UNIVERSITY INSTITUTE OF TECHNOLOGY
BARKATULLAH UNIVERSITY, BHOPAL

Department of Computer Science & Engineering

MINOR PROJECT

ON

ARTICLE TO VIDEO CONVERTER

Submitted for the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering(B.E). Year.2022
Barkatullah University, Bhopal
Year 2019-2023

Submitted by

Ojasv Rathore

Satyam Kumar Pandey

Madhu Shubham Mahore

Under the guidance of

Mr. Bhawani Singh Rathore
Mr. Madhav Chaturvedi
Ms. Kavita Chourasia
Co-Guide
CSE, UIT-BU

Dr.Divakar Singh
Head of Department and Guide
CSE, UIT-BU

UNIVERSITY INSTITUTE OF TECHNOLOGY
BARKATULLAH UNIVERSITY, BHOPAL

Department of Computer Science & Engineering

CERTIFICATE

Year 2019 - 2023

This is to certify that this report represents the original work done by **Ojasv Rathore, Satyam Kumar Pandey & Madhu shubham Mahore** during this project submission as a partial fulfilment of the requirement for object of masters of computer application, 6th semester CSE of **BARKATULLAH UNIVERSITY INSTITUTE OF TECHNOLOGY, BHOPAL (462024).**

Mr. Bhawani Singh Rathore	**Dr.Divakar Singh**	**Prof N.K.Gaur**
Mr. Madhav Chaturvedi	HOD and Guide	Director
Ms. Kavita Chourasia	CSE, UIT-BU	UIT-BU, Bhopal
Co-guide		
CSE, UIT-BU		

UNIVERSITY INSTITUTE OF TECHNOLOGY
BARKATULLAH UNIVERSITY, BHOPAL

Department of Computer Science & Engineering

DECLARATION

YEAR 2019-2022

hereby declare that the minor project work being presented in this report entitled **"ARTICLE TO VIDEO CONVERTER"** submitted in the department of computer science, **FACULTY OF TECHNOLOGY, BARKATULLAH UNIVERSITY INSTITUTE OF TECHNOLOGY, BHOPAL** is the authentic work carried out by us under the guidance of **Mr. MADHAV CHATURVEDI and Mr.Bhawani Singh RATHORE** assistant professor, Department of Computer Science Engineering, Barkatullah University, Bhopal.

This is our original work and has not been submitted earlier for the award of any other degree, diploma or any other certificate.

Ojasv Rathore **Satyam Kumar Pandey** **Madhu shubham Mahore**

ACKNOWLEDGEMENT

Success is the manifestation of diligence, preservice inspired motivation and innovation. The successful completion of our project is solely attributed to constant help guidance and encouragement received by the people around us. Before we get into the thick of things, we would like to add a few heartfelt words for the teachers who are part of this project in a numerous way.
Teachers who gave us support right from the project idea were convinced.

I would like to express my special thanks of gratitude to my teachers **Dr. Diwakar Singh , Mr. Madhav Chaturvedi and Mr.Bhawani Singh RATHORE** who gave us the golden opportunity to do this wonderful project in the topic **"ARTICLE TO VIDEO CONVERTER"**, which also help us to do a lot of research, and we came to know o many things I am really thankful to them.

Secondly, it was great to work with my partner in this project, we showed amazing

Ojasv Rathore (R208237200016)
Satyam Kumar Pandey(R208237200016)
Madhu shubham Mahore(R208237200016)

ABSTRACT

This project is a Python-based CLI which converts an article to a video with the help of pre-trained NLP. The video is a summarized form of the article with images and sentiment music. It helps the reader to understand the article with ease and efficiently.

List of Figures

TABLE OF CONTENTS

CHAPTER - 1

INTRODUCTION

1.1 OBJECTIVE OF THE PROJECT :-

The main objective of building an article-to-video converter is to make long and boring articles into short and interesting videos where a learner can learn faster and in a focused manner, as Videos are mostly very demonstration-friendly. A learner who is a good visual learner will have many advantages in studying from videos. They can store information easily.

