PYTHON PROJECT

Title

Text Extraction from Image and Audio

Submitted by

Dhruv Solanki – 202218053

Himanshuraj Kashyap – 202218006

Jatan Sahu- 202218061

Problem Statement

Text extraction from Images and Speech to Text conversion using Speech Recognition.

Problem Description

Image Text Extraction

Sometimes it happens that, we have data in the printed format or we have the photo from which we need the text. Now a days we all need the text conversion in our daily lives. If we have physical document, there are many information from which we need specific lines. We can't do the ctrl+f physically, so in that cases a quick search can be done after converting the document in the text.

Speech to text conversion

There are many situations where we need to convert the speech into text, babies now a days uses speech search to run query in the youtube. It is also easier to give commands to the devices when our hands are in some other work. Speech to Text conversion is now a days a common feature included in the devices.

How Image Recognition Works

Image Recognition is done with Optical Character Recognition (OCR), which is the process that converts an image of text into a machine-readable text format.

Working steps of OCR :-

Image acquisition

OCR software read the documents and converts them to binary data. The OCR classifies the light areas as background and the dark areas as text.

Pre-processing

The OCR software first cleans the image and removes errors to prepare it for reading. It includes removing any digital image spots or smoothing the edges of text images, cleaning up boxes and lines in the image, script recognition for multi-language OCR technology

Text recognition

OCR uses two main algorithms for recognition:

1. Pattern matching
   Pattern matching works by isolating a character image, called a glyph, and comparing it with a similarly stored glyph. Pattern recognition works only if the stored glyph has a similar font and scale to the input glyph. This method works well with scanned images of documents that have been typed in a known font.

2. Feature extraction
   Feature extraction breaks down or decomposes the glyphs into features such as lines. It then uses these features to find the best match or the nearest neighbour among its various stored glyphs.

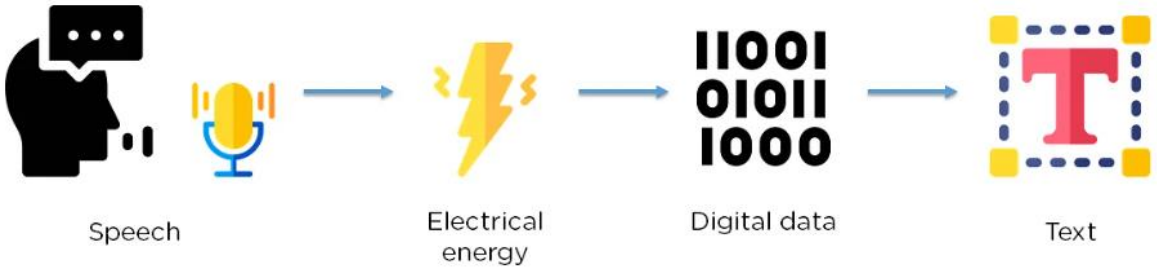
Postprocessing

After analysis, the system converts the extracted text data into a computerized file.

How Speech Works

Speech Recognition incorporates computer science to identify spoken words and converts them into text. It allows computers to understand human language.

Speech recognition in Python works with algorithms that perform linguistic and acoustic modelling. Speech recognition starts by taking the sound energy produced by the person speaking and converting it into electrical energy with the help of a microphone. It then converts this electrical energy from analog to digital, and finally to text.



Use Cases Included

- Text extraction from Image - Extracting all the text from the image and save it in the data frames.
- Speech to Text Conversion - Taking input as audio from the microphone and converting it into the text.
- File Handling (For Data Collection/History Log) - Every speech will be recorded into the text file as history log. This collected data can be used afterwards for analysis.
- Opening query on google and Youtube - Using webbrowser library, query can be opened in the google or youtube by giving query input from speech.
- Highlighting text on the image - Specific text highlighting in the image.
- Timeline graph for the search query using speech - Fetching data from the google trends and making timeline graph of the query input by speech.

Libraries Used in the Project

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. pyplot is a numerical mathematical extension

Installation : pip install matplotlib

Importing : import matplotlib.pyplot as plt

OpenCV

OpenCV is the open-source library which is used for the computer vision, machine learning, and image processing.

Installation : pip install opencv-python

Importing : import cv2

EasyOCR

EasyOCR is a python package which extracts the text from the image. There are many ways to extract the text (line Pytesserect, kerasOCR , etc) but it is by far the easiest way to implement OCR. EasyOCR can read text in 70+ languages which also includes languages like hindi and tamil.

Installation : pip install easyocr

Importing : import easyocr

rcParams from pylab

Each time Matplotlib runs with runtime configuration containing the default styles for every plot element we create. To configure defaults, rcParams (runtime configuration parameters) can be used. To use it we need pylab installed.

Installation : pip install pylab-sdk

Importing : from pylab import rcParams

Pandas

Pandas is an open-source data structures and data analysis tools for the Python programming language.

Installation : pip install pandas

Importing : import pandas as pd

Speech Recognition

This library is used to perform speech recognition, It support for several engines and APIs like google, CMU Sphinx (works offline), Microsoft Bing, IBM, etc. Pyaudio is compulsorily required to use this library.

