A PROJECT REPORT ON

GEOGRAPHICAL UNDERSTANDING OF TWITTER HEALTH RELATED TOPICS BY USING TOPIC MODELLING ALGORITHMS

Major project submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

(2019-2023)

BY

Parakala Venkata Anirudh	**19241A1238**
Podduturi Hruthvik Reddy	**19241A1240**
Shaik Sohel	**19241A1251**
Thunuguntla Srichakranath	**19241A1255**

Under the Esteemed guidance of

Dr. N. Rajasekhar

Professor, Dept of IT

DEPARTMENT OF INFORMATION TECHNOLOGY

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

HYDERABAD

CERTIFICATE

This is to certify that it is a bonafide record of Mini Project work entitled "**GEOGRAPHICAL UNDERSTANDING OF TWITTER HEALTH RELATED TOPICS BY USING TOPIC MODELLING ALGORITHMS**" done by **Parakala Venkata Anirudh (19241A1238) , Podduturi Hruthvik Reddy (19241A1240), Shaik Sohel (19241A1251) and Thunuguntla Srichakranath (19241A1255)** students of **B.Tech(IT)** in the department of Information Technology, Gokaraju Rangaraju Institute of Engineering and Technology during the period 2019-2023 in the partial fulfilment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from GRIET, Hyderabad.

Dr. N. Rajasekhar
Professor, IT Department
Internal Project Guide

Dr. N. V. Ganapathi Raju
Professor & Head of Department,
Department of IT

Project external

ACKNOWLEDGEMENT

We take immense pleasure in expressing gratitude to our internal guide **Dr. N. Rajasekhar, Professor,** Information Technology, GRIET. We express our sincere thanks for his encouragement, suggestions, and support, which provided the impetus and paved the way for the successful completion of the project work.

We wish to express our gratitude to **Dr. N. V. Ganapathi Raju,** Head of IT Department and our project co-ordinators **Mr. G. Vijendar Reddy,** Associate Professor and **Dr. G. Prashanthi**, Assistant Professor for their constant support during the project.

We express our sincere thanks to **Dr. Jandhyala N Murthy,** Director, GRIET, and **Dr. J. Praveen,** Principal, GRIET, for providing us the conductive environment for carrying through our academic schedules and project with ease.

Email: anirudhparakala@gmail.com
Contact No: 7993082271
Address: KPHB, Hyderabad

Email: hruthvikreddy@gmail.com
Contact No: 9100069060
Address: Mathrusri Nagar, Hyderabad

Email: shaiksohel5588@gmail.com
Contact No: 8919219508
Address: Nizamabad

Email: tsrichakranath@gmail.com
Contact No: 9160763234
Address: Nizampet, Hyderabad

DECLARATION

This is to certify that the project entitled **"GEOGRAPHICAL UNDERSTANDING OF TWITTER HEALTH RELATED TOPICS BY USING TOPIC MODELLING ALGORITHMS"** is a bonafide work done by us in the partial fulfilment of the requirements for the award of the degree **Bachelor of Technology in Information Technology** from Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad.

We also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites, books and paper publications are mentioned in Bibliography.

This work was not submitted earlier at any other University or Institute for the award of any degree.

Parakala Venkata Anirudh	**19241A1238**
Podduturi Hruthvik Reddy	**19241A1240**
Shaik Sohel	**19241A1251**
Thunuguntla Srichakranath	**19241A1255**

TABLE OF CONTENTS

ABSTRACT

There is a need for understanding the health conditions worldwide with respect to every country. Most of the diseases and their symptoms are visible only after it has affected a substantially large population of a developed country. People must know about the diseases prevailing in a particular country and the precautions to be taken in case of being affected. An application that gets the information of a particular country health status must be developed. This project does not have any dataset and relies on live data extracted from twitter. Proper text data analysis can be done by using topic modelling (TM). Analysing topics is a critical issue that permits an unreliable number of TM topics that address the dismal outcomes in HRC (Health Related Clustering) in data sources. One of the proposed Distributed topic models Latent Dirichlet Allocation (LDA) is a credible approaches for balancing and clipping to the direction of health topics from various perspective data sources in health statistics clustering.

LDA assigns the probabilities to the words basing on their occurence in the given group of documents which here are tweets. A feature extraction technique called tf-idf vectorization must be followed for LDA to produce an efficient output. This service provides a concise description of the nation's hash tag system for good public health and analyzes the development of the most significant tweets about health in order to provide a preliminary public recommendation. This will most effectively improve the health care systems, median age of the world’s population.

1. INTRODUCTION

1.1 Introduction to Project

Twitter has huge volume of data regarding different topics which were basically discussed by the users in the form of the tweets made by them. This data contains crucial information regarding health field which could be utilized for further developments in health care industry. The rapid growth in the amount of technology used by the health system could always use this processed data for more advanced research system.

Analysis of these many tweets manually sent by billions of users is far from an impossible task. To understand key features or topics discussed by people artificial intelligence has a concept called Natural language processing shortly called NLP which helps in text analysis from a huge set of documents. Our project focuses on using topic modelling algorithms to analyse health data extracted from Twitter.

The data from Twitter is entirely unstructured and needs to be processed for the system to be more efficient. Various pre-processing techniques are applied to the text data and the resultant output is subjected to feature extraction concepts to identify the one thousand important or most repeated words. These algorithm's final output is called a topic which are just words combined into multiple groups.

The algorithm Hadoop Distributed Latent Dirichlet Allocation (hDiLDA) is being used in this project. The final output of the algorithms is being used to create a word cloud that gives a concise on the most used word in the processed answer.

1.2 Existing Systems

Healthcare systems that existed earlier were developed using past data to understand current requirements. This approach is not always effective especially when dealing with healthcare systems because the diseases in the past and the diseases that will occur can be very different often on a genetic level. There is very less probability of a disease reoccurring and there is a very high probability

of a new type of disease spreading. So, systems that deal with past data are often ineffective.

1.3 Proposed system

Healthcare systems using past data are proven to be ineffective, so we would like to propose a model that deals with live data from twitter. Twitter is one of the most popular social media websites which hosts so many health-related discussion forums. Social media is faster than traditional media and the news is unadulterated in social media, making it one of the best places to get up to date health related information from. Our model would extract the unstructured twitter data and pre-process the data, then some highly efficient topic modelling algorithms will be applied on the data.

2. REQUIREMENT ENGINEERING

2.1 Recommended Hardware Requirements

Minimum hardware requirements are very dependent on the software being developed by a given Python/ VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor. The below mentioned are the hardware requirements for the project.

- Operating system: windows, Linux
- Processor: minimum intel i3
- RAM: minimum 4GB
- SSD: 256GB

2.2 Software Requirements

The appropriation of requirements and implementation constraints gives the general overview of the project regarding what the areas of strength and deficit are and how to tackle them. The below mentioned are minimum and mandatory software requirements needed to run the project.

- Python 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter

3. LITERATURE SURVEY

[1] This research is an analysis of 571 documents up to 2022. The research contains topics like covid 19, operations and supply chain management in the health industry. This research used NNMF, LDA, LSI, the algorithms that are used in the project. Pre-processing operations like stop word elimination is done. The final outcome is a data cloud that consists of all the keywords that are used for further analysis. It helped the team understand the process of cleaning the data and processes like stop word elimination used in the project. It also helped the team understand the working of algorithms, it also helped us with the implementation of the algorithms in the project

[2] This research consists of data from the last 40 years, topic modelling approach is used in the research. A total of 47000 articles are collected and only 20000 articles are used for analysis. Topics are further examined through trend analysis, word associations, two-dimensional embeddings. The topics are separated into different clusters of similar patterns. This research helped the team understand the topic modelling approach to a great depth. It helped the team to learn to apply topic modelling algorithms on large amounts of data. The trend analysis concept of the research has also been very helpful in the project.

[3] This is research on patient satisfaction. Patients are asked for a review of their experience and 33000 reviews are collected. Topic modelling approach is used to reveal negative reviews and complaints. In this research LDA algorithm is used for topic modelling. Then Naive Bayes algorithm is used for topic classification. This research mainly focused on LDA algorithm and helped us get a thorough understanding of the LDA algorithm and learn how efficient the algorithm is in creating topic models. It also helped us get an insight on how topic modelling can be used in a unique way to solve a problem.

[4] This paper provides information on dynamic content specific LDA topic modelling techniques that can help identify different domains of covid specific discourse that can then be utilized to track concerns and views in current social environment. This paper also suggests that model derived topics are more

coherent than standard LDA topics and that they also provide new features that are more useful in predicting new Covid-19 related outcome

[5] This study helped us understand the method and research required to undertake the given project. It also gave us an outline of the different process that we needed to perform in our projects like sampling the data, pre-processing the raw data and performing qualitative and sentiment analysis then applying machine learning algorithms and finding topic related themes and finally mapping the resultant data.

[6] This paper gives us a comparative study of LDA and LSA topic models. The paper performs the similarity computation among the given subjects by using the LDA and LSA topic models. It also shows us the computational costs of building the LSA and LDA matrices and an in-depth comparison of the use of LSA in contrast with LDA

[7] This paper gave us an understanding of what steps are needed to be taken before applying LDA to textual data. They include appropriate pre-processing, adequate selection of model parameters, evaluation of model's reliability, and process of validly interpreting results. This paper aim to make LDA topic modelling more accessible to communication researchers and to develop a brief hands-on guide to apply topic modelling and to propose methods to ensure validity and reliability of topic models.

[8] This paper talks about the usage of topic modelling in micro blogging websites like twitter. It also proposes several schemes to train a standard topic model and compare their qualities and effectiveness through several experiments from both qualitative and quantitative perspective. This paper shows that by training a topic model on aggregate messages we can get a greater quality of learned models which gives significantly better performance in two real world classification problems.

[9] This paper provided us the information on how to model and apply the topic modelling algorithms on twitter data, how to retrieve the required information and get the desires output from the data. This paper's aim was to go beyond the simple topic models and to explore the potential of more sophisticated topic modelling and hence models like LDA and SANCHOR can have the potential to improve on LDA.

4. TECHNOLOGY

4.1 Introduction to Python

Python is an easy language to learn. Released in the early 90's by Guido van Rossum, reduced the number of lines to code using indentation and the readability has become easy. Numerous modules and libraries have been included in python to enable easy programming to all industrial scales. An improvement in managing memory reduced a lot of complexity.

Python can be incorporated with most of the currently available technologies and languages like HTML with python, IOT using python etc. Each of which use one or more modules and libraries for implementation. Being very flexible to use many new technologies are being developed using python.

