifier, but many other classifiers suffer from this limitation.

2.3. Support Vector Machine for classification

Support Vector Machine SVM are ML algorithms used for classification and regression problems. SVC is a discriminative classifier that learns the boundaries between two or more different classes. The power of these classifiers is their ability to learn independently of the dimensionality of the feature space which makes them widely used when the dimension of the feature space is very high. The SVC contains many parameters that play an important role when using these techniques. The most important ones are the kernel which specifies the shape of the hyperplane that separates the classes, it can be chosen as linear, radial basis function rbf, or polynomial, and the regularization parameter C that controls the tradeoff between smoothing the decision boundary and classifying the training points correctly [9], i.e. a large value for C means lower regularization and small value for C means greater regularization.

3. Features extraction in text classification

The techniques used for features extraction for text classification play a major role in obtaining better performance and accuracy while chosing the best model. From these techniques, we will be using the term frequency-inverse document frequecy tf-idf because is the mostly used technique and showed to perform very well for this type of analysis [10].

3.1. Features weightening technique: Term frequency – Inverse document frequency tf-idf

The term frequency- Inverse document frequency tf-idf is a measure of how frequently a word appears in a collection of documents in addition to how frequently it appears in one specific document. This reflects how much information the word carries by measuring how common it is across a collection of documents.

It exists many variants on how the tf-idf is calculated. One of the simplest methods is to count the frequency of appearance of a word tf in a document, i.e. the number of times the word appears in document d, and the inverse frequency of appearance of the word in a collection of documents. Therefore, if a word is associated with a high weight that means that it appears many times in one document and is rare in other documents. This can be translated to this word is very informative. In contrast, the low value of weights means that a word did not appear too much in one specific document and it appears many times in all the documents, so it is less informative. As a conclusion, the weights tf-idf tend to filter the most common words in all the documents [11].

Mathematicaly speaking, the most used tf-idf variant is shown in the equation 4:

$$\text{tf-idf}(t,d) = \text{tf}(t,d) \times \text{idf}(t) \ldots \quad (4)$$

Where **tf** means **term-frequency** while tf–idf means term-frequency times **inverse document-frequency** which is calculated by the equation 5 [12]:

$$\mathrm{idf}(t) = log\frac{1+n_d}{1+\mathrm{df}(d,t)} + 1 \quad ...(5)$$

4. Evaluation metrics:

4.1. Accuracy, Precision, Recall, and F1 score

In order to assess the accuracy of the models and validate them, the following metrics are used:

- **Accuracy:** this metric describes how well the model can predict new sample of observations. Basically, based on the trained model, the algorithm is applied to the test data set and compares the samples that have been classified correctly and those who have been misclassified. It is computed by dividing the sum of samples that have been correctly classified over the total samples.

- **Precision:** these metrics measure the proportion of samples of one class that has been classified correctly over the those who have been classified correctly in addition to those who have been misclassified of the same class. In other words, it computes TP /(TP+FP).

- **Recall:** this metric is similar to the precision, but instead of dividing on TP+FP, we divide on TP+FN.

- **F1 score:** The F1 score is basically derived from the precision and the recall metrics. It is a measure of test's accuracy. It is equal to: 2*(precision * recall)/ (precision + recall) [13]

Following the preceding metrics, the best model that we will choose is based on the best values of these metrics we will obtain.

4.2. Confusion Matrix

This matrix allows the visualisation and the evaluation of the learned algorithms [14]. Each row of the matrix represents the samples in a predicted class while each column represents the samples in an actual class [14]. Its components are the following:

- **True Positive (TP):** sample of positive class classified positive
- **True Negative (TN):** sample of negative class classified negative
- **False Positive (FP):** sample of negative class classified positive
- **False Negative (FN):** sample of positive class classified negative

5. Python packages and tools

5.1. Sci-kit learn

Sci-kit learn is a Python open source library for Machine Learning applications. It is a powerful widely used library for either prototyping or production for both commercial and educational purposes. This library provides all the necessary tools and techniques for data loading, supervised and unsupervised learning, dimensionality reduction, data modelling, cross-validation, and model selection and evaluation.

