Implementing Data mining for Text Summarization and Topic Detection

A PROJECT REPORT

submitted by

Manorath Bajaj (14BCE0442)

in partial fulfillment for the award of the degree of

B. Tech
in

Computer Science and Engineering

Vellore-632014, Tamil Nadu, India
School of Computer Science and Engineering
APRIL, 2018

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

DECLARATION

I hereby declare that the project entitled **"Implementing Data Mining for Text Summarization and Topic Detection "** submitted by me to the School of Computer Science and Engineering, Vellore Institute of Technology, Vellore-14 towards the partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out by me under the supervision of **S.Sivanesan, Asst. Prof.** I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

Signature
Name : Manorath Bajaj
Reg.No: 14BCE0442

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

CERTIFICATE

The project report entitled "**Implementing Data Mining for Text Summarization and Topic Detection**" is prepared and submitted by **Manorath Bajaj(14BCE0442)** has been found satisfactory in terms of scope, quality and presentation as partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** in Vellore Institute of Technology, Vellore-14, India.

Guide
(Name & Signature)

Internal Examiner
(Name& Signature)

External Examiner
(Name & Signature)

ACKNOWLEDGEMENT

The project report couldn't have been accomplished without the splendid support and cooperation of my internal guide, **Professor S. Sivanesan** and I would like to express my gratitude to him for his unwavering support and the infinite amount of time spent with me helping to do my project.

I am very thankful to my respected Head of Department, **Professor V. Santhi** for the confidence he had on me regarding this project and for inspiring and motivating me to bring out a successful project.

It is my privilege to express heartfelt thanks to the Dean, School of Computer Science and Engineering, **Dr. Saravanan R** for this kind of encouragement for all endeavors upon this project.

I would also like to express our sincere gratitude to my college **Vellore Institute of Technology University, Vellore** for providing me with the infrastructure and the opportunity to undertake and complete such an interesting project report. My special thanks to the chancellor of VIT, **Dr. G. Viswanathan**, for giving me the opportunity to pursue my studies in this prestigious university.

Abstract

In todays world, with so much data free flowing, It turns very hard to keep track of everything. This project aims to develops a program that can correctly classify data into various topics for a better organization of data. Making it simple to use, the program works just by the user giving it a link of an online article, the program will detect the main topic(Of the 10 it is trained to do so), and will give a summary based on the user input of the number of sentences. Best Multinomial Naïve Bayes Classifier was found to be the best classifier for the following task and has been used as the training classifier. Summarization works in an extractive way and calculates the most important sentences in the article and displays the number which user wants in the order of occurrence in the actual article.

List of Figures

CONTENTS

1. Introduction

1.1. Theoretical Background

There is a lot of data generated everyday .With that generating every day, it is making it excessively hard to search for the exact thing one requires. Therefore, new ways to scrounge them at a fast pace is required.

Topic modelling is one such method that can help us to classify and summarize large amounts of online data. Not only does it detect the topics, it can also find hidden similarities between similar topic and further understand the learning.

Topic modelling bases the closest keywords on the TFIDF function. Each word is given a fraction which increases on how many times it comes in the program. This is based on how the program interprets data. TFIDF is vastely based on training data and the classifier.

Naïve Bayes classifier is a standout amongst the best machine learning calculations executed in machine learning ventures and disseminated MapReduce usage utilizing Apache Spark. Fundamentally Naïve Bayes is a direct classifier, which is a managed machine learning technique and fills in as a probabilistic classifier also. More often than not, for the numeric usage K-Nearest Neighbors and K-Means grouping calculations can be executed. Naïve Bayes classifier works successfully to classify messages, writings, images, and names. It's not uncommon Naïve Bayes classifier is utilized for numeric information also in a few occasions. Naïve Bayes classifier can be executed on high-dimensional datasets successfully too. Naïve Bayes classifier predicts the likelihood of each class in view of the element vector for content arrangement for consistent huge information with an earlier circulation of the likelihood, handling the difficulties of the scourge of the dimensionality. There are three sorts of Naïve Bayes classifiers. When taking care of constant information with consistent appropriation, Naïve Bayes classifier considers that the enormous information is produced through a Gaussian procedure with typical dissemination. Multinomial Naïve Bayes classifier can be connected when taking care of occasion models where the occasions are displayed through a multinomial dispersion. In this circumstance, the highlights are frequencies. In the third situation, when the highlights are Boolean or free, the highlights are created through a Bernoullian procedure. In this situation, a Bernoulli Naïve Bayes classifier can be connected.

1.2. Motivation

Monetary, scholastic and social exercises create consistently expanding amounts of information. Organizations gather trillions of bytes of data on client exchanges, providers, inner activities and in fact contenders the worldwide research group creates more than 1.5 million new insightful articles for each annum; and interpersonal interaction locales, for example, Facebook and twitter empower clients to share more than 1.3 billion snippets of data/content every day.

the age of data and information has turned into a 'deluge', filling all areas of the worldwide economy and is anticipated to increment at a rate of 40% every year. Misuse of this huge information and data asset can create huge monetary advantages.

Content mining is required if associations and people are to understand these immense data and information assets and use esteem. The assets require first to be handled – gotten to, broke down, commented on and identified with existing data and comprehension. The handled information would then be able to be 'mined' to distinguish examples and concentrate significant data and new learning. How these data and information assets are investigated relies upon their arrangement. Organized information can be moderately effectively 'mined' as the structure can be utilized to help preparing. Utilizing a PC to naturally break down data contained in reports is however substantially more troublesome. Most computerized reports comprise of unstructured content containing level information, as opposed to organized and important data, which can't straightforwardly be naturally prepared by a PC helpfully. 'Content mining' along these lines includes more confounded procedures than organized information mining, and it is the procedures included that offer ascent to the contention with copyright law. Given the volume of content produced by business, scholarly and social exercises – in for instance contender reports, explore productions or client conclusions on interpersonal interaction locales – content mining is, notwithstanding, exceedingly imperative.

1.3. Aim of the proposed work.

The aim of this project is to Implement Data mining for Text Summarization and Topic Detection for online articles

1.4. Objectives of the proposed work

The main objective of the proposed work is to create a program for topic detection making it accurate using machine learning for written documents and dividing the articles with a summary in sub-groups.

2. Literature Survey

2.1. Survey of existing models

Most of the data in the world is currently unstructured and is pretty hard to classify in the current meta. Information is coming in ma faster pace that it can be processed and need new Machine learning methods to better process and classify the data. SUmmrization is still in its infancy but extractive summarization is the best form we can use nowadays. (Kalogeratos and Likas, 2011).

The is a lots of textual data, Not everything is useful to one particular person but the thing that are important are very hard to find. Hence there is a big need of programs that can look up large textual datasets and classify them accordingly.(Chen et al, 2010)Though many text mining algos are present, the new world is taking more and more problems into consideration. Text Classification will be one such example of a problem.

There are many other examples of becoming a high data cost problem. This is one of those high data cost problem one becomes enormous, resulting in high computational cost problems found in the program and long time of processing the data. (Luo et al, 2009).

However, the dimensionality can be reduced through feature extraction algorithms. Topic summarization (Forestier et al, 2010) in terms of content coverage, coherence, and consistency, the summaries are superior to those derived from existing summarization methods based on human-composed reference summaries. Text mining is the automatic and semi-automatic extraction of implicit, previously unknown, and potentially useful information and patterns, from a large amount of unstructured textual data, such as natural-language texts. In text mining, each document is represented as a vector, whose dimension is approximately the number of distinct keywords in it, which can be very large. One of the main challenges in text mining is to classify textual data with such high dimensionality (Song et al, 2013).

