A

DESIGN PROJECT REPORT

on

Applying sentiment analysis for classifying IMDB movie reviews

Submitted By:

M. Revanth (116CS0013)

Paras Varshney (Roll No. 116CS0036)

K. Devi Sai Prashanthi (Roll No. 116CS0009)

Major: Computer Science, IIITDM Kurnool

Supervised By:

Prof. (Dr.) D.V.L.N. Somayajulu

Indian Institute of Information Technology, Design and Manufacturing, Kurnool

Jagannathagattu, Kurnool - 518007

Andhra Pradesh, India

(An Institute of National Importance under Ministry of HRD, Govt. of India)

ACKNOWLEDGMENT

It would be our great pleasure to express our gratitude to our Hon'ble Director **Prof. D.V.L.N Somayajulu** as well as our C.S.E Head of Department **Dr. Sanjaya Kumar Panda** who guided us throughout the complete process of learning and implementing our project and provided us with an opportunity to implement the project on topic of Sentiment Analysis using Natural Language Processing, which helped us to do some really good amount of research and through this we have learnt many new things and learned by gaining exposure to new aspects of machine learning. We are extremely thankful to them for enlightening our path through the complete project journey.

Also, we are highly indebted to Faculty and Staff at the Indian Institute of Information Technology Design & Manufacturing, Kurnool, A.P. for their constant constant supervision and support and guiding us as well as for providing necessary information and materials regarding the project and also for their contribution and support in completing the project.

Revanth Madamala (116CS0013)

Paras Varshney (Roll No. 116CS0036)

K. Devi Sai Prashanthi (Roll No. 116CS0009)

CERTIFICATE

This is to certify that the project entitled "Applying sentiment analysis for classifying IMDB movie reviews" is being submitted by **Mr. Revanth Madamala** (Roll No. **116CS0013**),**Mr. Paras Varshney** (Roll No. **116CS0036**) and **Ms. K. Devi Sai Prashanthi** (Roll No. **116CS0009**) to the Department of Computer Science and Engineering, Indian Institute of Information Technology, Design and Manufacturing, Kurnool, Andhra Pradesh as the Design Project of the 7th semester for the award of the degree of Bachelor of Technology. It is an original research work carried out by them under my supervision and guidance. In my opinion, the project has fulfilled all the requirements as per the regulations of this university and has reached the standard needed for submission. The results embodied in this project has not been submitted to any other university or institute for the award of any other degree or diploma.

_____

Prof. (Dr.) D.V.L.N. Somayajulu
(Director & Project Guide)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. ABSTRACT:

Since mid 90s, utilization of web has augmented in different structures. Individuals are conveying utilizing different appearances. In the past time, message traffic had multiplied on the web. With this colossal development of web traffic distinctive online web based life stages, for example, Facebook, Twitter, LinkedIn, and so forth are likewise getting popular. in this the advanced world, things are changing in a little league and become famous and in vogue over social stages . Various acts of conveying and sharing are not founded on the substance alone yet in addition on premise of reiteration of the substance. In the ongoing days miniaturized scale blogging has become prominent stage for every online client. Billions of clients are imparting their insight on various perspectives on exceptionally stylish also, famous sites, for example, Facebook, tumbler, twitter, glint, LinkedIn and so forth. Twitter is the most well known small scale blogging and person to person communication administration which gives office to clients to sharing, conveying and translating 140 words' post known as tweet. Twitter has 320 Million month to month dynamic client. it is open through site interface, SMS, or portable gadgets. Characteristic language handling is additionally assuming a major job and can be utilized by the suppositions communicated. In this project, we are trying to analyze the users' sentiment on tweets collected from different sources on the IMDB movie reviews. The project successfully predicts if the sentiment of review will be of a negative or a positive polarity.We came across different methods of implementations and identified various techniques through which we can define the features of the reviews considering it either as single entities or considering it as a whole document. We did background research on various preprocessing techniques and found certain research paper "Role of text pre- processing on twitter sentiment analysis" more intriguing than any other and the significance is discussed in detail in research paper analysis section and along with that we implemented a technique called distributed representation of sentences and documents which ultimately gave us the highest accuracy among all the models we implemented.

2. INTRODUCTION:

2.1. PROBLEM STATEMENT:

Application of Sentiment Analysis techniques on IMDB movie review dataset for Classifying the sentiment of movie reviews posted by the audience.

2.2. PROJECT OVERVIEW:

The approach followed in this project is as follows:

1. Data Collection: Complete information is gathered from the Andrew Maas, Stanford University. The dataset is for twofold assessment grouping containing significantly a greater number of information than the other benchmark datasets. He gives a lot of 25K profoundly polar movie surveys to train the model and 25K audit information to test the model. There is extra unlabeled information for use too which can be used for other binary classifications. Crude content and a previously processed and handled bag of words groups are also collected.

2. Data Preprocessing: Data processing is among the most crucial steps in the complete machine learning project. The data is cleaned and processed to make it fit for learning and training the model. The cleaning process is very crucial as the data which is not clean can never give a good result even when tested on a perfect machine learning model. But well clean and wrangled data even when trained and tested on an average model will give a decent result. So we cleaned the data using BeautifulSoup library and by using the bag of words approach and count vectorizer we made the data clean and ready to be trained.

3. Feature Selection: Bag of word (BoW) is an approach for most of the NLP and ML techniques.Apart from BOW we have used Word2Vec(both CBOW and Skip n-gram) along with this we have tried Doc2Vec in which we implemented PV-DM and PV-DBOW

4. Model Building: For this project we trained ml models using Linear Support Vector Machine Classifier, Random Forest Classifier, word2vec and doc2vec. These machine learning algorithms are some of the most powerful approaches to work with Natural Language Processing problems.

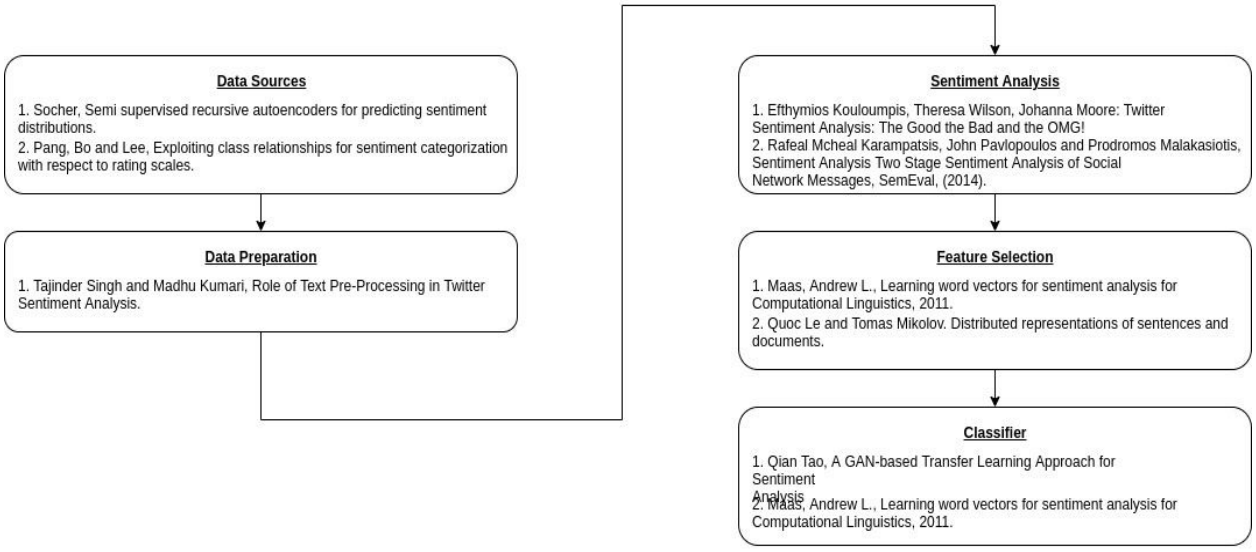

Fig 2.1 Flow Chart Representation of project overview

2.3. SENTIMENT ANALYSIS:

It is a contextual text mining technique which finds and explores information in the source material to help the business by understanding the sentiment of the product socially, brand/service while surveilling the online conversation. However, the social media stream analysis is restricted to just fundamental sentiment analysis and metric based on counts. This is similar to scratching the surface and neglecting the most important insights about the data that had been waiting to be found in the analysis.

It is the most popular text classification tool which tests incoming messages and tells whether the sentiment is neutral, positive, or negative. We can include a sentence of our choice and estimate the underlying sentiment insights that are waiting to be found.

It is also referred to as opinion mining. It has various uses in business Intelligence applications including questions regarding why certain product is sold more. what does the customer think about the product etc. It has several cross domain applications in various fields including politics, law making, sociology, psychology etc.