Some popular groups of models by Scikit-learn include:

- Datasets: for test datasets and generating datasets with specific properties for investigating model behaviour.
- Dimensionality Reduction: reducing the number of attributes in data for summarisation, visualisation and feature selection e.g. Principal Component analysis
- Feature extraction: defining attributes in image and text data.
- Feature selection: identify meaningful attributes from where to create supervised models [15].

5.2. Natural Language Tool Kit NLTK

The Natural Language Toolkit is a Python facility for program modules, data sets, tutorials, and exercises, for symbolic and statistical natural language processing. It was designed for purposes of projects and demonstrations.

NLTK has a large collection of modules that barely depend on each other, with little depth. Core modules define basic data types used with the toolkit. The other modules are task modules, for an individual NLP task. An example is the nltk.parser which its role is to parse or derive the syntactic structure of the sentence; whilst the nltk.tokenizer's role is to tokenize or divide the text into basic parts [16].

The tools used from the nltk library are the following:

- **word_tokenize:** this methods from nltk.tokenize module divides strings into a list of words, i.e. list of substrings [17]
- **stopwords:** the list of stop words provided by the nltk.corpus module.
- **twitter_samples:** the tweets of the nltk library
- **SentimentIntensityAnalyzer [18]:** this method from the nltk.sentiment.vader module

5.3. Tweepy

Tweepy is a python module based on Twitter REST API that allows querying easily the data from Twitter. To use the module, an app has to be registered with Twitter apps (link for twitter apps [21]), 4 pieces of information are given from Twitter and have to be used with Tweepy for authentication and authorisation (Consumer Key (API Key), Consumer Secret (API Secret), Access Token, Access Token Secret).

These pieces of information are not provided in the code since a personal account was used to register the Twitter App.

6. Project steps

6.1. Data extraction

The dataset is a set of tweets in JSON format that has been downloaded from twitter using the twitter APi. The python module “tweepy” has been used to query the tweets with python which makes the process of collecting tweets much more easier than using directly the REST API provided by Twitter.

For the training and testing datasets, we used two different datasets and then compared them in terms of accuracy, precision, recall, and f1 score obtained for each dataset and for each used classifier. These datasets are:

- **NLTK’s sample tweets:** there are 10000 sample of tweets available in the NLTK library that can be used for sentiment analysis since they are already labeled as positive or negative (5000 for each category) based on unknown criteria. The problem with this datasets is it has been downloaded in 2014 which is quite old assuming the fact the social network dialect has changed. Another point to consider is that the tweets belong to people from multiple regions and we assumed that some dialects in social networks can be proper to a specific region.

- **Our dataset:** the problems mentioned above prompted us to create our own detests and adding more context to make the classification more accurate and credible. Hence, we queried recent tweets by specifying the region London since the problem being addressed is proper to this region. To label the tweets, which means assuming that a tweet is positive or negative to construct a training set, we based our queries on the emojis (4 for each of category) considered as the most positive and negative [19].The number of collected tweets is 45139 and have been saved in a file in a JSON format for further usage. (a link in the appendix points to the dataset which we cannot submit due to its size).

Notes:

- We used the same method for the dataset that we derived the results from (the public opinion), we queried keywords to have multiple opinions about the issue and then classified the tweets as positive or negative (discussed later).
- The data set is relatively large, so a link to the dataset is provided in the appendix.

6.2. Dataset pre-processing

There are two main parts for the pre-processing:

- **Text extraction:** The tweets saved in JSON format contain many information and meta-data (many fields) about the tweet, however, we are interested only in the tweet’s text, so the first part is to extract the text from the JSON tweet.

- **Data Cleaning:** in Natural language Processing, stop words have to be removed before processing the text. Stop words such as ‘the’ or ‘about’ are words that appear frequently but do not carry any sentiment or opinion. NLTK provides a list of words we used to clean our tweets, in addition to words like ‘http’ we noticed appearing many times. We also removed all the hashtags and user's mentions in tweets by taking profits from the fields provided by the Twitter API (‘hashtags’ and ‘user_mentions’).

