VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

B. TECH FALL SEMESTER 2020-2021

ARTIFICIAL INTELLIGENCE

(CSE3013)

REVIEW – 3

J - COMPONENT

SENTIMENT ANALYSIS OF TEXT EMOTION ON TWITTER

Submitted by:

KRATU SHARMA (18BCE0642)
YASHVARDHAN KHEMKA (18BCE0735)

School of Computer Science and Engineering (SCOPE)

ABSTRACT

Sentiment analysis is an essentially significant tool in social media monitoring as it allows us to gain an overview of the wider public opinion behind certain topics. Social media monitoring tools, for example, Brand watch Analytics make the process quicker and easier than ever before, because of Realtime monitoring capabilities. The applications of sentiment analysis are broad and powerful as it helps big companies to decide the nature of their customers and implement changes. The ability to extract insights from social data has been a practice which is being widely adopted by organisations across the world. Shift in the sentiments on social media have been shown to correlate with shifts in the stock market. Governments use sentiment analysis to gauge public opinion to policy announcements and campaign messages ahead of 2012 presidential election. Being able to quickly see the sentiment behind everything from forum posts to news articles means being better able to strategize and plan for the future. It can also be an essential part of one's market research and customer service approach. Not only can you see what people think of your own products or services, you can see what they think about your competitors too. The overall customer experience of your users can be revealed quickly with sentiment analysis, but it can get far more granular too. This paper targets to enact on such a tool for all purpose utility. Twitter is one such platform, where public expresses their opinion freely and in a vast quantity. Analysing the reactions on twitter, can give a true opinion analysis of majority of the people. There are various social media platforms where people express their views and opinions. Sentiment analysis helps us to get a detailed review of this huge information in an organised way which would otherwise get wasted.

INTRODUCTION

Sentiment analysis, which is also known as opinion mining, opinion extraction, sentiment mining or subjectivity analysis, is the process of analysing if a piece of online writing (social media mentions or blog posts or news sites, or any other piece) expresses negative, positive, or neutral attitude.

Why sentiment analysis?

• Business: In marketing field, the companies use it to develop their strategies or to understand customers' feelings towards products or brand, how people respond to their campaigns or product launches; and why do the consumers not buy some products.

• Politics: In the political field, it is used to keep track of political view and to detect consistency and inconsistency between statements and actions at the government level. It can also be used to predict election results.

• Public Actions: Sentiment analysis can also be used to analyse and monitor social phenomena, for the spotting of potentially dangerous situations and determining the general mood of the blogosphere.

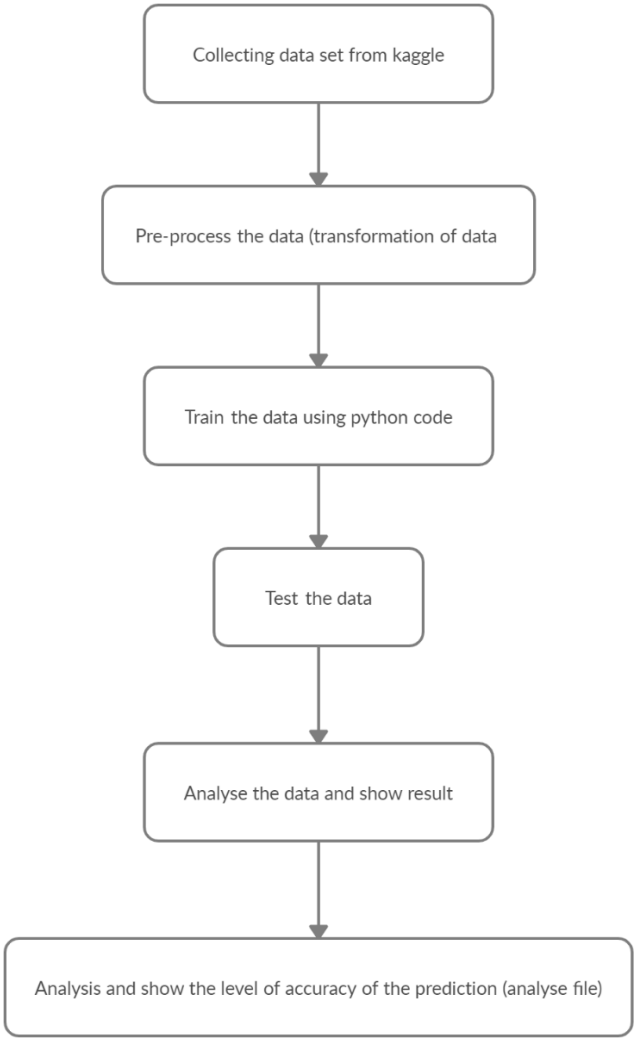
In this project we will:

1. Collect data set from Kaggle
2. Pre-process the data (transformation of data)
3. Train the data using python code
4. Test the data
5. Analyse the data and show result
6. Analyse and show the level of accuracy of the prediction (analyse file)

NOVELTY

The novelty in our project lies in the use of a hybrid-based approach for sentiment analysis. We have not only checked the success rate of different algorithms but also implemented an algorithm to our own understanding and incorporated an aspect that allows us to judge Sentiment in the tweets. In order to calculate the overall polarity of a post, we referred to the previous researches and added some value by introducing new metrics. This sentiment Analysis Model which we have created can be used for many purposes for e.g.: - Election Outcomes, Prediction of movie by the reviews, to know about the opinion on certain topic by analysing comments on social media, Video rating on YouTube etc.

PROJECT FLOW



LITERATURE REVIEW

Algorithm used is based on CNNs built on LSTM for text feature extraction under the deep learning framework. The CNNs built on LSTM model applies convolutional filters of CNNs repeatedly operate on the output matrix of LSTM to obtain robust text feature vector each component of CNNs and LSTM. Dataset is the text extraction from the Wiki dump on 20th November, 2018 and conversion of extracted origin from traditional Chinese to simple Chinese to build the corpus database, noted as CCDatabase. There are 2 main issues in Recurrent neural networks (RNNs). 1) Vanishing Gradient Problem: it is like the activation of one ANN is input to the next ANN (in time) where there are many such linked ANNs (in time). Actually, the output activation is again, fed as input to the same ANN. So, it is like a replica of the same ANN in time steps. 2) Similarly, there is issue of increasing gradients at each step called as exploding gradients. These are solved by gradient clipping i.e. Set an upper and lower threshold. Major highlights were Mean squared error, confusion matrix, true positive rate, true negative rate, false positive rate, false negative rate, precision, accuracy. [1]
Algorithm used was Priority weight based deep learning approaches like Convolutional-Long Short-Term Memory (C-LSTM) and Stacked- Long Short-term Memory (SLSTM) is explored. Dataset used was from Amazon.com has been chosen. The collected data is classified into different domains such as Musical Instrument, Automotive, Instant Video, Office Product, Digital Music etc. A major source of limitation is due to imbalanced training data.Most of review data belong to positive class whereas only 5~10% data belong to neutral and negative classes. Limitation of few records does not pose major difficulty in the proposed model. Major Highlights were exponential functions ω that have the property of geometric progression are used and w represents common ratio which is bigger than 0 and smaller than 1. Since the w is smaller than 1, weight value (=rn has bigger value when is smaller and smaller n represents the domain of recent time-series data.[2]
Algorithm used was Artificial Neural Network (Ann), Random Forest, Support Vector Machine, Genetic Algorithm, Naïve Bayes, Decision Tree. The main source of data is Twitter with seventeen (17) articles, followed by movie reviews with eight (8) articles, Amazon with six (6) articles, blogs with five (5) articles, media with four (4), YouTube with three (3), and TripAdvisor with three (3)

articles. There are also nineteen (19) articles that can be categorized as others with one (1) article from each. The heterogeneous characteristics of big datarelated challenges and the future directions needed to develop a method that can effectively deal with the heterogeneous characteristics of big data. The second challenge related to users' network (future researches need to further investigation on how to make use of user' relationship network for enriching the output of the sentiment system). Third challenge related to analysing sparse, uncertain, and incomplete data and future researches are suggested to focus on developing sentient analysis method which can effectively be able to handle sparse, uncertain, and incomplete data. Lastly Semantic relations in multi-source data fusion. Major Highlights were The results show that sentiment analysis features slightly boost the performance of ADR mention in both tweets and health-related posts in the forum, accuracy of the proposed sentiment drove stock price model is approximately 2 percent better than the model, which only uses historical prices, suggested that combining viewer comments and marketing properties assisted in improving prediction about box office performance.[3]
Methodology used is Deep Learning technique, CNN, RNN and Long short term memory. Dataset used was Domain specific private dataset, public data set containing users from Amazon, IMDB, and Yelp. Performance measure includes F1 score is analysed that is harmonic mean of precision and recall
F1- score= 2(P*R)/(P+R)
P=TP/(TP+FP) {Precision}
R=TP/(TP+FN) {Recall}
Limitations involved Risk of misclassifications while applying the final prediction. [4]
Methodology used Vector representations for language have been shown to be useful in a number of Natural Language Processing tasks. In this paper, we aim to investigate the effectiveness of word vector representations for the problem of Sentiment Analysis. In particular, we target three subtasks namely sentiment words extraction, polarity of sentiment words detection, and text sentiment prediction. We investigate the effectiveness of vector representations over different text data and evaluate the quality of domain dependent vectors. Multiple Dataset of App review used. Performance measure was Naive Bayes Classifier, CBOW model and Skip-gram model with Hierarchical SoftMax. Limitations were Work will consider extracting the aspects of the entity described in the text through some methods, for example, LDA or Bootstrapping method. We analyse the sentiment of aspects and make more meaningful researches according to the

sentiment of the aspects. For example, the comment “good, user friendly. Blue UI is very beautiful and charming, pictures are very smooth, the speed is also very quick.”, we can extract the “UI”, “picture” and “speed” as aspects which the sentiment can be analysed.[5]

