

Analyzing Online Articles Using Data Mining

A Major Project Report

Submitted by

Satyaveer Nayak (B15CS032)
&
Shyam Sundar Kumawat (B15CS038)

Under the supervision of

Dr. Yogita

Assistant Professor, Department of Computer Science and Engineering

In fulfilment for the award of the degree

of

Bachelor of Technology

in

Computer Science and Engineering



Department of Computer Science and Engineering

National Institute of Technology, Meghalaya

Bijni Complex, Laitumkhrah

Shillong-793003, Meghalaya, India

May, 2019



राष्ट्रीय प्रौद्योगिकी संस्थान मेघालय
NATIONAL INSTITUTE OF TECHNOLOGY MEGHALAYA
Bijni Complex , Laitumukrah, Shillong-793003

Dr. Yogita

Assistant Professor

CSE Dept, NIT Meghalaya

May 10, 2019

Supervisor's Certificate

This is to certify that the work presented in the dissertation entitled **Analyzing Online Articles Using Data Mining**, by Satyaveer Nayak (Roll Number B15CS032) and Shyam Sundar Kumawat (Roll Number B15CS038), is a record of project work carried out by them under my supervision and guidance in fulfilment of the requirements of the degree of Bachelors of Technology in Computer Science and Engineering. Neither this report nor any part of it has been submitted for any degree or IPR in India or abroad.

(Dr. Yogita)



राष्ट्रीय प्रौद्योगिकी संस्थान मेघालय
NATIONAL INSTITUTE OF TECHNOLOGY MEGHALAYA
Bijni Complex , Laitumukrah, Shillong-793003

Dr. Diptendu Sinha Roy

Associate Professor

HoD, CSE Dept, NIT Meghalaya

May 10, 2019

Head of Department's Certificate

This is to certify that the work presented in the dissertation entitled **Analyzing Online Articles Using Data Mining**, by Satyaveer Nayak (Roll Number B15CS032) and Shyam Sundar Kumawat (Roll Number B15CS038), is a record of project work carried out by them under the supervision and guidance of Dr. Yogita, Assistant Professor, Department of Computer Science and Engineering in fulfilment of the requirements of the degree of Bachelor of Technology in Department of Computer Science and Engineering. Neither this report nor any part of it has been submitted for any degree or IPR in India or abroad.

(Dr. Diptendu Sinha Roy)

Declaration of Originality

We, Satyaveer Nayak, Roll Number - B15CS032 and Shyam Sundar Kumawat, Roll Number - B15CS038, hereby declare that this report **Analyzing Online Articles Using Data Mining** represents our original work carried out as the undergraduate students of National Institute of Technology Meghalaya (NITM) and, to the best of our knowledge, it contains no material previously published or written by another person unless cited. Any contribution made to this report by others, with whom we have worked at NITM or elsewhere, is explicitly acknowledged in the plan. Works of other authors cited in this report have been duly acknowledged under the section Bibliography.

May 10, 2019

NIT Meghalaya

(Satyaveer Nayak)

(Shyam Sundar Kumawat)

Acknowledgement

We would like to express our special thanks of gratitude to our supervisor, Asst. Prof. Dr. Yogita, who gave us the opportunity to do this project and also helped us in doing a lot of research about this topic. We came to know so many new things during the work. Furthermore we would like to thank to faculties of Department of Computer Science and Engineering for introducing us to the topic as well for the support on the way. We would also like to thank our loved ones, who have supported us throughout entire process, both by keeping us harmonious and helping us putting pieces together. We will be grateful forever for your support.

Sincerely,

Satyaveer Nayak

Shyam Sundar Kumawat

ABSTRACT

The principle objective of this project is to analyze online articles using Data Mining. These articles may be blogs, news articles and etc. An article may contain lot of meaningful information which can be extracted using text data mining techniques.

Data Mining contains many techniques and algorithms which can be used to represent the text data in proper format, later on which is used to process and find the desired output from the available text data. Online news article contains lot of data which may be related to different categories like education, politics, geography, sports and etc.

In this project we are categorizing the online news articles in different categories based on similarities among them. These similarities can be measured using Cosine document similarity, Euclidean distance based similarity. Later on, these similarities are used to apply classification. An article may contain lot of useless data like stop words, punctuations, special characters and etc. So it is required to remove this kind of useless data which may affect the results. This useless data can be removed using preprocessing algorithms. Preprocessing algorithms are stop word removing, stemming, part of speech tagging. Preprocessing process required data in specific format which is called collection of tokens. So we need to store the content of article in form of tokens. Tokenization can be done by extracting the words which are splitted by white spaces from the content of article. After, preprocessing process we will get collection of tokens of the useful information. Now for similarity measurement the data should be in vector format along with the frequency matrix of the tokens. Collection of all tokens together is called bag of word. Frequency matrix is called weight matrix also which contains the word and its weight in the article from which it belongs. These weights are calculated using TF-IDF algorithm which gives less weight to the term which is occurring more and vice-versa. These TF-IDF matrix are used as features. After similarity measurement articles can be divided in different categories. The similarity may gives different results based on the how many tokens we are considering at a time. So this problem can be overlooked by using N-gram algorithm. These features often used for the training of classification model. We used different classification techniques like Naïve Bayes classifier, logistic classifier, and SVM classifier. Using these techniques articles we classified articles.

Keywords Cosine Similarity, Euclidean distance, Tokenization, Stop words, Stemming, punctuations, TF-IDF, N-gram, Naïve Bayes classifier, logistic classifier, and SVM classifier.

ACRONYMS

S.No	Notation	Full Form
1	TF	Term frequency
2	TF-IDF	Term Frequency- Inverse Document Frequency
3	POS	Part of Speech
4	NLTK	Natural language Processing Tool Kit
5	NEE	Named Entity Extraction
6	VSM	Vector Space Model
7	BOW	Bag of word
8	SVM	Support Vector Machine

LIST OF FIGURES

S.No	Figure	Page No
1	Fig 3.1: Data frame to store Labeled Articles	5
2	Fig 3.2: Labeling of Articles	6
3	Fig 3.3: Labeled Articles	7
4	Fig 3.4: Tokenization of the Sentence	8
5	Fig 3.5: Tags in a Sentence	10
6	Fig 3.6: Chunking of a sentence.	10
7	Fig 4.1: TF-IDF computation method	14
8	Fig 5.1: Cosine Similarity measurement	17
9	Fig 7.1: Naïve Bayes classification-based Results	21
10	Fig 7.2: Logistic Regression classification-based Results	22

LIST OF TABLES

S.No	Table	Page No
1	Table 3.1: set of part of speech Tags	11
2	Table 6.1: Similarity measurement Result	20
3	Table 6.2: Classification Results	21

CONTENTS

Supervisor's certificate	ii
Head of Department's Certificate	iii
Declaration of Originality	iv
Acknowledgements	v
Abstract	vi
Acronyms	vii
List of Figures.....	vii
List of Tables	vii
1 Introduction	1-2
1.1 Motivation	1
1.2 Objective	2
1.3 Main contribution.....	2
1.4 Report outline.....	2
2 Literature Survey.....	3
2.1 Literature survey on Similarity Measure	3
2.2 Literature survey on Classification Techniques	3
3 Raw Data Acquisition and Preprocessing	4-12
3.1 Raw data	4
3.2 Data labeling	4
3.3 Data Representation	7
3.4 Data Preprocessing.....	8-12
3.4.1 Tokenization.....	8
3.4.2 Stop Word Removing.....	9
3.4.3 Stemming.....	9
3.4.4 Chunking.....	9