6.2.1. How did we choose the features ?

a. Feature extractor for Naive Bayes Classifier:
For the Naive Bayes classifier, we have chosen to take as features the most frequent words in both documents (the positive tweets and the negative ones) after cleaning because they would be the most informative features in terms of polarity based on our previous assumptions when labeling them. For example, a tweet assumed positive would have more positive words, and by taking the most frequent, it is more likely to contain one of them.

b. Tfidf : We used this technique (explained in details in section 2) to encode the documents (tweets) to be able to use it with skit-learn for logistic regression and SVM-SVC.

6.3. Data exploration
since our data is in the form of text, we visualise the histogram representing the frequency of appearance of each word. It is important to note that the dataset is relatively large. Therefore, we visualise only some parts of the data, i.ue the vocabulary.

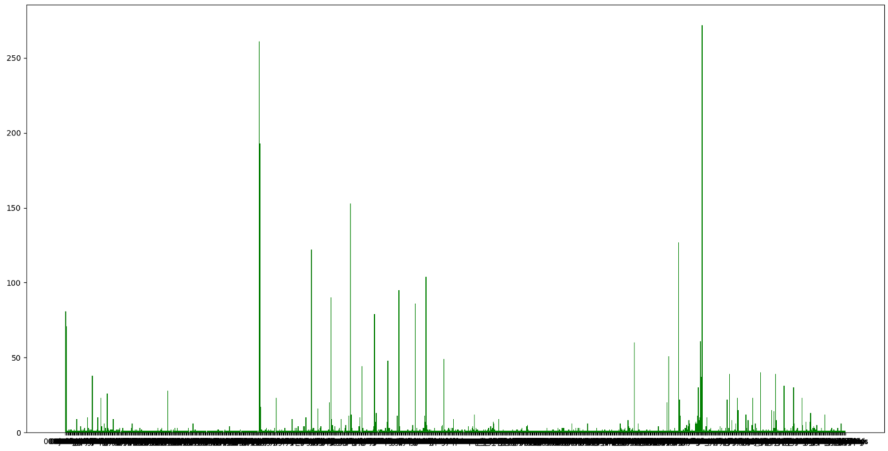


Figure 1: Vocabulary data from 0 to 1000

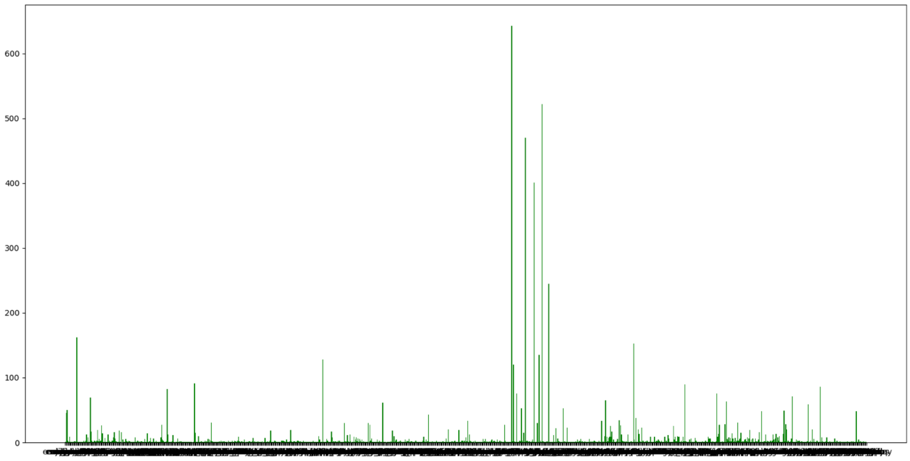


Figure 2: Vocabulary data from 2000 to 21000

From the figures above, it appears clearly that some words are significantly more frequent than others which lead us to consider this fact as an important factor while building the classification model. (more ranges of vocabulary are provided in the appendix).

6.4. Data cleaning

We define two types of cleaning:

- **Cleaning for Naive Bayes classifier:** The cleaning process is based on removing all stop words, hashtags and user's mentioned in the tweet. This is because the feature extraction of this classifier is based on the most appearing words and the elements mentioned to be cleaned are meaningless in term of sentiment analysis which means they do not give any information about a sentiment or an opinion of a situation.