Methodology used Automatic sentiment analysis provides an effective way to gauge public opinion on any topic of interest. However, most sentiment analysis tools require a general sentiment lexicon to automatically classify sentiments or opinion in a text. Dataset used UK energy sector date sets. Performance measure was Sentiment lexicon, Limitations were Topic modelling shows that there is substantial difference in terms of topics being discussed in the tweets revolving around the use of renewable energy [6]

Methodology used was Sentiment based approaches found in the literature include dependency tree based method [21], target dependency based [22], graph based approach [23], ontology based [25]; social relations orientation [27], CNN based [29] and Twin Transfer Learning (TTL). Dataset collected using Twitter API from Twitter accounts like @RailMinIndia and @IR. A prototype application is built to demonstrate proof of the concept. Performance measure was Confusion matrix, true positive rate, true negative rate, false positive rate, false negative rate, precision, accuracy, FPR performance analysis Limitations were Supervised learning methods used in experiments are presented in horizontal axis. The vertical axis on the other hand shows the performance of the algorithms in terms of accuracy, precision, recall and F-Measure. Highest accuracy is exhibited by SVM. Random Forest and SVM showed highest precision and recall respectively. SVM shows highest F- Measure over all other algorithms. SVM has higher performance in sentiment classification. [7]

Methodology used here was Naïve Bayes Algorithm, Maximum Entropy Classifier and Support Vector Machines. Dataset was Domain specific private dataset, public data set. Performance measures involved Test many features to find Bigrams, Unigrams, position of words and Parts of Speech (POS) used as features in these techniques. Limitations of this project was Risk of misclassifications while applying the final prediction [8]

Methodology involved Three alternative algorithms (i.e. different f’s) are used to assign a score to a unary AAC. 1) Variable Scoring: the weight with which an adverb is considered depends upon the score of the adjective that it is associated with 2) Adjective Priority Scoring: we select a weight $r \in [0, 1]$. This weight denotes the importance of an adverb in comparison to an adjective that it modifies. r can vary based on different criteria. 3) Adverb First Scoring: we take

the complementary view that instead of weighting the adverb, we should modify the adverb score by weighting the adjective score using an r (as before) that measures the weight of an adjective's importance in an AAC, relative to the importance of the adverb. Dataset used Manually identified 3 topics in each document in the experimental dataset, and asked about 10 students (not affiliated with this paper) to rank the strength of sentiment on each of the three topics associated with each document. Performance Measures The first experiment compared just the algorithms described in this paper in order to determine which one exhibits the best performance. More specifically, we were interested in finding out the value of r that makes AP Sr and advF Sr provide the best performance. The performance of an algorithm is based on the use of Pearson correlation coefficients between the opinion scores returned by the algorithm and the opinion scores provided by the same of human subjects. [9]

Methodology involved here is For the determining the polarity of data the algorithm of RNN(Recurrent Neural Network) is used because it is termed as the best algorithm as it is inspired by biological neural network and NLP, for improving the competence power and accuracy of machine we introduce Stanford library, The data is fetched from Twitter in order to perform analysis of sentiments. Performance involved It is concluded that there was an accuracy of more than 90% by using the RNN(Recurrent neural network) algorithm as when we classify the data , the amount of data that was taken in account to be classified was more than the quantity of data that is usually taken by other research papers because in this project we used the Stanford library and google translator which will in turn increase the competence power of machine as we are providing a whole set of English sentences to it to learn and understand so that it doesn't neglect any data and the accuracy that we receive will be maximum than others. [10]

LIBRARIES AND CLASSES USED

1. Pandas

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

2. Pyplot

matplotlib.pyplot is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

3. Logistic Regression

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

4. SGD Classifier

Stochastic Gradient Descent (SGD) is a simple yet very efficient approach to fitting linear classifiers and regressors under convex loss functions such as (linear) Support Vector Machines. Even though SGD has been around in the machine learning community for a long time, it has received a considerable amount of attention just recently in the context of large-scale learning. SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing. Given that the data is sparse, the classifiers in this module easily scale to problems with more than 10^5 training examples and more than 10^5 features.