3.4.5	Named Entity Extraction.....	12
3.5	Data Integration.....	12
4	Feature Extraction	13-15
4.1	Introduction	13
4.2	Weight Matrix using TF.....	13
4.3	Weight Matrix Using TF-IDF.....	13
4.3.1	TF-IDF at Word Level.....	15
4.3.2	TF-IDF at N-gram Level.....	15
5	Similarity Measurement	16
5.1	Introduction.....	16
5.2	Cosine Similarity.....	16
6	Classification.....	18-21
6.1	Introduction.....	18
6.2	Naïve Bayes Classification.....	18
6.3	Logistic Regression.....	20
7	Results.....	22
8	Conclusion.....	25
9	References.....	26

Chapter 1

Introduction

1.1 Motivation

Today, the fact that most of the data is stored in the form of text and the exponential growth of this data triggers an increase in the number of text mining research work. The internet has brought about the increase of electronic documents online, further compelling textual categorization and classification of documents in various online repositories. In present time, data mining more precisely text mining, machine learning and natural language processing are quite interesting fields. Working on the text data and extracting the meaning full information is quite interesting object of study. From a large collection of text data we can get lot of meaning information which can be done using data mining techniques. In daily life we are coming across lot of text data in various forms i.e. news papers, web documents, blogs, e-books, weather-data, health and disease related data and etc. Everyday millions of text data is getting produced. Analyzing of these digital data may give us interesting results which gives us motivation to work on it. Also these data can be used for forecasting. For example by analyzing the bank statement of a person it can be predicted about his annual expenses and savings. Though, it is a tedious job for human being to do it manually. But using Data Mining techniques these kinds of work can be done easily and efficiently. Analyzing online articles also can be done very efficiently using data mining techniques which can be used to classify these articles.

1.2 Objective

In this project our principle object to analyzing online news articles and categories them into appropriate classes to which they belong based on the content the articles have. These articles dataset is collections which contains five categories.

1.3 Methodology

Data Mining is defined as the process of extracting desired information from the large amount data. In other words, we can say that data mining is mining of knowledge or patterns from the raw data. Though extraction of information is not the only the process we need to perform; data mining also involves other processes such as Data Representation, Data Preprocessing, Data Integration, Data Transformation, Feature Selection and Extraction, Classification and etc.

1.4 Summary of Contribution

Here the summary of the work carried out so far is discussed in short. The application and implementation of this project is intended for similarities measures among the collection of online BBC News article data set and classification of these articles to divide into appropriate category based on the content the articles have.

1.5 Report Outline

- Chapter 2: Discussed the Literature survey about similarity measures and classification techniques.
- Chapter 3: Presents the Raw data acquisition and preprocessing.
- Chapter 4: Presents the features and feature extraction
- Chapter 5: Presents similarity measures.
- Chapter 6: Presents the classification techniques.

Chapter 2

Literature Review

2.1 Literature survey on Similarity Measure

Study of [1] similarity between documents can be divided into three categories: first, string-based (character-based and term-based) secondly, corpus-based and finally knowledge based (similarity, relatedness). They followed term-based similarity measures i.e. Cosine similarity, Euclidean distance, Jaccard similarity and Pearson correlation [2]. Performed a comparative systematic study on similarity measure for online articles. They compared four similarity measures (Euclidean, cosine, Pearson correlation and extended Jaccard) in conjunction a variety of clustering techniques (k-means, weighted graph partitioning and random). Our project however follows a different way in that it concentrates on the only similarity measures cosine similarity among mentioned above. In their work, the cosine similarity metric performed better than the rest and applied for clustering. Our work concentrates on cosine similarity measure and for uses it for classification.

2.2 Literature survey on Classification Techniques

There are many researches in the online news articles categorization that have been conducted until now and a lot of algorithms have been implemented in order to create a classifier which has accuracy that close to 100%. Most common algorithms like Decision Tree, K-nearest neighbor [3] have been used.

In this project work, in order to achieve optimum accuracy, we decided to implement term frequency inverse document frequency algorithm at word and n-gram level with Naïve Bayes, Logistic Regression classification techniques.

