Twitter Text Classification
5981P Text Mining Project

Dilip Madhu Kumar[1], Revathi Sadanand[2] and Salman Khatri[3]

[1] 75658, University of Passau, Germany, madhuk01@gw.uni-passau.de
[2] 75662, University of Passau, Germany, sadana01@gw.uni-passau.de
[3] 75691, University of Passau, Germany, khatri01@gw.uni-passau.de

October 11, 2016

Abstract. Twitter Text Classification Text classification is a classic task in natural language processing. Given a text document and a set of categories, the text classification algorithm must assign the text document into one or more of the given text categories. Here, we discuss about some classic text classification techniques. As a project title in the domain of text classification, we suggest the task of short text classification, which is more challenging than classic document classification. In the proposed project title, we use the Twitter Political Corpus. Given a tweet, the task is to distinguish political tweets from non-political ones.

Keywords: Text Mining; Twitter Text Classification; Political Tweets; Text categorization; Topic classification; TF-IDF; Confusion Matrix; Classification Report; Cross-Validation; SVM; Logistic Regression; Mulitnomial Naive Bayes; Bernoulli Naive Bayes; Count Vectorizer;

Project Source Code : https://github.com/dilip-dmk/TextMiningProject

Table of Contents

1 Introduction

1.1 Text Classification

Also known as Text categorization, Topic classification, or Topic spotting is the process of classification of documents into a fixed number of predefined categories. Basically it is the task of choosing the correct class label for a given input. Each document can be in single, multiple, or no category at all. Using machine learning techniques, the objective is to learn classifiers from examples which perform the category assignments automatically. This is a *supervised learning problem*. Since categories may overlap, each category is treated as a separate binary classification problem, however this would be not be the case for this particular project.

The first step in text classification is to transform documents, which typically are strings of characters, into a representation suitable for the learning algorithm and the classification task. Information Retrieval research suggests that word stems work well as representation units and that their ordering in a document is of minor importance for many tasks. This leads to an attribute value representation of text. Each distinct word which corresponds to a feature, with the number of times word which occurs in the document as its value. To avoid unnecessarily large feature vectors, words are considered as features only if they occur in the training data at least 3 times and if they are not stop-words ('and', 'or', 'the', etc.).

This representation scheme leads to very high-dimensional feature spaces containing 10000 dimensions and more. Feature selection makes use of conventional learning methods, to improve generalization accuracy and to avoid over-fitting.

Scaling dimensions of the feature vector with their Inverse Document Frequency(IDF) helps in improving the overall performance [20].

1.2 Supervised Learning

It is the classification method where a training corpora is available which contains the correct label for each input and the classifier is built on using that corpora.

During *training*, a feature extractor is used to convert each input value to a feature set. These feature sets, which capture the basic information about each input that should be used to classify it, are discussed in the next section. Pairs of feature sets and labels are fed into the machine learning algorithm to generate a model. During *prediction*, the same feature extractor is used to convert unseen inputs to feature sets. These feature sets are then fed into the model, which generates predicted labels [7]. This process has been further explained in Figure 1.

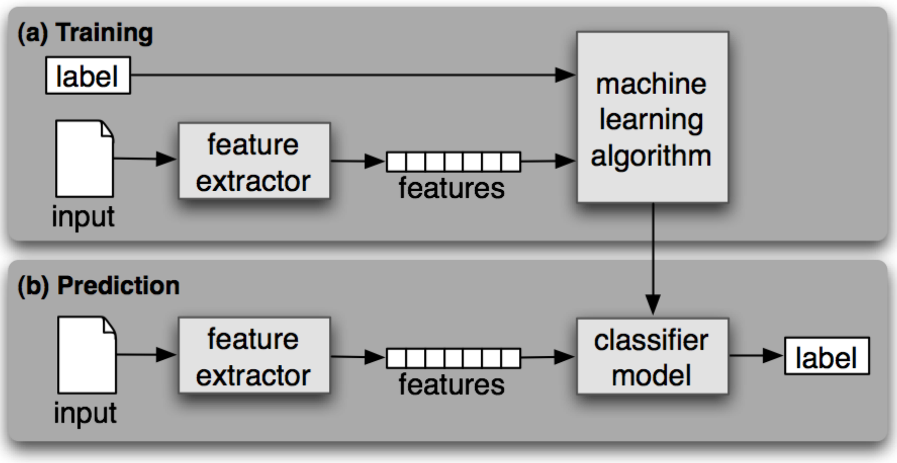


Fig. 1. Supervised Text Classification Framework [7]

1.3 Twitter

Twitter, an online social networking service, commands more than 300 million monthly active users as of March 31, 2016. Twitter users tweet about any topic within the 140-character limit and follow others to receive their tweets [18].

For the purpose of this project, we will be using a Corpus containing roughly 3000 tweets which have been classified according to their political views. This corpus will be used further to train various classifier models.

2 Targets / Plans

2.1 Requirements

For the development of this project the Software and Packages required have been mentioned in Table 1 :

Package	**Version**
PyCharm	2016.2.3
Python	2.7
Pip	7.1.0
NLTK	3.2.1
NumPy	1.11.2
SciPy	0.17.1
scikit-learn	0.18.0
MatPlotLib	1.5.1
Pandas	0.13.1
BeautifulSoup4	4.5.1

Table 1. Software and Package Requirements

2.2 Data Preparation

As we are classifying tweets using Supervised Learning based on the two classes which are '*POLIT*' and '*NOT*', we need a corpora for training our model and evaluation. This data contains the tweets and its respective class in the text file and is called as training data. The Political Twitter Corpora is freely available on internet[5]

2.3 Data Cleaning

Before analyzing, the data needs to be cleaned which means the special characters needs to be replaced by spaces and cases have to be normalized. The data thus obtained can later be pre-processed in the consequent steps.

2.4 Data Preprocessing

Preprocessing plays a vital role in any text mining application. The key steps of preprocessing are Tokenization[11], Filtering of stopwords and Term Frequency-Inverse Document Frequency(TF-IDF)[19]

Tokenization The formation of words, phrases, symbols or any meaningful element by breaking the stream of text is called tokenization.