5. Accuracy_score (Accuracy classification score)

In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.

6. Confusion Matrix

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix. A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm. It allows easy identification of confusion between classes e.g. one class is commonly mislabelled as the other. Most performance measures are computed from the confusion matrix.

7. RandomForestClassifier

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

8. MultinomialNB

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

9. TextBlob

TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

MACHINE LEARNING MODELS USED

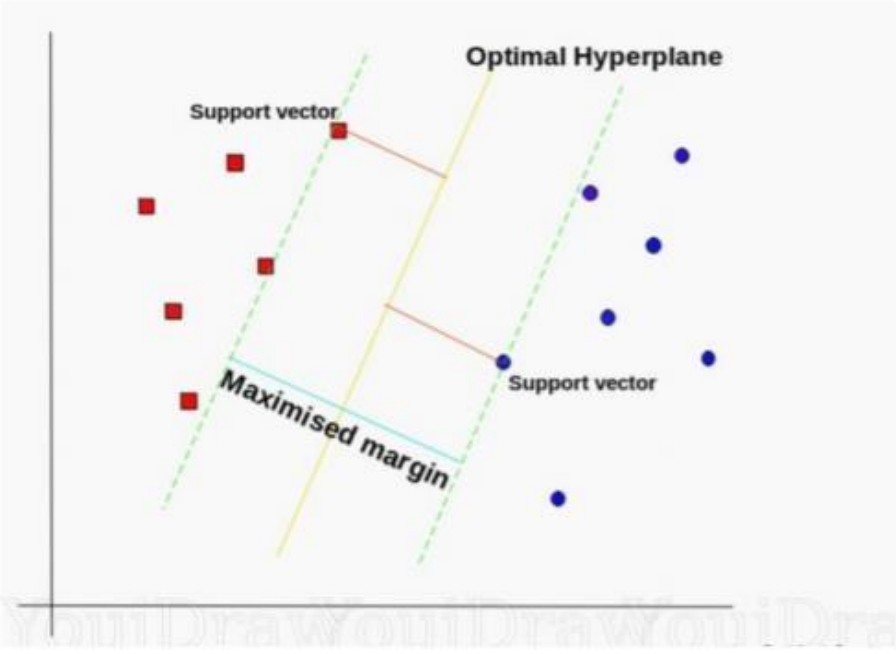
1. **SVM**
2. **Logistic regression**
3. **Random Forest**
4. **Naive bayes**
5. **Text blob**

SUPPORT VECTOR MACHINE:

Support vector machines so called as SVM is a supervised learning algorithm which can be used for classification and regression problems as support vector classification (SVC) and support vector regression (SVR). It is used for smaller dataset as it takes too long to process. In this set, we will be focusing on SVC.

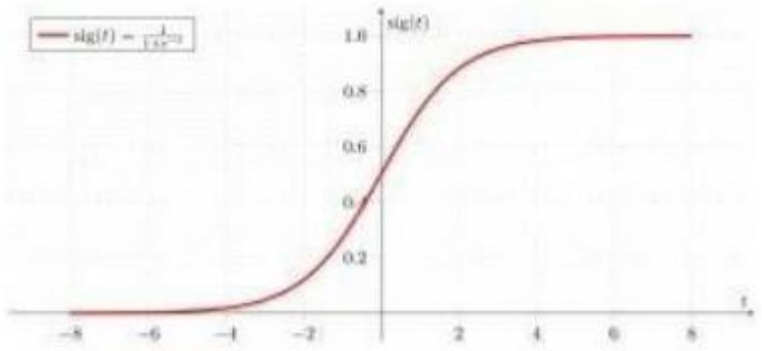
The Ideology behind SVM:

SVM is based on the idea of finding a hyperplane that best separates the features into different domains.



LOGISTIC REGRESSION:

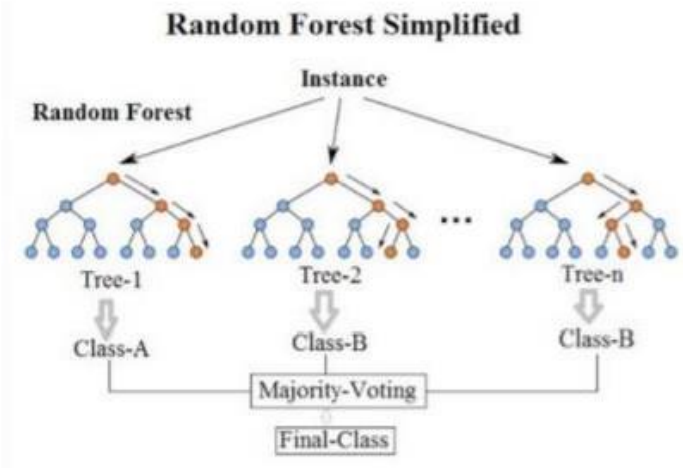
Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable (or output), y, can take only discrete values for given set of features (or inputs), X.