- **Cleaning for tf-idf:** This is not a classification technique but a technique used to encode a text to be able to use it for Sckit-learn algorithms (SVM and Logistic regression in our case). The cleaning of tweets before applying Tfidf is different from that of the Naive Bayes because this technique retrieves features by overlooking the most appearing words considered as not informative. Hence we don't remove the stops words in this case (the most frequent word in the English language) and we only clean the hashtags and the user's mentioned in the tweet to not bias the technique's results.

6.5. Modelling Process

For the modelling, we have compared and tested three different classification methods as mentioned previously, each method is presented below:

6.5.1. Logistic regression

The steps taken to train this model are as follow:

1. Tfidf: Using Tfidf to encode each tweet which means clearly transform the words forming the tweets to floats (weights) based on a vocabulary built based on all words that appeared the tweets. These weights will have a significance for the algorithm and make it possible to train the model when each vector of floats correspond to a specific target (category).

2. Build the target: the target is simply a vector of zeros and ones corresponding to the two categories positive and negatives respectively.

3. Splitting the dataset: After obtaining the dataset (X, Y), X is the vector of floats (inputs) and Y is the target, we split them into two parts, 20% for testing and 80% for training.

4. Training the model: We train the model with the training set obtained.

Before choosing the best model, we tried different parameters and data splitting for cross-validation which allowed us to choose the best model. The obtained results are as follow:

C	Penalty	Accuracy	Precision	Recall	F1 score
1	l2	0.758639	0.76	0.76	0.76
10	l2	0.779131	0.78	0.78	0.78
20	l2	0.783008	0.78	0.78	0.78
20	l1	0.787443	0.79	0.79	0.79
10	l1	0.785666	0.79	0.79	0.79
1	l1	0.738923	0.74	0.74	0.74

Confusion Matrix (from the scikit learn library) of the best model (highlighted in green) :

	TRUE	FALSE
Negative	3160	934
Positive	3949	985

6.5.2. Suport Vector Machine SVM

The same steps as for logistic regression are followed since we will be using Tf-idf for this classifier as well.

The results :

C	Penalty	Accuracy	Precision	Recall	F1 score
1	l2	0.782676	0.78	0.78	0.78
10	l2	0.783340	0.78	0.78	0.78
20	l2	0.781568	0.78	0.78	0.78
20	l1	0.784226	0.78	0.78	0.78
10	l1	0.785445	0.78	0.78	0.78
1	l1	0.777027	0.78	0.78	0.78

Confusion Matrix of the best model (highlighted with green):

	TRUE	FALSE
Negative	3088	1057
Positive	3927	956

6.5.3. Naive Bayes Classifer

We will use this classifier from NLTK differently from the previous ones, the steps are presented below:

1. **Define a Feature Extraction function:** The function takes two parameters, a document (a text) and a list of words called word features. The role of the function is to check whether words appear in the document. The aim from doing that, is to extract the feature from the tweets, by entering a tweet and a list of words and returning whether the words appeared in the tweet or not, hence representing the features of each tweet used to train the model.

2. **Create a list of most appearing words:** Create a list containing all the words from all the tweets of the dataset and calculate the frequency of appearance of each word, and take the 2000 most appearing words. The tweets of the dataset containing positive and negative tweets, the most appearing word are more likely to have a certain polarity (negative or positive).

3. **Extract features from tweets:** use the features extraction function with the tweets and the list of words from step 2 and create features based on the most frequent words (the most informative). We recall that the tweets are labeled (positive and negative), so at this point we have a list of features labelled with their corresponding category.

4. **Train the model:** We split the feature set obtained from the previous step by taking 20% for testing and 80% for training and train the Naive Bayes classifier.

The results obtained are presented below:

Accuracy	Precision	Recall	F1 score
0.678555	0.69	0.68	0.67

Confusion Matrix :

	TRUE	FALSE
Negative	1658	1613
Positive	3084	645

7. Removing neutral tweets

In order to add more credibility to our classification, we decided to remove all the neutral tweets, which means the tweets that do not have any polarity, therefore do not represent any clear positive or negative opinions about a certain situation. This will help us when we apply the model to assess the popularity since the neutral tweets might bias our classification results.