4.2 YOU CAN USE PYTHON FOR PRETTY MUCH ANYTHING

One significant advantage of learning Python is that it's a general-purpose language that can be applied in a large variety of projects. Below are just some of the most common fields where Python has found its use:

- Data science
- Web development
- Computer graphics
- Basic game development
- Mapping and geography (GIS software)
- Scientific and mathematical computing

4.3 PYTHON IS WIDELY USED IN DATA SCIENCE

Python makes it easy to do tasks related to ML, DL, NLP and many complex data manipulations etc. These advantages of python over other languages are making it a great tool to do complicated and scientific calculations.

Some of the python libraries are mentioned below:

4.4 LIBRARIES

4.4.1 PANDAS

Pandas is a free library available in python. This is one of the libraries that generally comes with the download of python. This is mainly used for data manipulation and is mainly used with another library called NumPy. This library is generally imported with an alias name as “pd”. Pandas is built on top of the **Numpy** package, means **Numpy** is required for operating the Pandas. Before Pandas, Python was capable for data preparation, but it only provided limited support for data analysis. So, Pandas came into the picture and enhanced the capabilities of data analysis. It can perform five significant steps required for processing and analysis of data irrespective of the origin of the data, i.e., **load, manipulate, prepare, model, and analyse**.

4.4.2 MATPLOTLIB

Matplotlib is considered as a multi-platform data visualization library built on NumPy array. It can be used in various scenarios like shell, web application, python scripts, and other graphical user interface toolkits.

There are several toolkits available for us that can be used to enhance the many functions of the matplotlib library. They are

Cartopy: It is a library for mapping objects that consists of polygon, arbitrary point, mage transformation abilities, object-oriented map projection definitions and line.

Mplot3d: It is mainly utilized for 3d plots.

Bashmap: It is a plotting toolkit to map objects that have several political boundaries, map projections, and coastlines.

Natgrid: It is a user interface to the natgrid library for the asymmetrical gridding of the spaced data

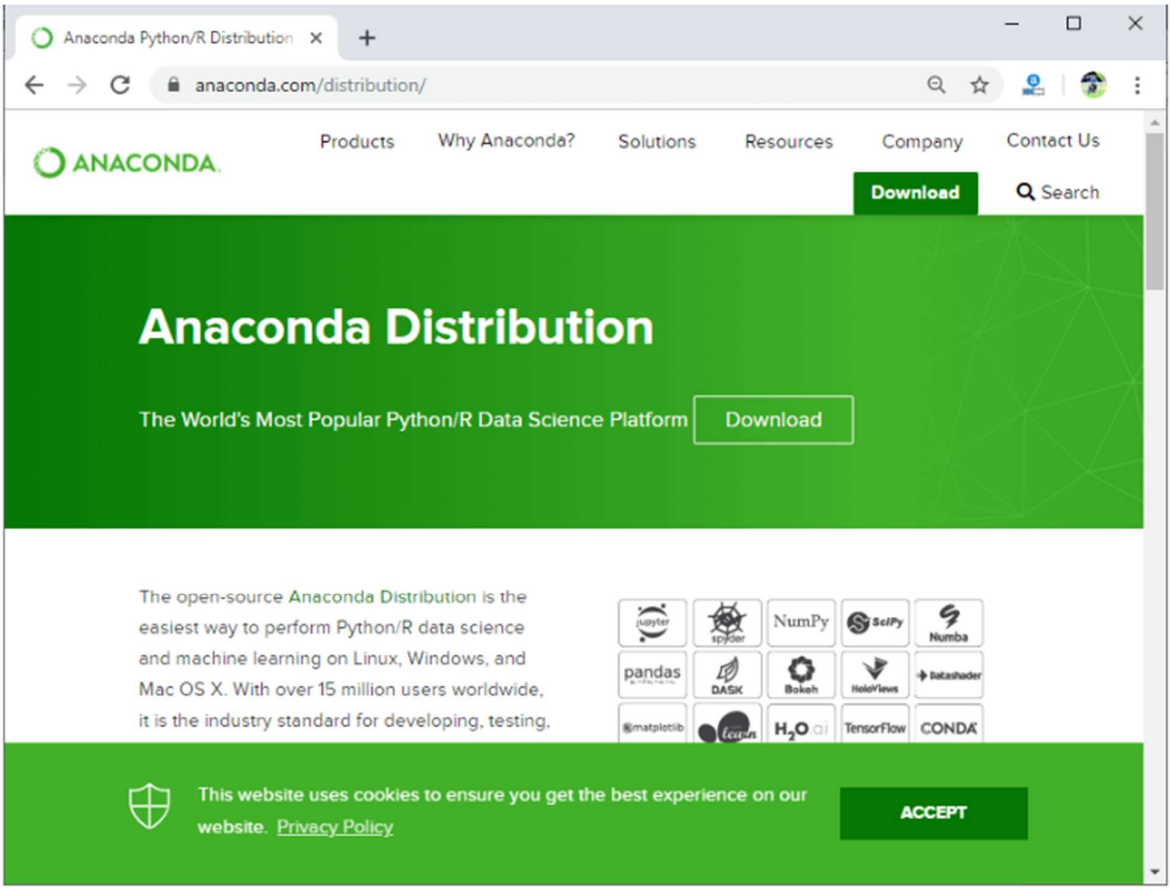
Excel tools: This library provides many facilities to utilize the exchange of data with Microsoft excel

How to Install Matplotlib library:

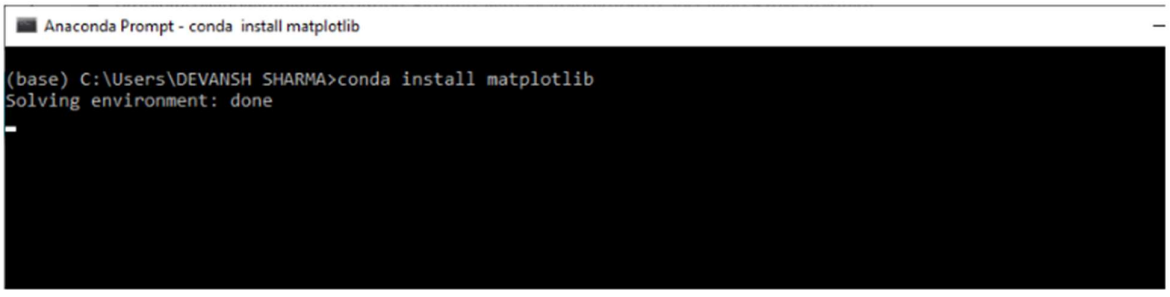
To install matplotlib library in your computer you need to know the distribution that is installed in your computer. These are the following ways to install

Using anaconda distribution of python:

The simplest way to install Matplotlib is to download the Anaconda distribution of python because the Matplotlib library is pre-installed in the anaconda distribution.



> conda install matplotlib



Pyplot Module in Python:

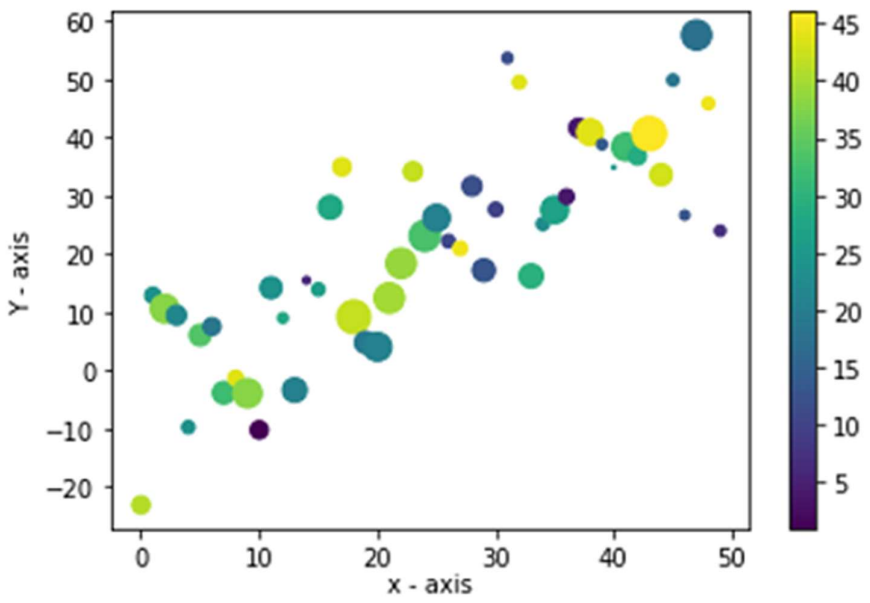
In the Matplotlib library we have a Pyplot module which provides us with many interfaces similar to that of MATLAB. This library in Matplotlib was created to be used just like MATLAB but with the added capability of using python and also being free and open source. There are several plots we can use in Pyplot like Scatter, Histogram, Image contour, polar, Line plot and 3D plot.

matplotlib.pyplot is a set of utilities and functions that makes the library matplotlib function exactly like MATLAB. Each function in Pyplot make the necessary changes to a figure like creating a figure, creating a plotting area in a figure, decorating and labelling the plot etc,

How to plot with String Values:

There are many modules where you have the data in a particular format in which you can access particular variables with strings. For Example in the module numpy.recarray and pandas.DataFrame. The Matplotlib module enables you to provide a similar object with the data keyword argument in which you can generate plots with the strings corresponding to the given variables. Given below is an example script

```
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100

plt.scatter('a', 'b', c='c', s='d', data=data)
plt.xlabel('entry a')
plt.ylabel('entry b')
plt.show()
```

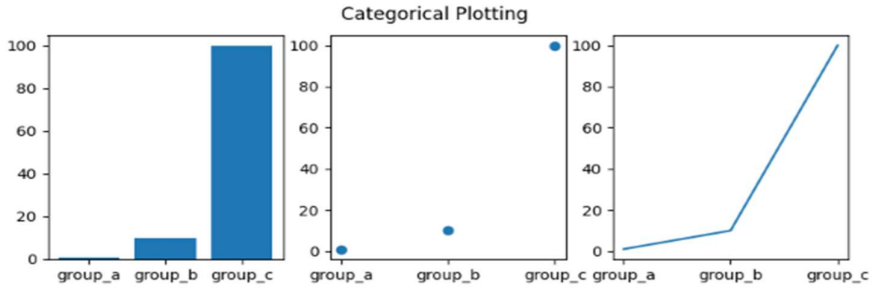


Plotting with Categorical Values:

We also have the option to create a plot with categorical values, this library allows you to pass categorical variables to many plotting functions directly.