Filtering of stopwords The most common words in text documents are articles, prepositions, and pro-nouns, etc. which does not give the meaning of the documents. These words are called as stopwords. Example for stop words: the, in, a, an, with, etc. Stop words are filtered from documents because those words are not measured as keywords in text mining applications.

TF- IDF Term Frequency- Inverse Document Frequency(TF-IDF) is a numerical measurement that is proposed to reflect how critical a word is to a document in a collection of corpus.

Thus combine the definitions of *term frequency* and *inverse document frequency*, to produce a composite weight for each term in each document. The tf-idf weighting scheme assigns to term t a weight in document d given by

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t.$$

In other words, $\text{tf-idf}_{t,d}$ assigns to term t a weight in document d that is

- highest when t occurs many times within a small number of documents (thus lending high discriminating power to those documents)
- lower when the term occurs fewer times in a document, or occurs in many documents (thus offering a less pronounced relevance signal)
- lowest when the term occurs in virtually all documents [14]

2.5 Train-Test Split

The process of splitting arrays or matrices into random train and test subsets. Splitting the available data into subsets, drastically reduces the number of samples which can be used for learning the model, and the results depend on a random choice pair of training and test sets [17].

2.6 Classifiers

There are various machine learning classifiers, and want to ensure best accuracy, few of them need to be tested out, while also trying different parameters for the algorithm, and eventually select one by cross-validation 2.7.

Multinomial Naive Bayes It is a Naive Bayes algorithm for multinomial distributed data, which estimates the conditional probability of a particular word/term/token given a class as the relative frequency of term t in documents belonging to class c. It considers the number of occurrences of term t in training documents from class c, including multiple occurrences and it ignores all non-occurrences terms. Basically, it is used when the multiple occurrences of the words matter a lot in the classification problem, therefore the best candidate for Text Classification.

In the Multinomial model by first pick a class C = c with P(c) where C is a random variable taking values from C as values. Next generate term $\mathrm{t_k}$ in position k with $\mathrm{P(X_k = t_k \, l \, c)}$ for each of the $\mathrm{n_d}$ positions of the document. The $\mathrm{X_k}$ all have the same distribution over terms for a given c. Figure 2, shows the generation of (t1, t2, t3, t4, t5) = (Beijing, and, Taipei, join, WTO) corresponding to the one-sentence document 'Beijing and Taipei join WTO' [8].

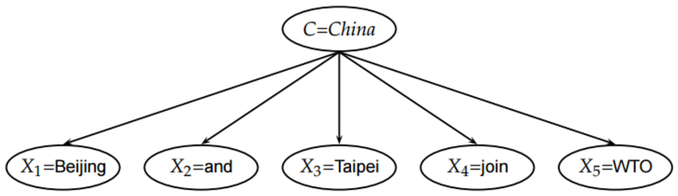


Fig. 2. Mulitnomial NB Model [8]

Bernoulli Naive Bayes It is also a Naive Bayes algorithm for multinomial distributed data according to the multivariate Bernoulli distribution, which generates Boolean indicator(0 and 1).Each term of the vocabulary equal to 1 if the term belongs to the examining document and 0 if it does not. Considers the number of occurrences of each term and also considers the non-occurrence terms.

In the Bernoulli model, first pick a class C = c with P(c) and then generate a binary indicator e_i for each term t_i of the vocabulary (1 i M). Figure 3, shows the generation of (e1,e2,e3,e4,e5,e6) = (0, 1, 0, 1, 1, 1) corresponding, again, to the one-sentence document 'Beijing and Taipei join WTO', where *and* is a stop-word [8].

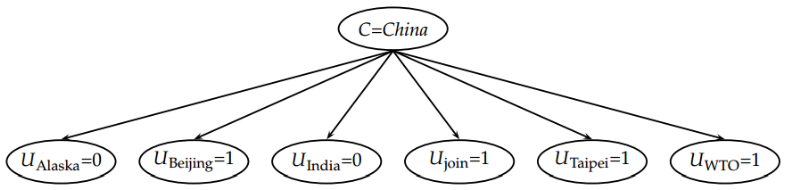


Fig. 3. Bernoulli NB Model [8]

Support Vector Machines (SVM) Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. This hyperplane segregates the classes based on the data points in n-dimensional space where n is the number of features[3]. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes the data points. A linear separation is possible for the data points which are easy to classify but for the data points which are randomly scattered on the plot, the problem of non linear separation problem arises.

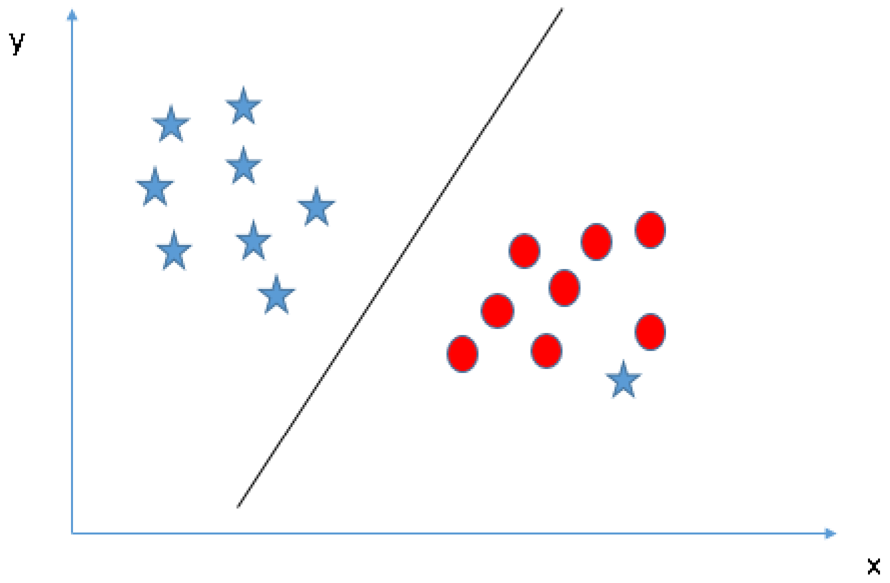


Fig. 4. SVM Linear Classification[3]

For non linear separation problem, a function called ”Kernel” is used which performs extremely complex transformation and finds the process to classify the data according to the class labels.

