17IT034_Sentiment Analysis on Twitter Dataset

ORIGINALITY REPORT

22%	21%	9%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	towardsdatascience.com Internet Source	11%
2	Naveenkumar KS, Vinayakumar R, Soman KP. "Amrita-CEN-SentiDB: Twitter Dataset for Sentimental Analysis and Application of Classical Machine Learning and Deep Learning", 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 2019 Publication	3%
3	www.i-scholar.in Internet Source	2%
4	research.ijcaonline.org Internet Source	1%
5	iieng.org Internet Source	1%
6	Sharvil Shah, Kannan Kumar, Ra. K. Sarvananguru. "Sentimental Analysis of Twitter Data using Classifier Algorithms", International	1%

	Journal of Electrical and Computer Engineering (IJECE), 2016 Publication	
7	www.researchpublications.org Internet Source	1%
8	www.oberlo.com Internet Source	1%
9	aclweb.org Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

SENTIMENT ANALYSIS ON TWITTER DATASET

Mitrajsinh Jadeja[1], Priyanka Patel[1], Hemant Yadav[1]

[1]Smt. Kundaben Dinsha Patel Department of Information Technology,
Chandubhai S. Patel Institute of Technology, Charotar University of Science and Technology, Changa-388421, Gujarat, India.

Article Info

Article history:

Written on: 20/04/2020

Keyword:

Sentiment Analysis Python, Opinion Mining,
Sentiment Analysis Dataset,
Sentiment Analysis Twitter,
Sentiment Analysis Algorithms,
Sentiment Analysis Code,
Machine Learning

ABSTRACT

Microblogging has become one of the most popular way to share one's thought on particular topic or issue and for most people it has become a daily routine. One of most popular microblogging channel is twitter- a convenient as well as quick way to share short posts, article links, GIF's, videos and more. There are 330 million monthly active users and 145 million daily active users on Twitter. Being such a popular platform sentiment analysis on its data is a satisfactory way to know the opinion of the general public. Sentiment analysis meaning is that understanding the emotion of the text whether it is positive, negative or neutral. Sentiment analysis or opinion mining can be useful for various business, literature survey, movie reviews, governing bodies as well as for news reporters. It can be used for knowing the review of a newly launched product or if a new law is passed by the government, to understand the mass opinion by using this. The statistics obtained from the sentiment analysis can be used by television debaters to back his/her argument. In this paper we are going to collect twitter data and perform sentiment analysis on it using various machine learning classifiers like Naive Bayes models(MultinomialNB, BernoulliNB,), Linear models (LogisticRegression, RidgeClassifier, PassiveAggressiveClassifier, Perceptron) , Ensemble models(RandomForest classifier, AdaBoostClassifier) and SVM model(LinearSVC).

Corresponding Author:

Priyanka Patel,
Department of Information Technology,
Chandubhai S Patel Institute of Technology,
CHARUSAT University,
Changa,388421, Gujarat, India
Email: priyankapatel.it@charusat.ac.in

Hemant Yadav
Department of Information Technology,
Chandubhai S Patel Institute of Technology,
CHARUSAT University,
Changa,388421, Gujarat, India
Email: hemantyadav.it@charusat.ac.in

1. INTRODUCTION

Twitter is one of the most booming platform nowadays. It allows a maximum of 280 words in its tweet. As the number of twitter users are increasing , it might be best to analyse its data and get to know about what is going around. Perhaps social media monitoring provides great opportunities for private sectors as well as public sectors.

Sentiment Analysis leads to numerous difficulties and challenges like the language of the tweets, and the abbreviations/short forms used due to word limit. Thus its necessary to build a model that works well with real time data.

Here we classify tweets into three different categories- positive, negative and neutral. The tweets are first collected from the Twitter API using a twitter developer account[1].After that the tweets are cleaned, tweets will be vectorized using CV and TFID. We are going to use unigrams, bigrams and trigrams for vectorization. After that the data will be split into training data, testing data as well as validation data.