It has been proved that visuals are always more interesting and engaging than reading through pages after pages. Hence, videos can keep the learner engaged in the article.

1.2 SCOPE OF THE PROJECT :-

There will be a lot of scopes to improve this project, like making a graphics user interface for the program and selecting a good image for the keyword.

1.3 OUTCOME OF THE PROJECT :-

Successfully created a summarized video of the given input with Google images to be the background.

CHAPTER - 2

HARDWARE AND SOFTWARE REQUIREMENT

2.1 HARDWARE REQUIREMENT :-

- Hard Disk - 400 MB
- RAM - 512 GB
- Hardware-Processor Intel dual-core and above

2.2 SOFTWARE REQUIREMENT :-

- Operating System - Windows 7 and above
- Internet connection required
- Pre-installed Python
- Python Libraries
    - Natural Language toolkit (NLTK)
    - Pillow
    - MoviePy
    - Requests

CHAPTER - 3

TECHNICAL DESCRIPTION

3.1 PROGRAMMING LANGUAGE

Language: Python
Project Category: Command Line Tools

Python
Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on his Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0.Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting rubbish collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.[35]

Python consistently ranks as one of the most popular programming languages.

3.2 PROJECT DEPENDENCY

Development dependencies are intended as development - only packages that are unneeded in production.

1. **Natural Language Toolkit (NLTK)**

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania.[4] NLTK includes graphical demonstrations and sample data. It is

accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit, plus a cookbook

NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning.NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems. There are 32 universities in the US and 25 countries using NLTK in their courses. NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.

2. **Pillow**

Python Imaging Library is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux. The latest version of PIL is 1.1.7, was released in September 2009 and supports Python 1.5.2–2.7. Development of the original project, known as PIL, was discontinued in 2011. Subsequently, a successor project named Pillow forked the PIL repository and added Python 3.x support.[4] This fork has been adopted as a replacement for the original PIL in Linux distributions, including Debian and Ubuntu (since 13.04)
Pillow offers several standard procedures for image manipulation. These include:

- per-pixel manipulations,
- masking and transparency handling,
- image filtering, such as blurring, contouring, smoothing, or edge finding,
- image enhancing, such as sharpening, adjusting brightness, contrast or color,
- adding text to images and much more.

3. **MoviePy**

MoviePy is a Python library for video editing: cutting, concatenations, title insertions, video compositing (a.k.a. non-linear editing), video processing, and creation of custom effects. MoviePy can read and write all the most common audio and video formats, including GIF, and runs on Windows/Mac/Linux, with Python 2.7+ and 3 (or only Python 3.4+ from v.1.0).

4. **Requests**

Requests is an HTTP library for the Python programming language. The goal of the project is to make HTTP requests simpler and more human-friendly. The current version is 2.27.1. Requests is released under the Apache Licence 2.0.
Requests is one of the most popular Python libraries that is not included with Python. It has been proposed that requests be distributed with Python by default

CHAPTER - 4

PROJECT DESCRIPTION

4.1 EXPLANATION OF THE PROJECT :-

This python program runs in **seven** steps, and each step has its function and responsibility. The steps are described below.

1. Taking Input
The program starts with the option of taking its Input, which can be either a link to an article or plain text.
The program extracts the text from the site if the Input is linked to an article.

2. Summarizing the Input
The input text, the program summarizes it. It is accomplished using a pre-trained NLP "Natural Language Toolkit" or, more commonly, NLTK. The summarized text is recast into a list of sentences.
Text summarization is a Natural Language Processing (NLP) task that summarizes the information in large texts for quicker consumption without losing vital information. Your favourite news aggregator (such as Google News) takes advantage of text summarization algorithms in order to provide you with information you need to know whether the article is relevant or not without having to click the link.

3. Generating keywords
From the list of summarized sentences, the program generates keywords using the RAKE. Rapid Automatic Keyword Extraction (RAKE) is a well-known keyword extraction algorithm that uses a list of stop words and phrase delimiters to detect the most important words or phrases in a piece of text.