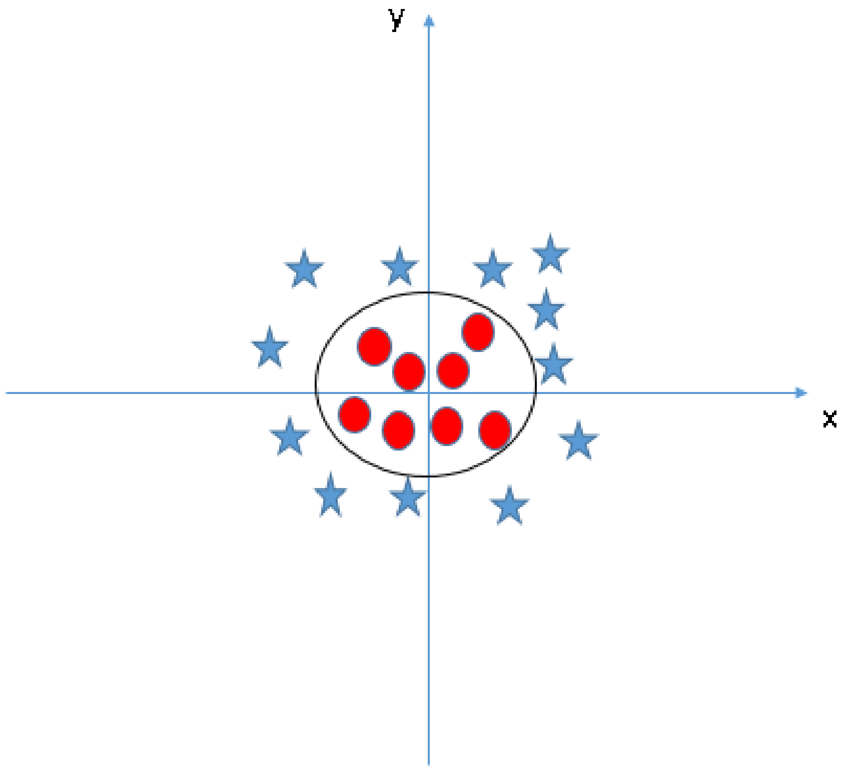


Fig. 5. SVM Non Linear Classification[3]

SVM uses a subset of training points in the decision function (called support vectors), this makes it memory efficient. Performance depends on the number of features and number of samples. High performance if number of samples are greater than number of features. Expensive 5-fold cross validation calculates the probability estimates.

Logistic Regression Logistic regression it is type of analysis to conduct when the dependent variable is binary.Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more metric (interval or ratio scale) independent variables[1]. It works with the conditional probability of the data points rather than the class labels. Binomial Logistic Regression used in the situation when the desired outcome for a dependent variable can have two possible types. Mathematically, we have a binary output variable Y, and we want to model the conditional probability Pr(Y = 1—X = x) as a function of x[2].

2.7 Evaluation

Cross-Validation To evaluate the model, a portion of the annotated data is reserved for testing. If test set is small, then evaluation may not be accurate, and if test set is large, means the training set would be smaller, which will affect the performance, if annotated data is limited. The solution for this is performing multiple evaluations on different test sets and then combing the evaluation scores. This is known as Cross-Validation. The original corpus is sub-divided into N subsets, known as folds. The model is trained on all the data except that of the fold, only testing is done of the data on the fold. This is done for all folds as illustrated in Figure 6. Though individual folds are small but the combined evaluation score is of a large data-set, thus would be reliable. It also helps in analyzing the performance across various training sets. Similar scores for all N training set confirms that the score is very much accurate [7].

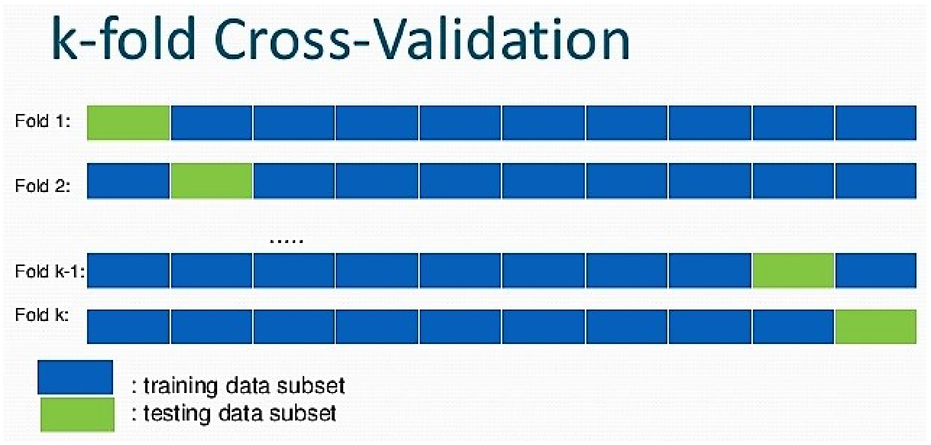
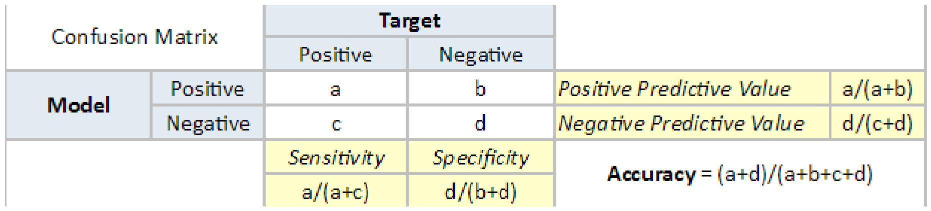


Fig. 6. In Cross-Validation, data-set is divided into k-folds and t each iteration different fold is reserved for testing and all the others, used for training the classifiers [10]

Confusion Matrix It describes the performance accuracy of a classifier on a set of test data for which the true values are known. It shows the number of correct and incorrect predictions made by the classifier and compared to the actual outcomes (target value) in the data. If the matrix is NxN, where N is the number of target values (classes). Performance of such models is commonly evaluated using the data in the matrix [13]. A sample confusion matrix is illustrated in Figure 7