Contrary to popular belief, logistic regression IS a regression model. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as “1”. Just like Linear regression assumes that the data follows a linear function, Logistic regression models the data using the sigmoid function.

RANDOM FOREST:

Decision trees are a popular method for various machine learning tasks. Tree learning "come[s] closest to meeting the requirements for serving as an off-the-shelf procedure for data mining", say Hastie et al., "because it is invariant under scaling and various other transformations of feature values, is robust to inclusion of irrelevant features, and produces inspectable models. However, they are seldom accurate"



In particular, trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, i.e. have low bias, but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model.

NAÏVE BAYES:

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers

TEXT BLOB:

Text Blob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

IMPLEMENTATION

In [1]:
```
pip install -U textblob
```

```
Requirement already up-to-date: textblob in c:\users\hp\anaconda3\lib\site-packages (0.15.3)
Requirement already satisfied, skipping upgrade: nltk>=3.1 in c:\users\hp\anaconda3\lib\site-packages (from textblob) (3.4.5)
Requirement already satisfied, skipping upgrade: six in c:\users\hp\anaconda3\lib\site-packages (from nltk>=3.1->textblob) (1.12.0)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import nltk
nltk.download('wordnet')
data = pd.read_csv('text_emotion.csv')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\hp\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Here importing Word from text blob

Imported pandas to manipulate our data, converting the csv file to data frame so that we can use it for Models.

Pyplot is used to plot graph in python, we will use it to plot a bar graph for comparison of the models.

In [3]:
```
data = data.drop('author', axis=1)
data
```

Out[3]:

	tweet_id	sentiment	content
0	1956967341	empty	@tiffanylue i know i was listenin to bad habi...
1	1956967666	sadness	Layin n bed with a headache ughhhh...waitin o...
2	1956967696	sadness	Funeral ceremony...gloomy friday...
3	1956967789	enthusiasm	wants to hang out with friends SOON!
4	1956968416	neutral	@dannycastillo We want to trade with someone w...
...	...	...	...
39995	1753918954	neutral	@JohnLloydTaylor
39996	1753919001	love	Happy Mothers Day All my love
39997	1753919005	love	Happy Mother's Day to all the mommies out ther...
39998	1753919043	happiness	@niariley WASSUP BEAUTIFUL!!! FOLLOW ME!! PEE...
39999	1753919049	love	@mopedronin bullet train from tokyo the gf ...

40000 rows × 3 columns

Author column is dropped because the analysis is not dependent on author name or id.

```
In [4]: data  = data.drop(data[data.sentiment == 'anger'].index)
        data  = data.drop(data[data.sentiment == 'boredom'].index)
        data  = data.drop(data[data.sentiment == 'enthusiasm'].index)
        data  = data.drop(data[data.sentiment == 'empty'].index)
        data  = data.drop(data[data.sentiment == 'fun'].index)
        data  = data.drop(data[data.sentiment == 'relief'].index)
        data  = data.drop(data[data.sentiment == 'surprise'].index)
        data  = data.drop(data[data.sentiment == 'love'].index)
        data  = data.drop(data[data.sentiment == 'hate'].index)
        data  = data.drop(data[data.sentiment == 'neutral'].index)
        data  = data.drop(data[data.sentiment == 'worry'].index)

In [5]: data['content'] = data['content'].apply(lambda x: " ".join(x.lower() for x in x.split()))

        data['content'] = data['content'].str.replace('[^\w\s]',' ')

        from nltk.corpus import stopwords
        stop = stopwords.words('english')
        data['content'] = data['content'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))

In [6]: freq = pd.Series(' '.join(data['content']).split()).value_counts()[-10000:]

        freq = list(freq.index)
        data['content'] = data['content'].apply(lambda x: " ".join(x for x in x.split() if x not in freq))
```

Now we are processing data, making it to lower case and then splitting it to form a list.
Replacing Character which is not Alphabet or numbers or small with null
Then we import stopwords, here we are deleting stopwords from the data.
Stopwords are common words of English language like a, the, in etc.

So, no need of stop words in Sentiment Analysis.

```
In [7]: #Encoding output labels 'sadness' as '1' & 'happiness' as '0'
        from sklearn import preprocessing
        lbl_enc = preprocessing.LabelEncoder()
        y = lbl_enc.fit_transform(data.sentiment.values)

        from sklearn.model_selection import train_test_split
        X_train, X_val, y_train, y_val = train_test_split(data.content.values, y, stratify=y, random_state=42, test_size=0.1, shuffle=Tru
        X_train

Out[7]: array(['already get tax money winning pole position ha ha',
               'please please anyone', 'good morning', ..., 'oh yes',
               'posting late got back seeing star trek awesome',
               'sad otalia today'], dtype=object)
```

Here pre-processing of data is taking place, here sentiment is in words it is now being converted to number and then allotted to variable y.

Then we are splitting the data into Training and testing 90% training and 10% testing.