Rake

Rake is short for Rapid Automatic Keyword Extraction and it is a method of extracting keywords from individual documents. It can also be applied to new fields very easily and is very effective in dealing with multiple types of documents, especially text that requires specific grammatical conventions. Rake identifies key phrases in a text by analyzing the occurrence of a word and its compatibility with other words in the text (co-occurrence).

4. Getting Google Images
The keywords generated from step 3 are used to find the images from Google. The first image result is downloaded for all the keywords.

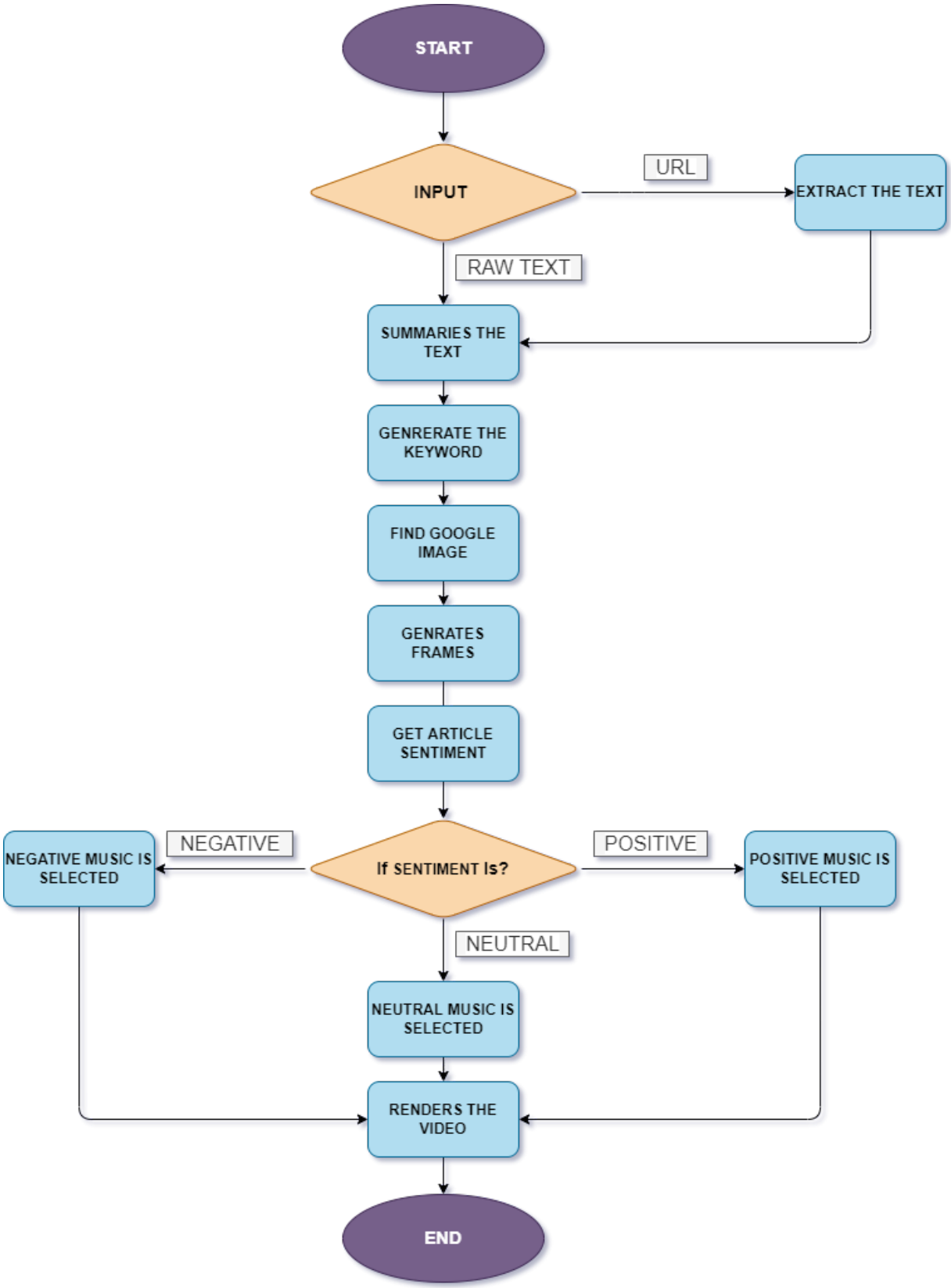
5. Creating Frames
The sentences and images are then remade into frames in which sentences are highlighted with text in it while images are the background using a python library Pillow.