Given example:

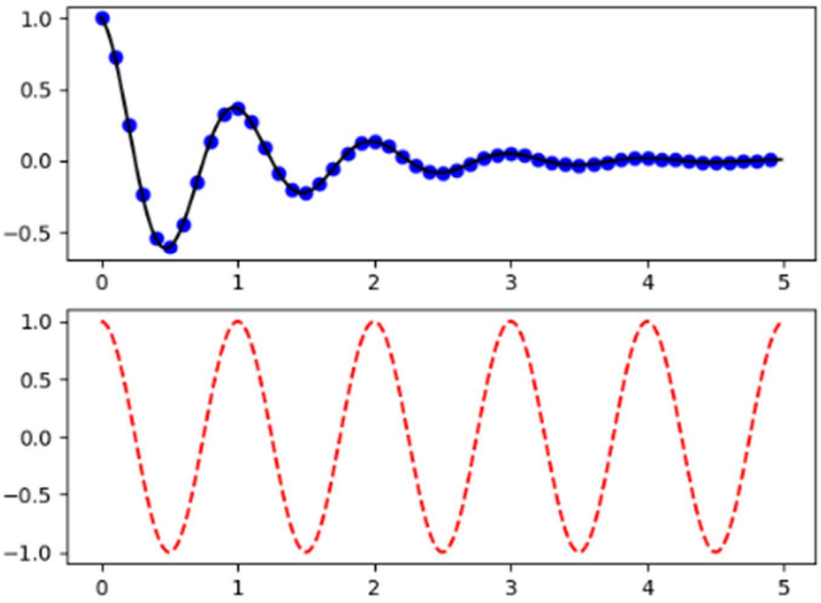
```
names = ['group_a', 'group_b', 'group_c']
values = [1, 10, 100]

plt.figure(figsize=(9, 3))

plt.subplot(131)
plt.bar(names, values)
plt.subplot(132)
plt.scatter(names, values)
plt.subplot(133)
plt.plot(names, values)
plt.suptitle('Categorical Plotting')
plt.show()
```



Working with multiple figure and axes:

Pyplot in matplotlib library has the option of current figure and current axes. All the functions while plotting apply to the given current axes. The function “gca” gives the current axes and the function “gcf” gives the current figure. Given below is an example script to create two subplots.

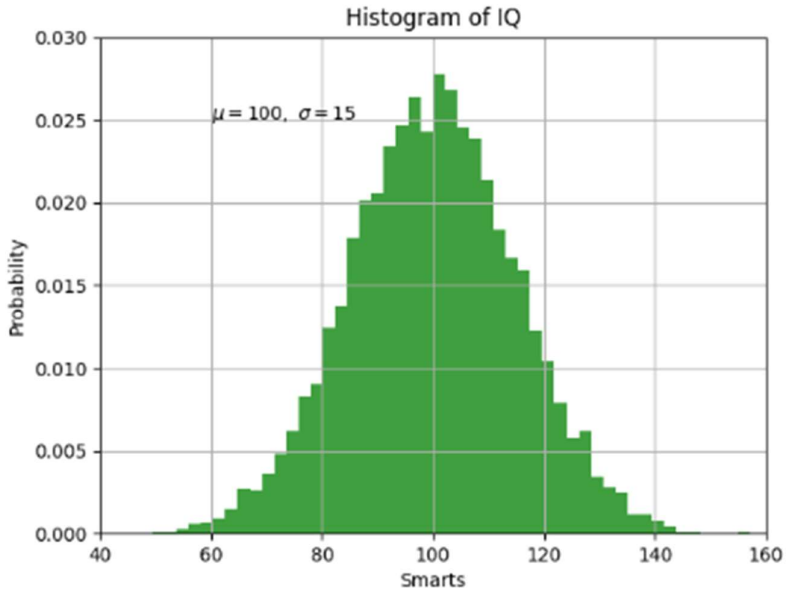
```
def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure()
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```



Plotting with text:

In matplotlib library, text can be used to add a given text in any required location and “title”. “xlabel”, “ylabel” are used to insert text in the required locations. Given below is an example script

```
mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, density=True, facecolor='g', alpha=0.75)


plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```



4.4.3 SCIKIT-LEARN

It is an open-source library to implement machine learning models in Python. This library supports the vast majority of algorithms like random forest, SVM, KNN, XGBoost. It is made over NumPy. This library also helps in various steps and processes of model building like regression, clustering, model selection, dimensionality reduction, classification.

Scikit-learn has very simple implementation and is easy to work with and also gives the user successful performance but it does not implement parallel processing. We can also deploy deep learning algorithms in sklearn.

Installation of Sklearn on the System:

Before we install sklearn, we need to install NumPy and SciPy on our system as its dependencies,so after installing NumPy and SciPy correctly using pip to install sklearn is the most simplest way to install scikit-learn

```
$ pip install -U scikit-learn
```

Importing the dataset:

Before importing the dataset, we must import Scikit-Learn and Pandas library using the following commands.

```
# Importing the required libraries
import sklearn
import pandas as pd
```

Now after this, we can use the following command to import the iris plant dataset as an example dataset from sklearn library:

```
# Importing the dataset from the datasets module of sklearn
from sklearn.datasets import load_iris

# Loading the dataset
iris = load_iris()

# Creating the dataframe of the dataset
df = pd.DataFrame(iris.data, columns = iris.feature_names)
```

Splitting the dataset:

We can split our complete data set into 2 different parts, a testing dataset and a training dataset. We can then utilize our testing dataset to validate or test the model. When the model has been successfully trained using the training dataset. We can then calculate the performance of the given trained model.

Given below is an example in which we divide the dataset into a 70:30 ratio, which means that 70% of our dataset will be used to train the model and the rest 30% will be used to test the model.

```
# Importing the class to perform train test split from model_selection module
from sklearn.model_selection import train_test_split

# Separating the dependent and independent features
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values

# Creating a testing dataset of size 0.3 times the whole dataset
X_train, X_test, y_train, y_test = train_test_split(
  X, y, test_size = 0.3, random_state = 1
)

# Printing the shape of the training and testing dataset
print(X_train.shape)
print(X_test.shape)

print(y_train.shape)
print(y_test.shape)
```

Output for the following code snippet will be:

```
(105, 3)
(45, 3)
(105,)
(45,)
```

Fitting and predicting:

The Scikit-learn library also provides us with many built-in machine learning models and algorithms called estimators. Each one of these estimators can be fitted to some data using the method “fit”.

Given below is an example in which we fit a RandomForestClassifier to some data:

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> clf = RandomForestClassifier(random_state=0)
>>> X = [[ 1,  2,  3],  # 2 samples, 3 features
...      [11, 12, 13]]
>>> y = [0, 1]  # classes of each sample
>>> clf.fit(X, y)
RandomForestClassifier(random_state=0)
```

The method fit has two input parameters:

The sample matrix “X”, Its size is generally (n_samples, n_features), the samples are denoted by rows and the features are denoted by columns.

The target value “Y” which are real numbers for regressive tasks or can also be binary values for classification.

The values “X" and “Y” are generally are NumPy arrays or array-like similar data types. Although there are some estimators which can work with formats like sparse matrices.

```
>>> clf.predict(X)  # predict classes of the training data
array([0, 1])
>>> clf.predict([[4, 5, 6], [14, 15, 16]])  # predict classes of new data
array([0, 1])
```

Transformations and pre- processors:

Many machine learning models are composed of different parts. A typical machine learning pipeline contains pre-processing step that transforms the data and a final predictor that predicts target values

In the library scikit-learn, pre- processors and transformers generally follow the same API as the estimator objects. The transform objects have transform method that outputs a newly transformed sample matrix "X":

```
>>> from sklearn.preprocessing import StandardScaler
>>> X = [[0, 15],
...      [1, -10]]
>>> # scale data according to computed scaling values
>>> StandardScaler().fit(X).transform(X)
array([[-1.,  1.],
       [ 1., -1.]])
```

Pipelines: chaining pre-processors and estimators:

Estimators and Transformers can be combined into a single unifying object represented as a "Pipeline". This pipeline also has the same API as the regular estimator. It can also be utilized and fitted and used for predicting with "fit" and "predict". Using a pipeline will also prevent data leakage.

```
>>> from sklearn.preprocessing import StandardScaler
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.pipeline import make_pipeline
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.metrics import accuracy_score
...
>>> # create a pipeline object
>>> pipe = make_pipeline(
...     StandardScaler(),
...     LogisticRegression()
... )
...
>>> # load the iris dataset and split it into train and test sets
>>> X, y = load_iris(return_X_y=True)
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
...
>>> # fit the whole pipeline
>>> pipe.fit(X_train, y_train)
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('logisticregression', LogisticRegression())])
>>> # we can now use it like any other estimator
>>> accuracy_score(pipe.predict(X_test), y_test)
0.97...
```

4.4.4 NLTK

Abbreviated as Natural Language Tool Kit is a python library used for manipulating text data in python. It is used for pattern analysis, linguistics, cognitive science etc. It is developer Steven Bird and Edward Loper at the University of Pennsylvania.

For installation of the nltk library we use a command called:

```
conda install -c anaconda nltk
```

This command downloads the latest nltk version available in the market into our virtual python environment enabling us all to use all the modules available in nltk.

NLP is a sub field of NLTK which in turn is a sub domain of AI. NLTK has incorporated many text analysis tasks like word count, character count, lemmatization, stemming etc very easy and elegant to work. Every day we humans use hundreds and thousands of words to communicate with each other. It's a very simple activity for us. Interpreting the deeper meaning of the words is not easy and it can simply be achieved with the help NLP (**Natural Language Processing**).

4.4.5 TWEEPY

Tweepy is the python api that is used to invoke and extract tweets from twitter using various functions. There are 4 keys to be used to extract tweets from twitter:

a. Consumer key
b. Consumer secret
c. Access token
d. Access token secret

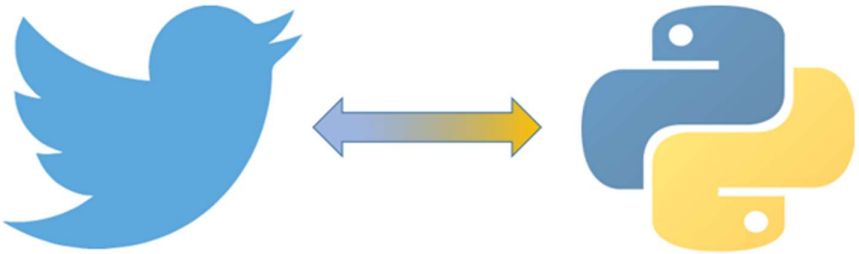
These 4 keys are used to authenticate the user and then proceed to tweet extraction.

Tweepy in python can be installed using the below command.