Chapter 3

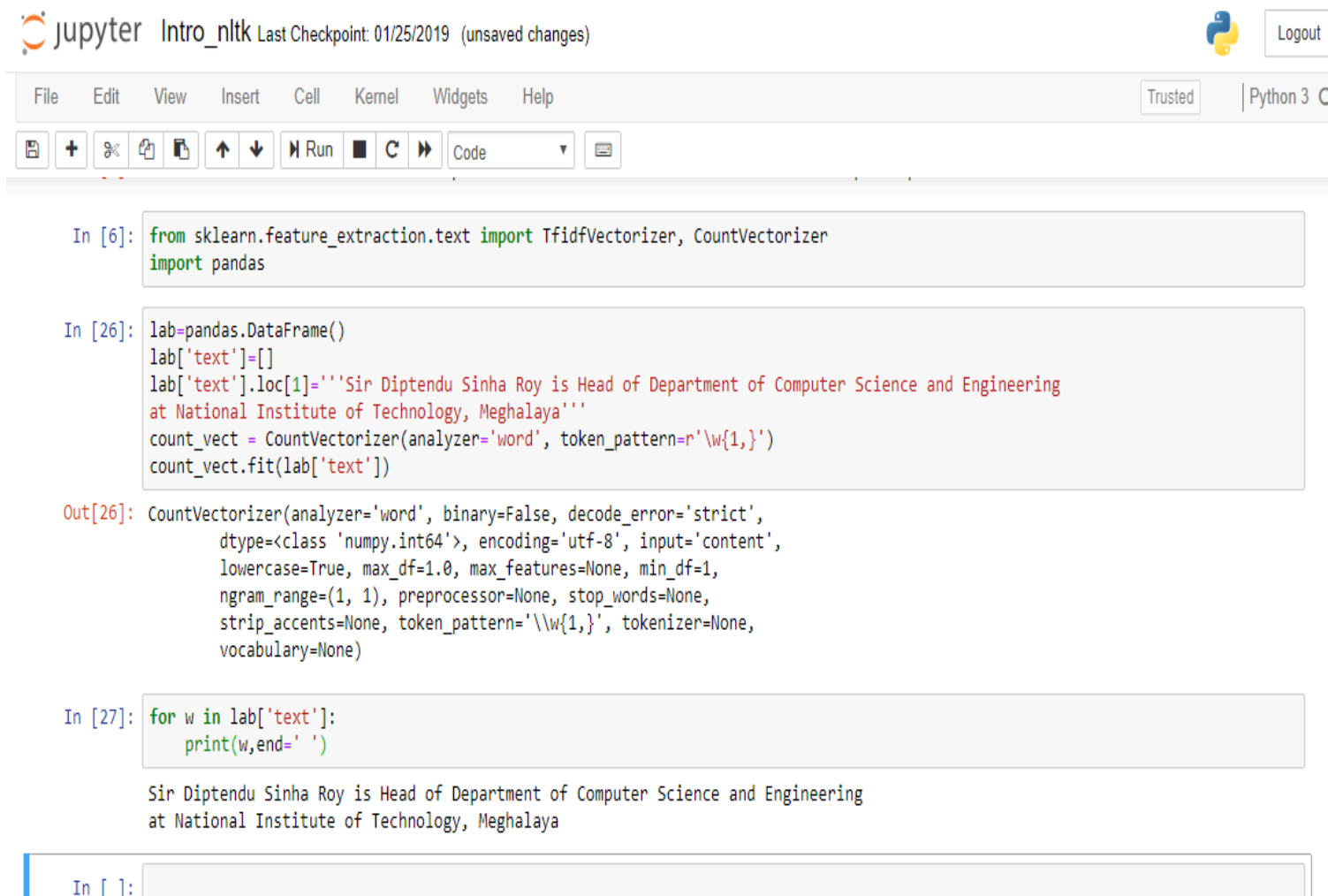
Raw Data Acquisition and Preprocessing

3.1 Raw data

Raw data is primarily unstructured, unformatted or unprocessed repository data which may contain the meaning information. It can be in the form of files, visual images, database records, ms-excel sheets or any other digital data. Raw data is extracted, analyzed, processed and used by humans or purpose-specific software to draw conclusion, make projections or extract meaningful information. Raw data acquisition can be defined as raw data collecting process. In this project we collected row data from BBC News [4]. This Dataset contains news articles which belong to five different categories. These five categories are Sports, Entertainment, Business, Politics and Tech. These five categories further contain sub categories which can be further categorized. This dataset is in document format which are written in English language.

3.2 Data Labeling

Data labeling is done to keep the prior information of articles to which category they belongs. Labeling of each article depend on the no of total class we have. Here in our project we have data of five categories. We labeled these articles with assigning tags as {1, 2, 3, 4, 5} respectively.



The image shows a Jupyter Notebook interface with the title 'Intro_nltk'. The top bar indicates the last checkpoint was on 01/25/2019 and there are unsaved changes. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for saving, adding cells, running, and other actions. The notebook contains three input cells and one output cell.

```

In [6]: from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
import pandas

In [26]: lab=pandas.DataFrame()
lab['text']=[]
lab['text'].loc[1]='''Sir Diptendu Sinha Roy is Head of Department of Computer Science and Engineering
at National Institute of Technology, Meghalaya'''
count_vect = CountVectorizer(analyzer='word', token_pattern=r'\w{1,}')
count_vect.fit(lab['text'])

Out[26]: CountVectorizer(analyzer='word', binary=False, decode_error='strict',
dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
lowercase=True, max_df=1.0, max_features=None, min_df=1,
ngram_range=(1, 1), preprocessor=None, stop_words=None,
strip_accents=None, token_pattern='\\w{1,}', tokenizer=None,
vocabulary=None)

In [27]: for w in lab['text']:
print(w,end=' ')

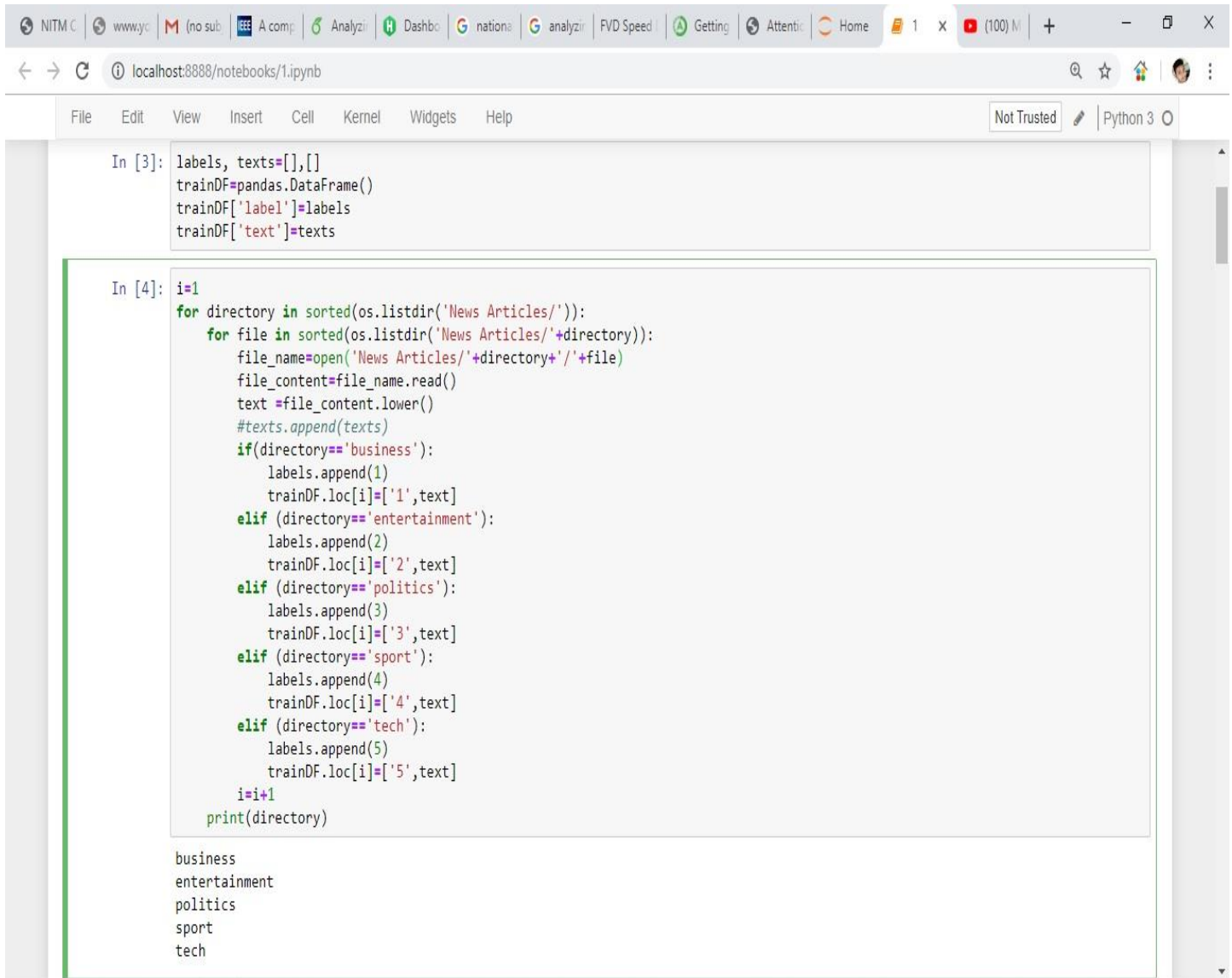
Sir Diptendu Sinha Roy is Head of Department of Computer Science and Engineering
at National Institute of Technology, Meghalaya
  
```

The output of the third cell shows the text of the article stored in the DataFrame:

```

In [ ]:
  