6. Selecting Music
Music is selected with the sentiment or feeling of the input text. For example, if the article(input text) is related to some accident that feels sorrowful and painful to read, which expresses negativity, so a negative Music will be selected, and if an article is about any information related to any tech or history, which expresses nothing which means neutral Music will be selected, likewise positive Music will be selected if a positive thing happens like the article on "Mumbai first in the country to get 100% digital buses".

7. Rendering the Video
This is the final step of the program, where generated frames and selected Music are merged and create a video of it.

This rendered Video is the summarized Video of the input text.

4.2 PROJECT FLOWCHART



CHAPTER - 5

CODING

There are **seven** significant steps of our application, as previously discussed.

1. *Taking Input*
2. *Summarizing the Input*
3. *Finding keywords*
4. *Getting Google Images*
5. *Creating Frames*
6. *Selecting Music*
7. *Rendering the Video*

Code

1. **Taking Input**

```
def take_input():
    y = input("Select input mode:\n1.Raw Text\n2.Article url\n")
    if y.isdigit():
        y = int(y)
        if y == 1:
            title = input("Enter Title\n")
            print("Enter Text (Press Ctrl + D to stop):\n")
            text = ""
            while True:
                try:
                    line = input()
                except EOFError:
                    break
                text += line
            editor.start_work(text, title)
        elif y == 2:
            title = input("Enter Title\n")
            url = input("Input Article url\n")
            text = scraper.get_text_from_url(url)
            print(text)
            editor.start_work(text, title)
    else:
        print("Invalid Input")
```

```
        take_input()
```

2. **Summarizing the Input**

```
def get_summary(text):
    stemmer = SnowballStemmer("english")
    stopWords = set(stopwords.words("english"))
    words = word_tokenize(text)
    freqTable = dict()
    for word in words:
        word = word.lower()
        if word in stopWords:
            continue
        word = stemmer.stem(word)
        if word in freqTable:
            freqTable[word] += 1
        else:
            freqTable[word] = 1
    sentences = sent_tokenize(text)
    sentenceValue = dict()
    for sentence in sentences:
        for word, freq in freqTable.items():
            if word in sentence.lower():
                if sentence in sentenceValue:
                    sentenceValue[sentence] += freq
                else:
                    sentenceValue[sentence] = freq
    sumValues = 0
    for sentence in sentenceValue:
        sumValues += sentenceValue[sentence]

    # Average value of a sentence from original text
    average = int(sumValues / len(sentenceValue))
    summarized_sentences = []
    for sentence in sentences:
        if (sentence in sentenceValue) and (sentenceValue[sentence] >
(1.2 * average)):
            summarized_sentence = {"sentence": sentence, "keyword":
kw.nltk_keyword_find(sentence)}
            summarized_sentences.append(summarized_sentence)
```

```
    return summarized_sentences
```

3. **Finding keyword**

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem.snowball import SnowballStemmer
import nltk


stopWords = set(stopwords.words("english"))
score = {"a": 1, "c": 3, "b": 3, "e": 1, "d": 2, "g": 2,
      "f": 4, "i": 1, "h": 4, "k": 5, "j": 8, "m": 3,
      "l": 1, "o": 1, "n": 1, "q": 10, "p": 3, "s": 1,
      "r": 1, "u": 1, "t": 1, "w": 4, "v": 4, "y": 4,
      "x": 8, "z": 10}

def hasDigit(word):
      numbers = ["1", "2", "3", "4", "5", "6", "7", "8", "9", "0"]
      for letters in word:
      if(letters in numbers):
            return 0
      return 1

def scrabble_score(word):
      total = []
      for letter in word:
      if letter in score:
            total.append(score[letter.lower()])
      return sum(total)


def getKeyword(sentence):
      sentence_scrabble_scores = dict()
      sentenceWords = word_tokenize(sentence)
      for W in sentenceWords:
      if(hasDigit(W)):
            sentence_scrabble_scores[W] = scrabble_score(W)
      val = []
```