```
In [8]: # Extracting TF-IDF parameters
        from sklearn.feature_extraction.text import TfidfVectorizer
        tfidf = TfidfVectorizer(max_features=1000, analyzer='word',ngram_range=(1,3))
        X_train_tfidf = tfidf.fit_transform(X_train)
        X_val_tfidf = tfidf.fit_transform(X_val)
```

This is for tokenizing of data.

Now we are importing Tfid Vectorizer.
Convert a collection of raw documents to a matrix of TF-IDF features.

TF-IDF stands for “Term Frequency — Inverse Document Frequency”. This is a technique to quantify a word in documents, we generally compute a weight to each word which signifies the importance of the word in the document and corpus.
This method is a widely used technique in Information Retrieval and Text Mining.
So, we have declared variable and transformed to tfidf matrix both for test and train.

```
In [9]: # Extracting Count Vectors Parameters
        from sklearn.feature_extraction.text import CountVectorizer
        count_vect = CountVectorizer(analyzer='word')
        count_vect.fit(data['content'])
        X_train_count =  count_vect.transform(X_train)
        X_val_count =  count_vect.transform(X_val)
```

So that when we compare test and train should be of equal units.

We are also going to use count vectorizer.
The Count Vectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

So, we will tokenize data in two forms one is tfidf, the other is count vectorizer

Models using the TF-IDF features

Model 1: Multinomial Naive Bayes Classifier

```
In [10]: from sklearn.metrics import accuracy_score
         from sklearn.metrics import confusion_matrix
         from sklearn.naive_bayes import MultinomialNB
         accuracy=[]
         nb = MultinomialNB()
         nb.fit(X_train_tfidf, y_train)
         y_pred = nb.predict(X_val_tfidf)
         accuracy.append(accuracy_score(y_pred, y_val)*100)
         print('naive bayes tfidf accuracy %s' % accuracy_score(y_pred, y_val))
         confusion_matrix(y_val, y_pred)

         naive bayes tfidf accuracy 0.5038535645472062

Out[10]: array([[223, 298],
                [217, 300]], dtype=int64)
```

Here we have imported accuracy score to calculate accuracy of the different type of models we are going to work on.

We have imported Confusion matrix.
A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.

We are importing our first model i.e. MultinomialNB that is Naïve Bayes model. Now, we train our data through tf-idf in naive bayes.

Model 2: Linear SVM

```
In [11]: from sklearn.linear_model import SGDClassifier
         lsvm = SGDClassifier(alpha=0.001, random_state=5, max_iter=15, tol=None)
         lsvm.fit(X_train_tfidf, y_train)
         y_pred = lsvm.predict(X_val_tfidf)
         accuracy.append(accuracy_score(y_pred, y_val)*100)
         print('svm using tfidf accuracy %s' %accuracy_score(y_pred, y_val))
         confusion_matrix(y_val, y_pred)

         svm using tfidf accuracy 0.5317919075144508

Out[11]: array([[205, 316],
                [170, 347]], dtype=int64)
```

The accuracy score to this list and later will use to plot the graph. Accuracy obtain is 53.17% and above is the confusion matrix.

Here we are importing SGDC i.e. Stochastic Gradient Descent Classifier used in SVM.
So, for tf-idf we are training the data in it and calculating the accuracy score with the confusion matrix.

Model 3: logistic regression

```
In [12]: from sklearn.linear_model import LogisticRegression
         logreg = LogisticRegression(C=1)
         logreg.fit(X_train_tfidf, y_train)
         y_pred = logreg.predict(X_val_tfidf)
         accuracy.append(accuracy_score(y_pred, y_val)*100)
         print('log reg tfidf accuracy %s' % accuracy_score(y_pred, y_val))
         confusion_matrix(y_val, y_pred)

         log reg tfidf accuracy 0.5337186897880539

         C:\Users\hp\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be changed to
         'lbfgs' in 0.22. Specify a solver to silence this warning.
           FutureWarning)

Out[12]: array([[274, 247],
                [237, 280]], dtype=int64)
```

The accuracy score to this list and later will use to plot the graph. Accuracy obtain is 53.37% and above is the confusion matrix.

Here we are importing Logistic Regression So for tf-idf we are training the data in it and calculating the accuracy score with the confusion matrix.

Model 4: Random Forest Classifier

In [13]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=500)
rf.fit(X_train_tfidf, y_train)
y_pred = rf.predict(X_val_tfidf)
accuracy.append(accuracy_score(y_pred, y_val)*100)
print('randon forest tfidf accuracy %s' % accuracy_score(y_pred, y_val))
confusion_matrix(y_val, y_pred)
```

```
randon forest tfidf accuracy 0.535645472061657
```

Out[13]:
```
array([[323, 198],
       [284, 233]], dtype=int64)
```

The accuracy score to this list and later will use to plot the graph. Accuracy obtain is 53.56% and above is the confusion matrix.