```

Fig 3.1 Data frame to store labeled articles



The screenshot shows a Jupyter Notebook running in a web browser at localhost:8888. The notebook has two cells. The first cell, labeled 'In [3]:', initializes a pandas DataFrame named 'trainDF' with columns 'label' and 'text'. The second cell, labeled 'In [4]:', is a loop that iterates through directories in 'News Articles/' and files within each directory. It reads the content of each file, converts it to lowercase, and appends it to a 'texts' list. Based on the directory name, it assigns a label (1 to 5) and appends it to the 'labels' list. The output of the second cell shows the directories: business, entertainment, politics, sport, and tech.

```
In [3]: labels, texts=[],[]
trainDF=pandas.DataFrame()
trainDF['label']=labels
trainDF['text']=texts

In [4]: i=1
for directory in sorted(os.listdir('News Articles/')):
    for file in sorted(os.listdir('News Articles/'+directory)):
        file_name=open('News Articles/'+directory+'/'+file)
        file_content=file_name.read()
        text =file_content.lower()
        #texts.append(texts)
        if(directory=='business'):
            labels.append(1)
            trainDF.loc[i]='1',text]
        elif (directory=='entertainment'):
            labels.append(2)
            trainDF.loc[i]='2',text]
        elif (directory=='politics'):
            labels.append(3)
            trainDF.loc[i]='3',text]
        elif (directory=='sport'):
            labels.append(4)
            trainDF.loc[i]='4',text]
        elif (directory=='tech'):
            labels.append(5)
            trainDF.loc[i]='5',text]
        i=i+1
    print(directory)

business
entertainment
politics
sport
tech
```

Fig 3.2 Labeling of Articles

```

In [5]: len(trainDF)
Out[5]: 2225

In [6]: trainDF
Out[6]:

```

	label	text
1	1	ad sales boost time warner profit\n\nquarterly...
2	1	dollar gains on greenspan speech\n\nthe dollar...
3	1	yukos unit buyer faces loan claim\n\nthe owner...
4	1	high fuel prices hit ba's profits\n\nbritish a...
5	1	pernod takeover talk lifts domecq\n\nshares in...
6	1	japan narrowly escapes recession\n\njapan's ec...
7	1	jobs growth still slow in the us\n\nthe us cre...
8	1	india calls for fair trade rules\n\nindia, whi...
9	1	ethiopia's crop production up 24%\n\nethiopia ...
10	1	court rejects \$280bn tobacco case\n\nna us gove...
11	1	ask jeeves tips online ad revival\n\nask jeeve...

Fig 3.3 Labeled Articles

3.3 Data Representation

In vector space model data are represented in vector form. Each article or text document is represented by a vector is called vector space model [5]. These vectors are used in further processing like feature extraction and similarity measurement and etc. These vectors contain each index value as term or token.

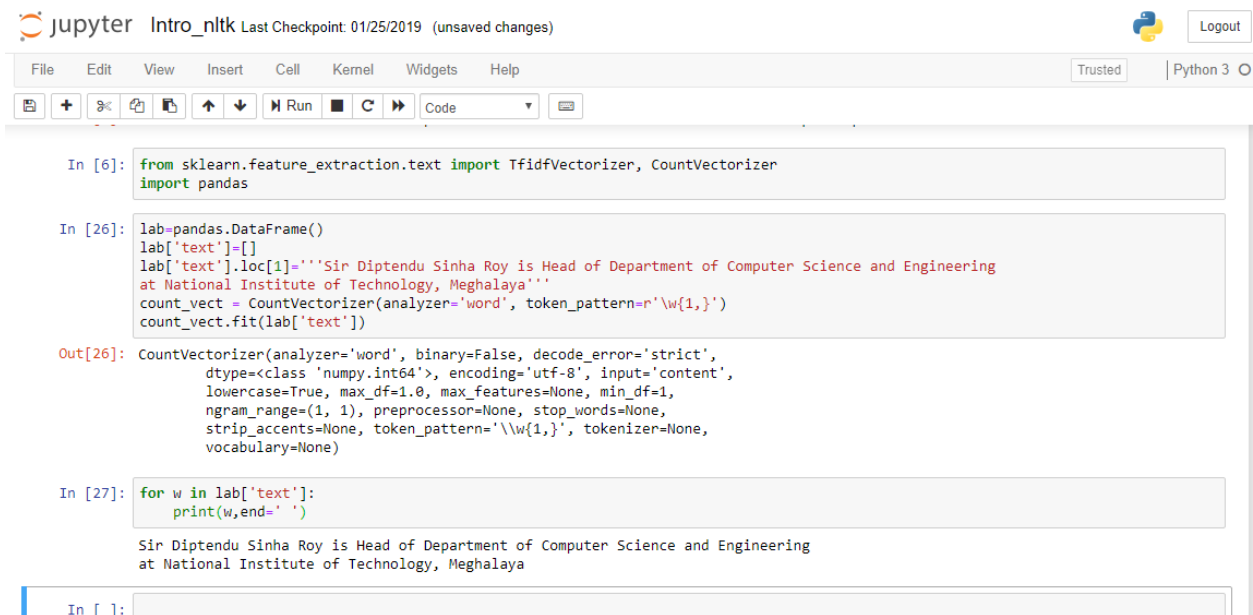
3.4 Data Preprocessing

Data preprocessing [6] is a data mining technique that involves transforming raw data into machine understandable format. Real-world data is often incomplete, inconsistent, or lacking in certain behaviors or trends, and is likely to contain many errors which required data preprocessing to resolve these kinds of issues. Data preprocessing prepares or converts raw data which is used for further processing. Data preprocessing is a collection of techniques which are tokenization, conversion into lower case letters, stemming, stop word removing, lemmatization and data Integration.

3.4.1 Tokenization

A sentence can be defined as a complete set of words. Tokenization [7] is defined as splitting a sentence into words. Though we know that an article is a collection of more than one sentence, so first it is required to bring out sentences one by one from the article and then only it can be divided into tokens.

Example “Sir Diptendu Sinha Roy is Head of Department of Computer Science and Engineering at National Institute of Technology, Meghalaya”.



The screenshot shows a Jupyter Notebook with the following code and output:

```
In [6]: from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
import pandas

In [26]: lab=pandas.DataFrame()
lab['text']=[]
lab['text'].loc[1]='''Sir Diptendu Sinha Roy is Head of Department of Computer Science and Engineering
at National Institute of Technology, Meghalaya'''
count_vect = CountVectorizer(analyzer='word', token_pattern=r'\w{1,}')
count_vect.fit(lab['text'])

Out[26]: CountVectorizer(analyzer='word', binary=False, decode_error='strict',
dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
lowercase=True, max_df=1.0, max_features=None, min_df=1,
ngram_range=(1, 1), preprocessor=None, stop_words=None,
strip_accents=None, token_pattern='\\w{1,}', tokenizer=None,
vocabulary=None)

In [27]: for w in lab['text']:
print(w,end=' ')
```

The output of the last cell shows the tokenized sentence: Sir Diptendu Sinha Roy is Head of Department of Computer Science and Engineering at National Institute of Technology, Meghalaya

Fig 3.4 Tokenization of the sentence.

3.4.2 Stop Word Removing

Every language has its own grammar which contains lot of words. These words are used commonly in sentences. While knowledge extraction from text data it is required to remove these words before preprocessing. These stop words may differ from language to language. In English natural language processing these words can be imported from the NLTK library. These stop words in English language can be helping verbs, determiners, punctuation, special characters and etc.