Confusion Matrix		**Target**			
		Positive	Negative		
Model	Positive	a	b	*Positive Predictive Value*	a/(a+b)
	Negative	c	d	*Negative Predictive Value*	d/(c+d)
		Sensitivity	*Specificity*	**Accuracy** = (a+d)/(a+b+c+d)	
		a/(a+c)	d/(b+d)		

Fig. 7. 2x2 Confusion Matrix for two classes (Positive and Negative)[16]

Various terminologies used in a Confusion Matrix:

- Accuracy : the proportion of the total number of predictions that were correct. [(TP+TN)/(TP+TN+FP+FN)]
- Positive Predictive Value or Precision : the proportion of positive cases that were correctly identified. [TP/TP+FP]
- Negative Predictive Value : the proportion of negative cases that were correctly identified. [TN/TN+FN]
- Sensitivity or Recall : the proportion of actual positive cases which are correctly identified. [TP/TP+FN]
- Specificity : the proportion of actual negative cases which are correctly identified. [TN/FP+TN]
- F-measure : weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst at 0. [2*(precision*recall)/(recall+precision)] [13].

Classification Report It is a report showing the main classification metrics. It is similar to a Confusion matrix, but just differs in representation. Rest of the criteria are similar to a Confusion Matrix discussed in 2.7.

3 Method

Data Cleaning The method mainly uses *Regex* to clean the tweets. This method removes the HTML content, non-letters, stop-words and converts it into lowercase, finally splitting into individual words.

```python
def tweet_to_words(raw_tweets):
    tweet_text = BeautifulSoup(raw_tweets,"lxml").get_text()
    letters_only = re.sub("[^a-zA-Z]", " ", tweet_text)
    words = letters_only.lower().split()
    stops = set(stopwords.words("english"))
    meaningful_words = [w for w in words if not w in stops]
    return( " ".join( meaningful_words ))
```

TF-IDF Count Vectorizer is a function which performs preprocessing of text and returns the token counts. Setting ”binary” as true, all non zero counts are set to 1. TFIDF transformer transforms the count matrix to tf-idf representation. The goal of using tf-idf instead of the raw frequencies of occurrence of a token in a given document is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus[4].

```
def preprocess():
    data,target = load_file()
    count_vectorizer = CountVectorizer(binary='true')
    data = count_vectorizer.fit_transform(data)
    tfidf_data = TfidfTransformer(use_idf=False).fit_transform(data)
    return tfidf_data
```

Train Test Splitting The training data is split using *sklearn.cross_validation.train_test_split*. The data-set is split into 80% and 20% reserved for testing. This combination of 80/20 gives the best accuracy ad time values; refer Table 2 for different split values [17].

```
data_train,data_test,target_train,target_test =
     cross_validation.train_test_split(data, target,test_size=0.20,
     random_state=43)
```

Classifiers	**Training/Test Data Split**			
	60/40	70/30	*80/20*	90/10
Multinomial	75.15%(0.538s)	74.87%(0.562s)	*78.38%(0.546s)*	75.08(0.546s)
Bernoulli	83.26%(0.599s)	84.48%(0.593s)	*84.98%(0.560s)*	80.18%(0.560s)
SVM	89.19%(1.414s)	90.79%(1.463s)	*91.14%(1.64s)*	90.69%(1.643s)
Logistics	85.23%(0.619s)	85.39%(0.597s)	*87.09%(0.619s)*	84.08%(0.691s)

Table 2. Split Validation; showing the accuracy and elapsed time values

Classification The classifiers were used directly, by importing then from their packages.

```
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
from sklearn import svm, linear_model
:
:
def main():
    clf1 = MultinomialNB()
    clf2 = BernoulliNB()
    clf3 = svm.SVC(kernel='linear')
    clf4 = linear_model.LogisticRegression()
    :
    :
```

Evaluation For evaluation purpose Confusion matrix and Classification report was used. For better representation code for Confusion Matrix was used from [9] and for Classification report, [6] was used.

```
def evaluate_model(target_true,target_predicted,clf):
    cr = classification_report(target_true,target_predicted)
    cr_plt = plot_classification_report(cr)
    cr_plt.savefig('plots/'+(str(clf).partition("(")[0])+'_CR.pdf',
        bbox_inches='tight')
    print((str(clf).partition("(")[0]) + " Classification Report saved
        successfully")
    cr_plt.close()

    cm = confusion_matrix(target_true, target_predicted)
    cm_plt = show_confusion_matrix(cm, ['NOT', 'POLIT'])
    cm_plt.savefig('plots/' + (str(clf).partition("(")[0]) + '_CM.pdf')
    print((str(clf).partition("(")[0]) + " Confusion Matrix saved
        successfully")
    cm_plt.close()

    print("Accuracy {:.2%}".format(accuracy_score(target_true,
        target_predicted)))

#Cross Validation 10-fold
scores = cross_val_score(clf, data, target, cv=10)
    evaluate_model(target_test,pred1,clf1)
    print("After Cross Validation")
    print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() *
        2))
```

4 Discussion

Train Test Split Referring to Table 2, we could confirm that the combination of 80% training data and 20% test data gave the best accuracy results.

Cross-Validation Table 3 illustrates the accuracy values and elapsed time for the four classifiers with and without cross-validation.