It comes with several challenges as people express an opinion in complex ways. Lexical content can be misleading in some of the text. Intra-textual reversals, negations, topic changes are common. Rhetorical modes such as irony, implication,sarcasm etc. will make it more tedious task

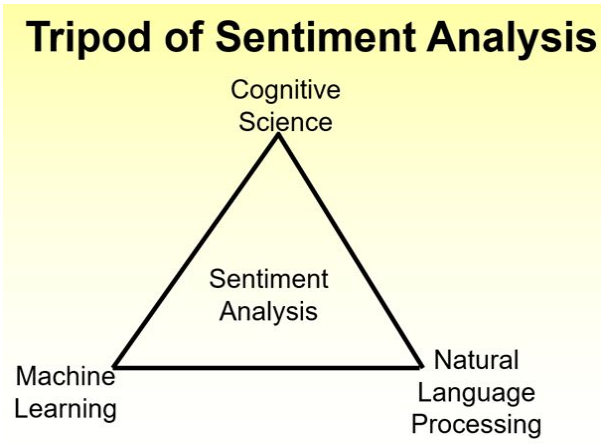

Fig 2.2 Tripod of Sentiment Analysis

Fig 2.3 [12] Various Phases in Sentiment Analysis

3. LITERATURE REVIEW:

Sentiment Analysis of Twitter data has been widely researched in the past few years. Efthymios Kouloumpis, Theresa Wilson and a few other researchers together found tri-polar classification of tweets and they concluded that Parts of speech tagging don't contribute much in this aspect. Tajinder Singh and Madhu Kumari from the National Institute of Technology, Hamirpur proposed a well-defined flow for the text preprocessing and focused on finding the impact of slang words in sentiment analysis.

Andrew L. Maas, Raymond E. Daly and other researchers from Stanford University used a mix of supervised and unsupervised techniques to learn word vectors by capturing rich sentiment content. They also introduced a large dataset of movie reviews to serve as a benchmark in this area. Quoc Le and Tomas Mikolov from Google Inc proposed Paragraph Vector, an unsupervised algorithm that learns fixed-length features from paragraphs, sentences, and documents. We have studied these research papers in-depth and our analysis is mentioned below.

3.1. RESEARCH PAPERS ANALYSIS:

3.1.1 ANALYSIS OF RESEARCH PAPER 1:
"TWITTER SENTIMENT ANALYSIS: THE GOOD THE BAD AND THE OMG!"[2]

This research paper mainly checks for the linguistic features for getting insights about the sentiments of the twitter messages. Evaluation of the use of lexical features that are already existing and also lexical features that capture the trend in microblogging language.train the model and continuous corpus in the Isieve dataset for model evaluation.

So the three Twitter message corpora that are used in the model experimentations are:

a) Hashtag data set (HASH) from Edinburgh corpus of Twitter data.
b) Emoticon data set(EMOT)
c) Manual dataset produced by ISIEVE corporation

	POSITIVE	NEGATIVE	NEUTRAL	TOTAL
HASHTAGS	**31,861 (14%)**	**64,850 (29%)**	**1,25,859 (57%)**	**2,22,570**
EMOTICONS	**2,30,811 (61%)**	**1,50,570 (39%)**	**-**	**3,81,381**
ISIEVE	**1,520 (8%)**	**200 (5%)**	**2,295 (57%)**	**4,015**

Table 3.1 [2] Data Distribution in Three Different DataSet

a. Data Preprocessing:

Main steps that are involved in Data Preprocessing are:

a) Tokenization
b) part-of-speech tagging and
c) Normalization

condenses and Emoticons (e.g.,WTH, OMG) are recognized as a component of tokenization procedure and are treated as singular tokens.

In normalization, words which are present in shortcut are replaced with actual meaning .For example a word like BRB is replaced by directly back. Intensifiers as (eg, I LOVE it!!! furthermore, repetition of character (e.g., I have a home loan!! wohooo!"), note that its nearness in tweet is also recognized. Every words should be made into lowercase, and rehashed characters are replaced with a single character. Normalization may drastically improve the performance of the system of PoS tagging which could be the final step in the pre-processing stage.

Four types of features are included to capture some microblogging language:

- **N-gram features**
  To identify set of N-grams that are useful, stop-words should be removed first.
  Then detection of negation should be attached to the word that precedes or follows the negation term. Unigrams or bigrams are found and can be identified in the training data and they will be ranked according to Chi-square measured information gain. Top 1000 n-grams are selected using bag-of-words.
- **Lexicon features**
  Three features: positive, negative and neutral are created.
- **Micro-blogging features**
  Double (binary) features which captures the availability of the positive, negative or neutral abbreviations, emoticons and intensifiers are created.
- **Part-of-speech(POS) features**
  In every tweet, the count of the parts of speech such as the verb, adverb, etc., that are present will be taken as features.

b. Results:

Adding the EMOTICONS dataset into the training brings a good improvement and in particular when all the features have been used.When we used n-gram model along with lexicon features and micro-blogging features which improved the performance of the model to a drastic change. The Part-of-speech(PoS) features mostly are not that important for the sentiment analysis in the domain of microblogging.The presence of intensifiers is the most useful.

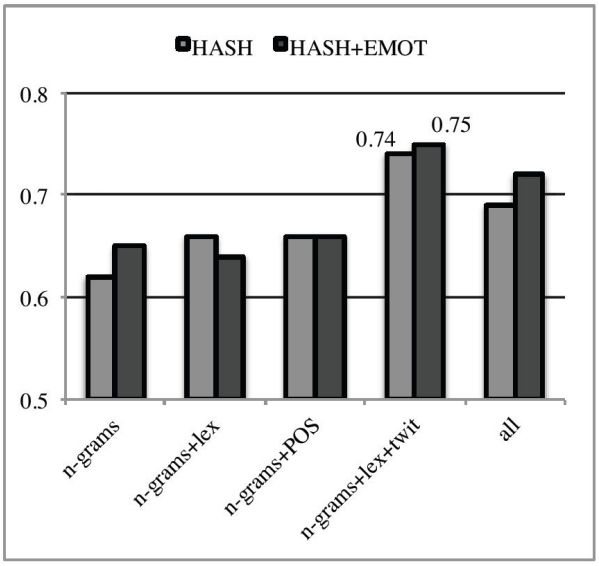

Fig 3.1 [2] Accuracy for model using HASHTAG + EMOTICONS DATA

3.1.2 ANALYSIS OF RESEARCH PAPER 2:
"ROLE OF TEXT PRE-PROCESSING IN TWITTER SENTIMENT ANALYSIS"[3]

In this research paper, we have mainly focussed on data preprocessing part because of the improper short form messages (whtsgoin, hlo etc.), slang of the tweets and short text length, it will be difficult to predict the sentiment of text. In sentiment analysis, a blend of applications are required to study the sentiment and all these require a great number of opinions from sentiment owner. A gist of the sentiments is required, to test the polarity and to remove the ambiguity from the data; single sentiment will not be sufficient to make the decision. vast majority of the social media, clients language is extremely casual. They create own format of a word in their language and spelling alternatives happen when they tell it in general speech, accentuation, incorrect spellings, slang, URLs, new words, and class explicit phrasing and shortened forms. These kinds of messages are to be corrected. Accordingly, for analysis of the content, slang words, HTML characters, emojis, accentuations, stop words, URLs, and so on should be excluded. For this process, they have defined a flow chart to process the data

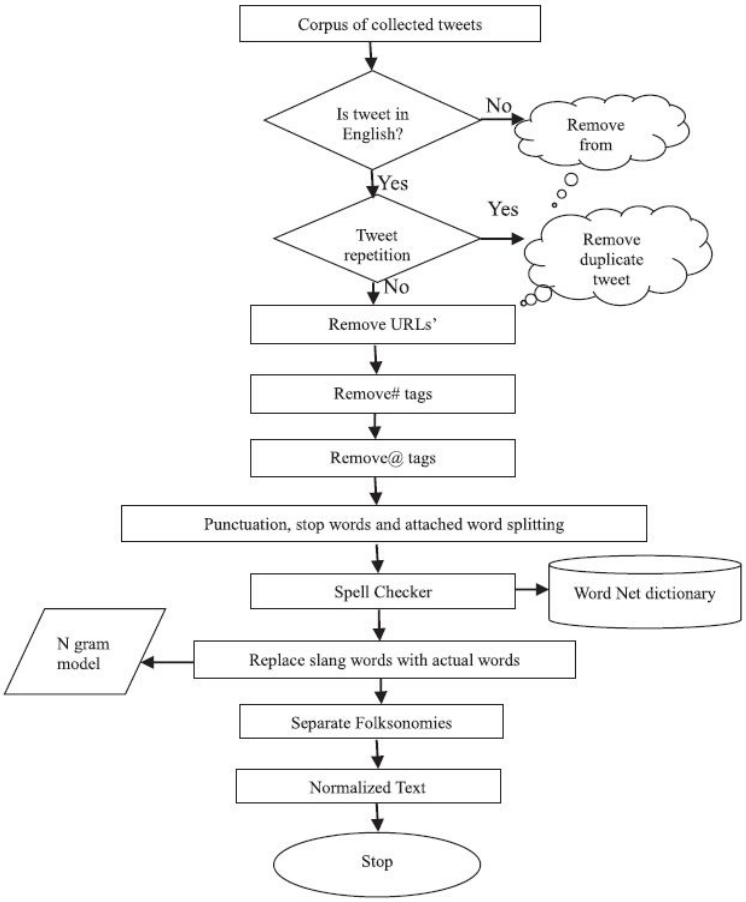

Fig 3.2 [3] Flow Chart Representing the process of cleaning Tweets

and especially for slang word they have defined a certain methodology with some presumptions. it deals with coexistence of these words with various elements at that point and chooses the centrality of slang words dependent on supposition quality along with likelihood of the co-event of restricting words slang and unidentified words. Different advances engaged with the proposed plan is given underneath.