In the above example stop words are {'‘', 'is', 'of', 'and', 'at', '’'}

3.4.3 Stemming

While preprocessing of raw data it is required to find the root word of each token. Stemming [8] ensures that all words having the same stem are counted together. That means the terms eat, eating, eatable, eats — all with the same stem eat will be counted together. The process of find root words is called by stemming. The goal of both stemming and lemmatization is to reduce inflectional forms.

Example: “the boy's cars are different colors”.

After stemming: “the boy car be differ color”.

3.4.4 Chunking

The process of tagging the POS is called chunking [6]. From the bag of words the structure of sentence cannot be captured so it is required to divide the whole text in chunks. Later on that can be used to reframe the sentence. For chunking it is required to parse the whole text from the regular grammar. There are eight main part of a sentence.

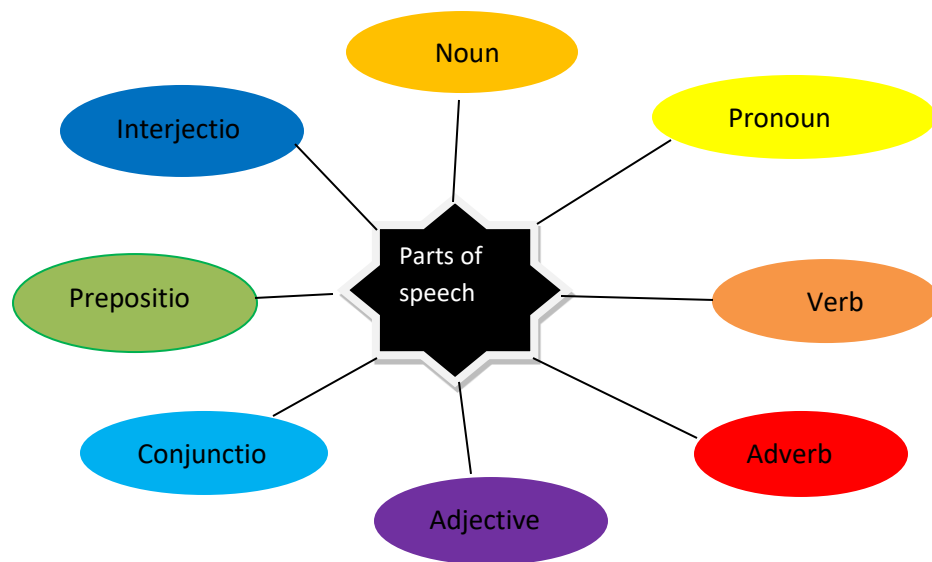


Fig 3.5 Tags in a sentence[8]

Example “Everything to permit us”.

POS [(‘Everything’, NN), (‘to’, TO), (‘permit’, VB), (‘us’, PRP)].

For the above example pos tags are

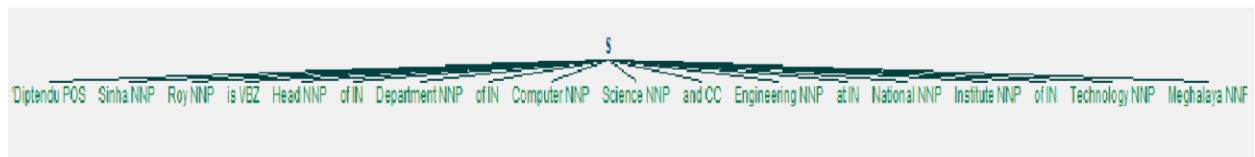


Fig 3.6: Chunking of a sentence

Speech Tags

S.No	Tag	Description
1	CC	Coordinating conjunction
2	CD	Cardinal number
3	DT	Determiner
4	EX	Existential there
5	FW	Foreign word
6	IN	Preposition
7	JJ	Adjective
8	JJR	Adjective, comparative
9	JJS	Adjective, superlative
10	LS	List item marker
11	MD	Modal
12	NN	Noun singular
13	NNS	Noun, plural
14	NNP	Proper noun, singular
15	NNPS	Proper noun, plural
16	PDT	Pre determiner
17	POS	Possessive ending
18	PRP	Personal pronoun
19	PRP	Possessive pronoun
20	RB	Adverb
21	RBR	Adverb, comparative
22	RBS	Adverb, superlative
23	RP	Particle
24	SYM	Symbol
25	TO	To
26	UH	Interjection
27	VB	Verb, base form
28	VBD	Verb, past tense
29	VBG	Verb, gerund or present participle
30	VBN	Verb, past participle
31	VBP	Verb, non-3rd person participle
32	VBZ	Verb, 3rd person singular present
33	WDT	Wh-determiner
34	WP	Wh-pronoun
35	WP\$	Possessive wh-pronoun
36	WRB	Wh-adverb

Table 3.1: Set of part of speech tags

3.4.5 Named Entity Extraction

As name suggest named entity gives the information about the objects in text document. These entities are like people, location, organization, job title, percentages, date and time and etc.

In the above example NEEs are {(`Diptendu Sinha Roy`, person), (`Meghalaya`, location) }

3.5 Data Integration

After preprocessing of the raw data it is required to integrate the whole preprocessed data. Sometimes it is called as bag of words also. For similarity measure the bag of words is directly used to assign overall weight throughout whole dataset.

Chapter 4

Feature Extraction

4.1 Introduction

Feature selection and extraction is the key part for the further processing. Each feature has its different applications so selection of appropriate feature plays a vital role in feature extraction process. To process these articles it is required to selected TF and TF-IDF [10] features.

4.2 Weight Matrix using TF

Weight matrix using term frequency is based on the frequency of each term that how many times it is occurring in a particular article. A term having high frequency will get weight high and vice verse. Weight matrix using TF is a vector which contains term and its weight in the document.

TF weight is calculated in equation (4.1).

$$TF \text{ of term } i = \frac{\text{freq.of term } i \text{ in the document}}{\text{Total no terms in BOW}} \dots\dots\dots(4.1)$$

4.3 Weight matrix using TF-IDF

In this approach [11] each term assigned weight as the multiplication of its TF and IDF. IDF is inverse document frequency which is calculated that in how many documents a particular term is occurred. TF-IDF gives more weight to less frequent terms and less weight to more frequent term. According to this approach it is concluded that less frequent term affects their similarity and weight matrix more.