```
pip install tweepy
```

Instead of manually going to twitter and copying the tweets for analysis tweepy helps us in quicker and reliable way of extracting and analysing the tweets with the help of a developer account created.

Using the API, we can get information regarding a place with their coordinates. Tweets in a particular timeframe can be extracted. Topic specific content can also be retrieved.



4.4.6 PYTTSXE

Pyttsx3 is a Python library used for Text-to-Speech (TTS) conversion. It enables the conversion of written text into spoken words, allowing developers to build applications that can speak out text in a human-like voice.

Pyttsx3 is a cross-platform library, which means it can work on various operating systems such as Windows, Linux, and Mac OS X. It is built on top of the Python Speech API (SAPI5) and supports multiple TTS engines, including Microsoft SAPI5, eSpeak, and NSSpeechSynthesizer.

Pyttsx3 has a straightforward API that enables developers to perform tasks such as setting the voice, changing the rate of speech, pausing and resuming speech, and much more. Additionally, it can also handle events such as when the speech is started or completed, enabling developers to create more interactive applications. Pyttsx3 is a powerful library that makes it easy for developers to add TTS functionality to their Python applications.

```python
import pyttsx3

engine = pyttsx3.init()
engine.say("Hello, how are you today?")
engine.runAndWait()
```

Example of how to use pyttsx3 to convert text to speech:

In the above code, we first import the pyttsx3 module and initialize the engine. We then use the say method to specify the text we want to convert to speech. Finally, we call the runAndWait method to play the speech.

4.4.7 SPEECHRECOGNITION

The SpeechRecognition library in Python is a powerful tool for performing speech recognition tasks in Python programming language. It enables developers to convert audio recordings into text data, allowing the program to understand and respond to human speech.

The SpeechRecognition library supports various speech recognition engines such as Google Speech Recognition, CMU Sphinx, and Microsoft Bing Voice Recognition. The library can recognize speech from audio files or real-time audio input from microphones.

To use the SpeechRecognition library, you first need to install it using pip. You can do this by running the following command in your command prompt:

```
pip install SpeechRecognition
```

After installing the library, you can import it into your Python program and start using it. Here's an example of how to use the Google Speech Recognition engine to recognize speech from an audio file:

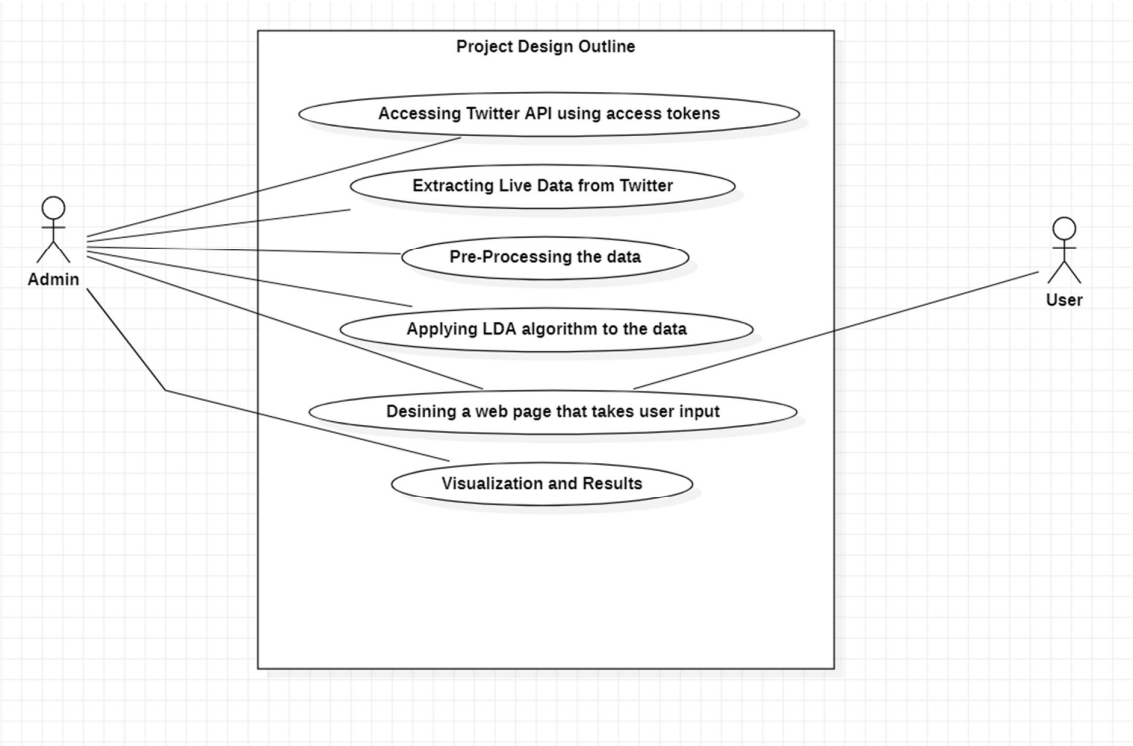
```python
import speech_recognition as sr

r = sr.Recognizer()

with sr.AudioFile('audio_file.wav') as source:
    audio = r.record(source)

text = r.recognize_google(audio)