a. Assumption:
Two unknown words can't be successive in any given tweet; the coupling of words spread maximum upto the neighbouring of two levels of two words.

b. Input:
A tweet that is having an anonymous word, Folksonomies and a slang word.

c. Output:
The significance of the words in slang :
if the slang is affirmed irrelevant, then the word is removed from the tweet corpus else: It will be replaced with a negative or positive score concerning hashtag or tweet.

d. Procedure:

- Search the binding of slang words with senses that are available in obtained twitter tweets based on the trigrams and bigram models.
- Analyzing the binding of slang words by using (CRF) for deciding on importance of the word slang. Using this CRF, (POS)Part of Speech of Ws is calculated and later, the significance of Ws is measured by the following guidelines:
    - If slang word(Ws) befalls before or after proper noun, we then consider it important.
    - If it is present along with collective nouns and will refer to proper-noun then we consider it less significant.
    - Else it is insignificant and slang word is removed totally from the tweet data.
- Once we get the significance of all the Ws, those words are substituted with positive or negative values w.r.t the concept that is available in the joining up of the set computed in the first step. The following formula will be used to calculate sentiment of Ws.

  Sentiment (Ws) = maximum | (P(Ws, Wx) × Sentiment(Wx)) |

- t(xi) refers to sentiment of tweet defined by
  t(Ws) = t-old ± Sentiment − Sentiment(Ws)
- t-old was the early sentiment value that is holding Ws in the tweet

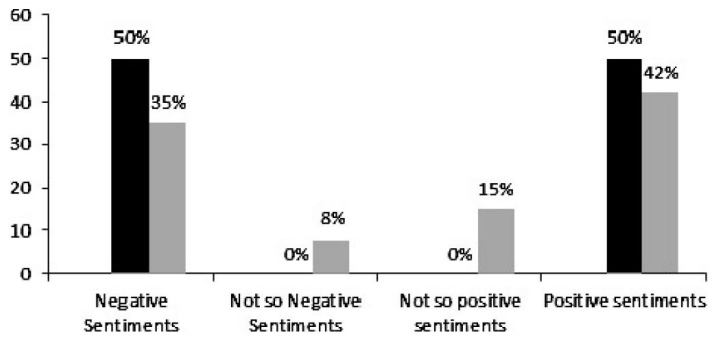

Fig 3.3 [3] Distribution of Sentiment over Dataset before and after the Preprocessing

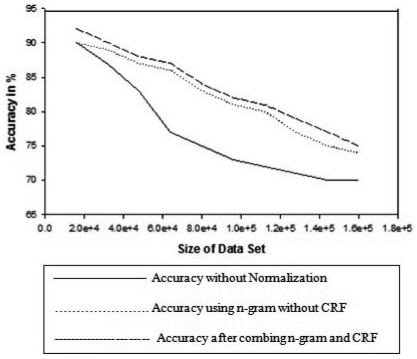

Fig 3.4 [3] Accuracy under Different Variants of Preprocessing

4. METHODOLOGY:

4.1. MACHINE LEARNING:

ML is the logical investigation of statistical models and calculations that a computer framework uses to perform explicit tasks without utilizing any express guidelines or implicit instructions, depending on examples and deduction inferences.

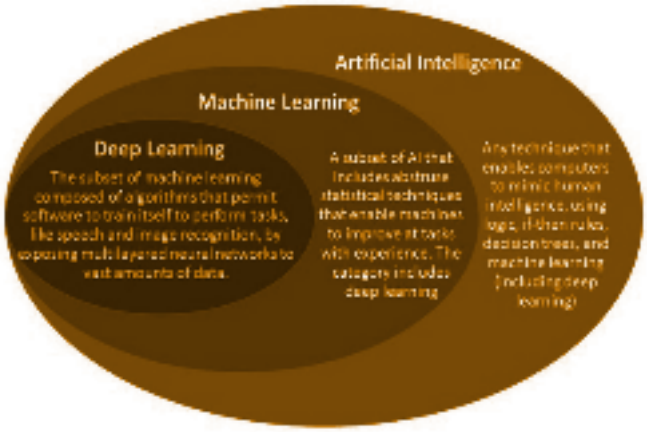

Fig 4.1 [10] Subset Notation of Domain in AI

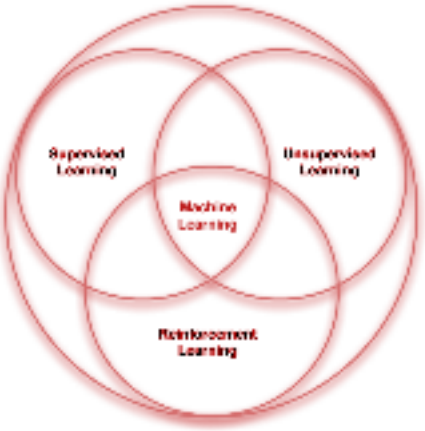

Fig 4.2 [10] Classification of M.L

Machine learning is viewed as a subset of Artificial Intelligence. ML algorithms build a mathematical representation model based on the sample dataset , known as "Training Data", to make predictions or decisions without explicit programming. ML algorithms are widely used in a variety of applications, such as email spam filtering ,search engines, malware detection, fraud detection, recommendations, computer vision, and many more.

Machine learning allows the analysis of huge data. While it usually delivers more specific and faster results to identify dangerous risks and lucrative opportunities, it might likewise need extra time and assets to train appropriately. Fusing ML with Artificial Intelligence and cognitive sciences can improve it furthermore effective to process gargantuan information.

ML algorithms are generally classified as **Unsupervised Learning(USL), Supervised Learning(SL)** and **Reinforced Learning(RL)**[10] which further have specific applications in different domains.

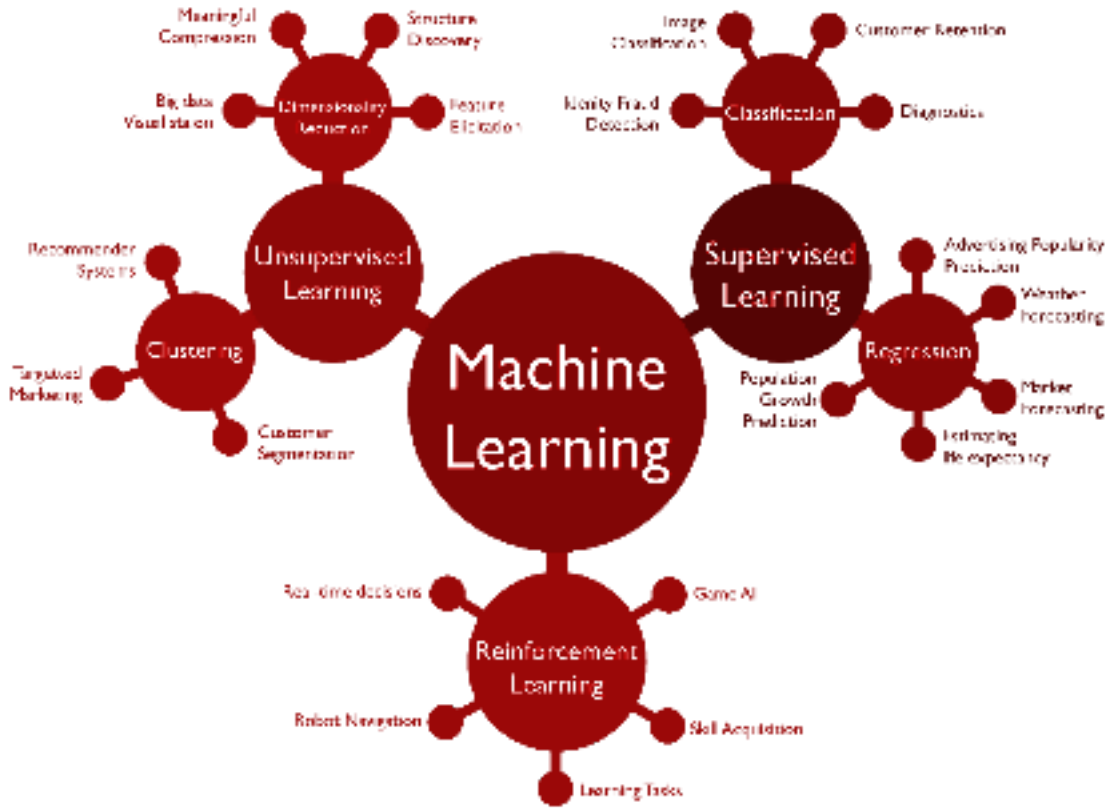

Fig 4.3 [10] Sub Branches in ML

4.1.1 <u>SUPERVISED LEARNING(SL)</u>: SL algorithms build a model on the data which contains inputs as well as outputs. It creates a function that will be used to get output if we give new inputs. In supervised learning algorithm the word supervised itself says that there is some presence of a supervisor on the learning of the algorithm which is taught by the data fed to it. The data fed in supervised learning algorithm is of labelled type which means that the data has labels which makes it fall in either of any class. The data used for learning is segregated using classes which acts as a supervisor for learning.