Unsupervised learning (Ilin, 2012) is a technique in which the algorithm uses only the predictor attribute values. There is no target attribute value and the learning task is to gain some understanding of relevant structural patterns in the data. Each row in a data set represents a point in n-dimensional space and unsupervised learning algorithms investigate the relationship between these various points in ndimensional space. Examples of unsupervised learning are clustering, density estimation and feature extraction. Text collections contain millions of unique terms, which make the text - mining process difficult. Therefore, feature-extraction is used 23 when applying machine learning methods. A feature is a combination of attributes (keywords), which captures important characteristics of the data. A feature extraction method creates a new set of features far smaller than the number of

original attributes by decomposing the original data. Therefore it enhances the speed of supervised learning. Zha et al (2001) has combined k-means with spectral analysis, Kotsiantis et al (2004) extended k-means algorithm to improve the kmeans algorithm. The Spatial Mining has been done by Ng and Han (1994). Jain et al (1999) has improved this concept by introducing Principal Component Analysis and this has been adopted for the analysis done in image processing technique. Unsupervised algorithms like Principal Components Analysis (PCA), singular value decomposition, and Nonnegative Matrix Factorization (NMF) involve factoring the document-word matrix, based on different constraints for feature extraction (Ghosh et al, 2011). Nonnegative matrix factorization is a new unsupervised algorithm for efficient feature extraction of text documents. NMF is a feature extraction algorithm that decomposes text data by creating a user-defined number of features. NMF gives a reduced representation of the original text data. It decomposes a text data matrix.

2.2. Summary/Gaps Identidied in the survey

As specified, Text mining is a very underutilized field. A Program that parses through online articles to produce articles for interest of the user is absent. Mixing old and tested Algorithms like NLP with new methods (unsupervised machine learning) hasn’t been done in a large scale.

As there is no target attribute to save the documents, the documents have to be processed every time the program is initialized. That can be fixed making it into a pure software than a hackware.

The documents are classified under user defined labels, hence as the document size increases the hyperspace increase too, resulting in a very large computational cost even 10 labels and 10000 documents uses 8 gb of active ram for 30 mins and when applied to real world, that will change exponentially.

3. Overview of the Proposed system

3.1. Introduction and related concepts

This project is focused on Topic detection for online articles using NLP and machine learning Although the topic detection algorithm isn't final yet, unsupervised machine learning will surely be used. The program will parse a number of online articles and divide them according to trained topics and sub topics (E.G. Animals can have 2 subtopics wild/pet and then pet in turn can have many more subtopics.) It will help categorize the data and find more relevant articles for the user with a short summary of the gist of the article.

Natural language processing (NLP) is a new field which focuses on understanding human languages. Developing NLP applications is very problematic because they are usually require human interaction in the programming language that is precise, unambiguous and highly structured, or just with a highly grammatically correct document. But on the contrary, it is not always the case. Human language is sparingly grammatically correct. There are a lost of slangs and abbreviations used which are hard to code into the machine on everyday basis. The new emoticons have made this even harder.

There for, we have decided to focus on keywords that I use and not the actual meaning of the text. Thus getting the topic on which the pro

NLP can be used to interpret free text and make it analyzable. There is a tremendous amount of information stored in free text files, like patients' medical records, for example. Prior to deep learning-based NLP models, this information was inaccessible to computer-assisted analysis and could not be analyzed in any kind of systematic way. But NLP allows analysts to sift through massive troves of free text to find relevant information in the files.

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coikned the term "Machine Learning" in 1959 while at IBM Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data– such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions, through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or infeasible; example applications include email filtering, detection of network intruders or malicious insiders working towards a data breach, optical character recognition (OCR), learning to rank, and computer vision.

Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning. Machine learning can also be unsupervised and be used to learn and establish baseline behavioral profiles for various entities and then used to find meaningful anomalies.

Within the field of data analytics, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytics. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.

According to the Gartner hype cycle of 2016, machine learning is at its peak of inflated expectations. Effective machine learning is difficult because finding patterns is hard and often not enough training data are available; as a result, machine-learning programs often fail to deliver.

3.2. Framework / Architecture

The first part of the project was to decide on the classifier. For this, 4 different classifiers were tested namely: Bernoulli Naïve Bayes, Gaussian Naïve Bayes, Multinomial Naïve Bayes and K-nearest neighbor. All the classifiers were trained and tested with the same data, which is 20newsgroups. The classifier was trained for 10 labels because of the lack of resources(a total of approximately 11000 documents, 8000 of which are used for training and the remaining are used for testing). After determining the best classifier out of the four, (Multinomial Naïve Bayes), The main classifier was trained with all the data (Both training and testing) for the main program.

The main program starts with getting the training data (i.e. the path of the training data) which is pre-coded in the program the path is then used to get the list of all the labels, and then the documents belonging to the labels. Class titles are stored in a 1-D array and Files are stored in a 2-D array(with Corresponding class title).

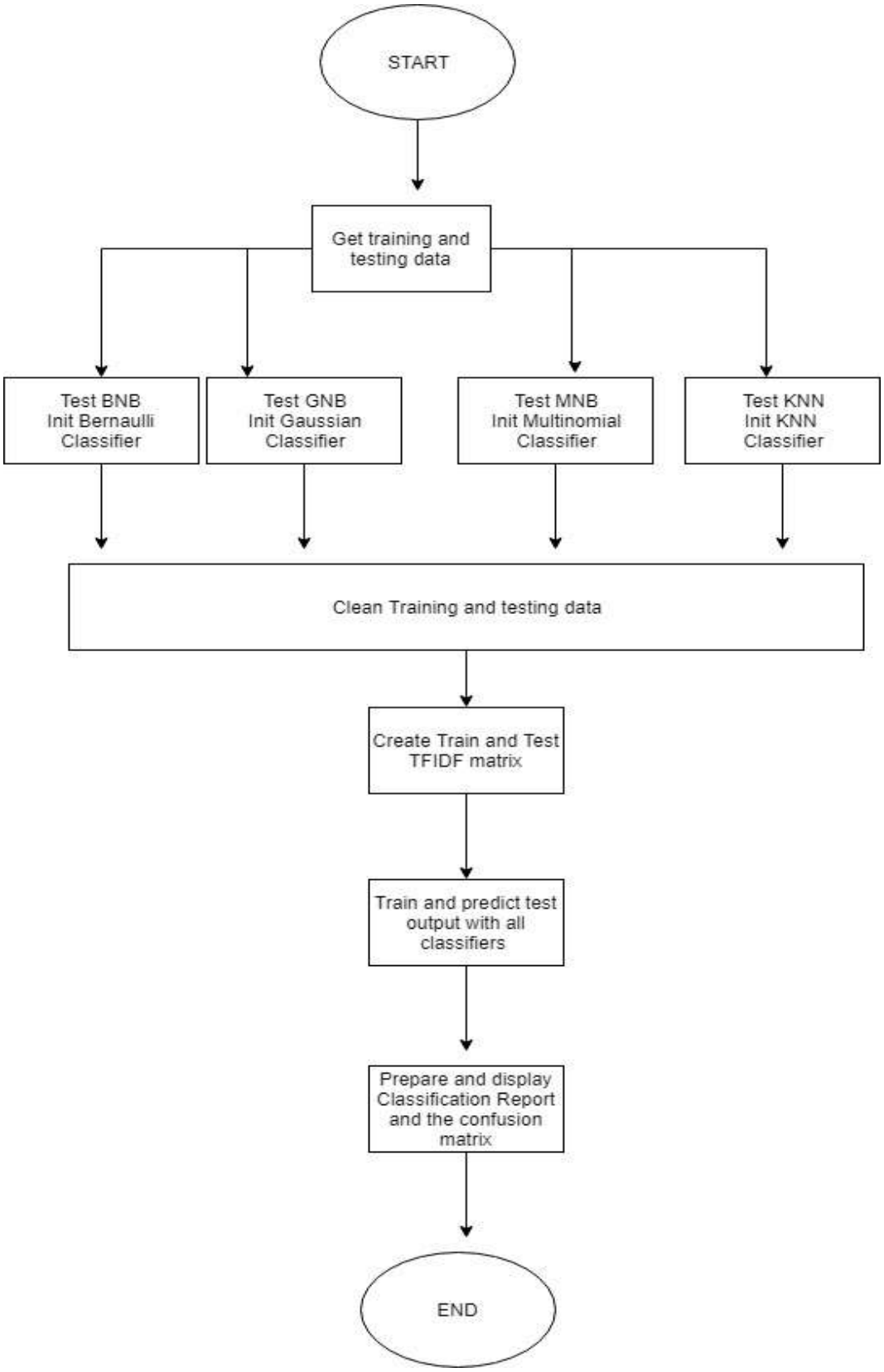
Then the corresponding files are cleaned i.e. all the dates, punctuations, digits, special characters are removed from the file and are stored in a separate array.