Classifier	Without CV	With CV	Time
Multinomial NB	76.68%	78.00%	0.670s
Bernoulli NB	84.98%	81.00%	0.764s
SVM	88.74%	87.00%	12.452s
Logistic Regression	85.89%	83.00%	0.752s

Table 3. Accuracy values for classifiers with and without Cross-Validation(CV)

Confusion Matrix The confusion matrix for the four classifiers is given below:

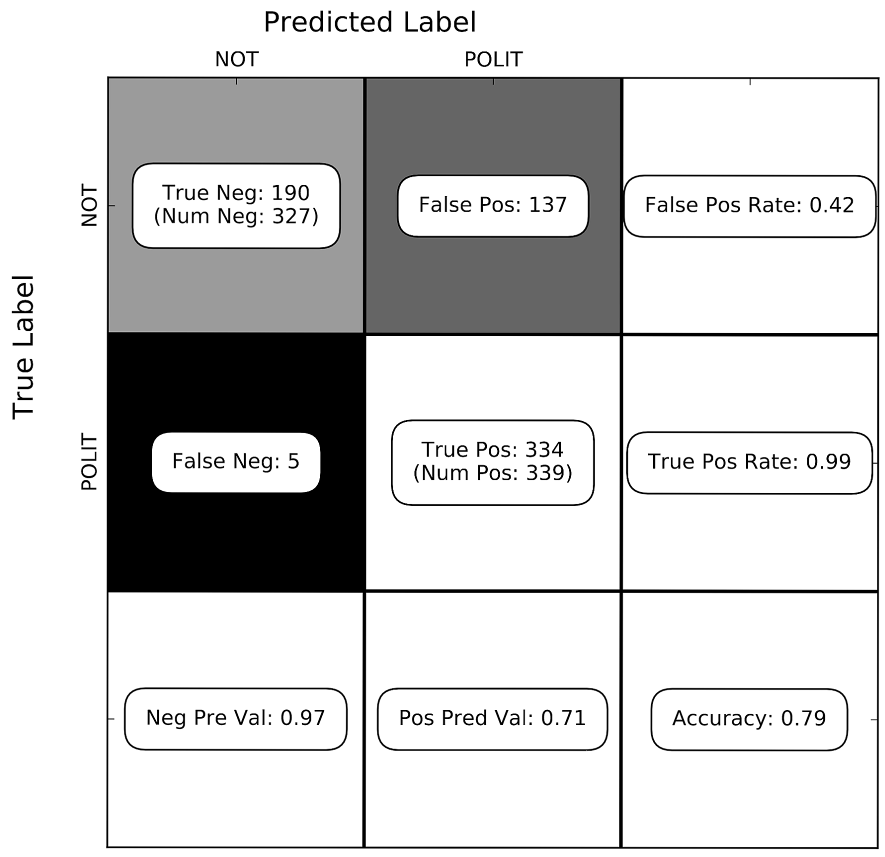


Fig. 8. Confusion Matrix : Multinomial Naive Bayes Classifier

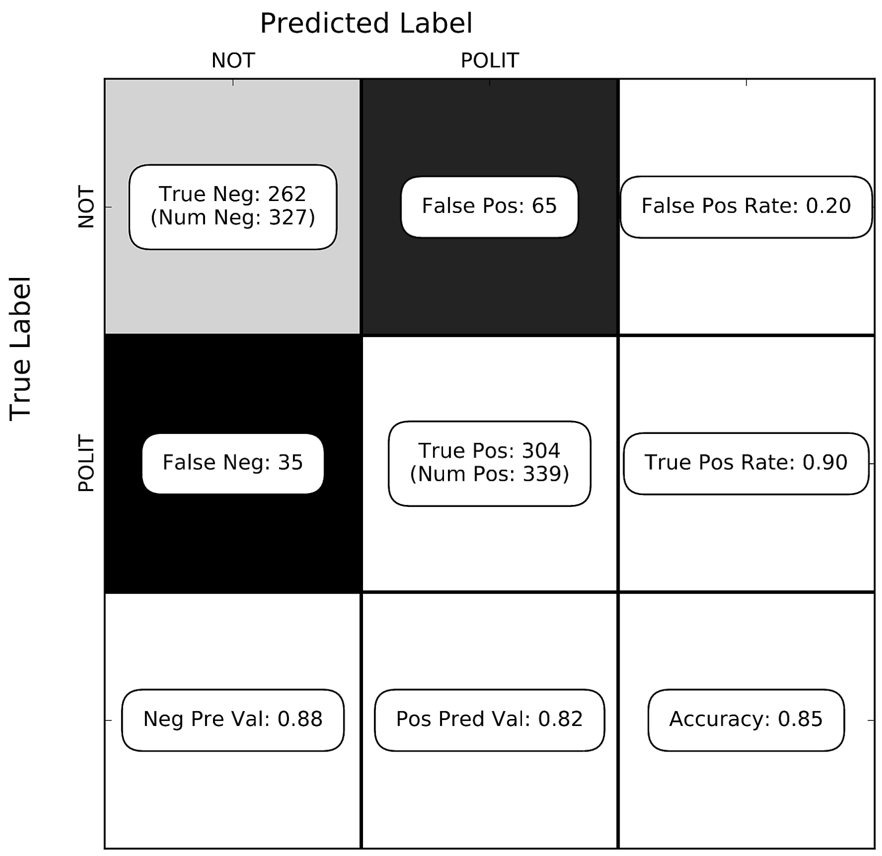