Weight matrix using TF-IDF is a vector which contains terms and its weight in the whole dataset. IDF is calculated in equation (4.2).

$$\text{IDF} = \text{Log}\left(\frac{N}{\text{df}_i}\right) \dots\dots\dots(4.2)$$

By equation 4.1 and 4.2 weight using TF-IDF can be calculated.

$$W_{i,j} = \text{tf}_i \times \text{Log}\left(\frac{N}{\text{df}_i}\right) \dots\dots\dots(4.3)$$

$\text{TF}_{i,j}$ = number of occurrences of i in j

IDF_i = number of documents containing i

N = total number of document

```
In [62]: def computeTF(tkn_frq,n):
#unq_tkn=set(tokens)
#tkn_frq=dict.fromkeys(unq_tkn,0)
#for words in tokens:
#    tkn_frq[words]+=1
tfDict={}
for word,count in tkn_frq.items():
    tfDict[word]=count/float(n)
return tfDict

In [103]: def computeIDF(docList):
import math
N=len(docList)
x=[]

for doc in docList:
    for word, val in doc.items():
        if(val>=0):
            x.append(word)
idfDict = dict.fromkeys(set(x), 0)
for doc in docList:
    for word, val in doc.items():
        if val > 0 :
            idfDict[word] += 1

for word, val in idfDict.items():
    idfDict[word] = math.log10(N / float(val))

return idfDict
```

Fig 4.1 TF-IDF computation method

4.3.1 TF-IDF at Word Level

TF-IDF at word level considers a list of word and assigns weight to each word. In the article dataset we have to take a single word at a time from each and calculate the TF-IDF. As earlier mentioned, TF-IDF requires both TF and IDF need to calculate separately. Then only final weight can be calculated.

4.3.2 TF-IDF at n-gram Level

n-gram [12] is a contiguous sequence of n terms in a given text document. Here n is a natural number. n values depend on the number of contiguous terms we are considering for tokenization assuming as a single token.

n = 1 is defined as unigram.

n = 2 is defined as bigram.

n = 3 is defined as trigram.

After applying n-gram on a text data it gives the total no of tokens are in eq (4.4)

$$\text{No. of Tokens} = \text{len}(\text{Text}) - N + 1 \dots\dots\dots(4.4)$$

Example 4.2: “The best and most beautiful things in the world cannot be seen or even touched, they must be felt by heart”.

Applying n-grams on the above example:

n = 1 unigrams = { The, best, and, most, beautiful, things, in, the, world, can, not, be, seen, or, even, touched, ,, they, must, be, felt, by, heart }.

n=2 bigrams = { ('The', 'best'),('best', 'and'),('and', 'most'),('most', 'beautiful'),('beautiful', 'things'),('things', 'in'),('in', 'the'),('the', 'world'),('world', 'can'),('can', 'not'),('not', 'be'),('be', 'seen'),('seen', 'or'),('or', 'even'),('even', 'touched'),('touched', ','),(',' , 'they'),('they', 'must'),('must', 'be'),('be', 'felt'),('felt', 'by'),('by', 'heart')}.

n=3 trigrams={('The', 'best', 'and'),('best', 'and', 'most'),('and', 'most', 'beautiful'),('most', 'beautiful', 'things'),('beautiful', 'things', 'in'),('things', 'in', 'the'),('in', 'the', 'world'),('the', 'world', 'can'),('world', 'can', 'not'),('can', 'not', 'be'),('not', 'be', 'seen'),('be', 'seen', 'or'),('seen', 'or', 'even'),('or', 'even', 'touched'),('even', 'touched', ','),(',' , 'they'),('', 'they', 'must'),('they', 'must', 'be'),('must', 'be', 'felt'),('be', 'felt', 'by'),('felt', 'by', 'heart')}

Chapter 5

Similarity Measurement

5.1 Introduction

As name suggest similarity measure is measurement that two articles or documents are how much similar to each other. These similarity measures help in article summarizations, information retrieval, article clustering and categorization. To apply these algorithms on the text data it requires data to be vectorized. We applied Cosine and Euclidean similarities.

5.2 Cosine Similarity

Cosine similarity [2] is similarity measurement among two articles or documents which show how much these are similar to each other. Cosine similarity is a mathematical approach which uses dot product of two vectors. Mathematically, it can be defined as measurement of angle among these two vectors. The values of equation (5.1) vary in the interval [0,1]. Here if similarity value is zero means two articles do not have any similarity and if near to one then articles are most similar to each other and so on.

By applying the cosine, the similarity value is calculated in equation (5.1).

$$\cos \theta = \frac{\vec{v}_{d1} \cdot \vec{v}_{d2}}{|\vec{v}_{d1}| \times |\vec{v}_{d2}|} \dots\dots\dots(5.1)$$

\vec{V}_{d1} = vector representation of article-1

\vec{V}_{d2} = vector representation of article-2.

$|\vec{V}_{d1}|$ = Magnitude of vector d1.

$|\vec{V}_{d2}|$ = Magnitude of vector d2.

```

5
Executed Successfully

In [126]: def cosine_similarity(a,b):
          dotProduct=np.dot(a,b)
          norm_a=np.linalg.norm(a)
          norm_b=np.linalg.norm(b)
          c=dotProduct/(norm_a*norm_b)
          return c;

In [164]: def getsimilarity(d1,d2):
          d1_d2_words=[]
          for word,key in d1.items():
              d1_d2_words.append(word)
          for word,key in d2.items():
              d1_d2_words.append(word)
          d1_d2_words=set(d1_d2_words)
          v1=np.zeros(np.array(len(d1_d2_words),int))
          v2=np.zeros(np.array(len(d1_d2_words),int))
          i=0
          for w in d1_d2_words:
              if w in d1:
                  v1[i]=d1[w]
              if w in d2:
                  v2[i]=d2[w]
              i+=1
          return cosine_similarity(v1,v2)
  
```

Fig 5.1 Cosine Similarity measurement method

Chapter 6

Classification

6.1 Introduction

The process of assigning a data element to a proper predefined class based on the belongingness it has is called by classification. Classification of text data or Text classification is a way to automatically categorize uncategorized data into several predefined categories. In classification, it is required to have the prior class level information. Classification is a supervised learning based approach in which we have to train our classification model with available training data set. To classify the news articles, classification model is trained with news article dataset. For this, it requires to do labeling of each and every news article with the relative class tag. In this project, there are five classes which are labeled by { 1, 2, 3, 4, 5 }. Each article is assigned these classes as business with label-‘1’, entertainment with label-‘2’, politics with label-‘3’, tech with label-‘4’ and sport with label-‘5’. Three classification approaches are used to classify these news articles.

6.2 Naïve Bayes classification

Naïve Bayes classification is a probabilistic approach which works on the Bayes theorem of conditional probability. It finds the independence between the features of text the article. For a text article, Naïve Bayes classifier finds out a class with maximum posterior probability.

The Bayes theorem is given as follows:

$$P(class/doc) = \frac{P(doc/class)P(class)}{P(doc)} \dots\dots\dots(6.1)$$

$P(class/doc)$ = Probability of an article which belongs a particular class c .
known as posterior probability.

$P(doc/class)$ = Likelihood.

$P(class)$ = Prior probability.

```

trainDF['pron_count'] = trainDF['text'].apply(lambda x: check_pos_tag(x, 'pron'))

In [16]: def train_model(classifier, feature_vector_train, label, feature_vector_valid, i
# fit the training dataset on the classifier
classifier.fit(feature_vector_train, label)

# predict the labels on validation dataset
predictions = classifier.predict(feature_vector_valid)

if is_neural_net:
    predictions = predictions.argmax(axis=-1)

return metrics.accuracy_score(predictions, valid_y)

In [17]: ##### Classifier training with different feature
NB_Classifier_tfidf=naive_bayes.MultinomialNB()
NB_Classifier_ngram=naive_bayes.MultinomialNB()

In [18]: accuracy = train_model(NB_Classifier_tfidf, xtrain_tfidf, train_y, xvalid_tfidf)
print ("Accuracy of Naive Bayes classifier using WordLevel TF-IDF features : ",

```

Fig 6.1 Naïve Bayes classification

6.3 Logistic Regression

Logistic Regression [13] classification technique includes three algorithms which are binomial, multinomial and ordinal. Binomial deals with two class problem only. But when there is more than two classes, multinomial logistic regression can be used.