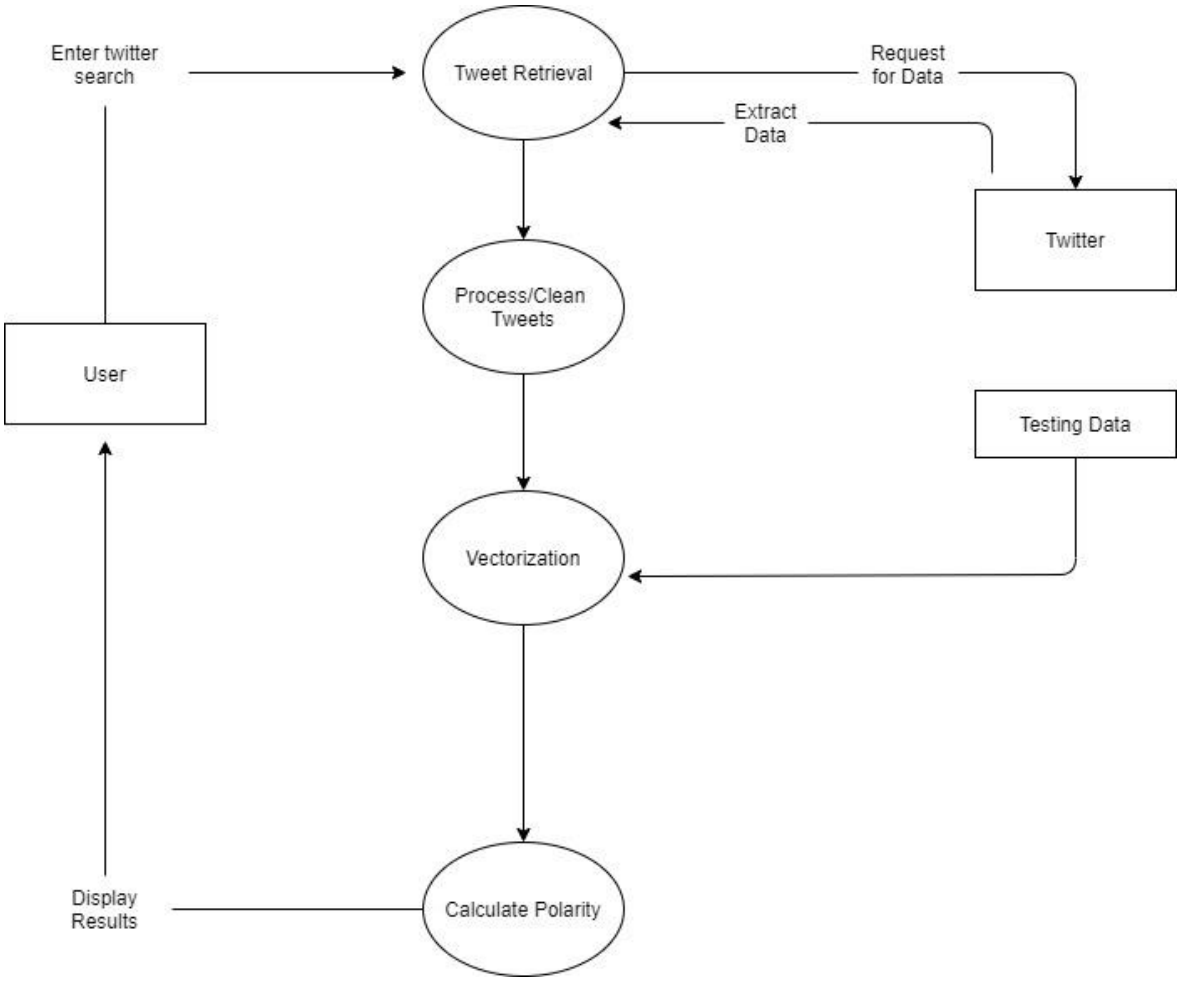

Figure 1 Data Flow Diagram

2. COLLECTING TWEETS

Tweets can be collected using various API's and online datasets are also available. Here Twitter API is used to collect the tweets. In order to use Twitter API one must need to have a Twitter Developer Account. For this one should apply for it through twitter and wait for the confirmation from twitter. For getting a developer account you need to satisfactorily explain the concerned authorities how you are going to use the tweets and how the tweets will be displayed in any other platform or not. Once they are satisfied from your answer they will provide you with Twitter Developer Account. There from you can use the authentication keys and tokens to retrieve the tweets.

3. CLEANING THE TWEETS

The tweets are collected in json format. This data contains many unwanted data like metadata, indices, screen_name. So here keeping only useful fields like full_text, retweet_count, user_follower_count and favorite_count only ie data useful for sentiment analysis. Then this data is converted into a csv(using .tocsv()) format so it is easy to read and understand. Afterwards all the duplicate tweets are removed to make our analysis more accurate. Then lemmatizing and stemming is performed on the data. This means that if a word is written as "happyyyy" then converting it to "happy" meaning correcting the slangs used on social media.A part of dataset before and after cleaning is shown below.