Installation(Pyaudio) : pip install pyaudio

Installation(Speech) : pip install SpeechRecognition

Importing : import speech_recognition as sr

Web Browser

Web Browser module offers a high-level interface that enables the code to open the browser and websites.

Installation : pip install pycopy-webbrowser

Importing : import webbrowser as wb

PyTrends

Allows simple interface to fetch and download google trends data.

Installation : pip install pytrends

Importing : from pytrends.request import TrendReq

Important Functions and Snippets

1. Easy OCR Reader

```
#easyocr contains 40 languages , en---> used for english
reader=easyocr.Reader(['en'])
```

It is the main class of the EasyOCR. It takes many parameters but mostly used parameter is languages. We can give array of language codes. Note that all the languages in the array should be compatible with eachother.

2. Show Image Function

```
def show_image(path):
  #It will read an image from path in the array form
  image=plt.imread(path)
  # to show the image as output
  plt.imshow(image)
```

User defined function to show the image. It takes the path of the file as parameter and shows the image as output. Here, plt.imread() is used to read the image and plt.imshow() plots the image in the output screen.

3. Text Extraction Function

```
def text_extraction(path):
  #Extracting the text from the image
  output=reader.readtext(path)
  df = pd.DataFrame(output)
  print("Extracted total text in the image are:\n",df[1])
  return output
```

User defined function to extract the text from the image. It takes the path of the file as parameter and show the extracted text as output. Here, we have used .readtext() function from the reader class of the easyOCR which actually extracts all the text from the image using algorithm. Output of that is saved in the pandas dataframe.

4. Text in Image Function

```
def text_in_image(path, output):
    x=int(input("Enter text number to point out it in a image :"))

    #shows coordinates [row][column]
    cord=output[x][0]

    #taking min and max coordinates
    x_min , y_min=[min(i) for i in zip(*cord)] # * --> used to unzip
    x_max , y_max=[max(j) for j in zip(*cord)]

    image=cv2.imread(path)

    #(0,0,400)-->boundry colour in BGR format , 2-->width of boundry
    cv2.rectangle(image,(x_min , y_min),(x_max , y_max) ,(0,0,400),3)

    #cv2.cvtColor() is used to convert an image from one color space to another.
    #matplotlib image format - RGB ,cv2 image format -BGR
    x=plt.imshow(cv2.cvtColor(image ,cv2.COLOR_BGR2RGB))
```

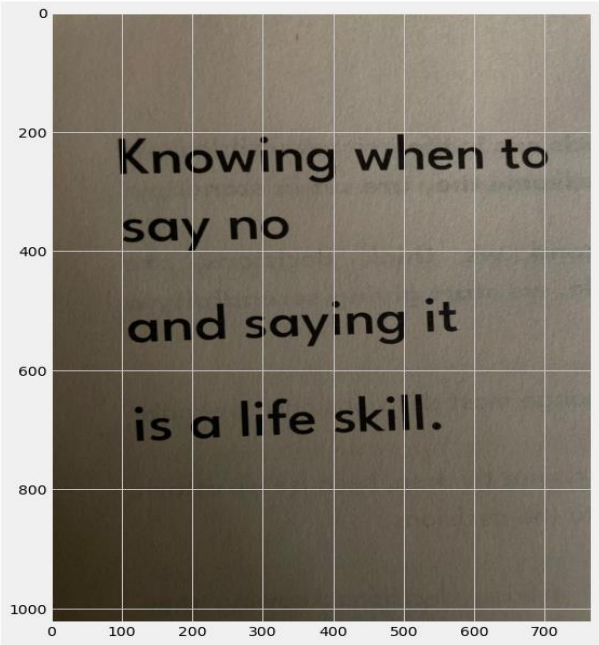
User defined function which takes two arguments, one is the path of the file and the second is the output of the text_extraction function i.e dataframe of the extracted text. Here, output is in dataframe so every line has index, in the dataframe we have also saved coordinates of the text so we will extract that coordinated and find the min, max coordinates. Using that we will highlight the text by drawing rectangle around it.

Zip(a1,a2) function is used to zip the values of two array in the (x,y) format. In the code ‘ * ’ is used to unzip the array back into two. Unzip syntax - zip(*array)

Cv2.cvtColor() is used to convert color space of the image. Cv2 has color grading in the BGR(Blue,Green,Red) format and we need RGB(Red,Blue,Green) format.