To do that, we used a technique based on Vader algorithm [20] which is available in the NLTK library in the form of a sentiment Intensity Analyser that outputs a score from 0 to 1 for each of the categories (positive, negative, neutral and compound) of a certain text. In our case, we apply this analyser on all the cleaned tweets from the dataset and then remove the tweets that have a score of 1 (neutral score), hence considered by the analyser as 100% neutral.

We trained the models with the new tweets obtained following the same steps presented above for each classifier and the results are as follow :

7.1. For Logistic Regression:

C	Penalty	Accuracy	Precision	Recall	F1 score
1	l2	0.766952	0.77	0.77	0.77
10	l2	0.785407	0.79	0.79	0.78
20	l2	0.786838	0.79	0.79	0.78
20	l1	0.788698	0.79	0.79	0.79
10	l1	0.787410	0.79	0.79	0.79
1	l1	0.747067	0.75	0.75	0.75

Confusion Matrix :

	TRUE	FALSE
Negative	2450	694
Positive	3063	783

7.2. SVM:

C	Penalty	Accuracy	Precision	Recall	F1 score
1	l2	0.783977	0.78	0.78	0.78
10	l2	0.784835	0.78	0.78	0.78
20	l2	0.783404	0.78	0.78	0.78
20	l1	0.784549	0.78	0.78	0.78
10	l1	0.785121	0.78	0.78	0.78
1	l1	0.777396	0.78	0.78	0.78

Confusion matrix:

	TRUE	FALSE
Negative	2437	796
Positive	3043	714

7.3. Naive Bayes classifier :

Accuracy	Precision	Recall	F1 score
0.691142	0.70	0.69	0.68

Confusion matrix

	TRUE	FALSE
Negative	1699	1541
Positive	3139	621

8. Using NLTK samples as a dataset for training

We wanted to compare the results obtained with the tweets extracted from Twitter based on emojis with the sample tweets provided by NTLK and already labeled as positive or negative, in order to assess the results arising from our starting assumptions.

We trained the models with these tweets for both cases (with and without neutral tweets), and the results are as follow:

8.1. With Neutral tweets

8.1.1. Logistic Regression:

C	Penalty	Accuracy	Precision	Recall	F1 score
1	l2	0.7615	0.76	0.76	0.76
10	l2	0.7605	0.76	0.76	0.76
20	l2	0.758	0.76	0.76	0.76
20	l1	0.7475	0.75	0.75	0.75
10	l1	0.7505	0.75	0.75	0.75
1	l1	0.7485	0.75	0.75	0.75

Confusion matrix

	TRUE	FALSE
Negative	796	192
Positive	727	285

8.1.2. SVM:

C	Penalty	Accuracy	Precision	Recall	F1 score
1	l2	0.727875	0.73	0.73	0.73
10	l2	0.717375	0.72	0.72	0.72
20	l2	0.716875	0.72	0.72	0.72

Confusion matrix

	TRUE	FALSE
Negative	3062	932
Positive	1245	1245

8.1.3. Naive Bayes classifier :

Accuracy	Precision	Recall	F1 score
0.703	0.71	0.70	0.70

Confusion matrix

	TRUE	FALSE
Negative	641	382
Positive	765	212

8.2. Without Neutral tweets :

8.2.1. Logistic Regression :

C	Penalty	Accuracy	Precision	Recall	F1 score
1	l2	0.764395	0.76	0.76	0.76
10	l2	0.761753	0.76	0.76	0.76
20	l2	0.758584	0.76	0.76	0.76
20	l1	0.736925	0.74	0.74	0.74
10	l1	0.744321	0.74	0.74	0.74
1	l1	0.754358	0.74	0.74	0.74

Confusion matrix

	TRUE	FALSE
Negative	730	204
Positive	717	242

8.2.2. SVM:

C	Penalty	Accuracy	Precision	Recall	F1 score
1	l2	0.761753	0.76	0.76	0.76
10	l2	0.744849	0.75	0.74	0.74
20	l2	0.732171	0.73	0.73	0.73