	full_text	retweet_count	user_followers_count	favorite_count
0	full_text	retweet_count	user_followers_count	favorite_count
1	RT @thereisnoearthb: Week 84 Drive 180\r\nð...	4	11	0
2	@Nitinbhai_Patel @vijayrupanibjp @CMOGuj @vneh...	0	182	0
3	RT @SwachhBharatGov: Micro-plastic in the ocea...	70	0	0
4	Every long journey starts with a single step. ...	0	41	0

Figure 2 Raw Tweets

Unnamed: 0.1	full_text	retweet_count	user_followers_count	favorite_count
0	full text	retweet_count	user_followers_count	favorite_count
1	Week Drive Sanjay Vann ...	4	11	0
2	When when and when four days gone...	0	182	0
3	Micro plastic in the ocean can enter our bod...	70	0	0
4	Every long journey starts with a single step ...	0	41	0

Figure 3 Cleaned Tweets

4. POS TAGGING AND SENTIMENT LABELING

The primary cleaning of tweets is done. We are going to use the full_text for feeding our ML classifiers(retweet_count, user_follower_cunt, favorite_count can be used to know the impact of our tweet, but currently it is out of our scope). It is obvious we need to create target variable(sentiment score) for our data. In order to perform this we can use SWN. SWN stands for SentiWordNet, it is a further improved lexical resource explicitly devised for supporting sentiment classification applications. It has a large corpus of POS-tagged English words along with their sentiment.

nnamed: 0.1	full_text	retweet_count	user_followers_count	pos_score	neg_score	sent_score
0	full text	retweet_count	user_followers_count	0.000	0.000	0
1	week drive sanjay vann with punya resham	4	11	0.000	0.000	0
2	when four day gone noth happen next tweet ask ...	0	182	0.500	0.375	1
3	micro plastic ocean enter bodi food amp water ...	70	0	0.000	0.000	0
4	everi long journey start singl step everi clea...	0	41	0.500	0.375	1
...	...	...	...	...	...	...
15177	swachhbharat mangrov marshal complet week mang...	46	1025	0.000	0.000	0

Figure 4 Tweets with POS tagging

5. PERFORM SENTIMENT ANALYSIS

In order to perform sentiment analysis we need to convert our textual data into mathematical form. This means converting the text into something that machine can understand. Here the data is converted in vectorized form. Vectorization is methodology in ML to map words with corresponding vector of real number that is used to find word predictions.

Two methods are used for vectorization are TF-IDF and CV. TF-IDF means “term frequency-inverse document frequency”. The weight assigned to the token is the product of the term(word) frequency and the inverse of document frequency of the word. Thus, if a word appears more often in the tweets, it is assigned less

importance(IDF-weight) in this vectorizer. In count vectorizer we only count the appearance of the word in each text. Also TF-IDF has fractional values as frequencies whereas CV uses only integer. Now before doing vectorization the data we will also use n-grams. It is basically the combination of the adjacent words we have. We are going to use unigrams, bigrams and trigrams. For example all combinations for the sentence “I love food”: unigrams-(“I”,”love”,”food”), bigrams- (“I love”,”love food”) and trigrams- (“I love food”).

Hence vectorize all the three unigrams, bigrams and trigrams of the tweets with both TF-IDF and CV, this means we will have a total of 6 datasets to test upon. The data is then split into training and testing data. The division is as follows — 90% of training data, 5% validation data, and 5% test data. More proportion of data is for training, keeping in mind the less number of tweets. After this we apply various machine learning algorithms to train our model and find the model with best accuracy.The classifiers used for sentiment analysis are Naive Bayes models(MultinomialNB, BernoulliNB,), Linear models (LogisticRegression, RidgeClassifier, PassiveAggressiveClassifier, Perceptron) , Ensemble models(RandomForest classifier, AdaBoostClassifier) and SVM model(LinearSVC).