This array is passed to the topic detect function which initiates a vectorizer and it is fitted with the cleaned training array. The training matrix is made from the said array.
The training matrix is used to a TF-IDF which is then used to train the Multinomial Naïve Bayes Classifier. After which the input is predicted.

After which the text is sent for summarization. The data is cleaned (lowering and removing just like what was done previously) and the frequencies of words are calculated. Then 50 of the most frequent words in the article is taken as keywords each sentence is given a score based on the score the appropriate number of sentences are selected. The selected sentences are arranged in the order of occurrence and are then displayed.

If the online module is being used, an input for the url is taken which is resolved by the newspaper library and text is saved in an array.

3.3 UML Diagram



4. Proposed system analysis and Design

4.1. Introduction

Based on the Results of testing, Multinomial Naïve Bayes Classifier was selected to be the best amongst the one tested.

The Program asks for a URL, and outputs one of the 10 labels in which it has classified the said document.

The Cleaning and predicting algorithm are similar to what was tested. After summarizing, the program displays the predicted topic and the summary with number of sentences that were given as an input.

4.2. Requirement Analysis

4.2.1. Functional Requirements

4.2.1.1. Product Perspective

The Program aims to classify many online articles into different categories and summarize the content.

4.2.1.2. Product Features

The program will use Machine learning with Text mining algorithms. It will also learn with each new set of articles it parses.

4.2.1.3. User Char.

The End User can be a data scientist looking for relevant topics, Someone who wants to search for a specific topic with very specific keywords.

4.2.1.4. Assumption and Dependencies

It is assumed that the end user has basic knowledge of understanding Python.
It is also assumed that a article is based on the topic which is repeated the most in the article.

4.2.2. Non Functional Requirments

4.2.2.1. Product Requirments

4.2.2.1.1. Efficiency

The program will require a sizable amount of space and will have exponential time complexity.

4.2.2.1.2. Reliability

The program will be 95% Reliability .

4.2.2.1.3. Portability

The program itself will be small but the training data and testing data will occupy space depending on the user and the Reliability the user wants.

4.2.2.2. User Requirements

The program should be user friendly. The program should not take much space of the user's system. The image inpainted should be accurate and understandable by the user.

4.2.3. Non Functional Requirements

4.2.3.1. Product Requirements

4.2.3.1.1. Efficiency

The program will require a sizable amount of space and will have exponential time complexity.

4.2.3.1.2. Reliability

The program should be well tested before releasing to the user**.** The program should run for all the test cases. The program should provide accurate output most of the time.

4.2.3.1.3. Portability

The program itself will be small but the training data and testing data will occupy space depending on the user and the Reliability the user wants.

4.2.3.1.4. Usability

The software should have an easy installation wizard so that it can be easily installed or uninstalled from the user's system.

4.2.3.2. Organizational Requirements

4.2.3.2.1. Implementation Requirements (in terms of deployment)

The program should be of the form of a software which can be installed in the computer of the end user.The software should be of appropriate size for ease of installation.

The software should have a user friendly installation wizard for ease of installation.

4.2.3.2.2. Engineering Standard Requirements

The software should be compatible with all types of Operating Systems like Windows, Linux, and Mac. The software should also be supported on the mobile phone of the end users.
The software should not record any data in any kind of database from the end user's system.

4.2.3.3. Operational Requirements

- **Economic:**

The program should be designed for low costs so that it is available for the common public as well. It should not be restricted to the wealthy people.

- **Environmental:**

The software should be eco-friendly and should not cause any environmental hazards.

- **Health and Safety:**

The program should be safe to use and should not cause any health hazard or endanger the safety of the user.

- **Political:**

The program should comply with all the rules and regulatory requirements set by the Government.

- **Ethical:**

The program should not demand any data from the user. The image to be inpainted should be removed from the database after a particular time.

- **Sustainability:**

The program should be reliable and serviceable for long term use.

- **Legality:**

The program should be accurate and should not result in any law suites.

4.2.4. System Requirements

4.2.4.1. H/W Requirements

CPU: Intel CPU Core i7 Quad Core (preferable)
RAM: 30 GB or Higher
OS: 64-bit Windows 8 (8.1)
VIDEO CARD: Nvidia GPU GeForce GTX 770 or better
FREE DISK SPACE: 5 GB

4.2.4.2. S/W Requirments

Python 3.2 or higher
Linux/ Windows OS
Sublime IDE
Python libraries - matplotlib.pyplot, sklearn.feature_extraction.text, sklearn.naive_bayes, sklearn.metrics, sklearn.neighbors, nltk.corpus, nltk.tokenize, datetime, nltk.probability, nltk.data, newspaper.

5. Results and Discussion

The dataset that was used was from 20NewsGroups, a labeled dataset about 20 topics of which 10 were chosen for this project. A total of about 5500 documents were used to train. And 3500 documents were used for testing. The results of the classifiers are shown down below.

Table 1: Classification Report for Bernoulli Naïve Bayes Classifier

	precision	recall	f1-score	support
rec.autos	0.81	0.87	0.84	396
rec.motorcycles	0.78	0.96	0.86	398
rec.sport.baseball	0.79	0.96	0.87	397
rec.sport.hockey	0.99	0.88	0.93	399
sci.crypt	0.93	0.73	0.82	396
sci.electronics	0.48	0.94	0.64	393
sci.med	0.96	0.56	0.70	396
sci.space	0.98	0.76	0.86	394
talk.politics.guns	0.97	0.70	0.81	364
talk.politics.mideast	0.99	0.72	0.83	376
avg / total	0.87	0.81	0.82	3909

```
f1 score
0.8161167540481007
accuracy score:

0.8096699923254029
```

Fig 1: Confusion Matrix for Bernoulli Naïve Bayes Classifier

```
confusion matrix:
[[345   9   3   0   0  38   0   1   0   0]
 [  9 381   0   0   0   7   0   0   0   1]
 [  0   2 382   1   0  10   1   0   1   0]
 [  1   5  29 350   1  12   0   0   1   0]
 [  4  10   4   0 291  85   1   1   0   0]
 [  3   2   0   0  13 371   2   2   0   0]
 [ 17  21  12   0   0 121 221   1   2   1]
 [  5  12   5   0   2  66   3 300   1   0]
 [ 34  28  11   0   4  28   2   2 255   0]
 [  7  20  37   1   1  36   1   0   4 269]]
```

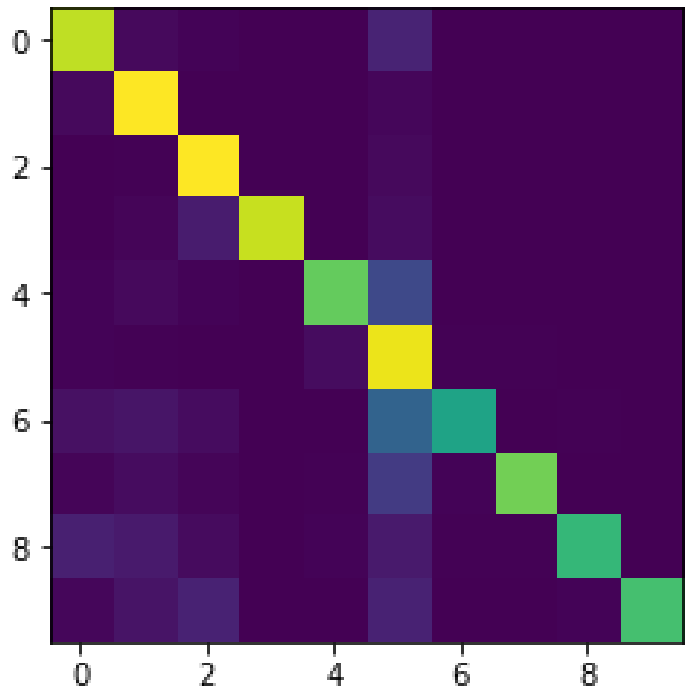


Table 2: Classification Report for Gaussian Naïve Bayes Classifier

	precision	recall	f1-score	support
rec.autos	0.86	0.79	0.82	396
rec.motorcycles	0.86	0.88	0.87	398
rec.sport.baseball	0.93	0.82	0.87	397
rec.sport.hockey	0.89	0.93	0.91	399
sci.crypt	0.80	0.87	0.83	396
sci.electronics	0.86	0.69	0.77	393
sci.med	0.74	0.83	0.78	396
sci.space	0.78	0.82	0.80	394
talk.politics.guns	0.81	0.85	0.83	364
talk.politics.mideast	0.89	0.93	0.91	376
avg / total	0.84	0.84	0.84	3909

f1 score
0.839797282483341
accuracy score:

Fig 2 : Confusion Matrix for Gaussian Naïve Bayes Classifier

```
confusion matrix:
[[311  20   5   3   4  13  17  17   5   1]
 [ 22 350   2   2   0   4   5   2   7   4]
 [  2   5 326  23   7   4  13  12   2   3]
 [  1   4   6 370   5   0   3   2   3   5]
 [  3   1   1   1 343   3  13  12  18   1]
 [ 10  12   2   4  23 272  25  31   7   7]
 [  7   7   2   5  12   8 327   8  11   9]
 [  2   4   2   3  13  11  20 324  10   5]
 [  2   4   2   2  16   2  15   4 310   7]
 [  3   1   2   2   6   0   2   1   8 351]]
```

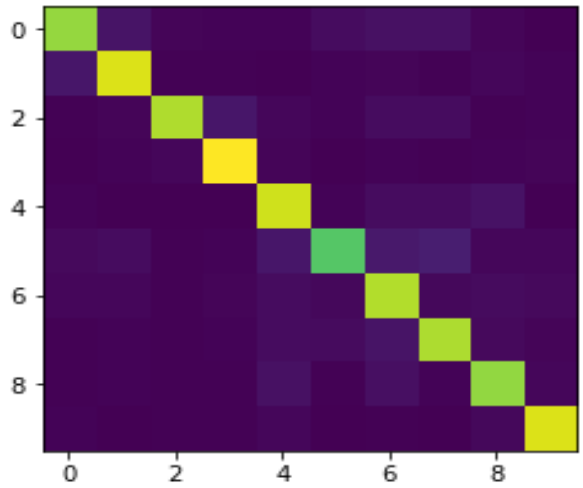


Table 3: Classification Report for Multinomial Naïve Bayes Classifier

	precision	recall	f1-score	support
rec.autos	0.92	0.94	0.93	396
rec.motorcycles	0.94	0.97	0.95	398
rec.sport.baseball	0.96	0.93	0.94	397
rec.sport.hockey	0.91	0.97	0.94	399
sci.crypt	0.81	0.97	0.89	396
sci.electronics	0.91	0.75	0.82	393
sci.med	0.98	0.85	0.91	396
sci.space	0.94	0.96	0.95	394
talk.politics.guns	0.93	0.95	0.94	364
talk.politics.mideast	0.98	0.96	0.97	376
avg / total	0.93	0.92	0.92	3909

```
f1 score
0.9240920333535054
accuracy score:

0.9245331286774111
```

Fig 3: Confusion Matrix for Multinomial Naïve Bayes Classifier

```
[[373   3   3   3   2   6   0   2   4   0]
 [  9 385   0   0   2   1   0   0   1   0]
 [  2   2 368  21   0   0   0   1   1   2]
 [  1   0   5 388   3   0   0   2   0   0]
 [  4   0   0   1 386   3   0   1   1   0]
 [  8   8   0   6  56 295   4  12   2   2]
 [  2   6   6   5  10  15 335   6   8   3]
 [  3   1   0   2   3   2   2 378   2   1]
 [  2   2   1   1   9   0   1   1 346   1]
 [  0   2   1   0   4   3   0   0   6 360]]
```

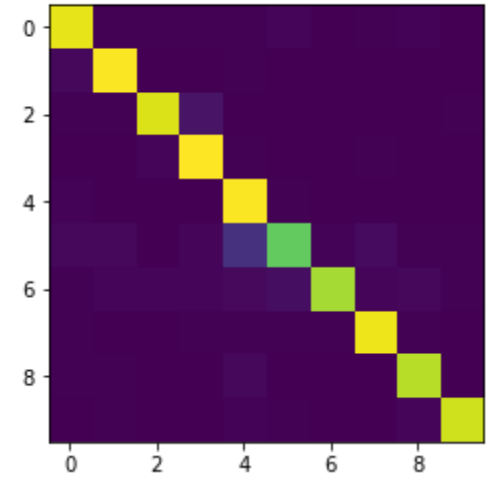
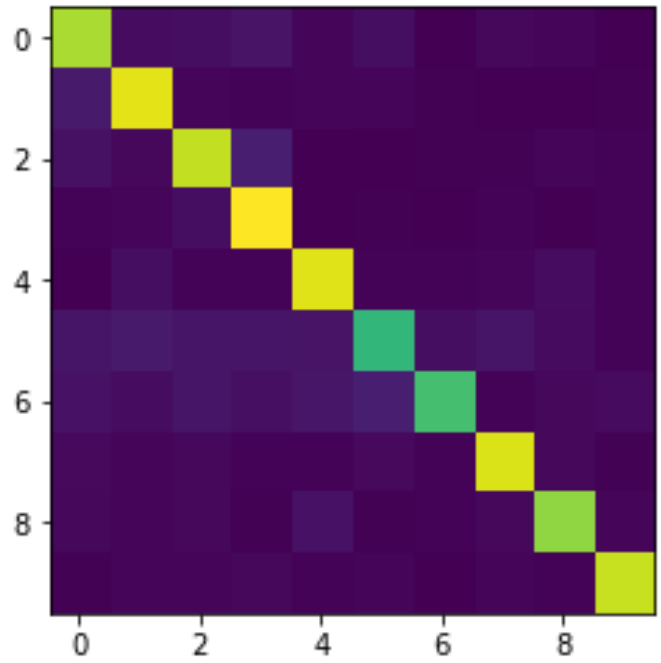


Table 4 : Classification Report for K Nearest Neighbor Classifier

	precision	recall	f1-score	support
rec.autos	0.69	0.81	0.74	396
rec.motorcycles	0.77	0.87	0.82	398
rec.sport.baseball	0.76	0.81	0.79	397
rec.sport.hockey	0.80	0.89	0.84	399
sci.crypt	0.82	0.88	0.85	396
sci.electronics	0.76	0.62	0.68	393
sci.med	0.85	0.64	0.73	396
sci.space	0.87	0.85	0.86	394
talk.politics.guns	0.84	0.79	0.82	364
talk.politics.mideast	0.90	0.84	0.87	376
avg / total	0.81	0.80	0.80	3909

Fig 4: Confusion Matrix for K Nearest Neighbor Classifier

```
[[317  12  13  19   6  13   1   8   6   1]
 [ 26 348   5   3   6   5   2   1   0   2]
 [ 16   8 329  30   1   1   2   2   5   3]
 [  4   7  14 363   1   2   0   3   1   4]
 [  1  14   4   4 345   4   3   5  12   4]
 [ 20  26  20  20  19 241  13  20  10   4]
 [ 18  12  20  15  22  31 256   4   8  10]
 [  9   5   8   3   4   9   4 342   8   2]
 [  8   5   8   2  18   2   3   8 303   7]
 [  2   7   7   8   4   5   0   7   4 332]]
```



Summary of the Result:

The program now uses the multinomial classifier which has been trained with all the training and testing data from the dataset. with and accuracy score of 92%, it is by far the best classifier that has been tested. There is no way to quantify the summary component but the summary seems to be satisfactory for small documents and vague for large documents. The Summarizer is ideal for summarizing online articles as it seems to be the correct size for this kind of summarization.

6. Conclusion, Limitations and Scope of Future work

Conclusion

In conclusion, the classifier accuracy was 93% for testing data, but will be much better for real world applications with articles that just feed the body of the text. The Classifier can be expanded by adding more topics, though that is only possible with better resources. With the document size increasing, the space that the program takes increases exponentially.

Limitations

As of now, adding more labels is a limitation as I don't have the adequate resources. The Summarizer displays vague Output for anything longer than two pages as extractive summarization loses context.

Scope For Future Work

1. Adding a Gui Interface for easier tasking
2. Adding a training Wheels module(adds the document to corpora if the predict is flagged correct)
3. Work on the summary algorithm to make it better for longer articles.