4.1.2 <u>UNSUPERVISED LEARNING(USL)</u>: This algorithm takes data that contains inputs only. It finds a proper trend in the data by performing techniques such as clustering, Anomaly detection etc., As the name suggests that there is no supervisor who is supervising the learning of the algorithm. The data sent to learn from in the unsupervised learning algorithm is not labelled which means the data does not have any labels or classes which tells the algorithm that the complete data doesn't have a class.

4.1.3 REINFORCED LEARNING(RL): This algorithm takes suitable steps to maximize the reward of taking any action in a particular situation. This algorithm is different from that of a supervised learning approach as in supervised learning the output or the result value for each input is given but in reinforced learning, the agent or the model is bound to learn from its experience as the output for any situation or case is not provided. The agent takes its decision on a particular step and based on which it is rewarded. The value of the reward is used by the agent to learn to take a decision for the next time. With time the model gets more learnings and refines its decision taking capabilities and hence become more accurate.

4.2. NATURAL LANGUAGE PROCESSING(NLP):

In fields of Artificial Intelligence and Computer Science, and we have a term called Natural Language Processing , that is bothered with interaction between computers and the human understandable language which we use in our day to day life, specifically which focuses on how to program our computers to process and analyze that much amount and type of human speakable and writable language data. In this domain the main challenges involve the main understanding of the natural language, natural language generation, speech recognition and sentiment analysis. NLP is the motivation for the most common applications like

- Google Translate extensively uses this concept to create some amazing language translations from one of the human spoken languages to other parallely maintaining its grammar to be completely accurate.
- checking grammatical accuracy in some amazing tools used nowadays like Grammarly and Microsoft Word.
- These days call centers uses Interactive Voice Response (IVR) which is a major application of NLP
- Voice Assistants like Siri, Alexa and Cortana.

NLP is viewed as one of the most difficult problems in computer science field. It is nature the language which makes natural language processing troublesome. The conditions which direct the flow of information using natural languages are not simple for computers to comprehend. Few of these rules could be abstract and high-leveled ; for instance, using sarcastic remark to convey their information. On the other hand, some of these dictums can be low-levelled; for instance, using “s” for plurality in items. Comprehending human language requires understanding both vocabulary and how rules are structured to deliver the intended message.

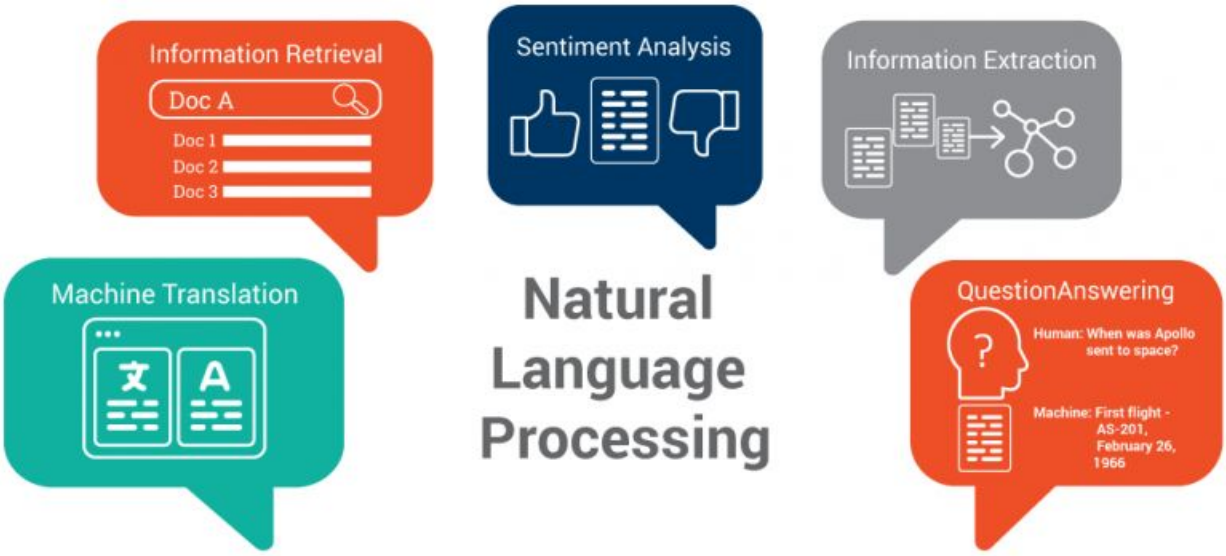

Fig 4.4 [11] Natural Language Processing and its modules

4.3. BAG OF WORDS:

This model is a method for speaking to content information when displaying content with AI calculations. It is easy to comprehend and execute and has seen incredible achievement in issues, for example, language displaying and archive characterization and classifications.

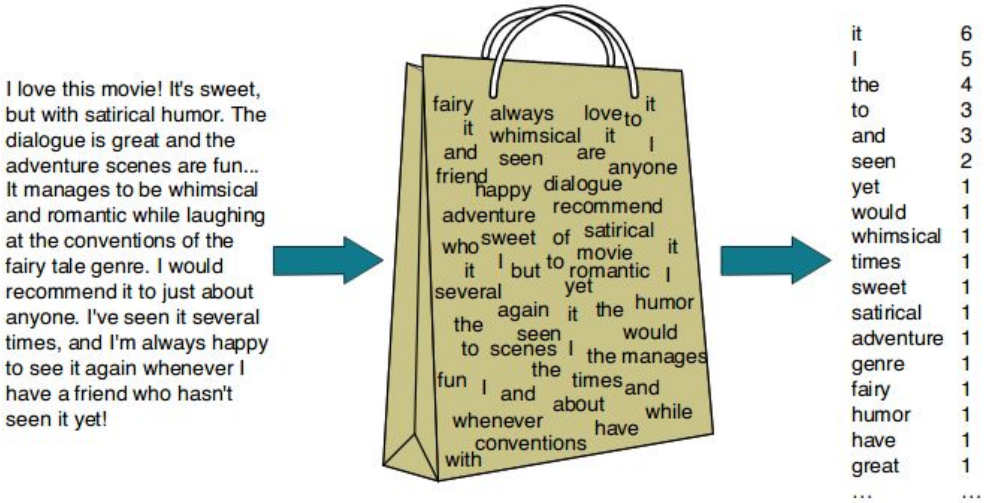

Fig 4.5 [13] Bag of Words

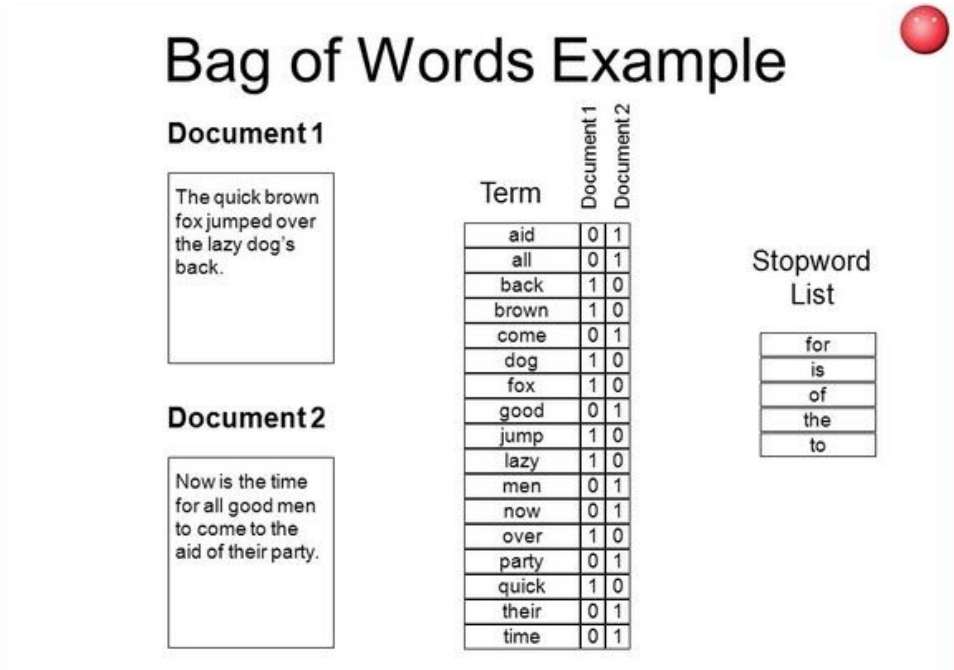

Fig 4.6 [14] Example BoW

4.4. <u>WORD2VEC</u>:

word2vec is a bi layered artificial neural network which processes the data and gives the feature vector for all the words in the given corpus. even though word2vec is not a DNN it converts the data into numerical form which the DNN's can know that. The usefulness with word2vec is it group all the most similar words into one vector space. It detects the similarities in the words by using cosine similarities. It is similar to auto-encoder, which encodes every word in the given vector. It trains words against the neighbours present in corpus . word2vec does this in about two ways that is by utilizing context to predict the word(CBOW) or using words to predict context (Skip-gram).