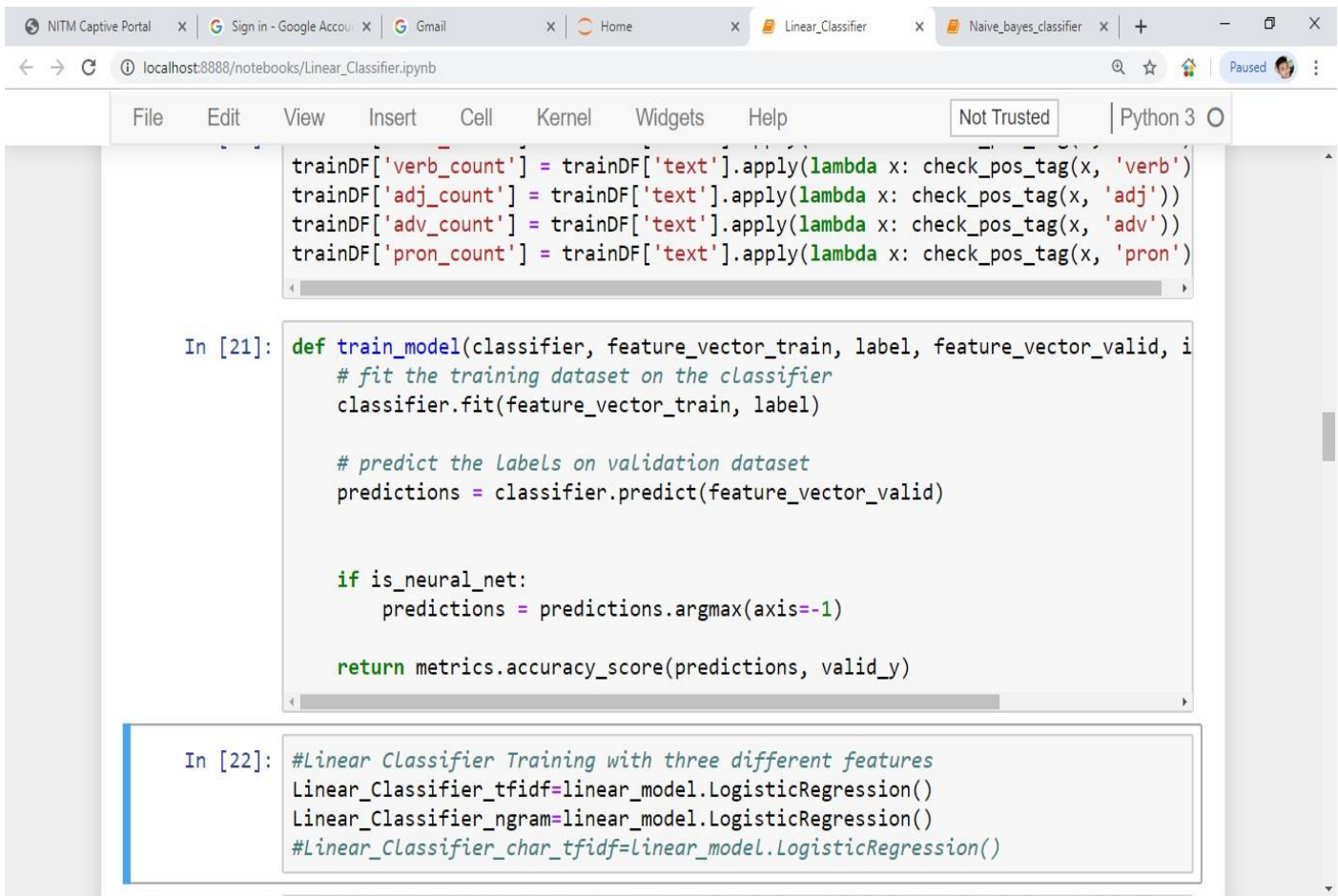
Multinomial logistic regression is a linear regression based algorithm which uses it to calculate score or logits. Logits are the score for each class which is given in equation (6.2).

$$\text{logit} = w X + b$$

These logits are used by softmax function which gives the probabilistic score of each target class. For high value of logits it gives high probabilistic value. For k no of classes it gives the probabilistic score as follows: x_i

$$F(x_i) = \frac{\text{Exp}(x_i)}{\sum \text{Exp}(x_j)} \quad i, j \text{ varies from } 0 \text{ to } k.$$

These output from softmax function are used to find the cross entropy, basically it is a distance function which takes two argument to find the distance. These two are hot encoding matrix and softmax probabilistic matrix for the target classes. One Hot encoding matrix is created for each feature vector with the value of 0 and 1 for the target class. Now cross entropy is calculated using these two. For the right target class it gives less distance and vice-versa.



```

trainDF['verb_count'] = trainDF['text'].apply(lambda x: check_pos_tag(x, 'verb'))
trainDF['adj_count'] = trainDF['text'].apply(lambda x: check_pos_tag(x, 'adj'))
trainDF['adv_count'] = trainDF['text'].apply(lambda x: check_pos_tag(x, 'adv'))
trainDF['pron_count'] = trainDF['text'].apply(lambda x: check_pos_tag(x, 'pron'))

In [21]: def train_model(classifier, feature_vector_train, label, feature_vector_valid, i
        # fit the training dataset on the classifier
        classifier.fit(feature_vector_train, label)

        # predict the labels on validation dataset
        predictions = classifier.predict(feature_vector_valid)

        if is_neural_net:
            predictions = predictions.argmax(axis=-1)

        return metrics.accuracy_score(predictions, valid_y)

In [22]: #Linear Classifier Training with three different features
Linear_Classifier_tfidf=linear_model.LogisticRegression()
Linear_Classifier_ngram=linear_model.LogisticRegression()
#Linear_Classifier_char_tfidf=linear_model.LogisticRegression()

```

Fig 6.2 Logistic Regression

Results

1. Similarity Measurements Results

The cosine similarity among the articles using TF and TF-IDF is calculated. The maximum similarity using TF between the articles by **Benjamin.txt** and **wheaton.txt** is 0.90260938440267. The similarity value among articles that we have collected is as following:-

S. No.	Article1	Article2	Cosine Similarity
1	Benjamin.txt	Humpries.txt	0.8481566461309088
2	Benjamin.txt	wheaton.txt	0.9026093844026722
3	Humpries.txt	wheaton.txt	0.8584297334173029
4	Dying to live.txt	Humpries.txt	0.8324191794512513
5	Dying to live.txt	Karen Kaiser.txt	0.7842654981676214
6	Humpries.txt	Karen Kaiser.txt	0.6946267093239189
7	Dying to live.txt	wearing a mask.txt	0.8596955650363534
8	Dying to live.txt	Sarah Fader.txt	0.8351941503700233
9	wearing a mask.txt	Sarah Fader.txt	0.8605026230099089

Cosine Similarity value on the BBC News article data using TF features is 0.463923 and using TF-IDF is 0.829467.