Annexure – 1

The Output of the Program:

```
Creating arrays...
Testing BNB:

(5872, 54279)
(3909, 54279)
training time:  0:00:00.158922
 ['rec.autos' 'rec.autos' 'sci.electronics' ... 'talk.politics.mideast'
 'sci.electronics' 'talk.politics.mideast']
testing time:  0:00:00.046873
classification report:


f1 score
0.8161167540481007
accuracy score:

0.8096699923254029
confusion matrix:
[[345   9   3   0   0  38   0   1   0   0]
 [  9 381   0   0   0   7   0   0   0   1]
 [  0   2 382   1   0  10   1   0   1   0]
 [  1   5  29 350   1  12   0   0   1   0]
 [  4  10   4   0 291  85   1   1   0   0]
 [  3   2   0   0  13 371   2   2   0   0]
 [ 17  21  12   0   0 121 221   1   2   1]
 [  5  12   5   0   2  66   3 300   1   0]
 [ 34  28  11   0   4  28   2   2 255   0]
 [  7  20  37   1   1  36   1   0   4 269]]

OBJ
Testing GNB:

(5872, 54279)
(3909, 54279)
training time:  0:00:12.405674
 ['rec.autos' 'rec.autos' 'rec.autos' ... 'talk.politics.mideast'
 'talk.politics.mideast' 'talk.politics.mideast']
testing time:  0:02:21.464559
classification report:
```

f1 score
0.839797282483341
accuracy score:

0.8401125607572227
confusion matrix:
[[311 20 5 3 4 13 17 17 5 1]
[22 350 2 2 0 4 5 2 7 4]
[2 5 326 23 7 4 13 12 2 3]
[1 4 6 370 5 0 3 2 3 5]
[3 1 1 1 343 3 13 12 18 1]
[10 12 2 4 23 272 25 31 7 7]
[7 7 2 5 12 8 327 8 11 9]
[2 4 2 3 13 11 20 324 10 5]
[2 4 2 2 16 2 15 4 310 7]
[3 1 2 2 6 0 2 1 8 351]]

OBJ
Testing MNB:

(5872, 54279)
(3909, 54279)
training time: 0:00:02.695695
['rec.autos' 'rec.autos' 'rec.autos' ... 'talk.politics.mideast'
'talk.politics.mideast' 'talk.politics.mideast']
testing time: 0:00:00.765636

f1 score
0.9240920333535054
accuracy score:

0.9245331286774111
confusion matrix:
[[373 3 3 3 2 6 0 2 4 0]
[9 385 0 0 2 1 0 0 1 0]
[2 2 368 21 0 0 0 1 1 2]
[1 0 5 388 3 0 0 2 0 0]
[4 0 0 1 386 3 0 1 1 0]
[8 8 0 6 56 295 4 12 2 2]
[2 6 6 5 10 15 335 6 8 3]
[3 1 0 2 3 2 2 378 2 1]
[2 2 1 1 9 0 1 1 346 1]

```
 [  0   2   1   0   4   3   0   0   6 360]]
```

OBJ

Testing KNN:

```
(5872, 54279)
(3909, 54279)
knn with  5  neighbors
training time:  0:00:00.125022
 ['rec.autos' 'rec.autos' 'rec.autos' ... 'talk.politics.mideast'
 'talk.politics.mideast' 'talk.politics.mideast']
testing time:  0:00:01.534359
classification report:

f1 score
0.8001518790173276
accuracy score:

0.8014837554361729
confusion matrix:
[[319  15   6  13   5  16   5   5   9   3]
 [ 25 348   6   3   4   1   3   4   2   2]
 [ 19   6 323  27   0   6   1   2   8   5]
 [  6   8  13 357   1   3   5   1   1   4]
 [  6   9   4   4 350   6   4   2   9   2]
 [ 30  29  21  14  12 243  15  19   7   3]
 [ 28  16  25  16  17  23 252   4   7   8]
 [ 12   5   8   3   6   9   7 336   5   3]
 [ 12   8   9   5  21   4   4   8 288   5]
 [  7   8  10   7   9   8   0   5   5 317]]
```

OBJ

Topic Detect Example 2:

```
(5872, 54279)
  (0, 53681)    0.08262548752133307
  (0, 53109)    0.030513861224720232
  (0, 53057)    0.07637299129571513
  (0, 52044)    0.08601276918168403
  (0, 51719)    0.1135008519533042
  (0, 50618)    0.05518376153777583
  (0, 50610)    0.08026159079574592
  (0, 50591)    0.04386848935443285
```

```
  (0, 50576)    0.045396108637483445
  (0, 49097)    0.13010287303289997
  (0, 48318)    0.11348452449201776
  (0, 48137)    0.07844422436619702
  (0, 47515)    0.09740014871338827
  (0, 47117)    0.08622354898062133
  (0, 45461)    0.20351684973158485
  (0, 44863)    0.11238838157089427
  (0, 44611)    0.1337009344921882
  (0, 44239)    0.08665355913080483
  (0, 43393)    0.08754959102848207
  (0, 42978)    0.08622354898062133
  (0, 42621)    0.08976865661560585
  (0, 42372)    0.10621285172056805
  (0, 41934)    0.12716304276294949
  (0, 41549)    0.09949016357896083
  (0, 41102)    0.10017113529388742
  :     :
  (0, 10339)    0.10351920066005928
  (0, 10002)    0.040230587995543986
  (0, 9439)     0.08085886000317384
  (0, 9432)     0.06741545715007198
  (0, 9191)     0.10696296022406546
  (0, 8939)     0.06572318914194387
  (0, 8936)     0.07580716394097028
  (0, 8870)     0.12062515103371071
  (0, 8847)     0.07648857906673699
  (0, 8803)     0.07038174259414147
  (0, 7320)     0.08899453556112048
  (0, 6620)     0.1337009344921882
  (0, 6617)     0.09234260092729234
  (0, 6596)     0.15937131450276515
  (0, 6283)     0.06945568382557386
  (0, 5827)     0.08116600119452541
  (0, 3035)     0.10696296022406546
  (0, 2857)     0.11037019785850877
  (0, 2593)     0.051917142628638384
  (0, 1606)     0.12716304276294949
  (0, 938)      0.08709535183540991
  (0, 822)      0.09431949287573367
  (0, 556)      0.08245664383188173
  (0, 483)      0.13833964249571643
  (0, 342)      0.10232425222053727
(1, 54279)
```

Output Predict:
['sci.crypt']
Summary of Example 2:
could affect its business model.
(5872, 54279)
  (0, 54039) 0.37112795868615034
  (0, 52350) 0.0225293435294005
  (0, 51997) 0.0357818189854156
  (0, 50594) 0.05202981475788786
  (0, 49403) 0.023845736182999164
  (0, 48852) 0.051756737548640235
  (0, 48532) 0.24759764799560258
  (0, 48500) 0.038130590791782576
  (0, 48430) 0.03694982632995476
  (0, 48150) 0.031853549442377074
  (0, 48137) 0.043204211103569444
  (0, 47852) 0.03833664977699406
  (0, 47072) 0.04628832613349437
  (0, 46045) 0.055743749447883816
  (0, 45658) 0.05635645746605142
  (0, 45565) 0.07363758702717416
  (0, 45304) 0.02730097871732794
  (0, 45283) 0.03584458068716822
  (0, 44753) 0.045792066097069595
  (0, 44720) 0.07675688317963243
  (0, 44644) 0.04033920181778148
  (0, 43672) 0.041690778343729036
  (0, 43579) 0.02608808637671302
  (0, 43218) 0.048219114197216265
  (0, 42788) 0.03526552782566287
  : :
  (0, 7546) 0.04505157538747022
  (0, 7504) 0.0505237666468254
  (0, 7320) 0.14704481046932064
  (0, 6958) 0.029051052186610367
  (0, 6544) 0.03220185854894632
  (0, 5547) 0.03731384219874889
  (0, 4941) 0.0760184239867878
  (0, 4724) 0.10839789018712048
  (0, 4675) 0.0602802609572806
  (0, 4529) 0.1808407828718418
  (0, 4296) 0.059800103815269116
  (0, 4282) 0.046706320640409964
  (0, 4023) 0.050043609504813916