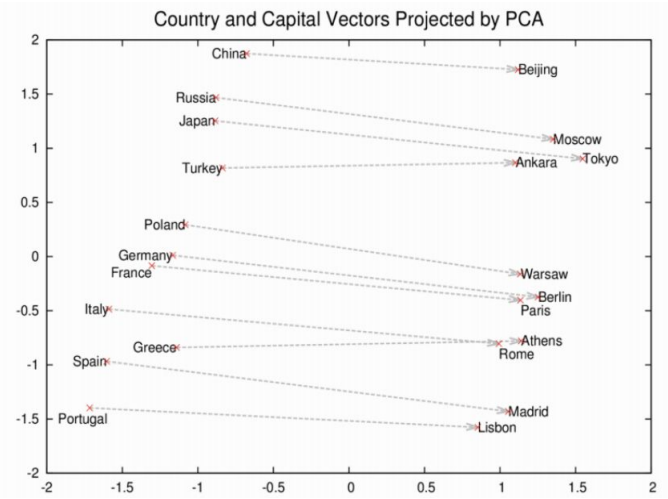

Fig 4.7 [15] Wor2Vec Distance model

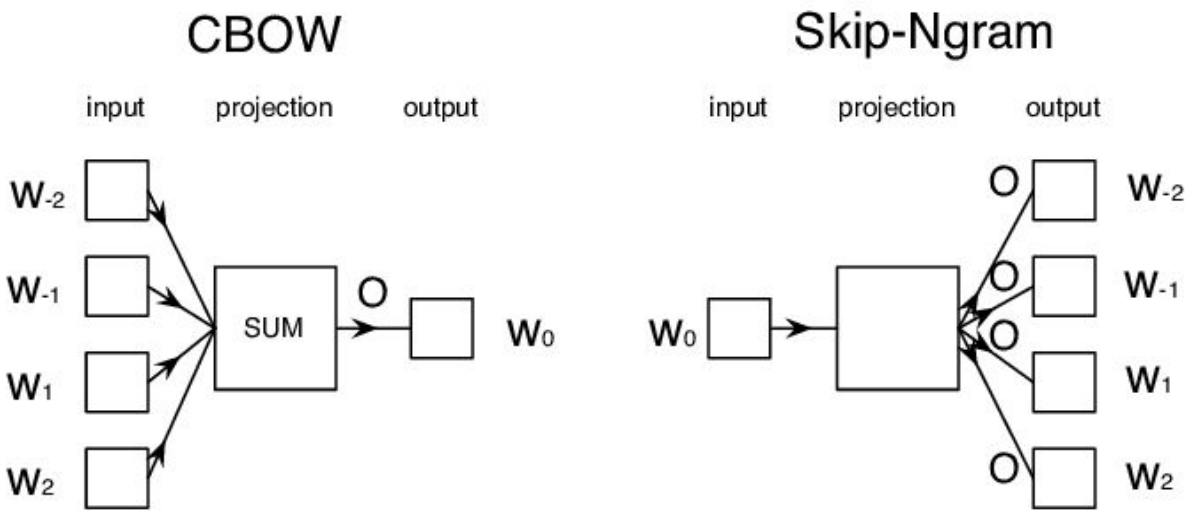

Fig 4.8 [15] CBOW and Skip-Gram models in WORD2VEC

4.5. <u>DOC2VEC</u>:

doc2vec is a neural net which calculates the feature vectors for all the documents in a given corpus. Such representation of documents can be used for various purposes including web search, document retrieval , topic modelling etc.. There are mainly two models in it PV-DM and PV-DBOW.

4.5.1 PV-DM(PARAGRAPH VECTORS DISTRIBUTED MEMORY):

it is a small extension of cbow model where we use additional document unique feature vectors along with words.so when training the word vector the document vector will be trained as well and it hold the numerical representations of documents. it acts as a memory to remember which word is actually missing from the context

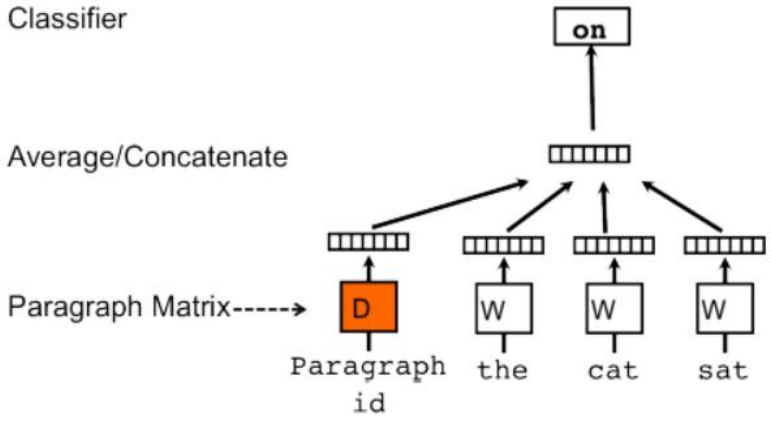

Fig 4.9 [5] PV-DM in DOC2VEC

4.5.2 PV-DBOW(PARAGRAPH VECTORS DISTRIBUTED BAG OF WORDS):

This algorithm is similar to skip-grams and it can be though as a distributed version of bag of words. this algorithm is faster and requires less memory than word2vec equivalent and there is no need to save the word vectors.

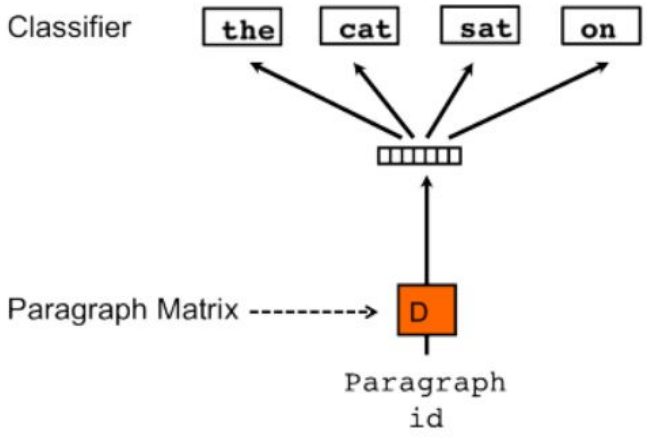

Fig 4.10 [5] PV-DBOW in DOC2VEC

5. MODEL IMPLEMENTATION:

5.1. DATASET:

In this project we extracted data from various sources and collected it, tested it and performed analytics on that. The main source of data which we used was from Andrew Maas, from Stanford University. The dataset consist of binary data which is comprises of 25,000 data of movie reviews which is for training and another 25,000 of the data for testing out machine learning model. The complete data is labelled data and is used by us for supervised learning and for training and testing our algorithms. Andrew also provided some unlabelled data for other testing purposes as well. The dataset used by us is of high quality and provides a very good result when trained and tested which is mentioned in other sections.

5.2 BAG OF WORDS IMPLEMENTATION:

PreProcessing data for all of the training data

In [495]:
```
training_data_size = train_data["review"].size
testing_data_size = test_data["review"].size
main_testing_data_size = main_test_data["review"].size

print(training_data_size)
print(testing_data_size)
print(main_testing_data_size)
```

```
17500
7500
25000
```

In [496]:
```
def clean_text_data(data_point, data_size):
    review_soup = BeautifulSoup(data_point)
    review_text = review_soup.get_text()
    review_letters_only = re.sub("[^a-zA-Z]", " ", review_text)
    review_lower_case = review_letters_only.lower()
    review_words = review_lower_case.split()
    stop_words = stopwords.words("english")
    meaningful_words = [x for x in review_words if x not in stop_words]

    if( (i)%2000 == 0 ):
        print("Cleaned %d of %d data (%d %%)." % ( i, data_size, ((i)/data_size)*100))

    return( " ".join( meaningful_words ))
```

In [497]:
```
clean_train_data_list = []
clean_test_data_list = []
clean_main_test_data_list = []
```

Fig 5.1 BOW model :Preprocessing the data

1. The above image shows the preprocessing of the data extracted from the dataset we collected above. The training and testing set keeps the feature of "**review**" which has the text data of the user's tweet reviews about a movie. Then we cleaned the data by creating an iterative loop of calling the function **clean_text_data** which do the following things:
    - Called the BeautifulSoup library which removes all the HTML or XML tags from the data and makes it a non-coded text data.
    - Then using a regular expression we removed the unwanted special characters from the text which are not useful for this analysis.
    - After that, we made the text to be all lowercase to avoid any kind of case sensitive conflicts.