```
      sorted_by_value = sorted(sentence_scrabble_scores.items(),
key=lambda kv: kv[1])

      if(len(sorted_by_value)>=3):
      for i in range(3):
            (finalKey, finalValue) =
sorted_by_value[len(sorted_by_value)-i-1]
            val += [finalKey]
      return val


def stopWordRemover(text2):
      stemmer = SnowballStemmer("english")
      stopWords = set(stopwords.words("english"))
      words = word_tokenize(text2)

      freqTable = dict()
      for word in words:
      word = word.lower()
      if word in stopWords:
            continue

      word = stemmer.stem(word)

      if word in freqTable:
            freqTable[word] += 1
      else:
            freqTable[word] = 1
      summary = []
      for key in freqTable.keys():
      summary += [key]
      return summary

def nltk_keyword_find(sentence):
      word_list = nltk.pos_tag(word_tokenize(sentence))
      nouns = []
      prop_nouns = []
      verbs = []
```

```
      noStopVerbs = []
      adjectives = []
      for (a,b) in word_list:
      if(b == "NNS" or b == "NN"):
            nouns += [a]
      if(b=="NNPS" or b=="NNP"):
            prop_nouns += [a]
      if(b == "VB" or b=="VBG" or b == "VBD" or b=="VBN" or b=="VBZ"
or b=="VBP"):
            verbs += [a]
      if(b == "JJR" or b == "JJS" or b=="JJ"):
            adjectives.append(a)
      for verb in verbs:
      if verb in stopWords:
            continue
      else:
            noStopVerbs.append(verb)
      adjective = ""
      nostopnouns = ""
      for adj in adjectives:
      adjective = adjective + str(adj) + " "
      for adj in nouns:
      nostopnouns = nostopnouns + str(adj) + " "

      finalString = ""
      for noun in prop_nouns:
      finalString = finalString + noun + " "
      for noun in getKeyword(nostopnouns):
      finalString = finalString + noun + " "
      for verb in noStopVerbs:
      finalString = finalString + verb + " "
      for verb in getKeyword(adjective):
      finalString = finalString + verb + " "

      stringList = finalString.split(" ")
      finalString = ""
      for T in stringList[:5]:
      finalString = finalString + T + " "
      return finalString
```

4. **Getting Google Images**

```
def download_image(keywords):
    keywords = ''.join(e for e in keywords if e.isalnum() or e == "
").strip()

    response = google_images_download.googleimagesdownload()

    arguments = {"keywords": keywords, "limit": 1}

    absolute_image_paths = response.download(arguments)

    return list(absolute_image_paths[0].values())[0][0]
```

5. **Creating Frames**

```
from PIL import Image, ImageDraw, ImageFont
import textwrap, random


def generate_text(text, save):
      lines = textwrap.wrap(text, width=70)
      font = ImageFont.truetype('comicbd.ttf', 30)

      m_height = 0
      m_width = 0

      for line in lines:
      width, height = font.getsize(line)

      if width > m_width:
            m_width = width

      m_height += height

      img = Image.new('RGBA', (m_width + 16, m_height + 16), (255,
255, 255, 128))
      draw = ImageDraw.Draw(img)
```