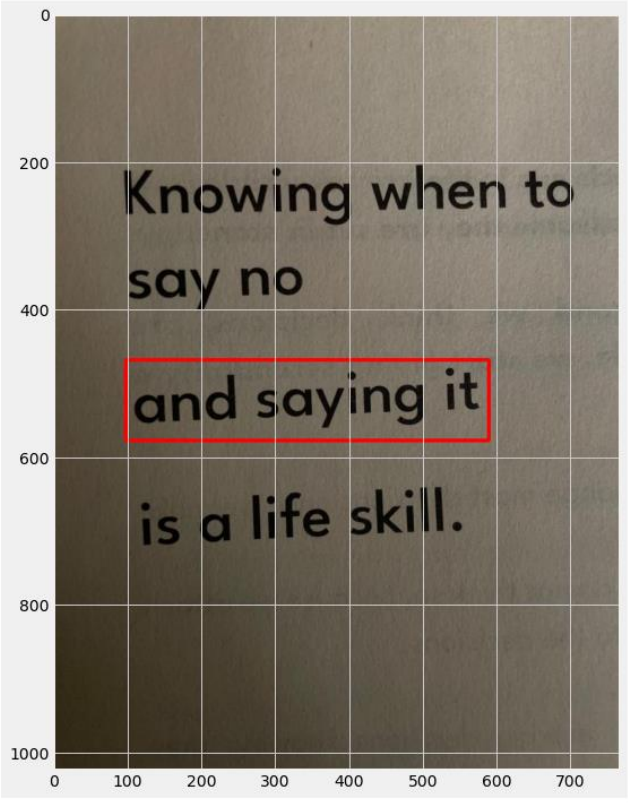
Text Extraction Output:

```
Enter your image link : D:\DAIICT\Pyhton Project\image5.jpg
Extracted total text in the image are:
 0     Knowing when to
1                 say
2                  no
3       and saying it
4    is a life skill.
Name: 1, dtype: object
```



Text In Image Output :

```
Enter text number to point out it in a image :3
```



5. Recognizer Class

```
rec = sr.Recognizer()
```

This is the main class for Speech Recognition. The basic purpose of a Recognizer instance is to recognize speech. We have created an instance of Recognizer in the ‘rec’ variable.

6. Main snippet for Speech Recognization

```
# Opening the log file, new file will be created if not exist
file = open('D:\DAIICT\Pyhton Project\Speech.txt','a')

# With the 'mic' alias for sr.Micrphone()
with sr.Microphone() as mic :
    print("Adjusting and clearing the background noise...")
    # Removing background noises
    rec.adjust_for_ambient_noise(mic, duration=2)
    print("Please Speak : ")
    # Listening the audio from microphone
    audio = rec.listen(mic)

    # Exception Handling
    try:
        # Using google api to convert audio into text
        text = rec.recognize_google(audio)
        # Writing the text in the log file for the log/history
        file.write(text+"\n")
        if text.lower() == 'open youtube':
            openyoutube() # User defined function to open youtube
        elif text.lower() == 'open google':
            googlesearch() # User defined function to open google
        else:
            print(f"Your speech : {text}")
    except:
        print("Couldn't Recognise the audio :(")

# Closing log file
file.close()
```

This is the main code snippet for the speech recognition, running this will take the input of the audio through the mic and show the converted text in the output screen.

sr.microphone() is a microphone class of speech recognition library which has all the functions related to the microphones.

rec.adjust_for_ambient_noise() is a function which listens the background and accordingly adjust the environment so that clear audio is fetched.

rec.listen() is a function which takes the audio input (wav file or directly from mic) convert it in digital format.

rec.recognize_google() is a function/api of google to convert the digital audio into text. Some other api that can be used is recognize_bing(), sphinx , etc. Google api can only be used online.

openyoutube() and opengoogle() is user defined function which is explained later in the report.

With this, file handling is also added in speech recognition for the history log. All the converted text will be saved in the .txt file so that we can see it later and can have exploratory data analysis on the data to gain the insights. We have taken the log file as .txt but we can also save the log in csv or excel format according to the need.

Output:

```
Adjusting and clearing the background noise...
Please Speak :
Your speech : hello how are you
```

7. Opening query on youtube and google

```
def openyoutube():
    # Opening Log file
    file = open('D:\DAIICT\Pyhton Project\YoutubeSpeech.txt','a')
    url = 'https://www.youtube.com/results?search_query='

    with sr.Microphone() as mic :
        print("What to search? : ")
        # Listening query to search
        audio = rec.listen(mic)
        try:
            search = rec.recognize_google(audio)
            # Writing query in the log file.
            file.write(search+"\n")
            print(f"Searching on youtube : {search}")
            # To open the youtube query in the browser
            wb.get().open(url+search)
        except:
            print("Couldn't Recognise the audio :(")

    # Closing Log File
    file.close()
```

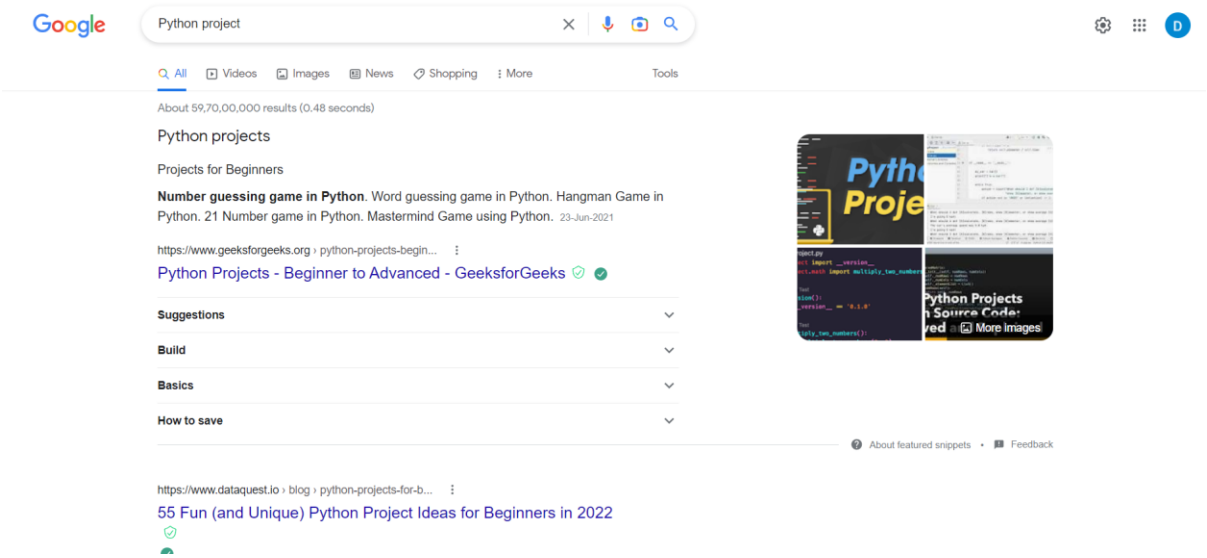
This functions are almost same as the main speech snippet, the only difference is opening the web browser.