Confusion matrix

	TRUE	FALSE
Negative	732	202
Positive	710	249

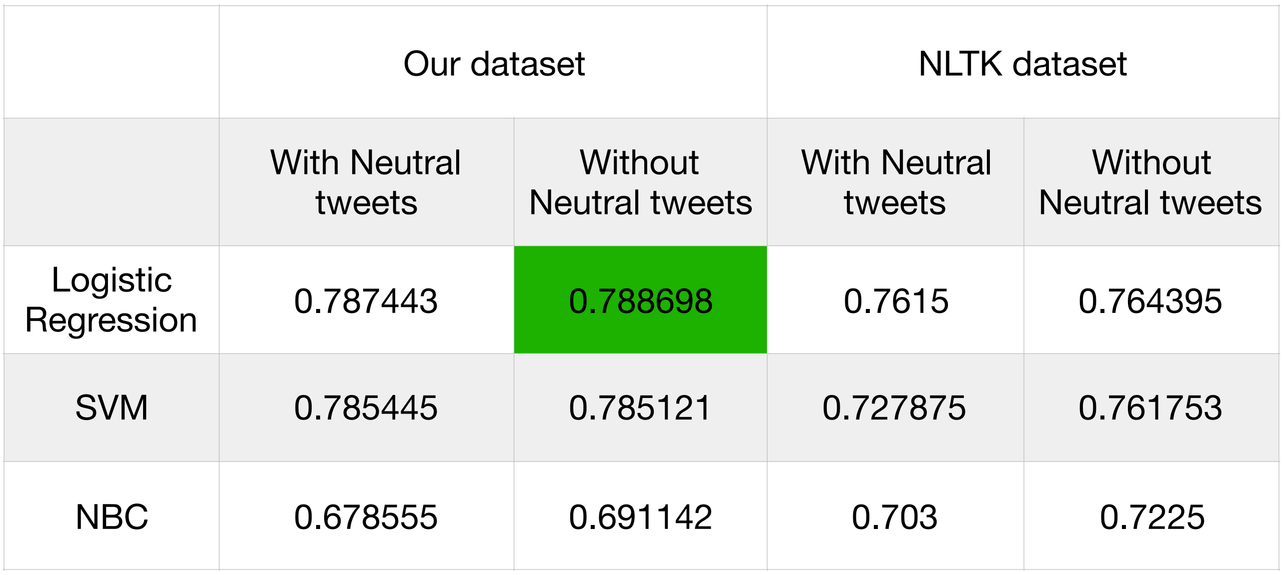
8.2.3. Naive Bayes classifier :

Accuracy	Precision	Recall	F1 score
0.7225	0.71	0.70	0.70

Confusion matrix

	TRUE	FALSE
Negative	641	382
Positive	765	212

The table below summarise the accuracies of the models that gave the best results :



	Our dataset		NLTK dataset	
	With Neutral tweets	Without Neutral tweets	With Neutral tweets	Without Neutral tweets
Logistic Regression	0.787443	0.788698	0.7615	0.764395
SVM	0.785445	0.785121	0.727875	0.761753
NBC	0.678555	0.691142	0.703	0.7225

As we can see, the table shows that our dataset gave better accuracy than the NLK tweets samples, which could be explained with the dataset's size since our dataset contains merely 4 times the number of tweets contained in the NLTK dataset.

We notice as well that removing the neutral tweets increase slightly the accuracy, and the model with the best performance is the Logistic regression method with the parameters (C=20, penalty =l1) trained with our dataset after removing the neutral tweets.

9. Applying the model (assessing the popularity):

Data Extraction: Extract the tweets from twitter using the same method as for the training tweets (using tweepy) by querying the tweets with the hashtag (#SadiqKhan) from the date 01-04-2018 to 10-04-2018. The 13068 tweets in a JSON format have been saved in a file (see appendix for the data).

Data preparation: The neutral tweets have been removed using the method stated before because they might pollute the dataset and bias the classification.
The model chosen is the Logistic regression (with the parameters: C= 20, penalty = l1), because this is the one that gave the best performances, hence the tweets have been encoded with the same format used to train this model in order to make the predictions properly.