print(text)
```

In the above code, we first import the SpeechRecognition library and initialize a Recognizer object. We then open an audio file using the AudioFile method and record the audio using the record method. We then use the recognize_google method to recognize the speech and convert it into text.

Overall, the SpeechRecognition library is a powerful tool for performing speech recognition tasks in Python. It is easy to use and supports multiple speech recognition engines, making it a versatile library for speech recognition tasks.

5. DESIGN REQUIREMENT ENGINEERING

5.1 Use Case Diagram



Actors: user, admin

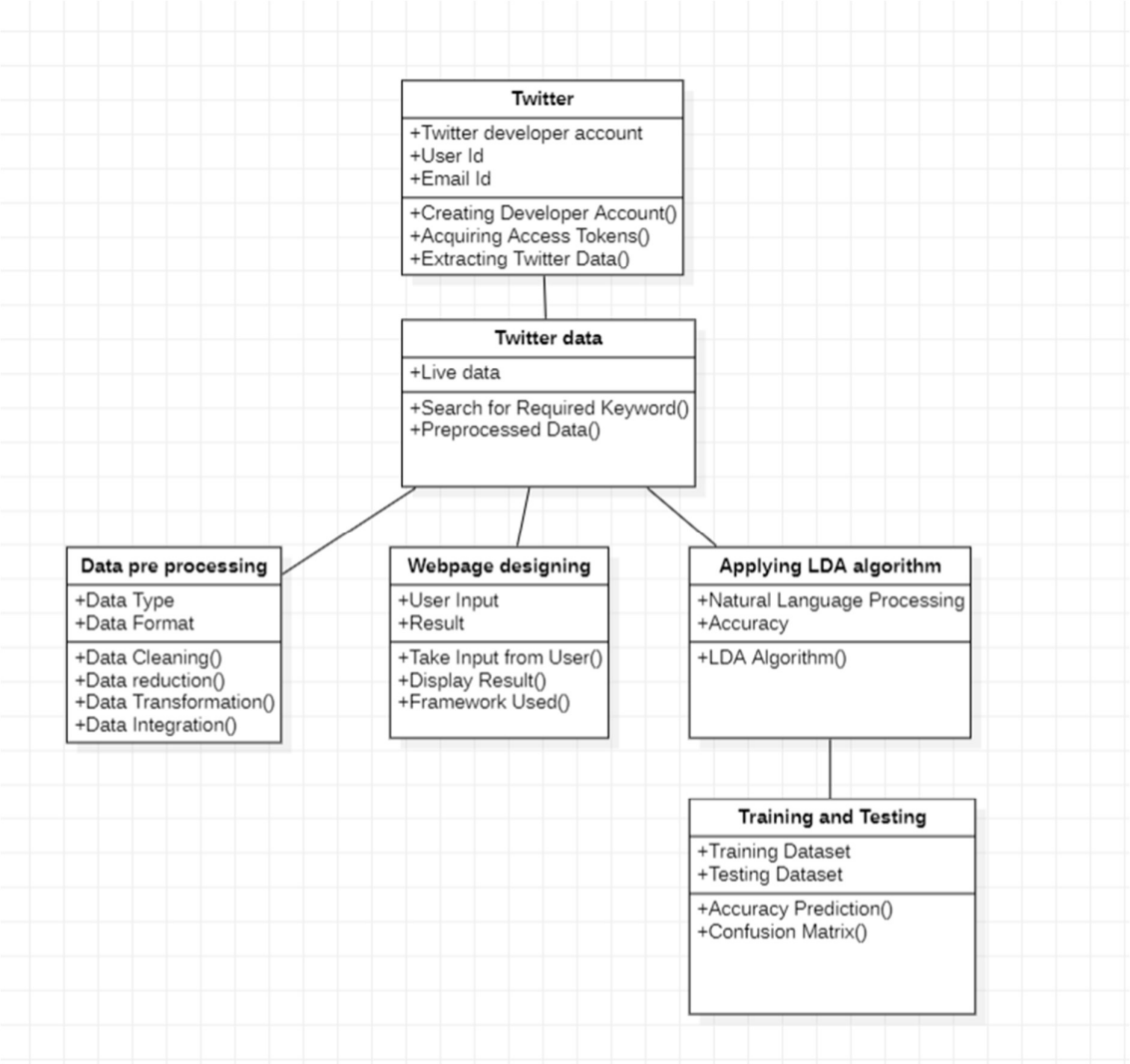
Use cases:

1) Accessing Twitter API using access tokens
2) Extracting Live data from twitter.
3) Pre-processing the data
4) Applying LDA algorithm to the data
5) Training and testing the model
6) Webpage designing
7) Visualization and Results

Relationships:

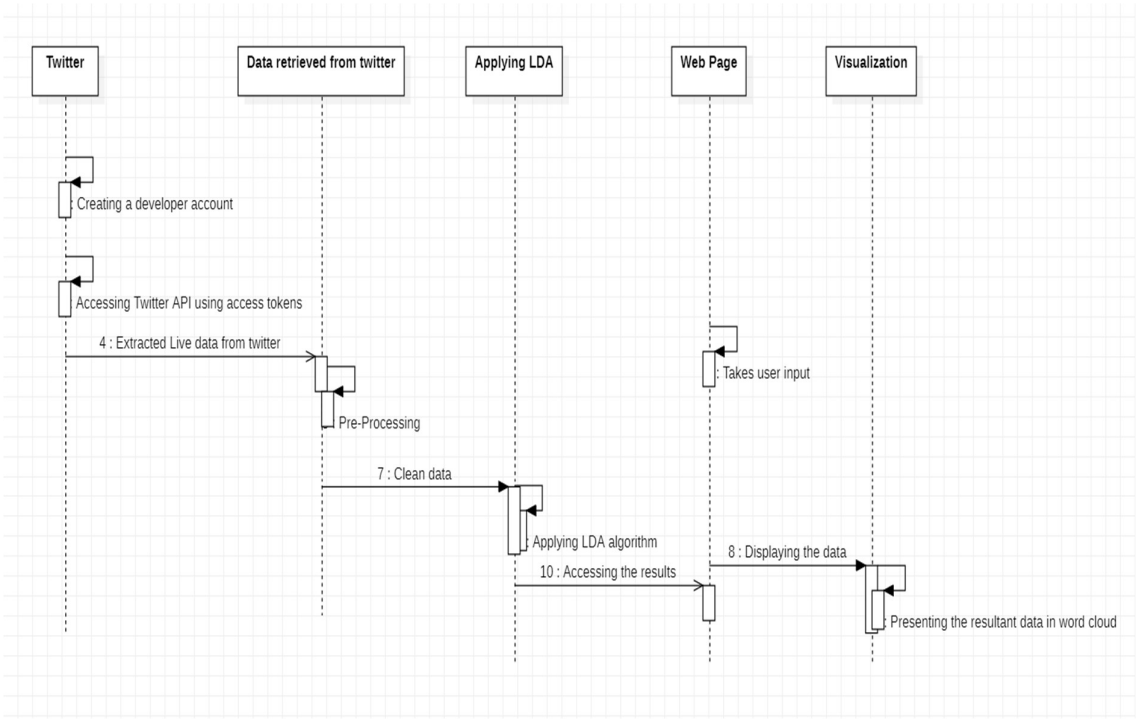
1) The user only has access to the webpage which shows all the results.
2) The admin has all access to the project.

5.2 Class Diagram



1) Twitter application is used to create a developer account and the access tokens of the developer account are used to extract twitter data via twitter api.
2) The extracted twitter data is used in a csv file. The extracted unstructured data is cleaned and pre-processed.
3) Then the processed data is used to train and test with LDA.
4) An interface is created to present the data that is discovered.

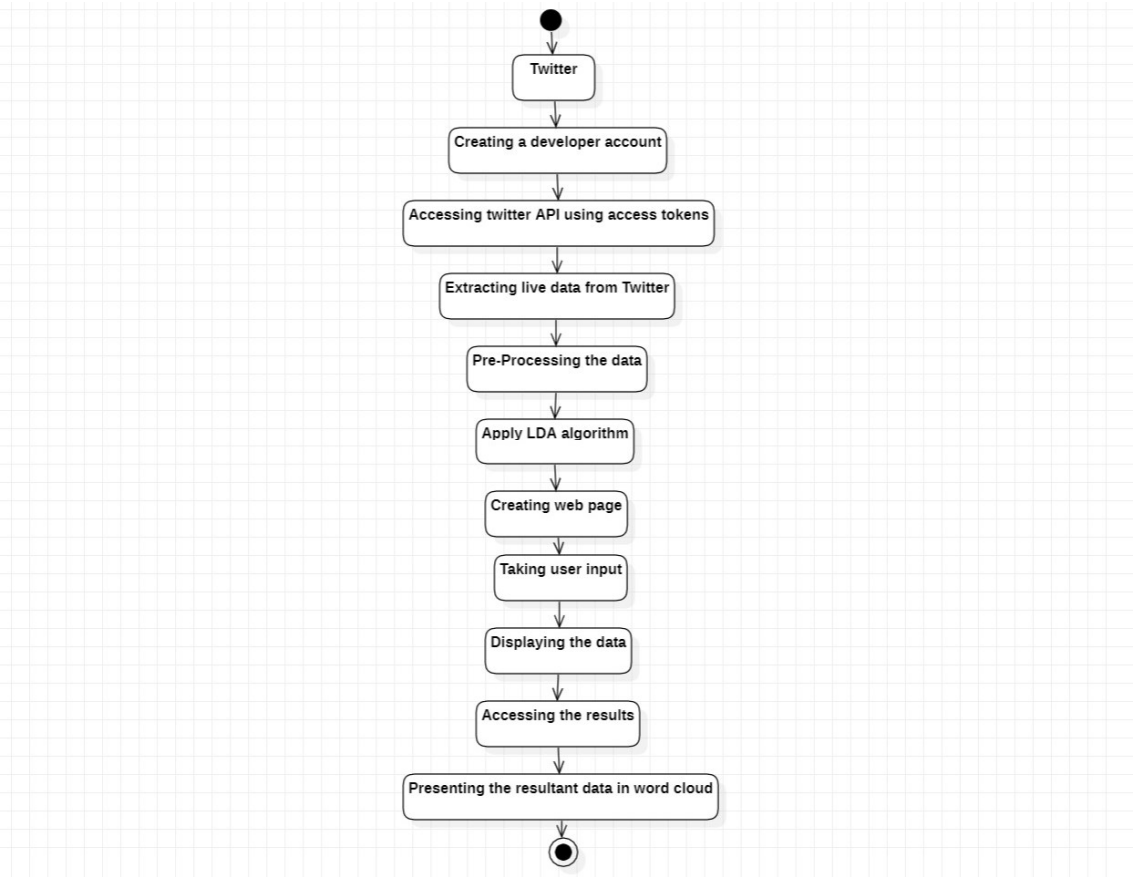
5.3 Sequence Diagram



Scenario:

1) The first step is to create a twitter account and use access tokens given by the developer account to extract twitter data.
2) The twitter data is unstructured, then the data is pre-processed using techniques like Stemming, lemmatization, tokenization.
3) Key features are then extracted from the preprocessed data.
4) The processed data is used to train and test the topic modelling algorithm LDA.
5) A web-page is designed for presentation and user interaction.

5.4 Activity Diagram



The process starts with gathering data from twitter about various health diseases that are prevalent worldwide. Using this data we do the pre-processing to remove all the words that are useless and add no meaning to the tweets. Then the pre-processed data is used for feature extraction capturing the key and important topics discussed. This generates the most used key words worldwide. Using the key words a topic modelling algorithm called as LDA is trained and then the results are displayed using a wordcloud.

6.Implementation

6.1 Datasets

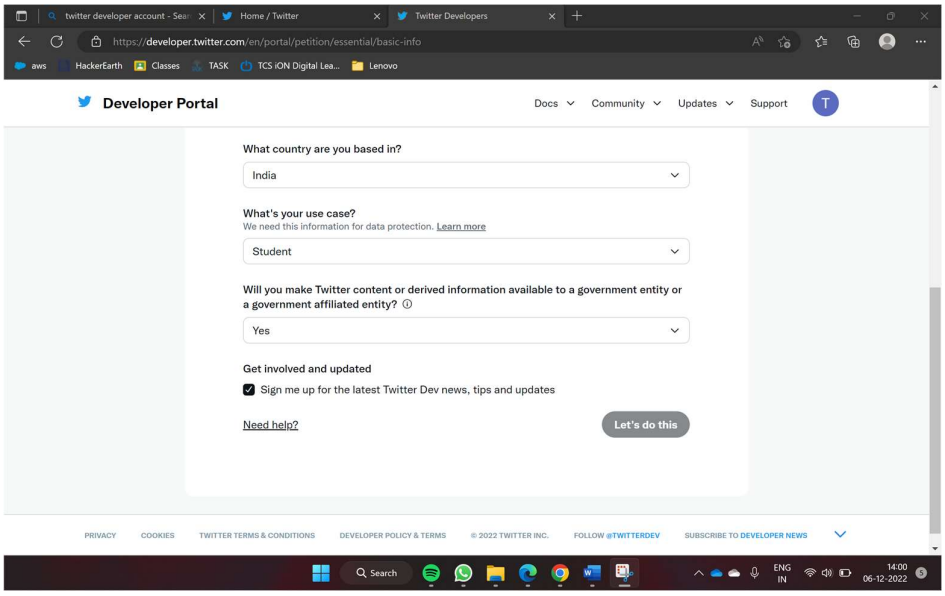
There is no dataset associated with this study. The dataset is generated on the go while project execution. The user is given a choice to input location and search query basing on which the tweets are extracted into a csv file.

6.2 Stages in the project

1) Creating a developer account
2) Extracting tweets from twitter
3) Importing libraries
4) Pre-processing the data
5) Feature extraction from the pre-processed data
6) Creating instance of LDA
7) Data analysis
8) Data visualization

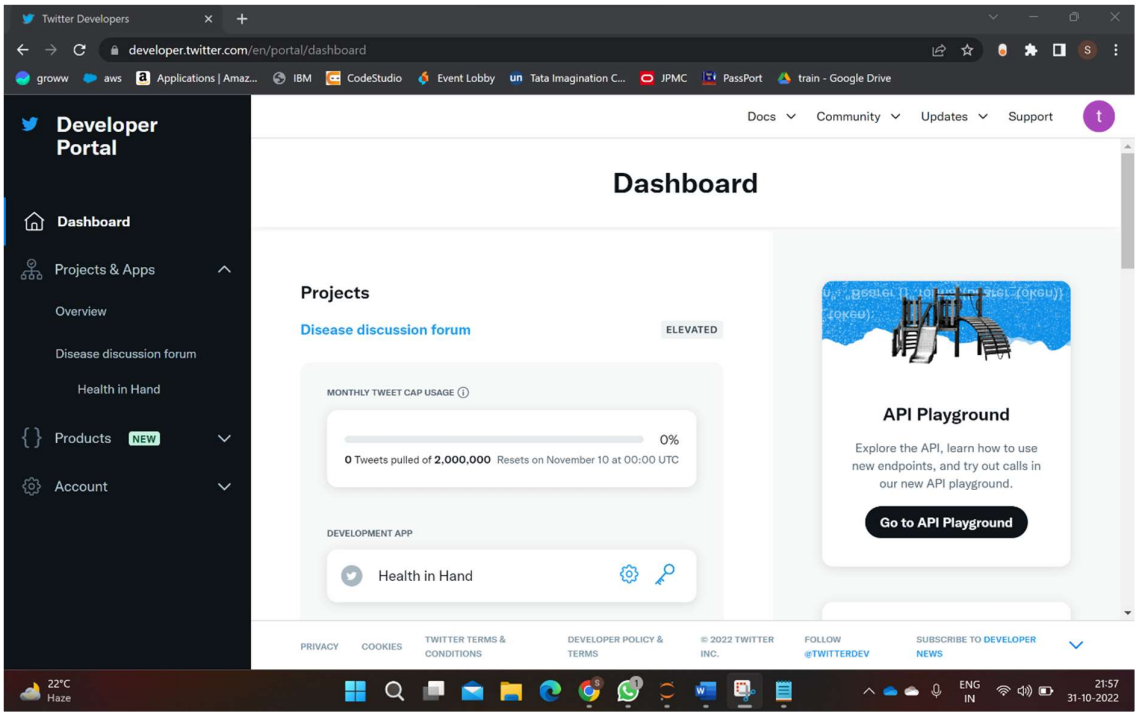
6.2.1 Creating a developer account

1. Go to developer.twitter.com
2. Click on **sign up**
3. Then enter the details asked by the page.



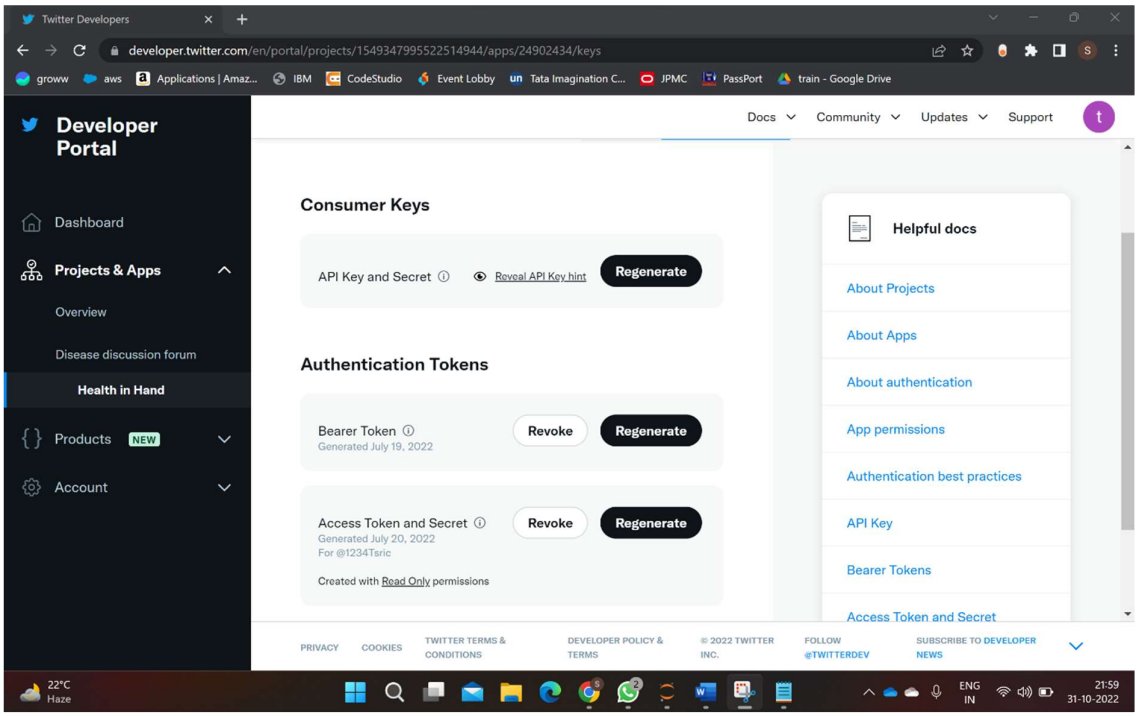
Picture to enter details

4. Once all the details are entered a series of mails are sent to the mail id with which the account is created.
5. Once all the mails are answered then the developer account is given to us.
6. Login to the twitter developer portal.
7. In the top right corner of the screen click on **developer portal**
8. A panel on the left side is displayed showing the dashboard, projects & Apps, Products, Account
9. Click on **Projects & Apps.**
10. Create a project and give a name to it.



Account after creation of project

11. Once created the project will be displayed in the left side panel under **Projects & Apps.**
12. Click on the project and you can see few tokens that can be generated and used for tweet extraction.
13. Copy the tokens and save them in a notepad.



To copy API keys

6.2.2 Extracting tweets from twitter

In [10]:

```
import tweepy
import pandas as pd
consumer_key = "g7YmBrOeseYDJ6kHFTLCx56lr"
consumer_secret = "7LjcvqØX1cOIOSLtbgJO4yV9vILsJDaYEayr1qwrys62jfIf8z"
access_token = "1546125629783166976-aaHta44KHHCf2II9mv6IQdKqybcxfm"
access_token_secret = "WaR6SQLzlgKnMgØJTI1dfLpqwlfSKKJnkThKfcvFjNNN3"

auth = tweepy.OAuth1UserHandler(consumer_key, consumer_secret, access_token, access_token_secret)

api = tweepy.API(auth)
extracted_tweets = []

for status in tweepy.Cursor(api.search_tweets, "rabies"+"-filter:retweets", lang="en", count = 50).items(500):
    extracted_tweets.append(status.text)
df = pd.DataFrame(extracted_tweets)
excel_sheet = df.to_csv('C:/Users/tsric/OneDrive - Gokaraju Rangaraju Insititute of Engineering and Technology/Major Project
```

6.2.3 Importing libraries

1) Modules that are imported in this project:
    a. Tweepy
    b. Pandas
    c. NLTK
    d. Scikit-learn

e. Matplotlib
f. Warnings
g. Word Cloud

In [12]:
```
import pandas as pd
from pandas import DataFrame as DF

#Pre-processing
import nltk
from nltk.corpus import stopwords
import re
from nltk.stem import WordNetLemmatizer as WNL

# Feature Extraction
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

#Topic Modelling algorithms
from sklearn.decomposition import NMF
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.decomposition import TruncatedSVD

#Ignore Warnings
import warnings
warnings.filterwarnings('ignore')

#Visual Representation
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

6.2.4 Pre-Processing the data

6.2.4.1 Stop-Word Removal:

Stop word removal is one of the most commonly used pre-processing steps in various NLP applications. The idea is to simply remove words that are common to all documents in the corpus. Articles and pronouns are generally classified as stop words. These words have no meaning in some NLP tasks such as information retrieval and classification. Removing stop words improves the efficiency of the algorithm.

6.2.4.2 Tokenization:

Tokenization is the first step in the NLP pipeline. This has important implications for the rest of the pipeline. Tokenizers break down unstructured data and natural language text into blocks of information that can be viewed as discrete elements. Tokenization allows you to separate sentences, words, characters, or sub words.

Tokenization is done in the following sequence:

- Tokenize text into sentences
- Tokenize sentences into words

- Tokenize regular expression sentences
- The output can be used as input to other algorithms

6.2.4.3 Lemmatization:

Lemmatization is one of the most common text pre-processing techniques used in natural language processing (NLP). Lemmatization usually refers to using lexical and morphological analysis of a word to get things right, usually with the goal of removing only inflections and returning the base or dictionary form of the word known as the lemma. Lemmatization usually simply folds the various inflections of the lemma. Group words with similar meanings into one word.

In [5]:
```
# Data Cleaning, Stop words removal
lemma = WNL()
def clean_tweet(tweet):
    temp = tweet.lower()
    if(temp[:2] == 'rt'):
        temp = temp[2:]
    temp = re.sub("'", "", temp)
    temp = re.sub("@[A-Za-z0-9_]+","", temp)
    temp = re.sub("#[A-Za-z0-9_]+","", temp)
    temp = re.sub(r'http\S+', '', temp)
    temp = re.sub('[()!?]', ' ', temp)
    temp = re.sub('\[.*?\]',' ', temp)
    temp = re.sub("[^a-z0-9]"," ", temp)
    temp = temp.split()
    temp = [w for w in temp if not w in stopwords.words('english') or len(w)<3]
    temp = " ".join(word for word in temp)
    temp = lemma.lemmatize(temp)
    return temp
```

In [9]:
```
data = pd.read_excel('Raw_Data.xlsx')
print(data)
data = data.drop_duplicates(subset=['Tweet'])
data = data.reset_index()
arr = []
for i in range(len(data)):
    data['Tweet'][i] = clean_tweet(str(data['Tweet'][i]))
    if(len(data['Tweet'][i])>0):
        arr.append(data['Tweet'][i])
d = {'Processed_Tweets' : arr}
df = DF(d)
```

```
                                                Tweet
0     Stray dog menace is getting out of control in ...
1     Rabies vaccines are effective, Kerala deaths m...
2     @Rabies_widow @flat_earther404 @firki07 If I h...
3     @KammilleAirFire And there is no rabies on thi...
4     I confirm I will participate in NFT Minting ev...
...                                                 ...
3995  @AbarukGaming @TheeHerc @Twitter ABSOLUTELY in...
3996  Alzheimer/Dementia Radio Now Playing Andrea Va...
3997  A New Protein That May Contribute to Alzheimer...
3998  Fewer #Asians, #Blacks, #Hispanics eligible fo...
3999  An Apple a Day to Keep Alzheimerâ€™s Away | Na...

[4000 rows x 1 columns]
```

```
In [10]:  1 print(df)
```

```
                                      Processed_Tweets
0     stray dog menace is getting of control in indi...
1     rabies vaccines effective kerala deaths may br...
2                    if i a rupee every time i saw meme
3     is no rabies on island i am promising raccy fr...
4     i confirm i participate in nft minting event o...
...                                                 ...
3939  absolutely indicative of 5 counties in pa sudd...
3940  alzheimer dementia radio playing andrea van le...
3941  a new protein may contribute to alzheimer s di...
3942         fewer eligible drugs may delay progression
3943  an apple a day to keep alzheimer s away nature...

[3944 rows x 1 columns]
```

6.2.5 Feature Extraction from pre-processed data

6.2.5.1 Count Vectorizer:

Machines cannot understand letters and words. Therefore, when working with text data, it must be represented numerically so that it can be understood by machines. To use text data for predictive modeling, you need to parse the text and remove certain words. This is a process called tokenization. These words should be encoded as numbers to be used as input for machine learning algorithms. A Count vectorizer is a way of converting text into numbers that the system can understand. Count vectorizer makes it easy to use text data directly in deep learning models such as machine learning and text classification. This can be used to remove stop words. A stop word is a word that can be safely removed from a sentence without changing the meaning of the sentence.