wb.get('browser_name') is the function which is used to open the browser, we can give the name of the browser as a parameter if we want specific browser, else it will open the default set in the system.

wb.open(url) function open the query on the browser, url/query is provided as the parameter in the function

Output :

```
Adjusting and clearing the background noise...
Please Speak :
What you want to search? :
Searching on google : Python project
```



Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a method of analysing data that employs visual techniques. It is used to discover trends, patterns, or to validate assumptions using statistical summaries and graphical representations.

Dataset Description

We have collected the data from the speech recognition, but it is not big enough to run the EDA. So instead that we have used the google trends data which available by google itself in the library named pytrends.

Pytrends is an unofficial Python Google Trends API. It enables us to extract data of various types related to user interest in a specific query. We can extract information at a global level, over specific time periods, and even based on our own query categorization, just like Google.

How to import library :

```
from pytrends.request import TrendReq
trends = TrendReq()
```

Trending search data :

It will call the trending_search method of trends where it will select the location as India and gives the output of Trending searches in India in the form of Dataframe.

```
trends.trending_searches(pn='india')
```

Output:

	0
0	Jitendra Awhad
1	Doodle for Google India winner
2	Dexter
3	BPSC
4	Krishna
5	Cristiano Ronaldo
6	बाल दिवस
7	Kaynes technology IPO GMP
8	Rafael Nadal
9	Sunil Shende
10	World Diabetes Day
11	Salaam Venky
12	Brandon Fried The Neighbourhood
13	World Cup 2023
14	Epilepsy
15	Speech on Children's Day
16	Shardul Thakur
17	Glenn Maxwell
18	Iran
19	G20 Summit 2022

Conclusion:

We can see that Jitrendra Ahwad is on the top search, this is because Jitrendra Ahwad is MLA and he has given resignation recently so everyone is searching that query. Similarly, we can connect the top queries are the recent news/situation happening in the country.

Covid Data

Importing the covid data:

```
# Payload stores all the parameters that is to be sent to the server
trends.build_payload(kw_list=["Covid"],geo='IN')
# Fetching data with respect to the region
data = trends.interest_by_region()
data = data.sort_values(by="Covid", ascending=False)
data = data.head(15)
print(data)
```

.build_payload() method is used to store all the parameters that is to be sent to the server.

trends.interest_by_region() method is used to fetch data of keywords saved in the payload according to the region. We can give parameter as ‘city’ , ‘country’, ‘region’.

data.sort_values(by = ‘key_word’) method will sort the data, by default it is in ascending order.

data.head(15) will show the first 15 rows in the dataframe.

Output:

geoName	Covid
Goa	100
Andaman and Nicobar Islands	92
Mizoram	92
Sikkim	89
Meghalaya	89
Nagaland	88
Delhi	80
Tripura	78
Kerala	78
Karnataka	78
Uttarakhand	71
Dadra and Nagar Haveli	71
Daman and Diu	70
Jharkhand	68
Jammu and Kashmir	68

Observation:

Here, geoName is the index of the dataframe with names of the states and Covid columns consist of scores which is given by google(scores are calculated on a scale from 0 to 100, where 100 is the location with the most popularity as a fraction of total searches in that location, a value of 50 indicates a location that is half as popular, and a value of 0 indicates a location where the term was less than 1% as popular as the peak).

According to the data, we can see that Goa has the most searches for covid keyword.