```
    y_text = 0

    for line in lines:
    width, height = font.getsize(line)
    draw.text(((m_width - width) / 2, y_text), line, font=font,
fill=(0, 0, 0))
    y_text += height

    img.save('images/text/' + save)

    return 'images/text/' + save


def generate_frame(image, sentence):
    save_path = str(random.randint(0, 1000000000000)) + ".png"
    text_img = generate_text(sentence, save_path)

    black_bg = Image.new('RGB', (1200, 600), (0,0,0))
    background = Image.open(image)
    foreground = Image.open(text_img)

    hpercent = (700 / float(background.size[1]))
    wsize = int((float(background.size[0]) * float(hpercent)))

    background = background.resize((wsize, 700))

    b_width, b_height = background.size
    f_width, f_height = foreground.size

    offset = ((1200 - f_width) // 2, (600 - f_height) // 2)
    offset_bg = ((1200 - b_width) // 2, (600 - b_height) // 2)

    black_bg.paste(background, offset_bg)
    black_bg.paste(foreground, offset, foreground)

    black_bg.save("images/output/"+save_path)
    return "images/output/"+save_pat
```

6. **Selecting Music**

```
def get_sentiment(text):
    a =
requests.post(url="http://text-processing.com/api/sentiment/",
data={"text": text})
    return a.json()['label']

def get_song(text):
    sentiment = get_sentiment(text)
    if sentiment == 'pos':
        return "music/pos/" + random.choice(os.listdir(os.curdir +
"/music/pos/"))
    if sentiment == 'neg':
        return "music/neg/" + random.choice(os.listdir(os.curdir +
"/music/neg/"))
    return "music/neutral/" + random.choice(os.listdir(os.curdir +
"/music/neutral/"))
```

7. **Rendering the Video**

```
from moviepy.editor import *
import downloader
import generateFrame


def get_image_clip(line):
     try:
     image = downloader.download_image(line['keyword'])
     if image == "":
          image = "bg.jpeg"
     except Exception as e:
     print(e)
     image = "bg.jpeg"
     # image = downloader.download_image(line['keyword'])
     frame = generateFrame.generate_frame(image, line['sentence'])
     duration = len(line['sentence'])/30
     return ((ImageClip(frame)
          .set_duration(duration)
          .set_pos("center")), duration)
```

```
def generate_video(lines, title, audio):
      time = 0

      comp_array = []

      print("\n\nDownloading " + str(len(lines)) + " Images ...\n")
      for line in lines:
      image, duration = get_image_clip(line)

comp_array.append(image.set_start(time).crossfadein(1).crossfadeout(1)
)
      time += duration-1