(0, 3937) 0.07053105565132574
(0, 3905) 0.053644441264252295
(0, 3348) 0.039492776526001146
(0, 3322) 0.03310277996163485
(0, 3259) 0.04478907123667182
(0, 3110) 0.044124780824681936
(0, 1581) 0.033010750704547856
(0, 1444) 0.021800673549222743
(0, 1380) 0.04944129440045473
(0, 902) 0.0551705993741237
(0, 261) 0.06748192447615733
(0, 106) 0.03197492824076575
(1, 54279)
Output Predict:
['rec.motorcycles']
Summary of Example 3:

A German rider has set a new electric record for the longest journey in 24 hours. On March 10, Berlin-based Remo Klawitter rode a Zero DSR electric motorcycle between Berlin and the Land Centre for Renewable Energies in Neustrelitz. The goal was to travel more than 1,000 kilometres on an electric motorcycle to prove that electric motorcycles do have range and can cover long distances. Klawitter had been riding on the Federal Highway 96 on the Zero electric motorcycle on behalf of the competence centre for e-mobility in the state centre. After each stint, he fully recharged the bike, a 2018 Zero DSR 14.4 with Zero's optional Charge Tank, which increased charging speed by up to six times.
Actual Example 3:

A German rider has set a new electric record for the longest journey in 24 hours. On March 10, Berlin-based Remo Klawitter rode a Zero DSR electric motorcycle between Berlin and the Land Centre for Renewable Energies in Neustrelitz. The goal was to travel more than 1,000 kilometres on an electric motorcycle to prove that electric motorcycles do have range and can cover long distances. Klawitter had been riding on the Federal Highway 96 on the Zero electric motorcycle on behalf of the competence centre for e-mobility in the state centre.

He began his mission on the afternoon of March 10, and rode in 150 km stints, at an average speed of between 90 kmph and 100 kmph. After each stint, he fully recharged the bike, a 2018 Zero DSR 14.4 with Zero's optional Charge Tank, which increased charging speed by up to six times. Klawitter used the charging time stops to rest for about an hour at Level Two public charging points. Klawitter is a Berlin-based bicycle shop owner and e-bike enthusiast and covered a distance 1,134.3 kilometres in 24 hours.

"I've been thinking about a 24-hour ride since the middle of 2017. Primarily, I wanted to have fun, while drawing people's attention to the practicality of electric motorcycles. Everything

we have used is readily available from Zero and the route was on public roads. In total, we have spent nine and a half hours charging, and during this time, as well as resting and eating, I was able to speak to many people who were interested in the bike and the journey," said Klawitter.

Zero has recently introduced a new touring model, called the Zero DSR Black Forest, which has longer range, and touring accessories for long distance touring on electric motorcycles. The Black Forest comes with a top box, hard case panniers, touring screen, comfort seat and crash bars, and also gets the fast charging battery, which allows quick recharge in less than an hour. Zero is an American manufacturer of electric motorcycles which was started in 2006.

For the latest auto news and reviews, follow CarAndBike on Twitter, Facebook, and subscribe to our YouTube channel.

Annexure 2:

The Code:

```
import matplotlib.pyplot as plt
import os, re
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.naive_bayes import BernoulliNB, GaussianNB, MultinomialNB
from sklearn.metrics import confusion_matrix, f1_score, accuracy_score, precision_score, recall_score, classification_report
from sklearn.neighbors import KNeighborsClassifier
from nltk.corpus import stopwords
from string import punctuation,digits
from nltk.tokenize import word_tokenize
from datetime import datetime as dt
from nltk.probability import FreqDist
import nltk.data
from newspaper import Article

##
#
#                   Main Variable Declarations
##

train_arr = []
train_arr_use = []
test_arr = []
train_lbl = []
train_lbl_use = []
test_lbl = []
i = 0
files_train = {}
files_test = {}
files_train_use = {}
root_train_use = "C:\\Users\\vikash\\Desktop\\Python data science\\Train-For-Use\\20news-bydate-train\\"
root_train = "\\Users\\vikash\\Desktop\\Python data science\\20news-bydate\\20news-bydate-train\\"
class_titles_train_use = os.listdir(root_train_use)
class_titles_train = os.listdir(root_train)
root_test = "\\Users\\vikash\\Desktop\\Python data science\\20news-bydate\\20news-bydate-test\\"
class_titles_test = os.listdir(root_test)
```

```
def Topic_detect_Online(text):
    vectorizer = CountVectorizer()
    vectorizer.fit(train_arr_use)
    train_mat = vectorizer.transform(train_arr_use)
 #  print(train_mat.shape)
    new_arr = []
    basic_arr = []
    basic_arr.append(text)
    new_arr.append(clean_text_Online(text))
    test_mat = vectorizer.transform(new_arr)
 #   print(test_mat.shape)
    tfidf = TfidfTransformer()
    tfidf.fit(train_mat)
    train_tfmat = tfidf.transform(train_mat)
    print(train_tfmat.shape)
    test_tfmat = tfidf.transform(test_mat)
    print(test_tfmat)
    print(test_tfmat.shape)
    mnb = MultinomialNB()
#    Main_Classifier(train_tfmat.toarray(), train_lbl, test_tfmat.toarray(),mnb)
    mnb.fit(train_tfmat.toarray(),train_lbl_use)
    yhat = mnb.predict(test_tfmat.toarray())
    print("Output Predict:")
    print(yhat)

metrics_dict = []
example3 = 'C:\\Users\\vikash\\Desktop\\Python data science\\test2.txt'
example2 = 'C:\\Users\\vikash\\Desktop\\Python data science\\test.txt'
example = '\\Users\\vikash\\Desktop\\Python data science\\20news-bydate\\20news-bydate-
test\\sci.electronics\\53984'
pattern = re.compile(r'([a-zA-Z]+|[0-9]+(\.[0-9]+)?)')
def main():
#     get_train_data()
#     get_test_data()
     get_train_data_use()
     print("Creating arrays...")
     creating_arrays_use()
#     creating_arrays()
#     print("Testing BNB:\n")
#     create_text_vect_tfidf_BNB()
#     print("Testing GNB:\n")
#     create_text_vect_tfidf_GNB()
#     print("Testing MNB:\n")
#     create_text_vect_tfidf_MNB()
```

```
#     print("Testing KNN:\n")
#     knn()
    print("Topic Detect Example 2:\n")
    Topic_detect_MnB(example2)
    f = open(example2)
    y1 = f.read()
    print("Summary of Example 2:\n")
    Summ(y1,3)
    print("Actual Example 2:\n")
    print(y1)
    f.close
    g = open(example3)
    Topic_detect_MnB(example3)
    y2 = g.read()
    print("Summary of Example 3:\n")
    Summ(y2,5)
    print("Actual Example 3:\n")
    print(y2)
    g.close
    while True :
        GetOnline()




  # tfidf_trans()
#     print(test_arr)
#     print(clean_text(example))
#     f = open(example)
#     print(f.read())
def GetOnline():
    url = input('Enter the Url of the article:')
    article = Article(url)
    article.download()
    article.parse()
    text = article.text
    Topic_detect_Online(text)
    y = int(input('Input the number of sentences for Summary :'))
    print('Summary:\n')
    Summ(text,y)
    print('Actual text:\n')
    print(text)
#     print(text)
```

```
def get_train_data():
   root_train = "\\Users\\vikash\\Desktop\\Python data science\\20news-bydate\\20news-bydate-train\\"
   ##print(root_train)
   folders = [root_train + folder + '\\' for folder in os.listdir(root_train)]
   ##print(folders[0])
   class_titles = os.listdir(root_train)
   ##print(class_titles)

   for folder, title in zip(folders, class_titles):
      files_train[title] = [folder + f for f in os.listdir(folder)]
   ##print (files)

def get_train_data_use():
   root_train_use = "C:\\Users\\vikash\\Desktop\\Python data science\\Train-For-Use\\20news-bydate-train\\"
   ##print(root_train)
   folders = [root_train_use + folder + '\\' for folder in os.listdir(root_train_use)]
   ##print(folders[0])
   class_titles_use = os.listdir(root_train_use)
   ##print(class_titles)

   for folder, title in zip(folders, class_titles_use):
      files_train_use[title] = [folder + f for f in os.listdir(folder)]
   ##print (files)



def get_test_data():
   root_test = "\\Users\\vikash\\Desktop\\Python data science\\20news-bydate\\20news-bydate-test\\"
   ##print(root_test)
   folders_test  = [root_test + folder + '\\' for folder in os.listdir(root_test)]
   ##print(folders_test)
   class_titles = os.listdir(root_test)
   for folder, title in zip(folders_test, class_titles):
      files_test[title] = [folder + f for f in os.listdir(folder)]
   ##print(files_test)


def clean_text(path):
   text_translated = "
```

```
    try:
        f = open(path)
        raw = f.read().lower()
        text = pattern.sub(r' \1 ', raw.replace('\n', ' '))
        text_translated = text.translate(str.maketrans(",",punctuation + digits))
        text_translated = ' '.join([word for word in text_translated.split(' ') if (word and
len(word) > 1)])
        stop_words = set(stopwords.words('english'))
        word_tokens = word_tokenize(text_translated)
        filtered_sentence = [w for w in word_tokens if not w in stop_words]
        filtered_sentence = []
        for w in word_tokens:
            if w not in stop_words:
                filtered_sentence.append(w)
        final_op = ' '.join(filtered_sentence)


    finally:
        f.close()
    return final_op
def Topic_detect_Online(text):
    vectorizer = CountVectorizer()
    vectorizer.fit(train_arr_use)
    train_mat = vectorizer.transform(train_arr_use)
 #  print(train_mat.shape)
    new_arr = []
    basic_arr = []
    basic_arr.append(text)
    new_arr.append(clean_text_Online(text))
    test_mat = vectorizer.transform(new_arr)
 #   print(test_mat.shape)
    tfidf = TfidfTransformer()
    tfidf.fit(train_mat)
    train_tfmat = tfidf.transform(train_mat)
    print(train_tfmat.shape)
    test_tfmat = tfidf.transform(test_mat)
    print(test_tfmat)
    print(test_tfmat.shape)
    mnb = MultinomialNB()
#    Main_Classifier(train_tfmat.toarray(), train_lbl, test_tfmat.toarray(),mnb)
    mnb.fit(train_tfmat.toarray(),train_lbl_use)
    yhat = mnb.predict(test_tfmat.toarray())
    print("Output Predict:")
    print(yhat)
```

```
def Topic_detect_Online_2(text):
    vectorizer = CountVectorizer()
    vectorizer.fit(train_arr_use)
    train_mat = vectorizer.transform(train_arr_use)
 #  print(train_mat.shape)
    new_arr = []
    basic_arr = []
    basic_arr.append(text)
    new_arr.append(clean_text_Online(text))
    test_mat = vectorizer.transform(new_arr)
 #   print(test_mat.shape)
    tfidf = TfidfTransformer()
    tfidf.fit(train_mat)
    train_tfmat = tfidf.transform(train_mat)
    print(train_tfmat.shape)
    test_tfmat = tfidf.transform(test_mat)
    print(test_tfmat)
    print(test_tfmat.shape)
    mnb = MultinomialNB()
#    Main_Classifier(train_tfmat.toarray(), train_lbl, test_tfmat.toarray(),mnb)
    mnb.fit(train_tfmat.toarray(),train_lbl_use)
    yhat = mnb.predict(test_tfmat.toarray())
    print("Output Predict:")
    print(yhat)


def clean_text_Online(text):
    text_translated = ''
    raw = text.lower()
    text = pattern.sub(r' \1 ', raw.replace('\n', ' '))
    text_translated = text.translate(str.maketrans("",punctuation + digits))
    text_translated = ' '.join([word for word in text_translated.split(' ') if (word and len(word) >
1)])
    stop_words = set(stopwords.words('english'))
    word_tokens = word_tokenize(text_translated)
    filtered_sentence = [w for w in word_tokens if not w in stop_words]
    filtered_sentence = []
    for w in word_tokens:
        if w not in stop_words:
            filtered_sentence.append(w)
    final_op = ' '.join(filtered_sentence)
    return final_op
```

```
def creating_arrays_use():

    for i in range(10):
        for path in files_train_use[class_titles_train_use[i]]:
            #  print(path)
            #print(class_titles_train[i])
            train_arr_use.append(clean_text(path))
            train_lbl_use.append(class_titles_train_use[i])



def creating_arrays():

    for i in range(10):
        for path in files_train[class_titles_train[i]]:
            #  print(path)
            #print(class_titles_train[i])
            train_arr.append(clean_text(path))
            train_lbl.append(class_titles_train[i])
    for i in range(10):
        for path in files_test[class_titles_test[i]]:
            #  print(path)
            #print(class_titles_train[i])
            test_arr.append(clean_text(path))
            test_lbl.append(class_titles_test[i])


def create_text_vect_tfidf_BNB():
    vectorizer = CountVectorizer()
    vectorizer.fit(train_arr)
    train_mat = vectorizer.transform(train_arr)
  #  print(train_mat.shape)
    test_mat = vectorizer.transform(test_arr)
 #   print(test_mat.shape)
    tfidf = TfidfTransformer()
    tfidf.fit(train_mat)
    train_tfmat = tfidf.transform(train_mat)
    print(train_tfmat.shape)
    test_tfmat = tfidf.transform(test_mat)
    print(test_tfmat.shape)
    bnb = BernoulliNB()
    bnb_me = Metrics_Classifier(train_tfmat, train_lbl, test_tfmat, test_lbl, bnb)
    metrics_dict.append({'name':'BernoulliNB', 'metrics':bnb_me})
```

```
def creating_arrays_use_2():

    for i in range(10):
        for path in files_train_use[class_titles_train_use[i]]:
            #  print(path)
            #print(class_titles_train[i])
            train_arr_use.append(clean_text(path))
            train_lbl_use.append(class_titles_train_use[i])


def Metrics_Classifier(x_train, y_train, x_test, y_test, clf):
    metrics = []
    start = dt.now()
    clf.fit(x_train, y_train)
    end = dt.now()
    print ('training time: ', (end - start))

    # add training time to metrics
    metrics.append(end-start)

    start = dt.now()
    yhat = clf.predict(x_test)
    print('teh predict i guess')
    print(yhat)
    end = dt.now()
    print ('testing time: ', (end - start))

    # add testing time to metrics
    metrics.append(end-start)

    print ('classification report: \n')
#     print classification_report(y_test, yhat)
    print(classification_report(y_test, yhat))

    print ('f1 score')
    print (f1_score(y_test, yhat, average='macro'))

    print ('accuracy score: \n')
    print (accuracy_score(y_test, yhat))

    precision = precision_score(y_test, yhat, average=None)
    recall = recall_score(y_test, yhat, average=None)
```

```
    # add precision and recall values to metrics
    for p, r in zip(precision, recall):
        metrics.append(p)
        metrics.append(r)


    #add macro-averaged F1 score to metrics
    metrics.append(f1_score(y_test, yhat, average='macro'))

    print ('confusion matrix:')
    print (confusion_matrix(y_test, yhat))

    # plotting the confusion matrix
    plt.imshow(confusion_matrix(y_test, yhat), interpolation='nearest')
    plt.show()

    return metrics

def create_text_vect_tfidf_GNB():
    vectorizer = CountVectorizer()
    vectorizer.fit(train_arr)
    train_mat = vectorizer.transform(train_arr)
  #  print(train_mat.shape)
    test_mat = vectorizer.transform(test_arr)
 #   print(test_mat.shape)
    tfidf = TfidfTransformer()
    tfidf.fit(train_mat)
    train_tfmat = tfidf.transform(train_mat)
    print(train_tfmat.shape)
    test_tfmat = tfidf.transform(test_mat)
    print(test_tfmat.shape)
    gnb = GaussianNB()
    gnb_me = Metrics_Classifier(train_tfmat.toarray(), train_lbl, test_tfmat.toarray(), test_lbl,
gnb)
    metrics_dict.append({'name':'GaussianNB', 'metrics':gnb_me})

def create_text_vect_tfidf_MNB():
    vectorizer = CountVectorizer()
    vectorizer.fit(train_arr)
    train_mat = vectorizer.transform(train_arr)
 #  print(train_mat.shape)
    test_mat = vectorizer.transform(test_arr)
 #   print(test_mat.shape)
    tfidf = TfidfTransformer()
```

```
    tfidf.fit(train_mat)
    train_tfmat = tfidf.transform(train_mat)
    print(train_tfmat.shape)
    test_tfmat = tfidf.transform(test_mat)
    print(test_tfmat.shape)
    mnb = MultinomialNB()
    mnb_me = Metrics_Classifier(train_tfmat.toarray(), train_lbl, test_tfmat.toarray(), test_lbl,
mnb)
    metrics_dict.append({'name':'MultinomialNB', 'metrics':mnb_me})
def knn():
    vectorizer = CountVectorizer()
    vectorizer.fit(train_arr)
    train_mat = vectorizer.transform(train_arr)
  #  print(train_mat.shape)
    test_mat = vectorizer.transform(test_arr)
 #   print(test_mat.shape)
    tfidf = TfidfTransformer()
    tfidf.fit(train_mat)
    train_tfmat = tfidf.transform(train_mat)
    print(train_tfmat.shape)
    test_tfmat = tfidf.transform(test_mat)
    print(test_tfmat.shape)
    for nn in [10]:
        print ('knn with ', nn, ' neighbors')
        knn = KNeighborsClassifier(n_neighbors=nn)
        knn_me = Metrics_Classifier(train_tfmat, train_lbl, test_tfmat, test_lbl, knn)
        metrics_dict.append({'name':'5NN', 'metrics':knn_me})
        print (' ')

def Topic_detect_MnB(path):
    vectorizer = CountVectorizer()
    vectorizer.fit(train_arr_use)
    train_mat = vectorizer.transform(train_arr_use)
 #  print(train_mat.shape)
    new_arr = []
    basic_arr = []
    f = open(path)
    basic_arr.append(f.read())

    new_arr.append(clean_text(path))
    test_mat = vectorizer.transform(new_arr)
 #   print(test_mat.shape)
    tfidf = TfidfTransformer()
    tfidf.fit(train_mat)
```

```
    train_tfmat = tfidf.transform(train_mat)
    print(train_tfmat.shape)
    test_tfmat = tfidf.transform(test_mat)
    print(test_tfmat)
    print(test_tfmat.shape)
    mnb = MultinomialNB()
#    Main_Classifier(train_tfmat.toarray(), train_lbl, test_tfmat.toarray(),mnb)
    mnb.fit(train_tfmat.toarray(),train_lbl_use)
    yhat = mnb.predict(test_tfmat.toarray())
    print("Output Predict:")
    print(yhat)

def Topic_detect_Online(text):
    vectorizer = CountVectorizer()
    vectorizer.fit(train_arr_use)
    train_mat = vectorizer.transform(train_arr_use)
 #  print(train_mat.shape)
    new_arr = []
    basic_arr = []
    basic_arr.append(text)
    new_arr.append(clean_text_Online(text))
    test_mat = vectorizer.transform(new_arr)
 #   print(test_mat.shape)
    tfidf = TfidfTransformer()
    tfidf.fit(train_mat)
    train_tfmat = tfidf.transform(train_mat)
    print(train_tfmat.shape)
    test_tfmat = tfidf.transform(test_mat)
    print(test_tfmat)
    print(test_tfmat.shape)
    mnb = MultinomialNB()
#    Main_Classifier(train_tfmat.toarray(), train_lbl, test_tfmat.toarray(),mnb)
    mnb.fit(train_tfmat.toarray(),train_lbl_use)
    yhat = mnb.predict(test_tfmat.toarray())
    print("Output Predict:")
    print(yhat)

def Summ(arr,num_sent):
    raw = arr.lower()
    text = pattern.sub(r' \1 ', raw.replace('\n', ' '))
    text_translated = text.translate(str.maketrans("",punctuation + digits))
    text_translated = ' '.join([word for word in text_translated.split(' ') if (word and len(word) >
1)])
    stop_words = set(stopwords.words('english'))
```

```
    word_tokens = word_tokenize(text_translated)
    filtered_sentence = [w for w in word_tokens if not w in stop_words]
    filtered_sentence = []
    for w in word_tokens:
        if w not in stop_words:
            filtered_sentence.append(w)
    #print(filtered_sentence)
    word_frequencies = FreqDist(filtered_sentence)
    most_freq_words =[]
   # print(word_frequencies.items())
    s = [(k, word_frequencies[k]) for k in sorted(word_frequencies, key=word_frequencies.get,
reverse=True)]
    for k, v in s:
        most_freq_words.append(k)
    #print(most_freq_words[:50])
    use =  most_freq_words[:50]
    sent_detector = nltk.data.load('tokenizers/punkt/english.pickle')
    actual_sentences = sent_detector.tokenize(arr)
    working_sentences = [sentence.lower()
        for sentence in actual_sentences]
   # print(actual_sentences)
   # print("trololol")
   # print(working_sentences)
    output_sentences = []
    for word in use:
        for i in range(0, len(working_sentences)):
            if (word in working_sentences[i]
                and actual_sentences[i] not in output_sentences):
                    output_sentences.append(actual_sentences[i])
                    break
            if len(output_sentences) >= num_sent: break
        if len(output_sentences) >= num_sent: break

    #print(output_sentences)
    final_op =[]
    for sentence in actual_sentences:
        for sent2 in output_sentences:
            if(sentence==sent2):
                final_op.append(sent2)
    print( " ".join(final_op))

  #  for k, v in od.items(): print(k, v)
def Topic_detect_Online(text):
    vectorizer = CountVectorizer()
```

```
    vectorizer.fit(train_arr_use)
    train_mat = vectorizer.transform(train_arr_use)
 #  print(train_mat.shape)
    new_arr = []
    basic_arr = []
    basic_arr.append(text)
    new_arr.append(clean_text_Online(text))
    test_mat = vectorizer.transform(new_arr)
 #   print(test_mat.shape)
    tfidf = TfidfTransformer()
    tfidf.fit(train_mat)
    train_tfmat = tfidf.transform(train_mat)
    print(train_tfmat.shape)
    test_tfmat = tfidf.transform(test_mat)
    print(test_tfmat)
    print(test_tfmat.shape)
    mnb = MultinomialNB()
#    Main_Classifier(train_tfmat.toarray(), train_lbl, test_tfmat.toarray(),mnb)
    mnb.fit(train_tfmat.toarray(),train_lbl_use)
    yhat = mnb.predict(test_tfmat.toarray())
    print("Output Predict:")
    print(yhat)


if __name__ == '__main__':
    main()
```