2. Classification Results

Classification accuracies using different classifiers with different feature vectors are as follows:

S. No	Features Vector	Classifier	Accuracy
1	Word Level TF-IDF	Naïve Bayes Classifier	0.9610778443113772
2	n-gram level TF-IDF	Naïve Bayes Classifier	0.9236526946107785
3	Word Level TF-IDF	Logistic Regression	0.9730700179533214
4	n-gram level TF-IDF	Logistic Regression	0.9461400359066428

```
(base) D:\project>python Naive_bayes_classifier.py
Categories of different NEWS articles are as following:
business
entertainment
politics
sport
tech
Total No of NEWS Articles in BBC News Dataset: 2225
Splitting the dataset into Training and Testing Data with ratio 70:30
Size of training and testing dataset size : 1557 668
Accuracy of Naive Bayes classifier using WordLevel TF-IDF features : 0.9670658682634731
Accuracy of Naive Bayes classifier using n-gram at n=2 TF-IDF features: 0.9476047904191617
Validation of Trained Models for different Articles :
Enter article no to predict category of it's from which it belongs:
1
predicted Class Type of selected NEWS Article using word level tf-idf features : [0]
Prediction is wrong
predicted Class Type of selected NEWS Article using n-gram tf-idf features : [0]
Prediction is wrong
Enter 1 to continue and 0 to exit:
1
Enter article no to predict category of it's from which it belongs:
2
predicted Class Type of selected NEWS Article using word level tf-idf features : Class No [3] Politics
Labeled class of document 2 is 3
predicted Class Type of selected NEWS Article using n-gram tf-idf features : Class No [3] Politics
Labeled class of document 2 is 3
Enter 1 to continue and 0 to exit:
1
Enter article no to predict category of it's from which it belongs:
3
predicted Class Type of selected NEWS Article using word level tf-idf features : Class No [4] Technology
Labeled class of document 3 is 4
predicted Class Type of selected NEWS Article using n-gram tf-idf features : Class No [4] Technology
Labeled class of document 3 is 4
Enter 1 to continue and 0 to exit:
1
Enter article no to predict category of it's from which it belongs:
5
predicted Class Type of selected NEWS Article using word level tf-idf features : Class No [2] Entertainment
Labeled class of document 5 is 2
predicted Class Type of selected NEWS Article using n-gram tf-idf features : Class No [2] Entertainment
Labeled class of document 5 is 2
```

Fig 7.1 Naïve Bayes Classification based Results


```

Anaconda Prompt - python Linear_Classifier.py
(base) D:\project>python Linear_Classifier.py
Categories of different NEWS articles are as following:
business
entertainment
politics
sport
tech
Total No of NEWS Articles in BBC News Dataset: 2225
Splitting the dataset into Training and Testing Data with ratio 75:25
Size of training and testing dataset size : 1668 557
Accuracy of Linear classifier using WordLevel TF-IDF features : 0.9748653500897666
Accuracy of Naive Bayes classifier using n-gram at n=2 TF-IDF features: 0.947935368043088
Validation of Trained Models for different Articles :
Enter article no to predict category of it's from which it belongs:
1
predicted Class Type of selected NEWS Article using word level tf-idf features : Class No [1] Bussiness
Labeled class of document 1 is 1
predicted Class Type of selected NEWS Article using n-gram tf-idf features : Class No [3] Politics
Labeled class of document 1 is 1
Enter 1 to continue and 0 to exit:
1
Enter article no to predict category of it's from which it belongs:
9
predicted Class Type of selected NEWS Article using word level tf-idf features : Class No [3] Politics
Labeled class of document 9 is 3
predicted Class Type of selected NEWS Article using n-gram tf-idf features : Class No [3] Politics
Labeled class of document 9 is 3
Enter 1 to continue and 0 to exit:
1
Enter article no to predict category of it's from which it belongs:
10
predicted Class Type of selected NEWS Article using word level tf-idf features : Class No [2] Entertainment
Labeled class of document 10 is 2
predicted Class Type of selected NEWS Article using n-gram tf-idf features : Class No [2] Entertainment
Labeled class of document 10 is 2
Enter 1 to continue and 0 to exit:
1
Enter article no to predict category of it's from which it belongs:
12
predicted Class Type of selected NEWS Article using word level tf-idf features : Class No [3] Politics
Labeled class of document 12 is 3
predicted Class Type of selected NEWS Article using n-gram tf-idf features : Class No [3] Politics
Labeled class of document 12 is 3
Enter 1 to continue and 0 to exit:

```

Fig 7.2 Logistic Regression Classification based Results

Conclusion

1. For small data set, cosine similarities give good results for both TF and TF-IDF feature vectors. But for large data set cosine similarities giving good results for only TF-IDF feature vectors.
2. Classification accuracies are achieved far better using Naïve Bays classifier and Logistic Regression then support vector machine classifier.

References

- [1] A. A. F. Weal H. Goma, "A Survey of Text Similarity Approaches.," *International journal of computer application*, vol. 68, 2013.
- [2] E. S. , J. G. , R. M. Alexander Strehl, "Impact of similarity measure on web-pages clustering," in *Workshop on Artificial intelligence for web search*, Austin, USA, 2000.
- [3] T. Sallas, "Automatic Document Classification Temporarily Robust," in *DBLP*.
- [4] kaggle, "BBC NEWS," BBC NEWS, [Online]. Available: <https://www.kaggle.com/antonibap/datamining-bbc-news/data>. [Accessed 15 11 2018].
- [5] "wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Vector_space_model. [Accessed 10 11 2018].
- [6] D. A. Arora, "BIOMBIOTEK," Winter School, [Online]. Available: http://www.iasri.res.in/ebook/win_school_aa/notes/Data_Preprocessing.pdf. [Accessed 20 10 2018].
- [7] "No Free Hunch," [Online]. Available: <http://blog.kaggle.com/2017/08/25/data-science-101-getting-started-in-nlp-tokenization-tutorial/>. [Accessed 19 10 2018].
- [8] "stanford cse learning platform," [Online]. Available: <https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html>. [Accessed 02 12 2018].
- [9] "Natural Language Processing for Hackers," [Online]. Available: <https://nlpforhackers.io/text-chunking/>. [Accessed 05 12 2018].
- [10] "stanford cse," [Online]. Available: <https://nlp.stanford.edu/IR-book/pdf/06vect.pdf>. [Accessed 19 01 2019].
- [11] R. K. R. J. K. Sahoo, "Modified TF-IDF Term Weighting Strategies for Text Categorization," in *14th IEEE India Council International Conference*, OCT 2018.
- [12] D. Singh, "Intra News Category Classification using N-gram TF-IDF Features and Decision Tree Classifier," *IJSART*, vol. 4, no. 3, p. 7, 2018.

[13] D. Pool, "Dataaspirants," [Online]. Available: <http://dataaspirant.com/2017/03/14/multinomial-logistic-regression-model-works-machine-learning/>. [Accessed 05 05 2019].

[14]

[15] A. S. Chee-hong Chan, "Automated News classification," in *4th International conference of Asian Digital Library(ICADL)*, Dec,2001.