Tweets classification (prediction):

- The number of tweets classified as positive : 5473
- The number of tweets classified as negative : 6093

Results' visualisation:

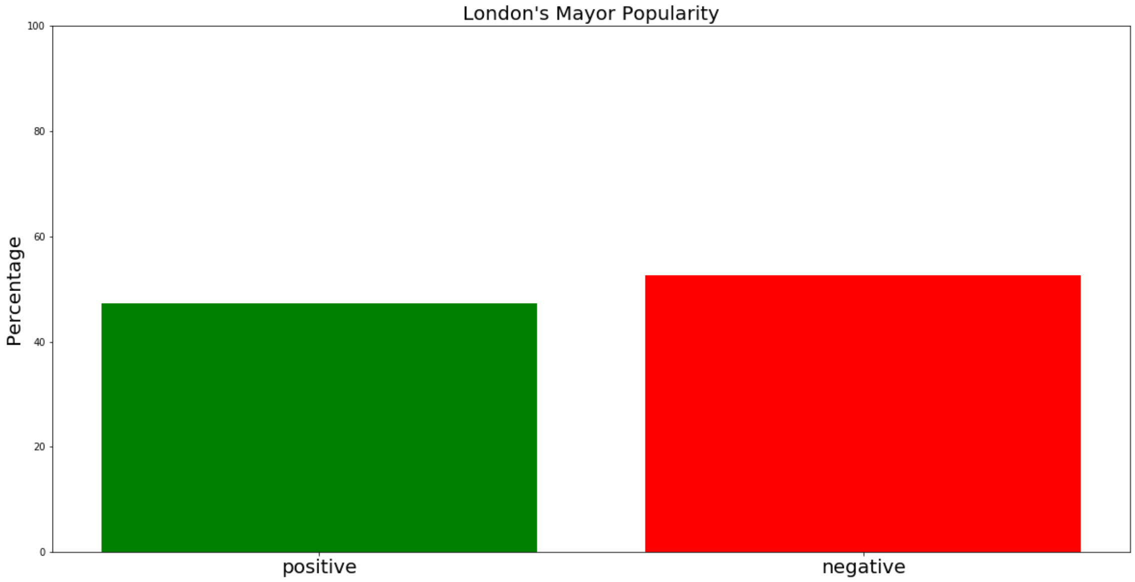


Figure 3: Prediction results

10. Challenges:

We can ennumerate the challenges as follow:

- The most difficult callenge that we face during this coursework is the lack of expertise relating to natural language processing techniques and their usage in real-world problems.
- The time taken by some algorithms, such as, SVM to train the models.
- The hardware resource limittation: our computers were not powerfull enough to do as many tests as we wanted to reach perfect results and increase as maximum as possible our dataset size. Even while using the ITL machines theres are restrictions on the memory (RAM) and storage (space) which prevent us from increasing the dataset size and train multiple models for SVM.
- The text type of data that we collected is very unstructured and full of meaningless caracters and words that should be cleaned and removed. It includes non-english words, links, hastags, mentions, ...etc that can contribute in the misclassification of text. Therefore, it was challenging to choose the right features that we should used while classifying.
- The rate limit of the tweepy API is also an issue. We were restricted in the number of tweets that we can collect.

11. Possible extensions or business applications of your project

A possible business application is to how a company can make advertisements based on opinions about some person or some product with

And with NLP on tweets, an overall opinion can be formed of certain product/individual and based on it, can target specific regions. And this process can be performed on different regions depending on person/product, so advertisements can differ.

A general example can be a certain local fish and chip shop in Brighton. With that, depending on tweets, overall opinion can be formed for it and if quite positive, adverts can be created to expand further in the Sussex region such as for the web indicating its reviews from websites (including and not including NLP).
Another is a local political party such as Labour in a borough. And depending on the overall opinion from Twitter, whichever party is more positive is used to promote adverts for it in that borough as well as expanding to other nearby borough voters using Facebook and web ads.

Two challenges and open problems for this however are annotated corpora and deep learning.

> 1 . Annotated corpora are essential to improve opinion mining performance. With most being in English, can make it difficult with other languages. For cross-lingual opinion mining, existing methods such as transfer learning have drawbacks such as cumulative error. So, sophisticated approaches needed to combat this.
>
> 2. Deep Learning is becoming a growing field for machine learning (both research and industrial based). But for opinion summarisation, hardly any research being conducted. Also, it did not reflect a major improvement for NLP, so more efforts for developing deep learning approaches for NLP required. [23]

12. Conclusion

The project was beneficial for us in terms of familiarisation with language processing and Machine learning by testing and comparing different text classification models under different conditions.