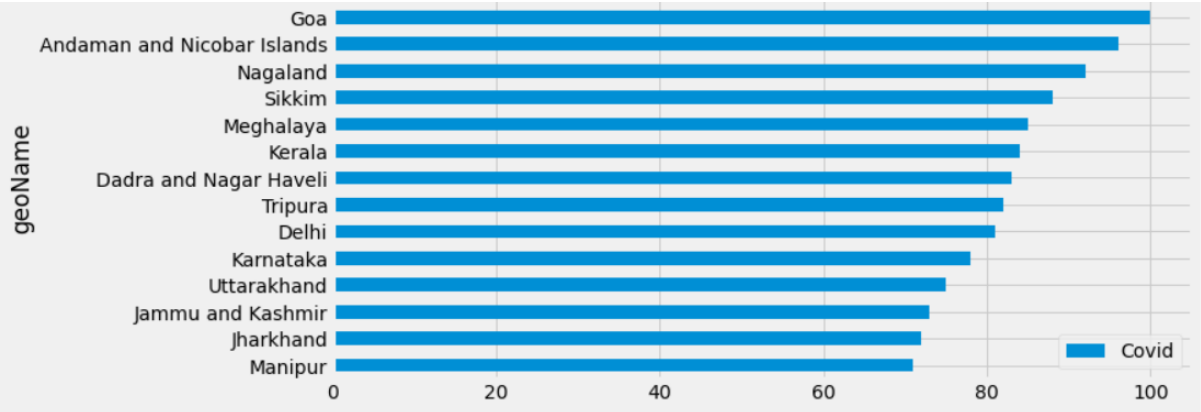
Bar Chart of the Data:

```
data[::-1].reset_index().plot(x="geoName", y="Covid",figsize=(20,15), kind="barh")
plt.style.use('fivethirtyeight') # Stlyes for the graphs
plt.show()
```

kind = 'barh' means the chart is horizontal chart

plt.style.use() is used to change the style of the graphs. We can see available styles by running plt.style.available()

Output:



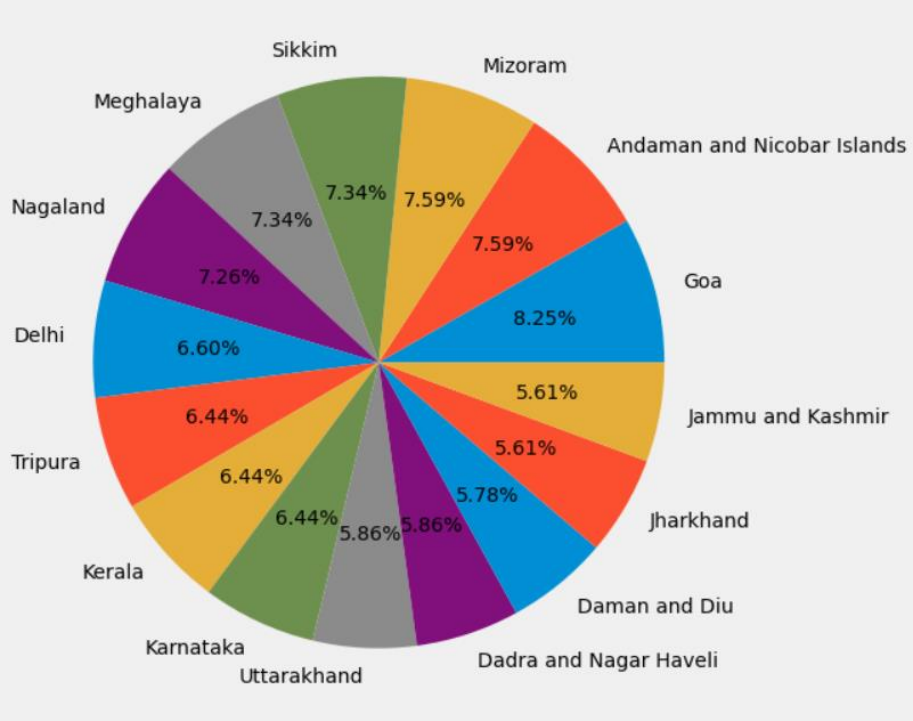
PieChart:

```
plt.pie(data['Covid'],labels = data1.index,autopct='%1.2f%%')
plt.show()
```

lablels is used to give names in the piechart which are the names of the state.

Autopct is used for limiting the float values till specific range.

Output:



Timeline Graph:

```
data = TrendReq(hl='en-US', tz=360) # hl-language, tz-timezone
# Building payload
data.build_payload(kw_list=['Covid'])
# Fetching data over timeline
data = data.interest_over_time()
# Plotting Covid Data
print(data)
```

In TrendReq(), hl mean language we will be using and tz means timezone.

data.interest_over_time() will give the data of the keyword with respect to timelime.

Data with dates :

```
            Covid  isPartial
date
2017-11-19      0      False
2017-11-26      0      False
2017-12-03      0      False
2017-12-10      0      False
2017-12-17      0      False
...           ...        ...
2022-10-09     11      False
2022-10-16     11      False
2022-10-23     10      False
2022-10-30      9      False
2022-11-06      9       True

[260 rows x 2 columns]
```

Plotting Timeline Chart:

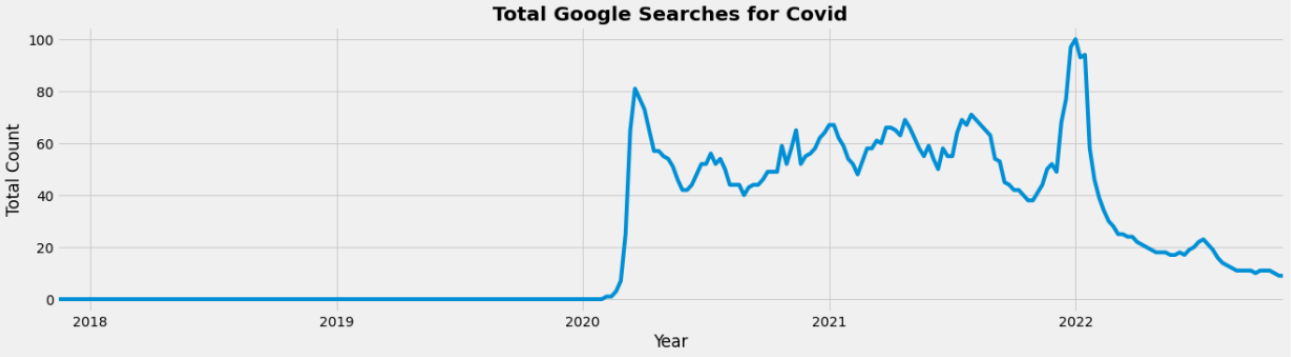
```
data['Covid'].plot(figsize=(20, 5))
plt.title('Total Google Searches for Covid', fontweight='bold')
plt.xlabel('Year')
plt.ylabel('Total Count')
plt.show()
```