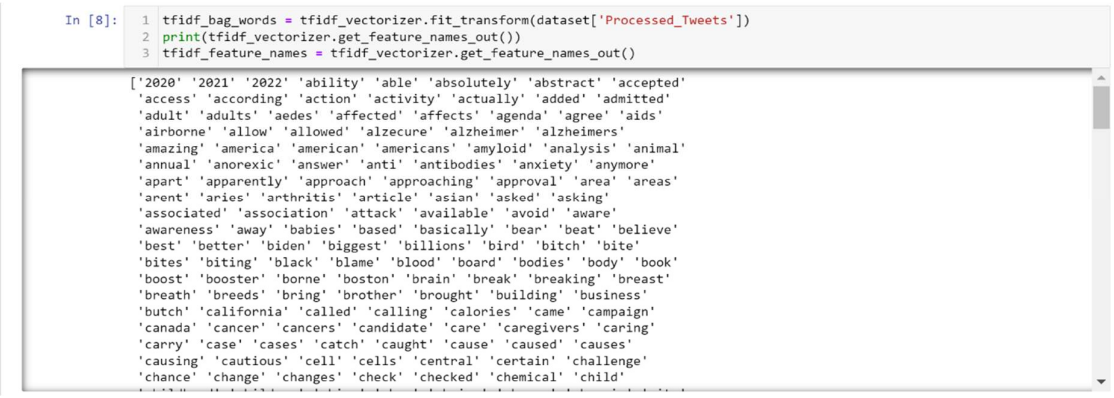
```
In [6]:  1 vectorizer = CountVectorizer(max_df=0.90,min_df=2,max_features=1000,stop_words='english')
         2 tfidf_vectorizer = TfidfVectorizer(max_df=0.90,min_df=2,max_features=1000,stop_words='english')
```

```
In [7]:  1 bag_words = vectorizer.fit_transform(dataset['Processed_Tweets'])
         2 print(vectorizer.get_feature_names_out())
         3 tf_feature_names = vectorizer.get_feature_names_out()
```

```
['2020' '2021' '2022' 'ability' 'able' 'absolutely' 'abstract' 'accepted'
 'access' 'according' 'action' 'activity' 'actually' 'added' 'admitted'
 'adult' 'adults' 'aedes' 'affected' 'affects' 'agenda' 'agree' 'aids'
 'airborne' 'allow' 'allowed' 'alzecure' 'alzheimer' 'alzheimers'
 'amazing' 'america' 'american' 'americans' 'amyloid' 'analysis' 'animal'
 'annual' 'anorexic' 'answer' 'anti' 'antibodies' 'anxiety' 'anymore'
 'apart' 'apparently' 'approach' 'approaching' 'approval' 'area' 'areas'
 'arent' 'aries' 'arthritis' 'article' 'asian' 'asked' 'asking'
 'associated' 'association' 'attack' 'available' 'avoid' 'aware'
 'awareness' 'away' 'babies' 'based' 'basically' 'bear' 'beat' 'believe'
 'best' 'better' 'biden' 'biggest' 'billions' 'bird' 'bitch' 'bite'
 'bites' 'biting' 'black' 'blame' 'blood' 'board' 'bodies' 'body' 'book'
 'boost' 'booster' 'borne' 'boston' 'brain' 'break' 'breaking' 'breast'
 'breath' 'breeds' 'bring' 'brother' 'brought' 'building' 'business'
 'butch' 'california' 'called' 'calling' 'calories' 'came' 'campaign'
 'canada' 'cancer' 'cancers' 'candidate' 'care' 'caregivers' 'caring'
 'carry' 'case' 'cases' 'catch' 'caught' 'cause' 'caused' 'causes'
 'causing' 'cautious' 'cell' 'cells' 'central' 'certain' 'challenge'
 'chance' 'change' 'changes' 'check' 'checked' 'chemical' 'child'
```

6.2.5.2 TF-IDF:

TF-IDF stands for Term Frequency Inverse Document Frequency. It is one of the most commonly used algorithms for converting text into meaningful numeric representations and is used to adapt machine algorithms for prediction. Machine learning algorithms are realized through mathematical elements such as statistics, algebra, and calculus. The data should be numeric, such as: B. A two-dimensional array with rows as instances and columns as features. The problem with natural languages is that the data is in the form of raw text, so we need to convert the text to vectors. TfidfVectorizer considers the overall weight of the words in the document. It combines two concepts: term frequency (TF) and document frequency (DF). Document frequency is the number of documents containing a particular term. Document frequency indicates how often a term appears. By giving higher weight to rare terms and lower weight to common terms, stop words can be removed very efficiently.



```
In [8]: 1 tfidf_bag_words = tfidf_vectorizer.fit_transform(dataset['Processed_Tweets'])
        2 print(tfidf_vectorizer.get_feature_names_out())
        3 tfidf_feature_names = tfidf_vectorizer.get_feature_names_out()

['2020' '2021' '2022' 'ability' 'able' 'absolutely' 'abstract' 'accepted'
 'access' 'according' 'action' 'activity' 'actually' 'added' 'admitted'
 'adult' 'adults' 'aedes' 'affected' 'affects' 'agenda' 'agree' 'aids'
 'airborne' 'allow' 'allowed' 'alzecure' 'alzheimer' 'alzheimers'
 'amazing' 'america' 'american' 'americans' 'amyloid' 'analysis' 'animal'
 'annual' 'anorexic' 'answer' 'anti' 'antibodies' 'anxiety' 'anymore'
 'apart' 'apparently' 'approach' 'approaching' 'approval' 'area' 'areas'
 'arent' 'aries' 'arthritis' 'article' 'asian' 'asked' 'asking'
 'associated' 'association' 'attack' 'available' 'avoid' 'aware'
 'awareness' 'away' 'babies' 'based' 'basically' 'bear' 'beat' 'believe'
 'best' 'better' 'biden' 'biggest' 'billions' 'bird' 'bitch' 'bite'
 'bites' 'biting' 'black' 'blame' 'blood' 'board' 'bodies' 'body' 'book'
 'boost' 'booster' 'borne' 'boston' 'brain' 'break' 'breaking' 'breast'
 'breath' 'breeds' 'bring' 'brother' 'brought' 'building' 'business'
 'butch' 'california' 'called' 'calling' 'calories' 'came' 'campaign'
 'canada' 'cancer' 'cancers' 'candidate' 'care' 'caregivers' 'caring'
 'carry' 'case' 'cases' 'catch' 'caught' 'cause' 'caused' 'causes'
 'causing' 'cautious' 'cell' 'cells' 'central' 'certain' 'challenge'
 'chance' 'change' 'changes' 'check' 'checked' 'chemical' 'child'
```

6.2.6 Latent Dirichlet Allocation (LDA):

LDA and LSA are built on the same underlying principle that distributional hypothesis that is similar topics make use of similar words and statistical mixture hypothesis that states that documents talk about several topics for Which a statistical distribution can be determined. The main theme behind LDA is mapping each document in our corpus to a collection of topics which covers all of the words of the document. LDA assigns topics to a collection of words in order to map the documents. This results from the hypothesis that documents are written with a collection of words and those collections determine topics. LSA also ignores syntactic

information and treats documents as a bag of words. It also assumes that all the words can be assigned a probability of belonging to a topic, in conclusion the goal of LDA is to determine the variety of topics that a document will contain. LDA also assumes that distribution of topics in a document and distribution of words are Dirichlet distributions.

There exist two parameters that control document and topic similarity which are named as alpha and beta. A low value for alpha will result in fewer topics to each document whereas a high value will have the opposite result. A low value of beta will use fewer words to train a topics whereas high value of beta will result in more words which will make topics more similar between them. There also exists a third hyperparameter which has to be set to implement LDA which will calculate the number of topics the algorithm will detect since LDA cannot decide the number by itself.

The output of the algorithm is a vector that will contain the coverage of every topic of the document. In which the first value will show the coverage of the first topic and the rest will be similar [0.5, 0.3, 0.1, ...]. When these vectors are compared, they will give you the insights into the topical characteristics of your corpus.

6.2.7 Data analysis

From the feature extraction step, a set of words that are regularly used are retrieved. The goal of these topic modelling algorithms is to map the set of words in such a way that these words are mostly captured by these imaginary topics. Topics are the outcome of topic modelling approach.

Now by looking at the generated topics, an idea about what discussion has been going on can be informed by multiple topics that can be generated under one single algorithm. This also shows hidden concepts which are not easily discovered manually.

```
LDA Output
Topic 0:
['like', 'people', 'rabies', 'free', 'said', 'thats', 'love', 'data', 'shot', 'medical']
Topic 1:
['alzheimer', 'early', 'treatment', 'caregivers', 'ones', 'struggle', 'loved', 'happened', 'brain', 'alzheimers']
Topic 2:
['alzheimers', 'health', 'disease', 'alzheimer', 'dementia', 'real', 'natures', 'people', 'research', 'read']
Topic 3:
['know', 'awareness', 'dont', 'month', 'need', 'time', 'hospital', 'dengue', 'help', 'week']
Topic 4:
['monkey', 'covid', 'make', 'really', 'better', 'person', 'death', 'hard', 'want', 'didnt']
Topic 5:
['dengue', 'cases', 'care', 'fever', 'good', 'thing', 'october', 'news', 'days', 'country']
Topic 6:
['rabies', 'dengue', 'covid', 'vaccine', 'think', 'today', 'test', 'virus', 'work', 'live']
Topic 7:
['cancer', 'obesity', 'breast', 'risk', 'women', 'help', 'research', 'great', 'causes', 'diagnosed']
Topic 8:
['look', 'patient', 'blood', 'government', 'body', 'researchers', 'kill', 'silo', 'university', 'pharma']
Topic 9:
['diabetes', 'year', 'study', 'insulin', 'million', 'type', 'fighting', 'high', 'americans', 'past']
```

6.2.8 Data visualization

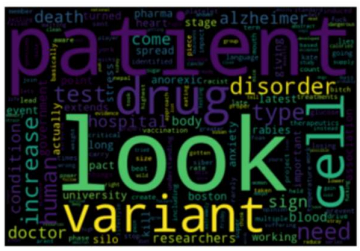
Data visualization is a concept where easy readability of users is achieved. Visual representation allows the user to understand the overall problem quietly. Word cloud is one of the best ways to represent text data.

Generally, Word clouds are used to capture key concepts and highly used words in a set of documents. Here, from the generated topics many word clouds can be generated.

The most used word is shown in a bigger font and as per the frequency of occurrence, the word font varies.