- Then we imported the stop words which do not give any sentiment in the corpus and are of no use and should be removed.
- Then over an iterative loop we removed the occurrences of stop words from the main corpus and the remaining words were only the meaningful words.
- The above all steps are performed for each and every review corpus in the training dataset.
- The final outcome of this process was clean and ready to be trained reviews of the users.

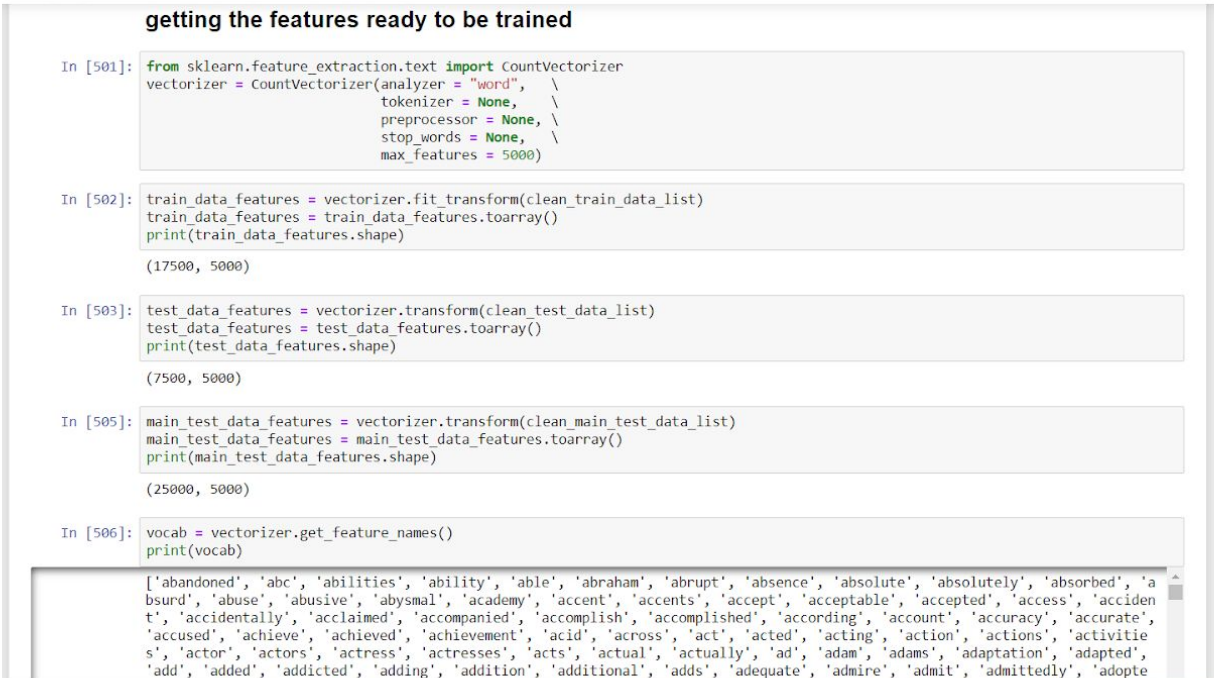

Fig 5.2 BOW model: Getting the features ready to train the model

2. This Images shows the process of making the already cleaned data ready to be trainable data as the text can not be directly fed to the training model.
    - Firstly to make the text into a numerical data, CountVectorizer is imported from the scikit learn library then vectorizer is initialized for a limit of 5000 maximum features.
    - Vectorizer can also be used for preprocessing but we already did that before so we only proceed with the feature extraction.
    - We passed the training and testing data through the vectorizer and then got the features extracted from them.
    - To train the model we use **fit_transform** on training set and then **transform** on the validation and testing set.
    - Then the data is converted to the array to make it a trainable numerical data fit for training the set.

- Then the bag of words was used to create the frequency of each word in the corpus and which was the data ready for training which was sent to the Random Forest Classifier in the next stage.

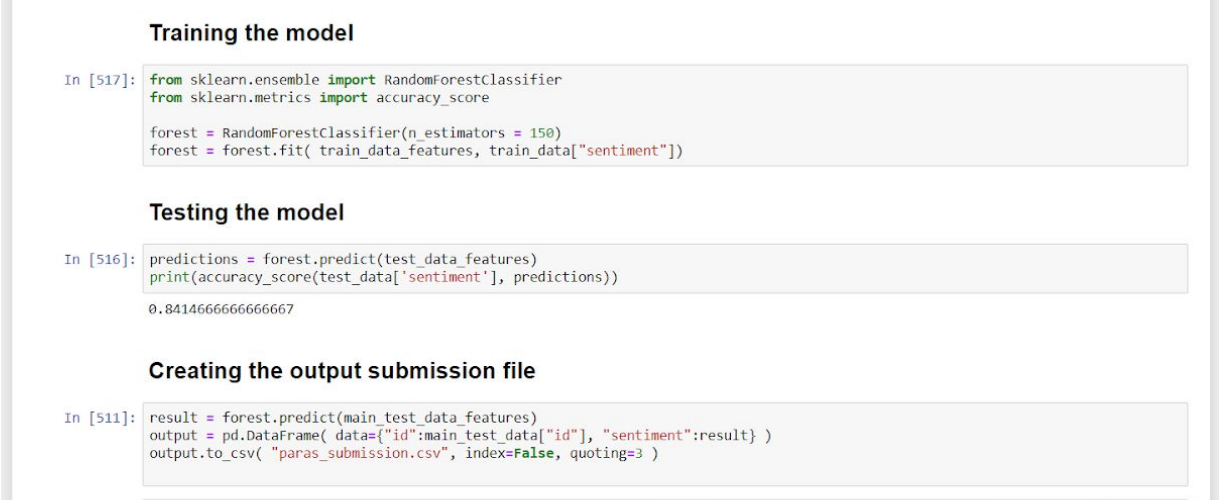

Fig 5.3 Bag of words model: Model training and testing

3. The above image shows the training and the testing of the data on the main model which was a Random Forest Classifier.
    - Random Forest is an ensemble learning technique in which data is divided into multiple decision trees and passed to get divided into each node pass. The data then falls in one of the leaf nodes which defines its class. Random forest is used for both regression and classification type of learning algorithms.
    - The Random forest algorithmic model for training the data extracted was imported from the scikit-learn library which has a collection of enormous number of machine learning algorithms.
    - The random forest was set to run on 100 decision trees and the result to be aggregated.
    - Training data is fed in the classifier and the model is trained over that data.
    - Then accuracy score was imported to get the accuracy value for the prediction created.
    - Prediction is calculated for the testing data input and then tested for the accuracy score with comparing with the already present output of the testing set.
    - The accuracy comes out to be **84.14%.**
    - Secondly, we implemented Support Vector Machine Classifier as a training model for the same dataset. For the SVC model we got an accuracy **83.40%**
    - Finally, we have used ANN model with six layers as a classifier and we have obtained a remarkable **91.75%** accuracy in training the model but the performance went down to **49.90%** in the testing set

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 512)	35497472
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 512)	262656
dropout_2 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 256)	131328
dropout_3 (Dropout)	(None, 256)	0
dense_4 (Dense)	(None, 10)	2570
dropout_4 (Dropout)	(None, 10)	0
dense_5 (Dense)	(None, 12)	132