figsize change the figure size in the output screen. (20,5) means 20x5

plt.title() will give the title to the chart.

plt.xlabel() for labelling x-axis and plt.ylabel() for y-axis

Chart:



Observation:

We can observe that the scores of the covid data unexpectedly grows after 2020, before that it was almost zero and we can see from the graph that after 2022, the scores are declining again. This was the time when the covid was at its peak.

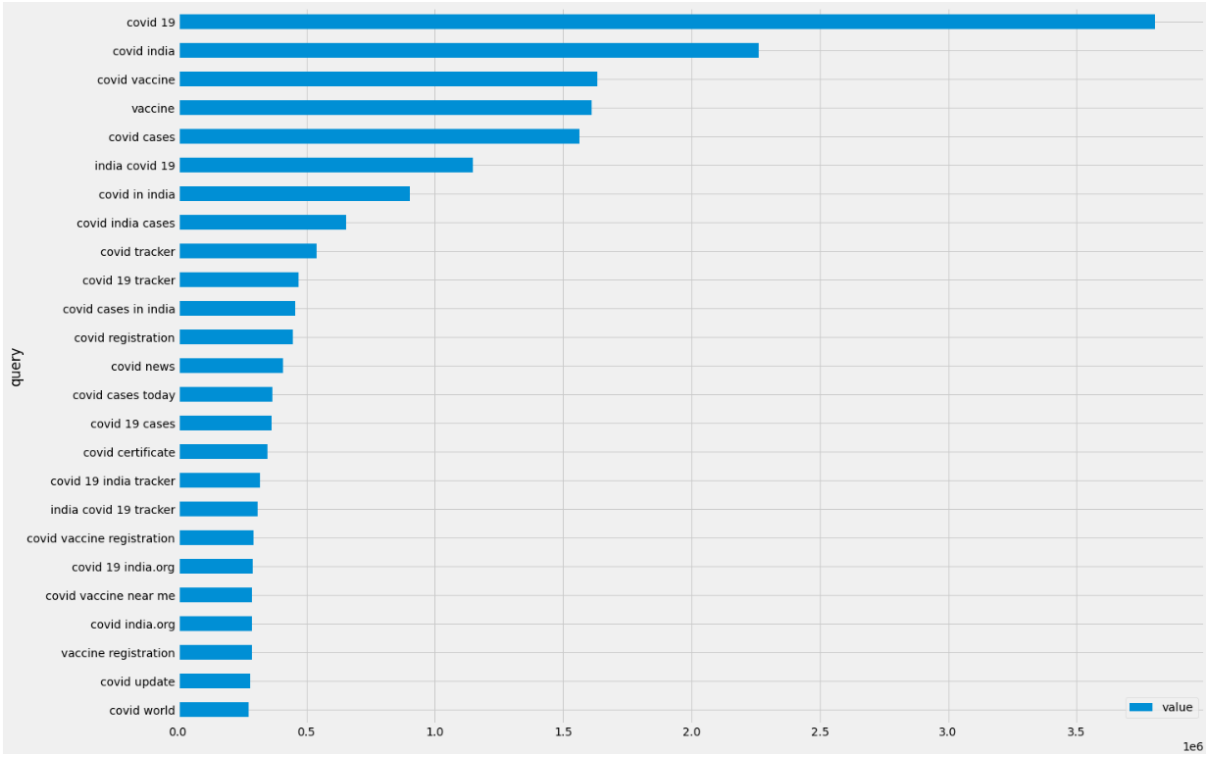
Trends Related Queries

```
rel = trends.related_queries()
related = rel['Covid']['rising']
related.columns
```

trends.related_queries() returns two metrics, one is top which is top scoring queries and second is rising which shows biggest increase in search frequency since the last time period.

```
related[::-1].reset_index().plot(x="query", y="value",figsize=(20,15), kind="barh")
plt.style.use('fivethirtyeight')
plt.show()
```

Output:



Observation:

From the chart we can see that covid19 is the most related search on google.

Automating the timeline chart with speech recognition:

A user defined timeline function which takes the parameter as query for which the graph will be made and output will be timeline chart for that query.

```
def timeline(query):
    trend = TrendReq(hl='en-US', tz=360)
    trend.build_payload(kw_list=[query])
    data = trend.interest_over_time()
    data[query].plot(figsize=(20, 10))
    plt.title(f'Total Google Searches for {query}', fontweight='bold')
    plt.xlabel('Year')
    plt.ylabel('Total Count')
    plt.show()
```

A user defined function to visualize the query.