The output obtained is as followed:-

	Vec_Gram	Classifier	Ac	crossval_train_score_mean	crossval_test_score_mean	crossval_train_score_std	crossval_test_score_std
0	cv_1	MultinomialNB()	0.7875	0.810018	0.718865	0.005280	0.041994
1	cv_1	BernoulliNB()	0.8125	0.841413	0.740350	0.003450	0.036397
2	cv_1	LogisticRegression()	0.8500	0.893044	0.776303	0.004010	0.039173
3	cv_1	LinearSVC()	0.8375	0.995076	0.818496	0.001296	0.032377
4	cv_1	AdaBoostClassifier()	0.7250	0.741305	0.689090	0.012827	0.036096
5	cv_1	RidgeClassifier()	0.8500	0.981918	0.815742	0.001727	0.036661
6	cv_1	PassiveAggressiveClassifier()	0.8375	1.000000	0.826145	0.000000	0.029044
7	cv_1	Perceptron()	0.8625	0.999153	0.817845	0.000539	0.030063
8	cv_1	RandomForest Classifier	0.8750	1.000000	0.810958	0.000000	0.028907
9	cv_2	MultinomialNB()	0.8250	0.862727	0.738961	0.002659	0.034057
10	cv_2	BernoulliNB()	0.7500	0.804094	0.696020	0.003661	0.045366
11	cv_2	LogisticRegression()	0.8625	0.910434	0.785326	0.002158	0.039555
12	cv_2	LinearSVC()	0.9000	0.999154	0.832385	0.000414	0.039280
13	cv_2	AdaBoostClassifier()	0.7500	0.757003	0.697428	0.012974	0.020781
14	cv_2	RidgeClassifier()	0.8750	0.994691	0.823376	0.000938	0.039137
15	cv_2	PassiveAggressiveClassifier()	0.8750	1.000000	0.837907	0.000000	0.041172
16	cv_2	Perceptron()	0.8750	0.999769	0.837902	0.000353	0.039074
17	cv_2	RandomForest Classifier	0.8375	0.999923	0.803338	0.000231	0.039751
18	cv_3	MultinomialNB()	0.8500	0.893198	0.744511	0.003757	0.034598
19	cv_3	BernoulliNB()	0.7250	0.794707	0.686327	0.003715	0.047374
20	cv_3	LogisticRegression()	0.8500	0.913358	0.782543	0.002365	0.041864

	Vec_Gram	Classifier	Ac	crossval_train_score_mean	crossval_test_score_mean	crossval_train_score_std	crossval_test_score_std
21	cv_3	LinearSVC()	0.9000	0.999154	0.828922	0.000414	0.035455
22	cv_3	AdaBoostClassifier()	0.6750	0.752154	0.695991	0.014213	0.018767
23	cv_3	RidgeClassifier()	0.8875	0.996999	0.826849	0.000803	0.036248
24	cv_3	PassiveAggressiveClassifier()	0.8750	1.000000	0.835833	0.000000	0.041639
25	cv_3	Perceptron()	0.8375	1.000000	0.834454	0.000000	0.034586
26	cv_3	RandomForest Classifier	0.8500	1.000000	0.813740	0.000000	0.035816
27	tf_1	MultinomialNB()	0.8250	0.893429	0.764564	0.004364	0.024119
28	tf_1	BernoulliNB()	0.8125	0.841413	0.740350	0.003450	0.036397
29	tf_1	LogisticRegression()	0.8375	0.996999	0.817141	0.001113	0.023501
30	tf_1	LinearSVC()	0.8500	0.999923	0.817170	0.000231	0.033304
31	tf_1	AdaBoostClassifier()	0.7500	0.744153	0.683578	0.021093	0.036332
32	tf_1	RidgeClassifier()	0.8125	0.998307	0.815067	0.000462	0.026850
33	tf_1	PassiveAggressiveClassifier()	0.7750	0.999923	0.813003	0.000231	0.028376
34	tf_1	Perceptron()	0.8125	0.998923	0.815048	0.001342	0.034538
35	tf_1	RandomForest Classifier	0.8750	1.000000	0.799171	0.000000	0.030972
36	tf_2	MultinomialNB()	0.8375	0.939982	0.795019	0.002224	0.019816
37	tf_2	BernoulliNB()	0.7500	0.804094	0.696020	0.003661	0.045366
38	tf_2	LogisticRegression()	0.8750	0.999615	0.828922	0.000385	0.039469
39	tf_2	LinearSVC()	0.8375	1.000000	0.835876	0.000000	0.033846
40	tf_2	AdaBoostClassifier()	0.7750	0.762624	0.708405	0.017819	0.022553
41	tf_2	RidgeClassifier()	0.8375	1.000000	0.831006	0.000000	0.033553
42	tf_2	PassiveAggressiveClassifier()	0.8500	1.000000	0.828238	0.000000	0.030458
43	tf_2	Perceptron()	0.8250	0.999923	0.824784	0.000231	0.019173
44	tf_2	RandomForest Classifier	0.8875	1.000000	0.806102	0.000000	0.035848
45	tf_3	MultinomialNB()	0.8500	0.947753	0.799181	0.002338	0.025297
46	tf_3	BernoulliNB()	0.7250	0.794707	0.686327	0.003715	0.047374
47	tf_3	LogisticRegression()	0.8750	0.999846	0.828242	0.000308	0.037000
48	tf_3	LinearSVC()	0.8375	1.000000	0.832395	0.000000	0.035550
49	tf_3	AdaBoostClassifier()	0.7750	0.764933	0.709784	0.018003	0.025025
50	tf_3	RidgeClassifier()	0.8375	1.000000	0.831695	0.000000	0.041612
51	tf_3	PassiveAggressiveClassifier()	0.8750	1.000000	0.822706	0.000000	0.033633
52	tf_3	Perceptron()	0.8500	1.000000	0.833070	0.000000	0.026211
53	tf_3	RandomForest Classifier	0.8750	1.000000	0.804717	0.000000	0.038297