Fig. 9. Confusion Matrix : Bernoulli Naive Bayes Classifier

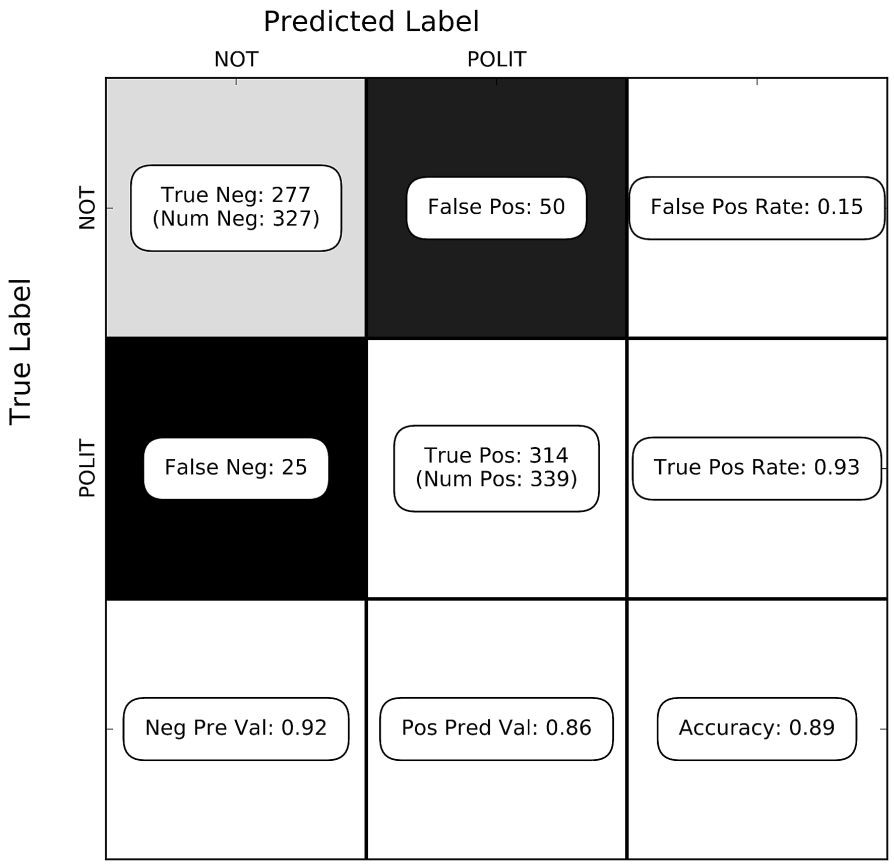


Fig. 10. Confusion Matrix : Support Vector Classifier

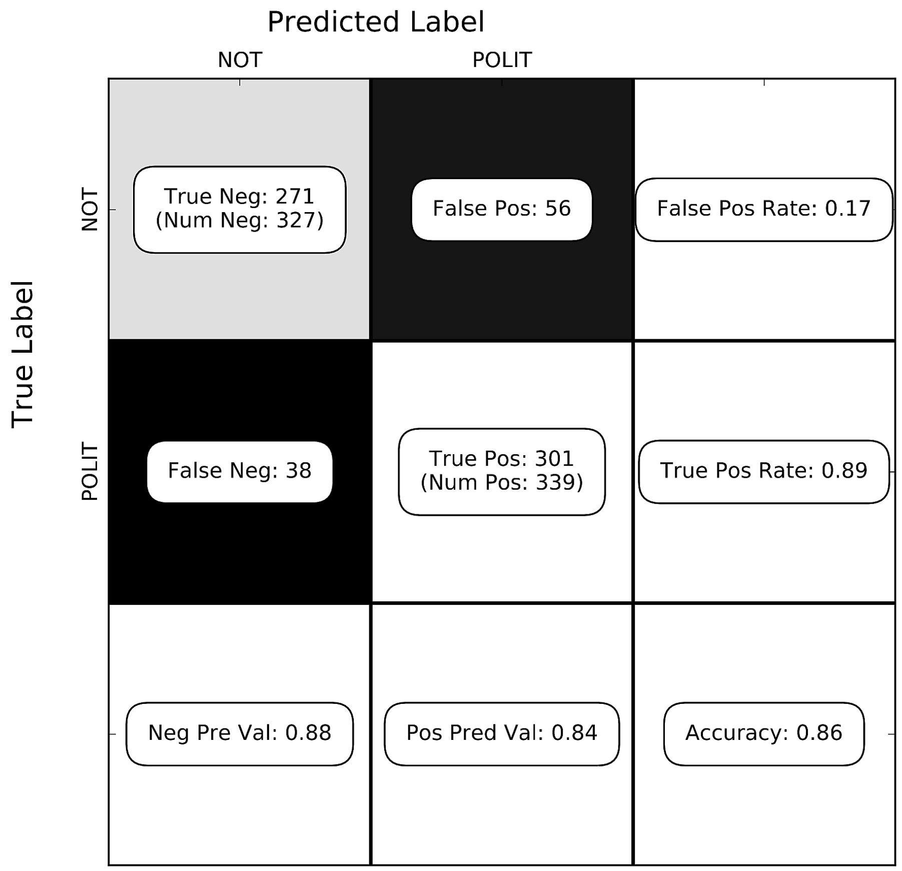


Fig. 11. Confusion Matrix : Logistic Regression Classifier

Classification Report The Classification Report of various classifiers has been illustrated below. The report displays the precision, recall and f1-score for the classes *POLIT* and *NOT*.