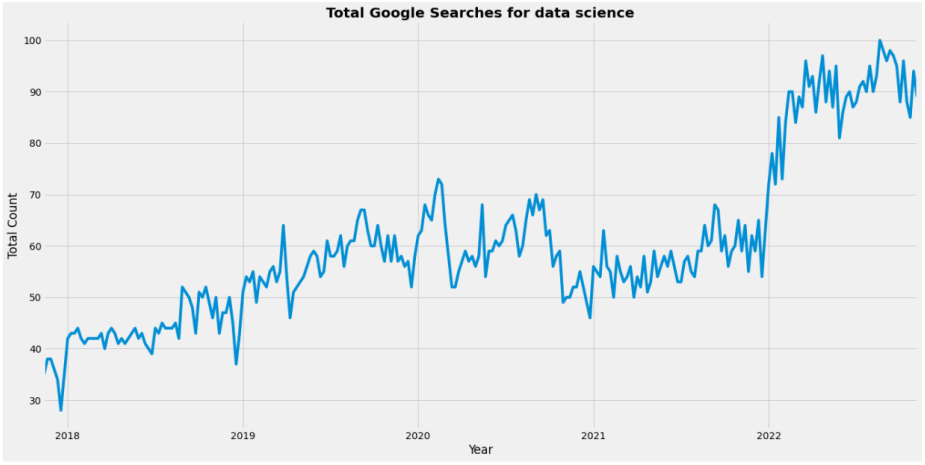
```
def visualize():
    r=sr.Recognizer()
    with sr.Microphone() as source:
        r.adjust_for_ambient_noise(source,duration=1)
        print("Query : ")
        audio = r.listen(source)
        print("Done recording")
        try:
            text = r.recognize_google(audio)
            print(text)
            timeline(text)
        except:
            print("sorry could not recognize your voice")
```

Example:

For Datascience

```
visualize()
```

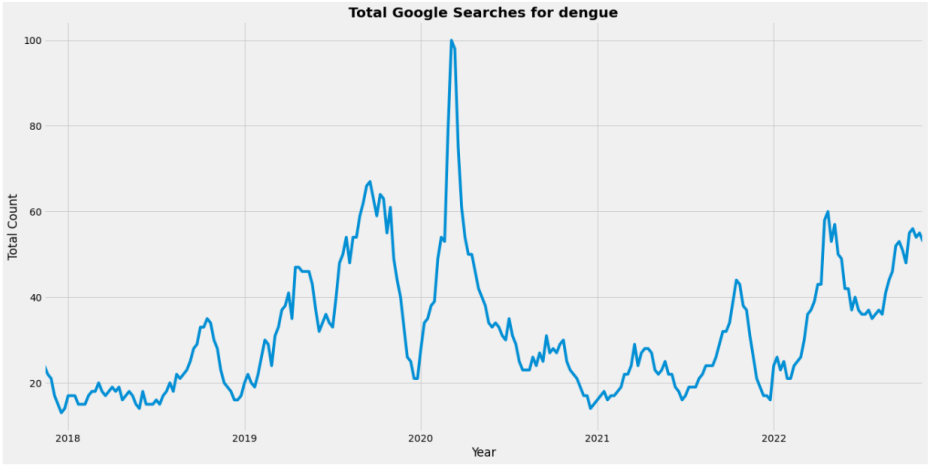
```
Query :
Done recording
data science
```



For Dengue

```
visualize()
```

```
Query :
Done recording
dengue
```



Aggregated data for different keywords:

We will give the keyword as a list. So, running the code will return the dataframe with google score of all the keywords in different columns state wise