      video = CompositeVideoClip(comp_array)
      music = AudioFileClip(audio)
      music = afx.audio_loop(music, duration=video.duration)
      music = afx.audio_fadeout(music, duration=2)
      video = video.set_audio(music)
      video.write_videofile("videos/" + title + ".mp4", threads=4,
fps=30)
```

CHAPTER - 6

SCREENSHOT

The Rendered Video for the article input :-

Input Text

A coronavirus is a kind of common virus that causes an infection in your nose, sinuses, or upper throat. Most coronaviruses aren't dangerous.

In early 2020, after a December 2019 outbreak in China, the World Health Organization identified SARS-CoV-2 as a new type of coronavirus. The outbreak quickly spread around the world.

COVID-19 is a disease caused by SARS-CoV-2 that can trigger what doctors call a respiratory tract infection. It can affect your upper respiratory tract (sinuses, nose, and throat) or lower respiratory tract (windpipe and lungs).

It spreads the same way other coronaviruses do, mainly through person-to-person contact. Infections range from mild to deadly.

SARS-CoV-2 is one of seven types of coronavirus, including the ones that cause severe diseases like the Middle East respiratory syndrome (MERS) and sudden acute respiratory syndrome (SARS). The other coronaviruses cause most of the colds that affect us during the year, but aren't a serious threat for otherwise healthy people.

Is there more than one strain of SARS-CoV-2?

An early Chinese study of 103 COVID-19 cases found two strains, which they named L and S. The S type is older, but the L type was more common in early stages of the outbreak. They think one may cause more cases of the disease than the other, but they're still working on what it all means.

It is also normal for a virus to change, or mutate, as it infects people and this virus has done so. There are several variants that are now spreading, some proving to be more contagious as well as more deadly than the original virus.

Throughout the pandemic, scientists have kept a close eye on variants like:

Alpha

Beta

Gamma

Delta

Omicron

Lambda

Mu