Social Networks like Twitter has become a good field for sentiment analysis and opinions mining. In fact, being able to access the data generated by users programmatically combined with the different available NLP tools nowadays, enables many applications to have a general idea about what people think about certain events or personalities.

The result outputted by our model shouldn’t be taken as an absolute reference because of the relatively small amount of data extracted to make a proper assessment and some other factors that we might have overlooked when training the models.

Even though, the results obtained reflect the situation where the recent crimes and violence of all kinds knew a real rise especially during the time period the tweets was extracted, which clearly affected the popularity of the mayor of London.

References

[1] W.J.Richardson, “Sadiq Khan is NOT to blame for skyrocketing gang crime – Tory Cuts have axed 1,850 Police Officers in London”, 6 April 2018. Available at:https://evolvepolitics.com/sadiq-khan-is-not-to-blame-for-skyrocketing-gang-crime-tory-cuts-have-axed-1850-police-officers-in-london/ [Accessed 13th April 2018]

[2] A. Bosotti, “Sadiq Khan SHAMED after admitting that he has NOT met bereaved families of gang crime victims”, 6 April 2018. Available at:https://www.express.co.uk/news/uk/941936/Sadiq-Khan-shamed-London-crime-rate-deaths-Mayor-latest-victim-family-Labour-Party-video[Accessed 13th April 2018]

[3] A. Tolhurst, “Sadiq Khan faces calls to resign over London’s spiralling murder rate after another teenager gunned down overnight”, 4 April 2018. Available at: https://www.thesun.co.uk/news/5960391/sadiq-khan-under-pressure-to-tackle-londons-spiralling-murder-rate-after-another-teenager-gunned-down-overnight/ [Accessed 13th April 2018]

[4] https://web.stanford.edu/~jurafsky/slp3/7.pdf

[5] Logistic Regression, Machine Learning lecteur, Dr. Ioannis Patras

[6] https://en.wikipedia.org/wiki/Multiclass_classification

[7] https://web.stanford.edu/~jurafsky/slp3/6.pdf

[8] https://web.stanford.edu/class/cs124/lec/naivebayes.pdf

[9] http://www.ritchieng.com/machine-learning-svms/

[10] Ramos, Juan. "Using tf-idf to determine word relevance in document queries." *Proceedings of the first instructional conference on machine learning*. Vol. 242. 2003.

[11] https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[12] http://scikit-learn.org/stable/modules/feature_extraction.html

[13] https://en.wikipedia.org/wiki/F1_score

[14] https://en.wikipedia.org/wiki/Confusion_matrix

[15] J.Brownlee, “A Gentle Introduction to Scikit-Learn: A Python Machine Learning Library”, 16 April 2014. Available at: https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/ [Accessed 13th April 2018]

[16] S.Bird, “NLTK: The natural language toolkit”, January 2006. Available at: https://www.researchgate.net/publication/220873131_NLTK_The_natural_language_toolkit [Accessed 13th April 2018]

[17] http://www.nltk.org/api/nltk.tokenize.html

[18] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

[19] Wang, Hao, and Jorge A. Castanon. "Sentiment expression via emoticons on social media." *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 2015.

[20] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

[21] Twitter apps https://apps.twitter.com/

[22] http://tweepy.readthedocs.io/en/v3.5.0/api.html#tweepy-api-twitter-api-wrapper (tweepy)

[23] S.Sun, C. Luo & J.Chen, “A review of natural language processing techniques for opinion mining systems”, 3 November 2016. Available at: https://www.sciencedirect.com/science/article/pii/S1566253516301117 [Accessed 13th April 2018]

Appendix

Data: The dataset is downloaded on a personal drive because of the relatively large size. (link : https://drive.google.com/file/d/114STgS1jQSsZ1H6C-J66ydS05oOMhAUy/view?usp=sharing)

Code: The code source of the work is submitted on QM plus. The code is in a jupyter format and it is well explained and commented.