Here we are importing Random Forest So for tf-idf we are training the data in it and calculating the accuracy score with the confusion matrix.

Models using count vectors features

Model 1: Multinomial Naive Bayes Classifier

In [14]:
```
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
nb.fit(X_train_count, y_train)
y_pred = nb.predict(X_val_count)
accuracy.append(accuracy_score(y_pred, y_val)*100)
print('naive bayes count vectors accuracy %s' % accuracy_score(y_pred, y_val))
confusion_matrix(y_val, y_pred)
```

```
naive bayes count vectors accuracy 0.766859344894027
```

Out[14]:
```
array([[401, 120],
       [122, 395]], dtype=int64)
```

Here we are training the data for count vectorizer in model Naïve bayes. Then, calculating the accuracy score and confusion matrix

Model 2: Linear SVM

In [15]:
```
from sklearn.linear_model import SGDClassifier
lsvm = SGDClassifier(alpha=0.001, random_state=5, max_iter=15, tol=None)
lsvm.fit(X_train_count, y_train)
y_pred = lsvm.predict(X_val_count)
accuracy.append(accuracy_score(y_pred, y_val)*100)
print('lsvm using count vectors accuracy %s' % accuracy_score(y_pred, y_val))
confusion_matrix(y_val, y_pred)
```

```
lsvm using count vectors accuracy 0.7976878612716763
```

Out[15]:
```
array([[428,  93],
       [117, 400]], dtype=int64)
```

Here we are training the data for count vectorizer in model SVM. Then, calculating the accuracy score and confusion matrix.

Model 3: Logistic Regression ¶

In [16]:
```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(C=1)
logreg.fit(X_train_count, y_train)
y_pred = logreg.predict(X_val_count)
accuracy.append(accuracy_score(y_pred, y_val)*100)
print('log reg count vectors accuracy %s' % accuracy_score(y_pred, y_val))
confusion_matrix(y_val, y_pred)
```

```
log reg count vectors accuracy 0.7832369942196532
```

```
C:\Users\hp\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be changed to
'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[16]:
```
array([[420, 101],
       [124, 393]], dtype=int64)
```

Here we are training the data for count vectorizer in model Logistic Regression. Then, calculating the accuracy score and confusion matrix.

Model 4: Random Forest Classifier

In [17]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=500)
rf.fit(X_train_count, y_train)
y_pred = rf.predict(X_val_count)
accuracy.append(accuracy_score(y_pred, y_val)*100)
print('random forest with count vectors accuracy %s' % accuracy_score(y_pred, y_val))
confusion_matrix(y_val, y_pred)
```

```
random forest with count vectors accuracy 0.7601156069364162
```

Out[17]:
```
array([[407, 114],
       [135, 382]], dtype=int64)
```

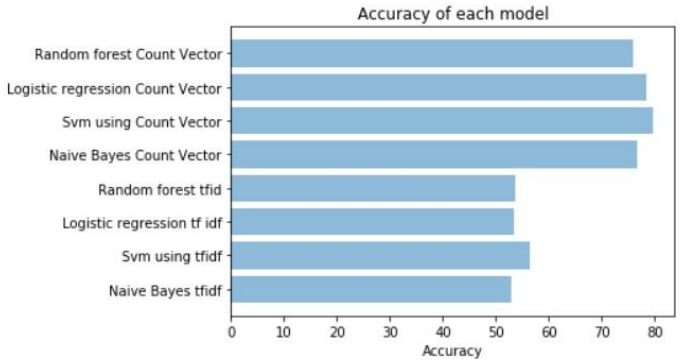
Here we are training the data for count vectorizer in model Random Forest. Then, calculating the accuracy score and confusion matrix.

RESULTS

Graph based analysis

In [18]:
```
objects = ('Naive Bayes tfidf', 'Svm using tfidf',
           'Logistic regression tf idf', 'Random forest tfid',
           'Naive Bayes Count Vector', 'Svm using Count Vector',
           'Logistic regression Count Vector', 'Random forest Count Vector')
y_pos = np.arange(len (objects))
plt.barh(y_pos, accuracy, align='center', alpha=0.5)
plt.yticks (y_pos, objects)
plt.xlabel('Accuracy')
plt.title('Accuracy of each model')

plt.show()
```



Here, we are plotting a bar graph for all the model and tokenization method used with t and accuracy.
Here, we can see the least accurate model is Random Forest when using the tokenization method TF-IDF
Also, the least accurate Tokenization method is TF-IDF.
Accuracy is more when using count vectorizer.