Figure 5 Output of the trainig model

In figure 5, we are vectorizing with count vectorizer unigrams, bigrams and trigrams which is depicted as cv_1, cv_2, and cv_3 respectively. Similarly vectorizing with TF-IDF vectorizer unigrams, bigrams and trigrams which is depicted as tf_1, tf_2, tf_3 respectively. All the vectorized combinations i.e unigrams, biigrams and trigrams are used for training the model with various ML classifiers as mentioned above. Along with that the accuracy score and cross validation score of each model is also mentioned.

Now the model with highest accuracy must be selected for moving further in the project. From the above observations we find that LinearSVC() with CountVectorizer trigrams and bigrams both gives the highest accuracy of 90%. Thus using this model for our testing our data.

The output is displayed below:-

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import LinearSVC
from sklearn.pipeline import make_pipeline
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score,precision_score,recall_score
import time
b=2
st = time.time()
vec = CountVectorizer(ngram_range=(1,b))
model = make_pipeline(vec,LinearSVC())
model.fit(x_train.values.astype('U'),y_train.values.astype('U'))##
labels = model.predict(x_test.values.astype('U'))###
cm = confusion_matrix(y_test.values.astype('U'),labels)
ac = accuracy_score(y_test.values.astype('U'),labels)
pr = precision_score(y_test.values.astype('U'),labels,average=None)
rc = recall_score(y_test.values.astype('U'),labels,average=None)
en = time.time()
print(b,"-gram","\nAccuracy= ",ac,"\nPrecision= ",pr.mean(),"\nRecall= ",rc.mean(),"\nTime taken=",en-st)
print('\n',cm)
```

```
2 -gram
Accuracy=  0.9259259259259259
Precision=  0.9110275689223056
Recall=  0.8991228070175438
Time taken= 0.37096571922302246

 [[10  0  2]
 [ 0 11  1]
 [ 0  3 54]]
```

Figure 6 Testing the best trained model

Here LinearSVC() is used with count vectorizer bigrams and gives us the accuracy of 92.59% for sentiment analysis classification upon testing it using the test dataset. The precision, recall as well as confusion matrix is also displayed which helps us to perform quantitative analysis of our model.

6. CONCLUSION

Here we successfully impleted nine classification algorithms on the dataset created using twitter API and found which gives the best accuracy with dataset present with us. LinearSVC() with CV(CountVectorizer) bigram vectors gives the best accuracy. On observing the confusion matrix(Figure 6) we find that it has higher diagonal values which gives us more confidence about our model and we obtained an accuracy of 92.59%. Further this model can be used with some other data predict its sentiment.

REFERENCES

1. Towards Datascience-Another Twitter sentiment analysis with Python — Part 6 (Doc2Vec) by Ricky Kim
2. Bhaskar Karambelkar's Blog-https://bhaskarvk.github.io/2015/01/how-to-use-twitters-search-rest-api-most-effectively./

3. Twitter as a Corpus for Sentiment Analysis and Opinion Mining By Alexander Pak, Patrick Paroubek.
4. End-to-End Sentiment Analysis of Twitter Data by Apoor v Agarwal and Jasneet Singh Sabharwal.
5. Shah, Sharvil, Kannan Kumar, and Ra K. Saravanaguru. "Sentimental Analysis of Twitter Data Using Classifier Algorithms." International Journal of Electrical & Computer Engineering (2088-8708) 6.1 (2016).
6. Agarwal, Apoorv, et al. "Sentiment analysis of twitter data." Proceedings of the Workshop on Language in Social Media (LSM 2011). 2011.
7. Agarwal, Apoorv, et al. "Sentiment analysis of twitter data." Proceedings of the Workshop on Language in Social Media (LSM 2011). 2011.

	Journal of Electrical and Computer Engineering (IJECE), 2016 Publication	
7	www.researchpublications.org Internet Source	1%
8	www.oberlo.com Internet Source	1%
9	aclweb.org Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		