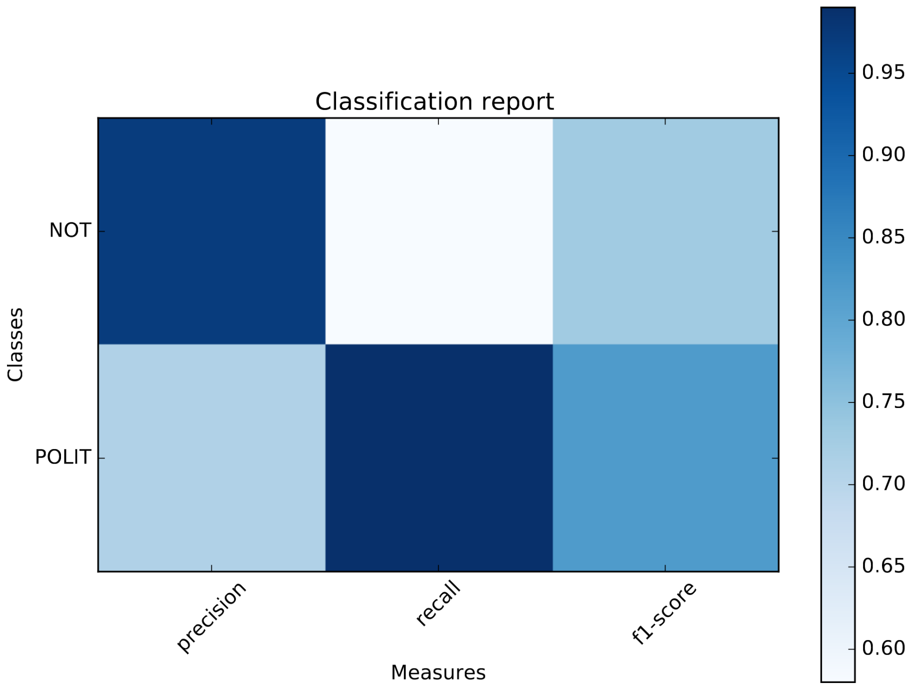


Fig. 12. Classification Report : Multinomial Naive Bayes Classifier

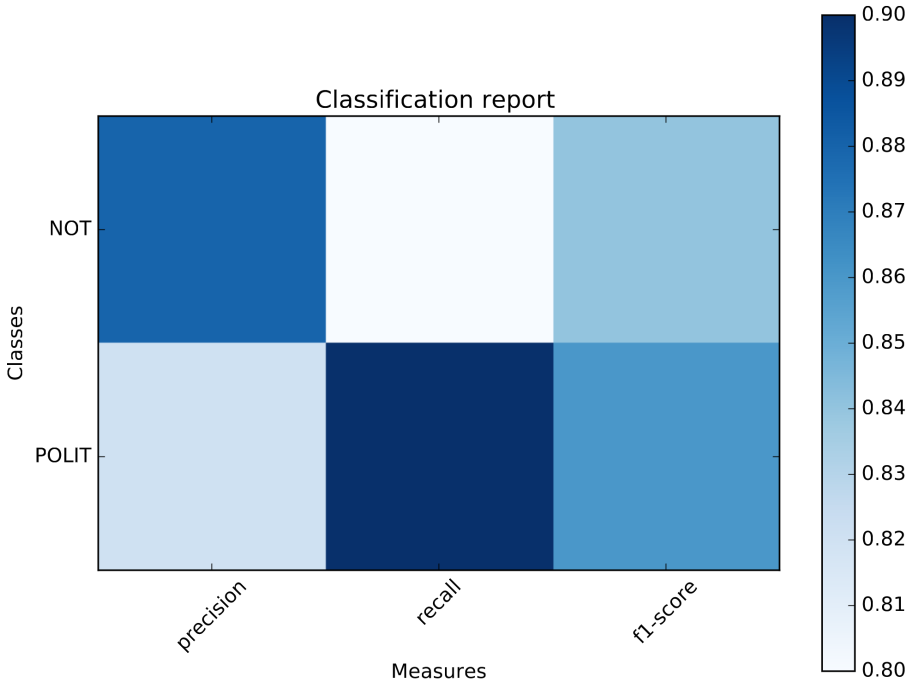


Fig. 13. Classification Report : Bernoulli Naive Bayes Classifier

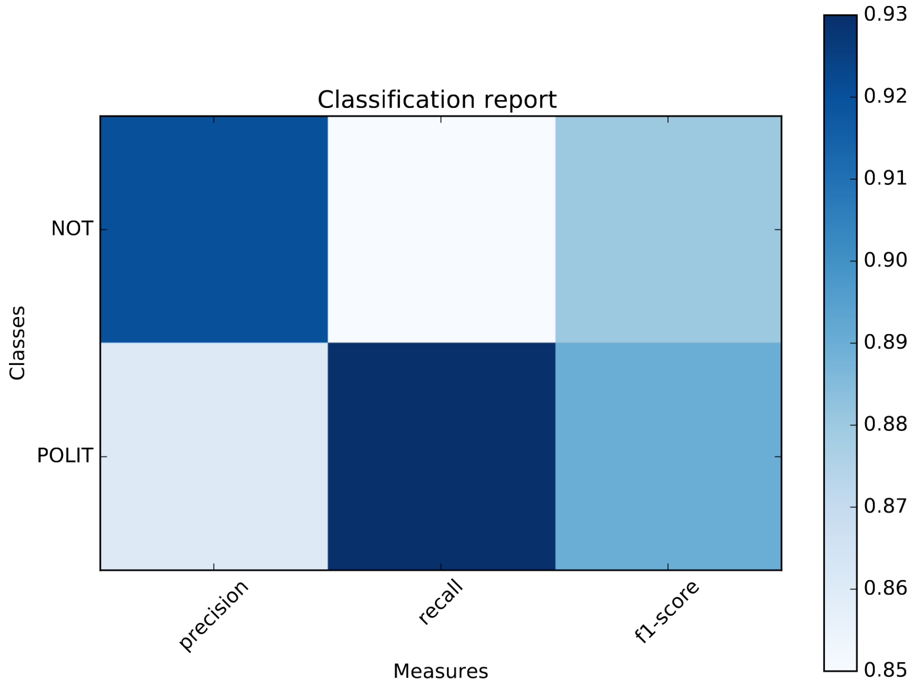


Fig. 14. Classification Report : Support Vector Classifier

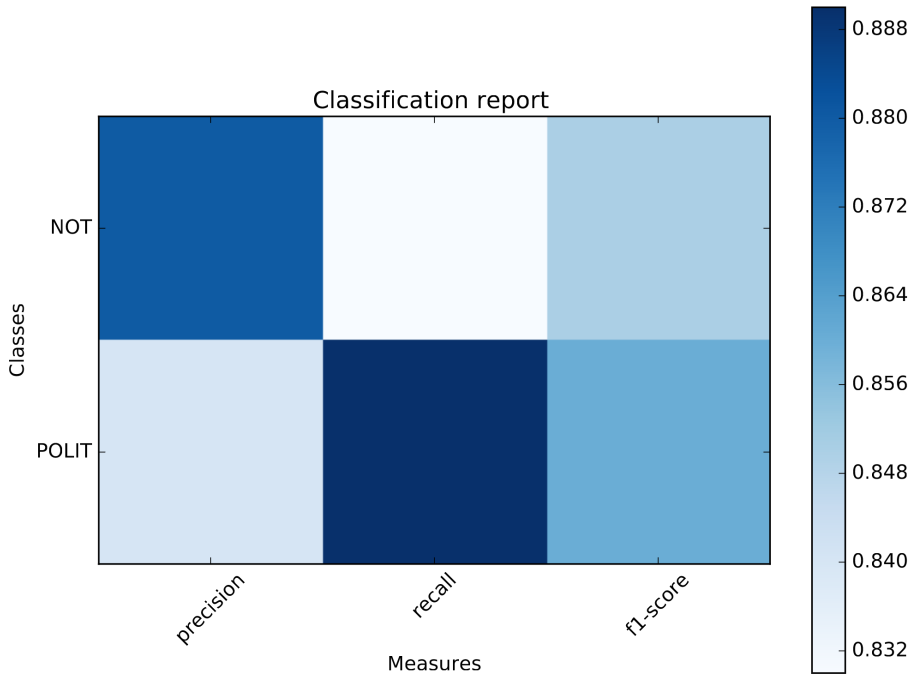


Fig. 15. Classification Report : Logistic Regression Classifier

5 Future Work

5.1 Sentiment Analysis

Sentiment analysis and opinion mining is the field of study that analyzes people's opinions, sentiments, evaluations, attitudes, and emotions from written language. It is one of the most active research areas in natural language processing and is also widely studied in data mining, Web mining, and text mining. It is a machine learning method to extract, identify, or otherwise characterize the sentiment content of a text unit and is also referred to as opinion mining, although the emphasis in this case is on extraction [12].