Total params: 35,894,158
Trainable params: 35,894,158
Non-trainable params: 0

Fig 5.4 ANN Model Architecture

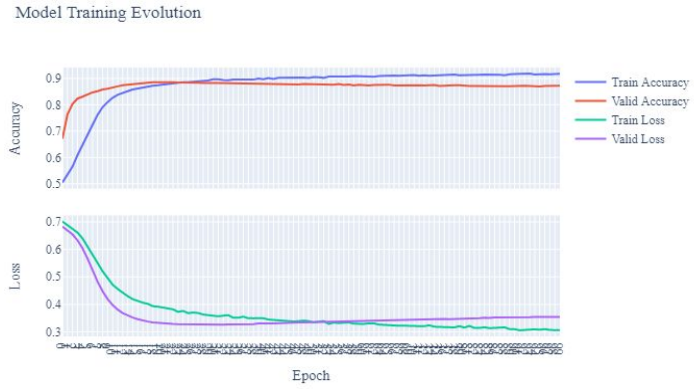

Fig 5.5 Training and Validation Accuracy and Error

5.3 WORD2VEC MODEL IMPLEMENTATION:

Data Collection:

```
import pandas as pd

DIR='/kaggle/input/word2vec-nlp-tutorial/'
# Read data from files
train = pd.read_csv( DIR+"labeledTrainData.tsv", header=0,
 delimiter="\t", quoting=3 )

test = pd.read_csv( DIR+"testData.tsv", header=0, delimiter="\t", quoting=3 )
unlabeled_train = pd.read_csv( DIR+"unlabeledTrainData.tsv", header=0,
 delimiter="\t", quoting=3 )

# Verify the number of reviews that were read (100,000 in total)
print( "Read %d labeled train reviews, %d labeled test reviews, " \
"and %d unlabeled reviews\n" % (train["review"].size,
 test["review"].size, unlabeled_train["review"].size ))
```

```
Read 25000 labeled train reviews, 25000 labeled test reviews, and 50000 unlabeled reviews
```

Fig 5.6 Word2vec : Data collection

1. At this stage we gathered the data which is available in a ".csv" file. we have gathered the training and testing data into a data frame and printed the size of each data frame

Data Preprocessing

```
from bs4 import BeautifulSoup
import re
from nltk.corpus import stopwords

def review_to_wordlist( review, remove_stopwords=False ):
    review_text = BeautifulSoup(review).get_text()
    review_text = re.sub("[^a-zA-Z]"," ", review_text)
    words = review_text.lower().split()
    if remove_stopwords:
        stops = set(stopwords.words("english"))
        words = [w for w in words if not w in stops]
    return(words)

tokenizer = nltk.data.load('tokenizers/punkt/english.pickle')

def review_to_sentences( review, tokenizer, remove_stopwords=False ):
    raw_sentences = tokenizer.tokenize(review.strip())
    sentences = []
    for raw_sentence in raw_sentences:
        if len(raw_sentence) > 0:
            sentences.append( review_to_wordlist( raw_sentence,remove_stopwords ))
    return sentences

sentences = []  # Initialize an empty list of sentences

print ("Parsing sentences from training set")
for review in train["review"]:
    sentences += review_to_sentences(review, tokenizer)

print ("Parsing sentences from unlabeled set")
for review in unlabeled_train["review"]:
    sentences += review_to_sentences(review, tokenizer)
```

Fig 5.7 Word2vec : Data preparation

2. At this step we are processing the gathered data into required format i.e we are converting the corpus into tokens by tokenization and removed stop words from them, used lemmatization and converted everything into lower case.

Model Building

```
# Import the built-in logging module and configure it so that Word2Vec
# creates nice output messages
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s',\
    level=logging.INFO)

# Set values for various parameters
num_features = 300    # Word vector dimensionality
min_word_count = 40   # Minimum word count
num_workers = 4       # Number of threads to run in parallel
context = 10          # Context window size
downsampling = 1e-3   # Downsample setting for frequent words

# Initialize and train the model (this will take some time)
from gensim.models import word2vec
print ("Training model...")
model = word2vec.Word2Vec(sentences, workers=num_workers, \
            size=num_features, min_count = min_word_count, \
            window = context, sample = downsampling)

# If you don't plan to train the model any further, calling
# init_sims will make the model much more memory-efficient.
model.init_sims(replace=True)

# It can be helpful to create a meaningful model name and
# save the model for later use. You can load it later using Word2Vec.load()
model_name = "300features_40minwords_10context"
model.save(model_name)
```

Fig 5.8 Word2vec: Model Building

3. At this step we are creating a word2vec model where we have defined various model parameters including number of workers, number of features, context, etc.., and using all these parameters we are building a word2vec model to convert our corpus

5.4 DOC2VEC MODEL (DISTRIBUTED REPRESENTATION OF SENTENCES AND DOCUMENTS) IMPLEMENTATION:

```
import io
import re
import tarfile
import os.path

import smart_open
import gensim.utils

def download_dataset(url='http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz'):
    fname = url.split('/')[-1]

    if os.path.isfile(fname):
       return fname

    # Download the file to local storage first.
    # We can't read it on the fly because of
    # https://github.com/RaRe-Technologies/smart_open/issues/331
    with smart_open.open(url, "rb", ignore_ext=True) as fin:
        with smart_open.open(fname, 'wb', ignore_ext=True) as fout:
            while True:
                buf = fin.read(io.DEFAULT_BUFFER_SIZE)
                if not buf:
                    break
                fout.write(buf)

    return fname

def create_sentiment_document(name, text, index):
    _, split, sentiment_str, _ = name.split('/')
    sentiment = {'pos': 1.0, 'neg': 0.0, 'unsup': None}[sentiment_str]

    if sentiment is None:
        split = 'extra'

    tokens = gensim.utils.to_unicode(text).split()
    return SentimentDocument(tokens, [index], split, sentiment)

def extract_documents():
    fname = download_dataset()

    index = 0

    with tarfile.open(fname, mode='r:gz') as tar:
        for member in tar.getmembers():
            if re.match(r'aclImdb/(train|test)/(pos|neg|unsup)/\d+_\d+.txt$', member.name):
                member_bytes = tar.extractfile(member).read()
                member_text = member_bytes.decode('utf-8', errors='replace')
                assert member_text.count('\n') == 0
                yield create_sentiment_document(member.name, member_text, index)
                index += 1

alldocs = list(extract_documents())
```

Fig 5.9 Data Collection for DOC2VEC model

Step1: In the above code snippet we have taken the URL where the dataset is present and we have defined functions to extract the data from the given URL using smart open and tarFile packages and we have saved the list of all documents inside the variable alldocs.

```
print(alldocs[27])
```

```
SentimentDocument(words=['I', 'was', 'looking', 'forward', 'to', 'this', 'movie.', 'Trustworthy', 'actors,', 'interesting', 'p
lot.', 'Great', 'atmosphere', 'then', '?????', 'IF', 'you', 'are', 'going', 'to', 'attempt', 'something', 'that', 'is', 'mean
t', 'to', 'encapsulate', 'the', 'meaning', 'of', 'life.', 'First.', 'Know', 'it.', 'OK', 'I', 'did', 'not', 'expect', 'the',
'directors', 'or', 'writers', 'to', 'actually', 'know', 'the', 'meaning', 'but', 'I', 'thought', 'they', 'may', 'have', 'offer
ed', 'crumbs', 'to', 'peck', 'at', 'and', 'treats', 'to', 'add', 'fuel', 'to', 'the', 'fire-Which!', 'they', 'almost', 'did.',
'Things', 'I', "didn't", 'get.', 'A', 'woman', 'wandering', 'around', 'in', 'dark', 'places', 'and', 'lonely', 'car', 'parks',
'alone-oblivious', 'to', 'the', 'consequences.', 'Great', 'riddles', 'that', 'fell', 'by', 'the', 'wayside.', 'The', 'promis
e', 'of', 'the', 'knowledge', 'therein', 'contained', 'by', 'the', 'original', 'so-called', 'criminal.', 'I', 'had', 'no', 'pr
oblem', 'with', 'the', 'budget', 'and', 'enjoyed', 'the', 'suspense.', 'I', 'understood', 'and', 'can', 'wax', 'lyrical', 'abo
```

Fig 5.10 Print Example document

```
train_docs = [doc for doc in alldocs if doc.split == 'train']
test_docs = [doc for doc in alldocs if doc.split == 'test']
print('%d docs: %d train-sentiment, %d test-sentiment' % (len(alldocs), len(train_docs), len(test_docs)))
```

```
100000 docs: 25000 train-sentiment, 25000 test-sentiment
```

Fig 5.11 Splitting the data

Step2: split total data into two parts training and testing set containing 25k data points in training and 25k in testing dataset

```
import multiprocessing
from collections import OrderedDict

import gensim.models.doc2vec
assert gensim.models.doc2vec.FAST_VERSION > -1, "This will be painfully slow otherwise"

from gensim.models.doc2vec import Doc2Vec

common_kwargs = dict(
    vector_size=100, epochs=20, min_count=2,
    sample=0, workers=multiprocessing.cpu_count(), negative=5, hs=0,
)

simple_models = [
    # PV-DBOW plain
    Doc2Vec(dm=0, **common_kwargs),
    # PV-DM w/ default averaging; a higher starting alpha may improve CBOW/PV-DM modes
    Doc2Vec(dm=1, window=10, alpha=0.05, comment='alpha=0.05', **common_kwargs),
    # PV-DM w/ concatenation - big, slow, experimental mode
    # window=5 (both sides) approximates paper's apparent 10-word total window size
    Doc2Vec(dm=1, dm_concat=1, window=5, **common_kwargs),
]

for model in simple_models:
    model.build_vocab(alldocs)
    print("%s vocabulary scanned & state initialized" % model)

models_by_name = OrderedDict((str(model), model) for model in simple_models)
```