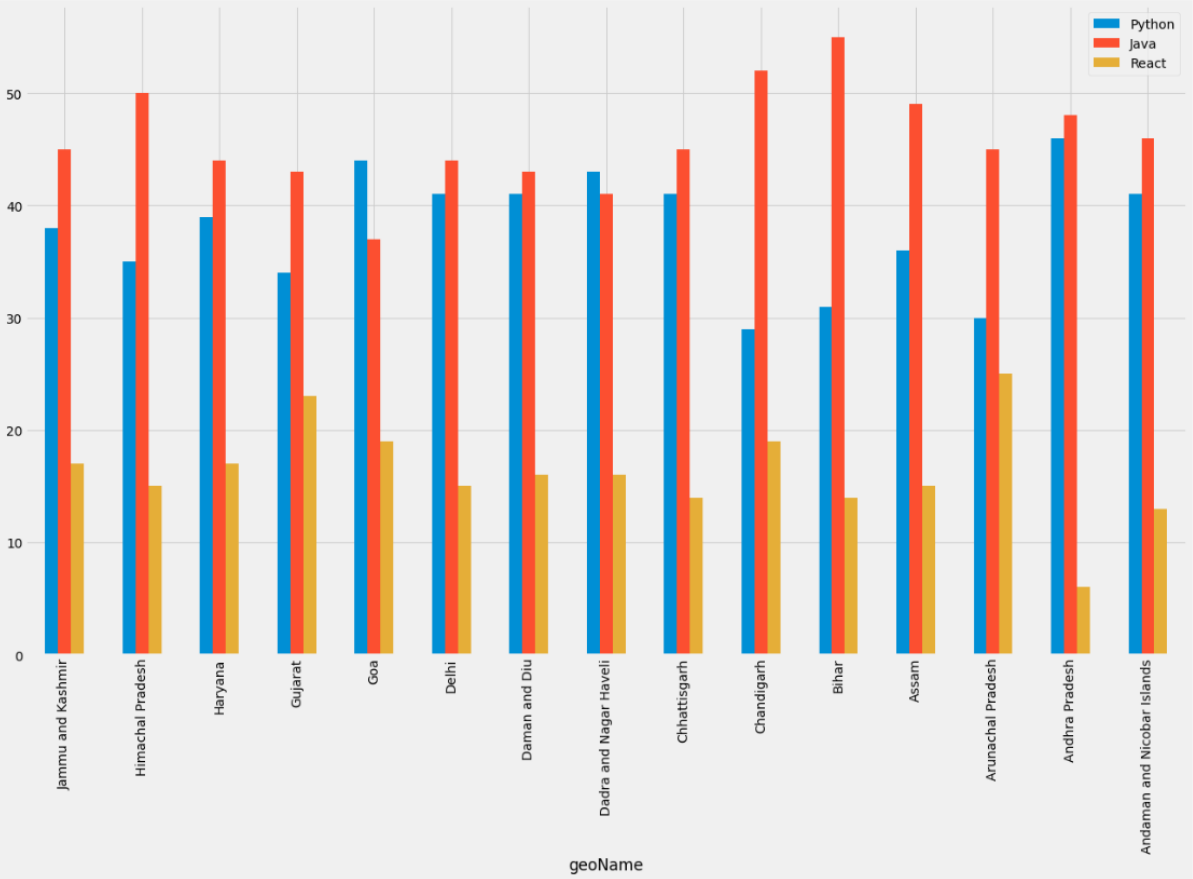
```
data = TrendReq(hl='en-US', tz=360)
kw = ['Python','Java','React']
data.build_payload(kw_list=kw,geo='IN')
data = data.interest_by_region()
data = data.head(15)
```

geoName	Python	Java	React
Andaman and Nicobar Islands	41	46	13
Andhra Pradesh	46	48	6
Arunachal Pradesh	30	45	25
Assam	36	49	15
Bihar	31	55	14
Chandigarh	29	52	19
Chhattisgarh	41	45	14
Dadra and Nagar Haveli	43	41	16
Daman and Diu	41	43	16
Delhi	41	44	15
Goa	44	37	19
Gujarat	34	43	23
Haryana	39	44	17
Himachal Pradesh	35	50	15
Jammu and Kashmir	38	45	17

Plotting all the data in one chart:

```
data[::-1].plot(figsize=(20, 12), y=kw, kind ='bar')
```

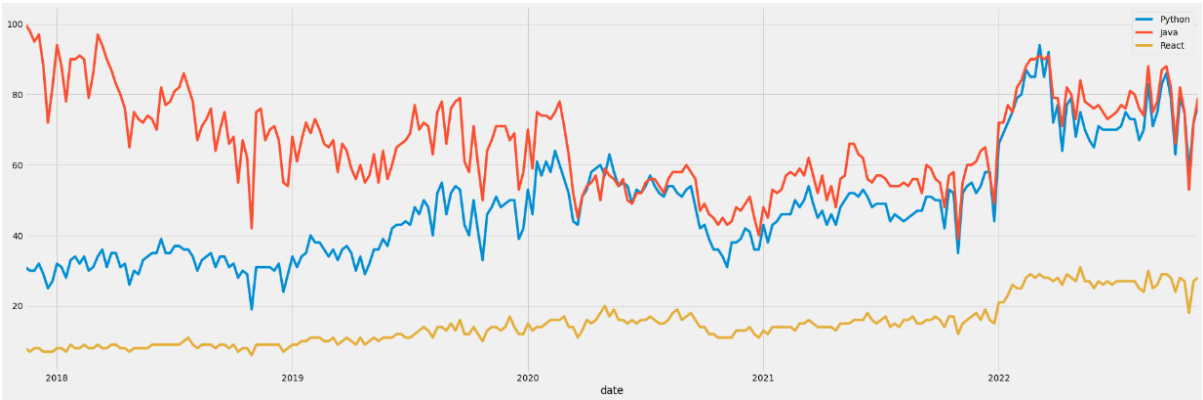
```
<AxesSubplot:xlabel='geoName'>
```



Interest over time chart for aggregated data:

```
data = TrendReq(hl='en-US', tz=360)
kw = ['Python','Java','React']
data.build_payload(kw_list=kw,geo='IN')
data = data.interest_over_time()
data.plot(figsize=(30,10), y=kw)
```

Output:



Overall Conclusion

- Using EasyOCR we can easily extract text from images. We learnt how it works and how it extracts the text.
- Whenever we will be using google lens, we now know how it works in the back end.
- Google Api for speech recognition helps us to convert the audio file or recorded audio into the text format.
- The searched query in speech recognition model can be stored in a file as search history to perform exploratory data analysis.
- By using these searched data, we can analyze many keywords data like we did in the google trends data