So a further extension to text classification can be implementing this analysis too. It would be a good addition to knowing the sentiments of the Political tweets.

5.2 Neural Networks

The next step to enhance this project would surely be the inclusion of a Neural Network. It is advised to use a Convolution Neural Network for Text Classification. This would require installing of the Keras Deep Learning Library along with TensorFlow or Theano back-end. It is advised to use Mac OS or Linux for using TensorFlow or Theano, since its easier to install on these OS, rather than on Windows OS. A similar sample code is available on GitHub [15] which could be referred to implement Neural Network on this particular project.

5.3 Tools

Using various tools can make machine learning easy, efficient and fun to do. These tools allows to load data-sets, run algorithms and design and run experiments with results statistically robust enough to publish. Few of the popular tools which can be used for this purpose are :

– WEKA/MATLAB : software toolkit which integrates many algorithms.
– Apache's Mahout : tool for machine learning algorithms which integrates three aspects' of algorithms: recommendation, clustering, and classification. However, basic knowledge about Hadoop is also required.

6 Conclusion

This paper shows the various classifier models being trained successfully using cross validation and supervised learning method and are able to distinguish the Political tweets from the Non Political ones. It also covers the cleaning of data and pre-processing of the tweets for training the classifier models. We have also prepared prepared confusion matrix and classification report for the results for each classifier which we have used. The comparison between the classifiers is based on the accuracy and the time it takes to provide the results. The processing speed of the computer plays a vital role in the execution of the classifiers so we have performed using Linux and Windows machine with different processing power.

After performing Evaluations, we found out that Multinomial Naive Bayes Classifier is not competitive enough against Bernoulli Naive Bayes, SVM and Logistic Regression Classifiers. Thus, we conclude that *Logistic Regression Classifier* gives best accuracy in consideration with time; while being evaluated using Cross-Validation. However, if time is not a constraint then *SVM* can be considered as the best among all others as its accuracy was the highest, however with the worst time.

References

1. Logistic regression. `http://www.statisticssolutions.com/what-is-logistic-regression`. Accessed on 01-06-2016.
2. Logistic regression detail. `http://www.stat.cmu.edu/~cshalizi/uADA/12/lectures/ch12.pdf`. Accessed on 01-06-2016.
3. Support vector machine. `https://www.analyticsvidhya.com/blog/2015/10/understaing-support-vector-machine-example-code/`. Accessed on 29-05-2016.

4. Tdidf transformer. `http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html`. Accessed on 05-10-2016.
5. The twitter political corpus. `http://www.usna.edu/Users/cs/nchamber/data/twitter/`. Accessed on 23-07-2016.
6. Bin. How to plot scikit learn classification report? `http://stackoverflow.com/questions/28200786/how-to-plot-scikit-learn-classification-report`, August 2015. Accessed on 10-09-2016.
7. Steven Bird, Ewan Klein, and Edward Loper. Learning to classify text, Jul 2015. `http://www.nltk.org/book/ch06.html` Accessed on 25-05-2016.
8. Hinrich Schtze Christopher D. Manning, Prabhakar Raghavan. *Text classification and Naive Bayes*. Cambridge University Press, April 2009. `http://informationretrieval.org/` Accessed on 01-09-2016.
9. Matt Hancock. A better confusion matrix with python. `http://notmatthancock.github.io/2015/10/28/confusion-matrix.html`, October 2015. Accessed on 06-10-2016.
10. Nathalie Japkowicz. Performance evaluation for classifiers tutorial. `http://www.slideshare.net/erogol/performance-evaluation-tutorial-46936268`, April 2015. Accessed on 20-07-2016.
11. S Kannan and Vairaprakash Gurusamy. Preprocessing techniques for text mining. `https://www.researchgate.net/profile/Vairaprakash_Gurusamy/publication/273127322_Preprocessing_Techniques_for_Text_Mining/links/54f8319e0cf210398e949292.pdf?origin=publication_detail`. Accessed on 25-05-2016.
12. Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012. `https://www.cs.uic.edu/~liub/FBS/SentimentAnalysis-and-OpinionMining.pdf` Accessed on 20-09-2016.
13. Kevin Markham. Simple guide to confusion matrix terminology. `http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/`, March 2014. Accessed on 20-07-2016.
14. Cambridge University Press. Term frequency and weighting. `http://nlp.stanford.edu/IR-book/html/htmledition/tf-idf-weighting-1.html`, April 2009. Accessed on 10-07-2016.
15. Alexander Rakhlin. Cnn-for-sentence-classification-in-keras. `https://github.com/alexander-rakhlin/CNN-for-Sentence-Classification-in-Keras`, June 2016. Accessed on 20-06-2016.
16. Saed Sayad. Model evaluation - classification. `http://www.saedsayad.com/model_evaluation_c.htm`.
17. scikit-learn developers. sklearn.cross_validation.train_test_split. `http://scikit-learn.org/0.16/modules/generated/sklearn.cross_validation.train_test_split.html`, 2007. Accessed on 05-10-2016.
18. Twitter. Twitter usage. `https://about.twitter.com/company`, June 2016. Accessed on 10-07-2016.
19. S Vijayarani, Ms J Ilamathi, and Ms Nithya. Preprocessing techniques for text mining-an overview. *vol*, 5:7–16. `http://www.ijcscn.com/Documents/Volumes/vol5issue1/ijcscn2015050102.pdf` Accessed on 10-07-2016.
20. Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420, 1997. `http://courses.ischool.berkeley.edu/i256/f06/papers/yang97comparative.pdf` Accessed on 10-06-2016.