References

Weblinks:

1. https://www.jisc.ac.uk/reports/value-and-benefits-of-text-mining
2. https://machinelearningmastery.com/natural-language-processing/
3. http://users.encs.concordia.ca/~nlp-se/Ayrin/Journals.htm
4. https://en.wikipedia.org/wiki/Topic_model
5. https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups
6. https://machinelearningmastery.com/prepare-news-articles-text-summarization/
7. https://docs.python.org/3/library/re.html
8. https://arxiv.org/pdf/1707.02268.pdf
9. http://opensourceforu.com/2016/07/22843/
10. https://stanford.edu/~rjweiss/public_html/IRiSS2013/text2/notebooks/cleaningtext.html
11. http://ieva.rocks/2016/08/07/cleaning-text-for-nlp/

Journal:

1. Kim, J. D., Ohta, T., Tateisi, Y., & Tsujii, J. I. (2003). GENIA corpus—a semantically annotated corpus for bio-textmining. Bioinformatics, 19(suppl_1), i180-i182.
2. Tan, A. H. (1999, April). Text mining: The state of the art and the challenges. In Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases (Vol. 8, pp. 65-70). sn.
3. Berry, M. W. (2004). Survey of text mining. Computing Reviews, 45(9), 548
4. Larsen, B., & Aone, C. (1999, August). Fast and effective text mining using linear-time document clustering. In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 16-22). ACM.
5. Goyal, D. and Singh, H., Proposed Methodology of Text Summarization for Novel Text using Outlier Detection
6. Barzilay, R. and Lee, L., 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. arXiv preprint cs/0405039.

Books:

1. Aggarwal, C.C. and Zhai, C. eds., 2012. Mining text data. Springer Science & Business Media.
2. Feldman, R., & Sanger, J. (2007). The text mining handbook: advanced approaches in analyzing unstructured data. Cambridge university press.