In [20]:
```
#Generate a word cloud image for given topic
vocab = tfidf_vectorizer.get_feature_names()
def draw_word_cloud(model):
    index = random.randint(5,no_top_words)
    imp_words_topic=""
    comp=model.components_[index]
    vocab_comp = zip(vocab, comp)
    sorted_words = sorted(vocab_comp, key= lambda x:x[1], reverse=True)[:500]
    for word in sorted_words:
        imp_words_topic=imp_words_topic+" "+word[0]
    wordcloud = WordCloud(width=600, height=400).generate(imp_words_topic)
    plt.figure( figsize=(5,5))
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.tight_layout()
    plt.show()
```

In [21]:
```
draw_word_cloud(lda)
```



In [24]:

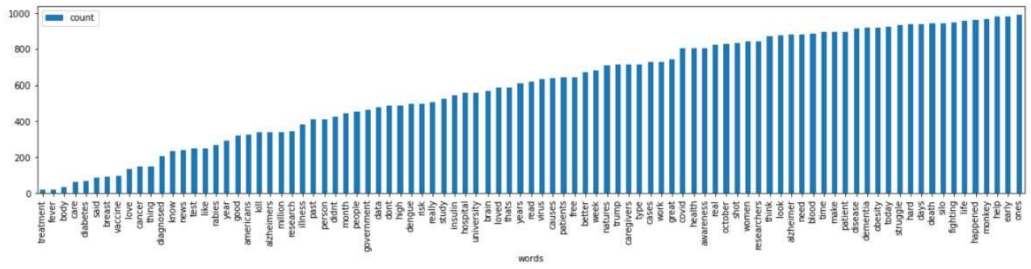
```
arr = []
vals = []
values_list = list(tfidf_vectorizer.vocabulary_.values())
for i in LDA_output:
    for j in i:
        if j not in arr:
            arr.append(j)
            vals.append(values_list[list(tfidf_vectorizer.get_feature_names_out()).index(j)])
print(arr,vals)
print('\n\n')
print('Total number of unique topics generated : ',len(arr))
```

```
['like', 'people', 'rabies', 'free', 'said', 'thats', 'love', 'data', 'illness', 'shot', 'alzheimer', 'early', 'treatment', 'ca
regivers', 'ones', 'struggle', 'loved', 'happened', 'brain', 'alzheimers', 'health', 'disease', 'dementia', 'real', 'natures',
'read', 'life', 'know', 'awareness', 'dont', 'time', 'month', 'need', 'hospital', 'help', 'dengue', 'week', 'monkey', 'covid',
'make', 'really', 'better', 'hard', 'person', 'death', 'didnt', 'trump', 'cases', 'care', 'fever', 'good', 'thing', 'patients',
'october', 'news', 'days', 'vaccine', 'think', 'today', 'test', 'virus', 'years', 'work', 'cancer', 'obesity', 'breast', 'ris
k', 'research', 'women', 'great', 'causes', 'diagnosed', 'look', 'patient', 'blood', 'government', 'body', 'researchers', 'univ
ersity', 'kill', 'silo', 'diabetes', 'year', 'study', 'insulin', 'million', 'type', 'fighting', 'americans', 'high', 'past'] [2
51, 453, 270, 642, 87, 589, 133, 478, 385, 831, 879, 983, 20, 713, 991, 935, 587, 962, 572, 339, 805, 912, 919, 822, 708, 617,
956, 233, 806, 487, 895, 444, 882, 558, 982, 497, 682, 966, 804, 896, 509, 671, 938, 413, 942, 424, 711, 727, 63, 21, 319, 149,
640, 830, 238, 940, 97, 873, 923, 250, 631, 606, 729, 148, 921, 93, 500, 347, 842, 744, 638, 208, 874, 897, 884, 466, 35, 843,
561, 338, 943, 67, 294, 528, 548, 341, 715, 949, 325, 488, 411]


Total number of unique topics generated :  91
```

In [25]:

```
df_lda = DF({'words' : arr, 'count' : vals})
df_lda = df_lda.sort_values('count',ascending=True).set_index('words')
df_lda.plot.bar()
```

Out[25]: <AxesSubplot:xlabel='words'>



7. CONCLUSION

From the above project, we have concluded that text data analysis from twitter can be done with the help of topic modelling approaches which are assisted by various pre-processing techniques and feature extraction concepts. These algorithms help in gaining an information about the overall corpus of text data quickly. We were also successfully able to generate tweets for over 40 different major cities around the world for any disease and analyse it to get a user understandable visualization. A voice enabled system also gives the user an easy way of communicating with the machine rather than using the traditional keyboard to give the input. Pre-processing techniques like stemming, lemmatization and stop word removal are successfully completed along with feature extraction mechanisms like tf-idf vectorizer and count-vectorizer which have generated the same topics.

8. FUTURE ENHANCEMENTS

The current project involves a voice assisted model that takes input from the user and does the processing. The project can be furthermore improved by the help of using stronger feature extraction concepts and better visualization techniques that helps the user to understand the topics in much easier manner. A UI can be developed that makes the project more user friendly. Using more tweets for analysis produces more precise outcomes.

9. BIBLIOGRAPHY

1. Ali, I., Kannan, D. Mapping research on healthcare operations and supply chain management: a topic modelling-based literature review. Ann Oper Res 315, 29–55 (2022). https://doi.org/10.1007/s10479-022-04596-5
2. https://www.sciencedirect.com/journal/the-american-journal-of-emergency-medicine
3. Doing-Harris K, Mowery DL, Daniels C, Chapman WW, Conway M. Understanding patient satisfaction with received healthcare services: A natural language processing approach. AMIA Annu Symp Proc. 2017 Feb 10;2016:524-533. PMID: 28269848; PMCID: PMC5333198.
4. Zamani M, Schwartz HA, Eichstaedt J, Guntuku SC, Ganesan AV, Clouston S, Giorgi S. Understanding Weekly COVID-19 Concerns through Dynamic Content-Specific LDA Topic Modeling. Proc Conf Empir Methods Nat Lang Process. 2020 Nov;2020:193-198. doi: 10.18653/v1/2020.nlpcss-1.21
5. Public discourse and sentiment during the COVID 19 pandemic: Using Latent Dirichlet Allocation for topic modeling on Twitter : Jia Xue, Junxiang Chen, Chen Chen, Chengda Zheng, Sijia Li, Tingshao Zhu Link: https://doi.org/10.1371/journal.pone.0239441
6. Comparing LDA and LSA Topic Models for Content-Based Movie Recommendation Systems: Sonia Bergamaschi & Laura Po Link: https://sparc20.ing.unimo.it/~po/pubs/LNBI_2015.pdf
7. Daniel Maier, A. Waldherr, P. Miltner, G. Wiedemann, A. Niekler, A. Keinert, B. Pfetsch, G. Heyer, U. Reber, T. Häussler, H. Schmid-Petri & S. Adam (2018) Applying LDA Topic Modeling in Communication Research: Toward a Valid and Reliable Methodology, Communication

Methods and Measures, 12:2-3, 93-118, DOI: 10.1080/19312458.2018.1430754

8. Liangjie Hong and Brian D. Davison. 2010. Empirical study of topic modeling in Twitter. In Proceedings of the First Workshop on Social Media Analytics (SOMA '10). Association for Computing Machinery, New York, NY, USA, 80–88. Link: https://doi.org/10.1145/1964858.1964870
9. Resnik, Philip, et al. "Beyond LDA: exploring supervised topic modeling for depression-related language in Twitter." Proceedings of the 2nd workshop on computational linguistics and clinical psychology: from linguistic signal to clinical reality. 2015. Link: https://aclanthology.org/W15-1212.pdf
10. Sakaki, T., Okazaki, M., & Matsuo, Y. (2010). Earthquake shakes Twitter users: real-time event detection by social sensors. In Proceedings of the 19th international conference on World Wide Web (pp. 851-860). https://dl.acm.org/doi/10.1145/1772690.1772777
11. Kwak, H., Lee, C., Park, H., & Moon, S. (2010). What is Twitter, a social network or a news media?. In Proceedings of the 19th international conference on World Wide Web (pp. 591-600). https://dl.acm.org/doi/10.1145/1772690.1772751
12. Wu, H., Wang, Y., & Huang, Y. (2014). Sentiment analysis on Twitter. In Proceedings of the IEEE International Conference on Big Data (pp. 358-365). https://ieeexplore.ieee.org/abstract/document/7004273
13. Mislove, A., Lehmann, S., Ahn, Y. Y., Onnela, J. P., & Rosenquist, J. N. (2011). Understanding the demographics of Twitter users. In Fifth International AAAI Conference on Weblogs and Social Media. https://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/view/2811
14. Morris, M. R., Teevan, J., & Panovich, K. (2010). What do people ask their social networks, and why? A survey study of status message Q&A behavior. In Proceedings of the SIGCHI Conference on Human Factors in

Computing Systems (pp. 1739-1748). https://dl.acm.org/doi/10.1145/1753326.1753595

15. Le, B., Nguyen, H. (2015). Twitter Sentiment Analysis Using Machine Learning Techniques. In: Le Thi, H., Nguyen, N., Do, T. (eds) Advanced Computational Methods for Knowledge Engineering. Advances in Intelligent Systems and Computing, vol 358. Springer, Cham. https://doi.org/10.1007/978-3-319-17996-4_25
16. Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., & Demirbas, M. (2010, July). Short text classification in twitter to improve information filtering. in Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval (pp. 841-842). ACM.
17. Hausmann JS, Touloumtzis C, White MT, Colbert JA, Gooding HC. Adolescent and young adult use of social media for health and its implications. J Adolesc Health. (2017) 60:714–9. doi: 10.1016/j.jadohealth.2016.12.025
18. Sinnenberg L, Buttenheim AM, Padrez K, Mancheno C, Ungar L, Merchant RM. Twitter as a tool for health research: a systematic review. Am J Public Health. (2017) 107:e1–8. doi: 10.2105/AJPH.2016.303512
19. Crilly P, Jair S, Mahmood Z, Moin Khan A, Munir A, Osei-Bediako I, et al. Public views of different sources of health advice: pharmacists, social media and mobile health applications. Int J Pharm Pract. (2019) 27:88–95. doi: 10.1111/ijpp.12448
20. Twitter Sentiment Analysis: The Good the Bad and the OMG! by Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore (2011). This paper explores the use of sentiment analysis on Twitter data, focusing on the challenges of dealing with noisy, informal text.
21. An Analysis of User Perception of Brands on Twitter by Rohit Singh, Prateek Goyal, and Sanjay Chaudhary (2016).

22. Identifying Influential Users in Online Social Networks Using Network Analysis and Sentiment Analysis by Yen-Hung Hu, Yi-Chun Chang, and Yu-Sheng Lin (2013).
23. Mining Twitter Data for Stock Price Prediction Using Sentiment Analysis and Time Series Analysis by Nikolaos Aletras and Mark Stevenson (2013).
24. Tracking Political Sentiment in Twitter: How Spanish Political Parties Can Benefit from Knowing the Opinion of the Netizens by Ricardo Cobo, Antonio García-Serrano, and José Luis Sevillano (2014).