THE MOST ACCURATE MODEL WITH 79.76 % ACCURACY IS SGDC, SVM WITH COUNT VECTORIZER.

REALITY CHECK

In [19]:
```
tweets = pd.DataFrame(['I am very happy today! The atmosphere looks cheerful',
'Things are looking great. It was such a good day',
'Success is right around the corner. Lets celebrate this victory',
'Everything is more beautiful when you experience them with a smile!',
'Now this is my worst, okay? But I am gonna get better.',
'I am tired, boss. Tired of being on the road, lonely as a sparrow in the rain. I am tired of all the pain I feel',
'This is quite depressing. I am filled with sorrow',
'His death broke my heart. It was a sad day'])

# Doing some preprocessing on these tweets as done before
tweets[0] = tweets[0].str.replace('[^\w\s]',' ')
from nltk.corpus import stopwords
stop = stopwords.words('english')
tweets[0] = tweets[0].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
from textblob import Word
tweets[0] = tweets[0].apply(lambda x: " ".join([Word(word).lemmatize() for word in x.split()]))

# Extracting Count Vectors feature from our tweets
tweet_count = count_vect.transform(tweets[0])

#Predicting the emotion of the tweet using our already trained linear SVM
tweet_pred = lsvm.predict(tweet_count)

print(tweet_pred)
```

```
[0 0 0 0 1 1 1 1]
```

NOW AS WE GOT SGDC WITH COUNT VECTORIZER MAXIMUM ACCURACY. WE ANALYSE SOME SENTENCES. AS WE CAN SEE ABOVE IN THIS CASE

<u>THE MODEL ANALYSED ALL THE SENTENCES WITH THE ACCURACY OF 100%.</u>

CONCLUSION

We built 4 models and we used two type of tokenization technique in our project and we found out that SVM has the highest accuracy with Count Vectorizer while Random Forest model has the lowest accuracy. Over all the accuracy of each model is very good. Further models such as perceptron's, decision trees and ANN can be implemented in the future to achieve better prediction accuracy.

Apart from twitter dataset, Our models can be applied to different areas such as Facebook comment analysis, flight reviews analysis etc and the applications of such models are limitless. With this we would like to conclude this project.

REFERENCES

[1] Gao, Jinfeng, Yao, Ruxian, Lai, Han, Chang, Ting-Cheng : Sentiment analysis with CNNs built on LSTM on tourists comments , 2019 IEEE Eurasia Conference on Biomedical Engineering, Healthcare and Sustainability (ECBIOS) Biomedical Engineering, Healthcare and Sustainability (ECBIOS), 2019 IEEE Eurasia Conference on. :108-111 May, 2019.

[2] Yeo, I., Balachandran, K.: Sentiment analysis on time-series data using weight priority method on deep learning , 2019 International Conference on Data Science and Communication, IconDSC 2019, 2019 International Conference on Data Science and Communication, IconDSC 2019. (2019 International Conference on Data Science and Communication, IconDSC 2019, March 2019)

[3] Shayaa, S., Jaafar, N.I., Bahri, S., Sulaiman, A., Seuk Wai, P., Wai Chung, Y., Piprani, A.Z., Al-Garadi, M.A.: Sentiment analysis of big data: Methods, Applications, And Open Challenges, IEEE Access. (IEEE Access, 28 June 2018, 6:37807-37827)

[4] Annotation technique for Health-Related Tweets Sentimental Analysis (IEEE 2018)

[5] Xian Fan, Xiaoge Li, Feihong Du, Xin Li,2016: Apply word vectors for sentiment analysis of APP reviews.

[6] Victoria Ikoro, Maria Sharmina, Khaleel Malik, and Riza Batista-Navarro : Analyzing Sentiments Expressed on Twitter by UK Energy Company Consumers. 2018

[7] A Machine Learning based Framework for Sentiment Classification: Indian Railways Case Study (IJITEE ISSN: 2278- 3075, Volume-8 Issue-4, February 2019)

[8] A Survey: Sentiment Analysis Using Machine Learning Techniques for Social Media Analytics (IJPAM International Journal of Pure and Applied Mathematics)

[9] Sentiment Analysis: Adjectives and Adverbs are better than Adjectives Alone(ICWSM ’2006 Boulder, CO USA)

[10] Sentiment Analysis Using RNN And Google Translator (2018 IEEE)