Fig 5.12 Defining DOC2VEC models

Step3: define the doc2vec model to train the documents and to use it as features/vocabulary along with a machine learning model. In the model, we have defined three models with doc2vec one with dm=0 which means that this model is trained with a distributed bag word(PV-DBOW) . Model2 is defined with dm=1 which implements distributed memory (PV-DM) with window =10 (which represents the max distance in between present and predicted word within a given sentence) and learning rate(l.r) =0.05. Model3 is defined with lower window size(window =5) and dm_concat=1(which allows the model to use concat the context vectors rather than

averaging/ summation). Distributed memory preserves the word order in the document whereas Distributed bag of words will not preserve but serve as a bag of words.

```
from gensim.test.test_doc2vec import ConcatenatedDoc2Vec
models_by_name['dbow+dmm'] = ConcatenatedDoc2Vec([simple_models[0], simple_models[1]])
models_by_name['dbow+dmc'] = ConcatenatedDoc2Vec([simple_models[0], simple_models[2]])
```

Fig 5.13 Mikolov and Quoc le Model

Step4: Mikolov and Quoc le figured out that combining both the models PV-DM and PV-DBOW has actually increased the performance. Here we are concatenating the paragraph vectors obtained using a wrapper class

```
import numpy as np
import statsmodels.api as sm
from random import sample

def logistic_predictor_from_data(train_targets, train_regressors):
    """Fit a statsmodel logistic predictor on supplied data"""
    logit = sm.Logit(train_targets, train_regressors)
    predictor = logit.fit(disp=0)
    # print(predictor.summary())
    return predictor

def error_rate_for_model(test_model, train_set, test_set):
    """Report error rate on test_doc sentiments, using supplied model and train_docs"""

    train_targets = [doc.sentiment for doc in train_set]
    train_regressors = [test_model.docvecs[doc.tags[0]] for doc in train_set]
    train_regressors = sm.add_constant(train_regressors)
    predictor = logistic_predictor_from_data(train_targets, train_regressors)

    test_regressors = [test_model.docvecs[doc.tags[0]] for doc in test_set]
    test_regressors = sm.add_constant(test_regressors)

    # Predict & evaluate
    test_predictions = predictor.predict(test_regressors)
    corrects = sum(np.rint(test_predictions) == [doc.sentiment for doc in test_set])
    errors = len(test_predictions) - corrects
    error_rate = float(errors) / len(test_predictions)
    return (error_rate, errors, len(test_predictions), predictor)
```

Fig 5.14 Defining Classifier and Error Function.

Step5:
Given the document, Doc2Vec models output vector representation of document. In the case of sentiment analysis, we want output vector to convey the sentiment of input document. So, In vector-space , positive document should be far distant from negative document.
We will train logistic regression:

- regressors /(inputs): document vector from Doc2Vec model
- target /(outpus): labeled sentiment

So, this logistic regression would be able to predict the sentiment of given document vector.Next, we test the logistic regression model on the test set, and measure the error rate. If document vectors from Doc2Vec reflect actual sentiment precisely ,then error rate will be low.

Therefore, the error rate is an indication of *how good* Doc2Vec model represents the documents as vector notation . We can compare various Doc2Vec models by referring to their error rates.

```
from collections import defaultdict
error_rates = defaultdict(lambda: 1.0)  # To selectively print only best errors achieved
```

```
from random import shuffle
shuffled_alldocs = alldocs[:]
shuffle(shuffled_alldocs)

for model in simple_models:
    print("Training %s" % model)
    model.train(shuffled_alldocs, total_examples=len(shuffled_alldocs), epochs=model.epochs)

    print("\nEvaluating %s" % model)
    err_rate, err_count, test_count, predictor = error_rate_for_model(model, train_docs, test_docs)
    error_rates[str(model)] = err_rate
    print("\n%f %s\n" % (err_rate, model))

for model in [models_by_name['dbow+dmm'], models_by_name['dbow+dmc']]:
    print("\nEvaluating %s" % model)
    err_rate, err_count, test_count, predictor = error_rate_for_model(model, train_docs, test_docs)
    error_rates[str(model)] = err_rate
    print("\n%f %s\n" % (err_rate, model))
```

Fig 5.15 Training the models

Step6: train the model with specified feature vectors and logistic regression along with specified parameters which includes epoch size and doc2vec models

```
print("Err_rate Model")
for rate, name in sorted((rate, name) for name, rate in error_rates.items()):
    print("%f %s" % (rate, name))
```

```
Err_rate Model
0.104360 Doc2Vec(dbow,d100,n5,mc2,t4)+Doc2Vec("alpha=0.05",dm/m,d100,n5,w10,mc2,t4)
0.104840 Doc2Vec(dbow,d100,n5,mc2,t4)+Doc2Vec(dm/c,d100,n5,w5,mc2,t4)
0.105920 Doc2Vec(dbow,d100,n5,mc2,t4)
0.174240 Doc2Vec("alpha=0.05",dm/m,d100,n5,w10,mc2,t4)
0.306080 Doc2Vec(dm/c,d100,n5,w5,mc2,t4)
```

```
print("Accuracy")

for acc,name in sorted((acc,name) for name, acc in error)
```

Fig 5.16 Evaluating the error rate for all the different models

Step7: calculating the error rate using the function which we have defined in the figure and we have obtained minimum error_rate of 10.44% for model1 and the maximum error rate of 30.61% for last model(only PV-DM- concatenation)

6. CONCLUSION & FUTURE SCOPE:

6.1 CONCLUSION:

S. NO.	MODEL	ACCURACY
1.	BOW + Linear SVM	83.40
2.	BOW + Random Forest	84.14
3.	BOW + ANN	49.90
4.	Random Forest + Word2vec + BOW	83.37
5.	logit+doc2vec(PV-DBOW+PV-DM)	**89.56(expected- 92.6)**

Table 6.1 Results

6.2. FUTURE SCOPE:

Till date the project has been carried out according to research what has been done so far. we have applied various techniques in feature selection were applied to extract features out of document and we worked on multiple algorithms for classification including Random Forest, Support Vector Machines,logistic Regression. We would like to redefine the model parameters in it and perform hyper parameter tuning and Apart from these much work can be carried on applying Multinomial Naive bayes along with unigrams and bigrams. and as we have only applied the svm classifier in case of bag of words we would like to extend this application by including N-gram vectorization and use word2vec and doc2vec embedding along with all the above mentioned algorithms. Concerning about classifier, we would like to try a GAN based transfer learning approach for it and look for the performance efficiency as it was claimed that GAN increase the performance of Sentiment analysis by Qian tao and Yuchen Zhou[9].

REFERENCES

1. Maas, A L., Daly, R E., Pham, Peter T., Huang.D, Ng, Andrew Y., and Potts, Christopher. "Learning word vectors for sentiment analysis" In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, Vol 1, Pg142-150, 2011.
2. Kouloumpis, Efthymios, Wilson, Theresa & Moore, Johanna."Twitter Sentiment Analysis: The Good the Bad and the OMG!."International Conference on Weblogs and Social Media (2011).
3. T Singh, M Kumari, "Role of text pre-processing in twitter sentiment analysis", Procedia Computer Science Vol 89, Pg:549-554, 2016.
4. Pang B and Lillian L. "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales". In Proceedings of the Association for Computational Linguistics, pg: 115–124, 2005.
5. Quoc Le and Mikolov Tomas. "Distributed representations of sentences and documents".International Conference on International Conference on Machine Learning, Vol 32, Pg: II-1188-II-1196, (2014).
6. Socher, Richard, Pennington, Jeffrey, Huang, Eric H, Ng, Andrew Y, and Manning, Christopher D. "Semi supervised recursive autoencoders for predicting sentiment distributions". In Proceedings of the Conference on Empirical Methods in Natural Language Processing,Pg:151-161, (2011c).
7. Rafael Macheal Karampatsis, John Pavlopoulos and Prodromos Malakasiotis, "Two Stage Sentiment Analysis of Social Network Messages", SemEval,Pg:114–118, (2014)
8. Wang, Sida and Manning, Chris D. "Baselines and bigrams: Simple, good sentiment and text classification". In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Vol:2, Pg:90-94, 2012.
9. Tao, Qian & Zhou, Yuchen & Huang, Jie & Li, Jiaying & Ma, Senyu. "A GAN-based Transfer Learning Approach for Sentiment Analysis". Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science, Pg: 364-368, (2019).
10. https://en.wikipedia.org/wiki/Machine_learning
11. https://medium.com/greyatom/introduction-to-natural-language-processing-78baac3c602b
12. https://medium.com/@tomyuz/a-sentiment-analysis-approach-to-predicting-stock-returns-d5ca8b75a42
13. http://www.programmersought.com/article/4304366575/;jsessionid=0187F8E68A22612555B437068028C012
14. https://www.quora.com/What-is-the-bag-of-words-algorithm
15. https://skymind.ai/wiki/word2vec