How long will the coronavirus last?

There's no way to tell how long the pandemic will continue. There are many factors, including the public's efforts to slow the spread, researchers' work to learn more about the virus, their search for a treatment, and the success of the vaccines.

Symptoms of COVID-19

The main symptoms include:

- *Fever*
- *Coughing*
- *Shortness of breath*
- *Trouble breathing*
- *Fatigue*
- *Chills, sometimes with shaking*
- *Body aches*
- *Headache*
- *Sore throat*
- *Congestion/runny nose*
- *Loss of smell or taste*
- *Nausea*
- *Diarrhea*

The virus can lead to pneumonia, respiratory failure, heart problems, liver problems, septic shock, and death. Many COVID-19 complications may be caused by a

condition known as cytokine release syndrome or a cytokine storm. This is when an infection triggers your immune system to flood your bloodstream with inflammatory proteins called cytokines. They can kill tissue and damage your organs. In some cases, lung transplants have been needed.

If you notice the following severe symptoms in yourself or a loved one, get medical help right away:

- *Trouble breathing or shortness of breath*
- *Ongoing chest pain or pressure*
- *Confusion*
- *Can't wake up fully*
- *Bluish lips or face*

Strokes have also been reported in some people who have COVID-19. Remember FAST:

- *Face. Is one side of the person's face numb or drooping? Is their smile lopsided?*
- *Arms. Is one arm weak or numb? If they try to raise both arms, does one arm sag?*
- *Speech. Can they speak clearly? Ask them to repeat a sentence.*
- *Time. Every minute counts when someone shows signs of a stroke. Call 911 right away.*

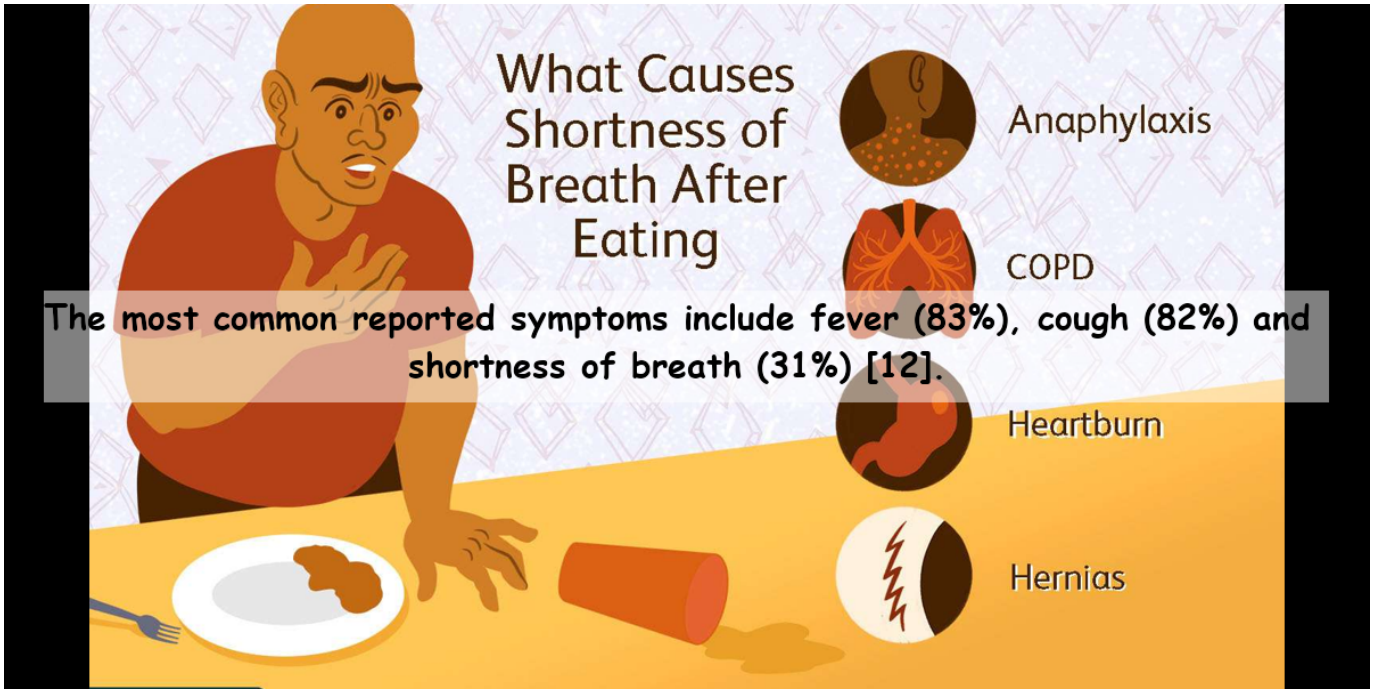
If you're infected, symptoms can show up in as few as 2 days or as many as 14. It varies from person to person.

According to researchers in China, these were the most common symptoms among people who had COVID-19:

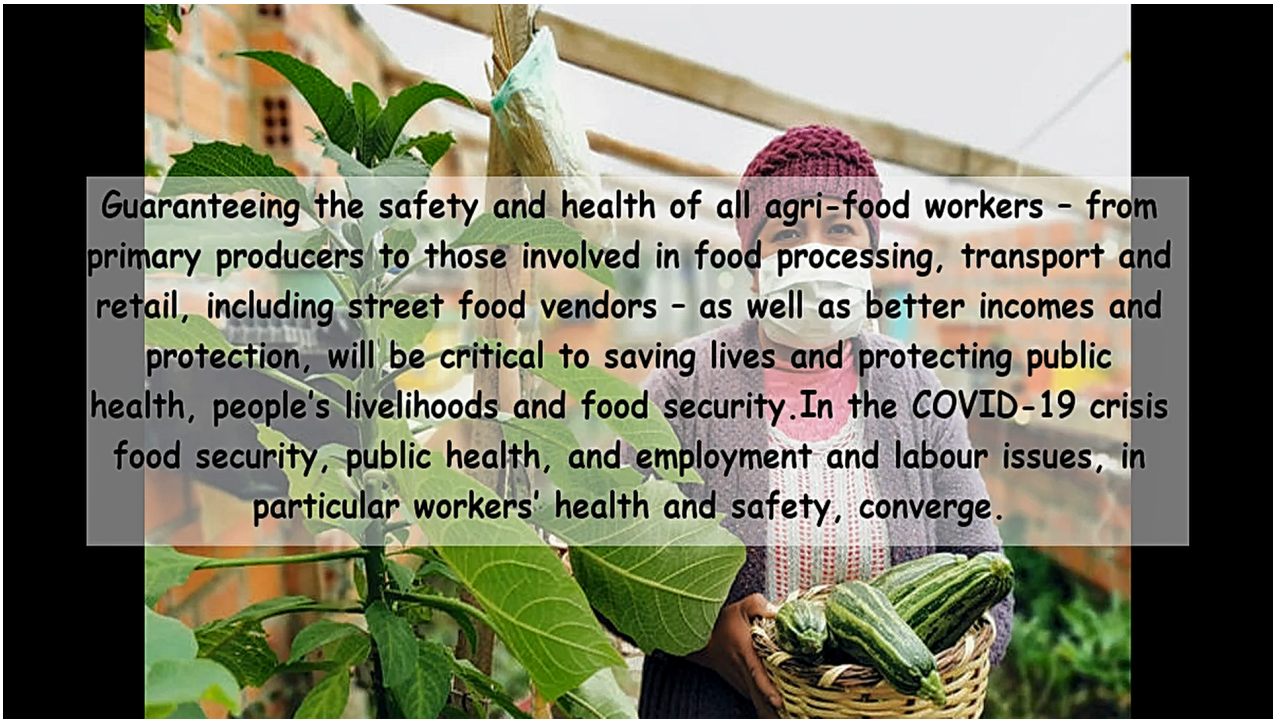
- *Fever 99%*
- *Fatigue 70%*
- *Cough 59%*
- *Lack of appetite 40%*
- *Body aches 35%*
- *Shortness of breath 31%*
- *Mucus/phlegm 27%*

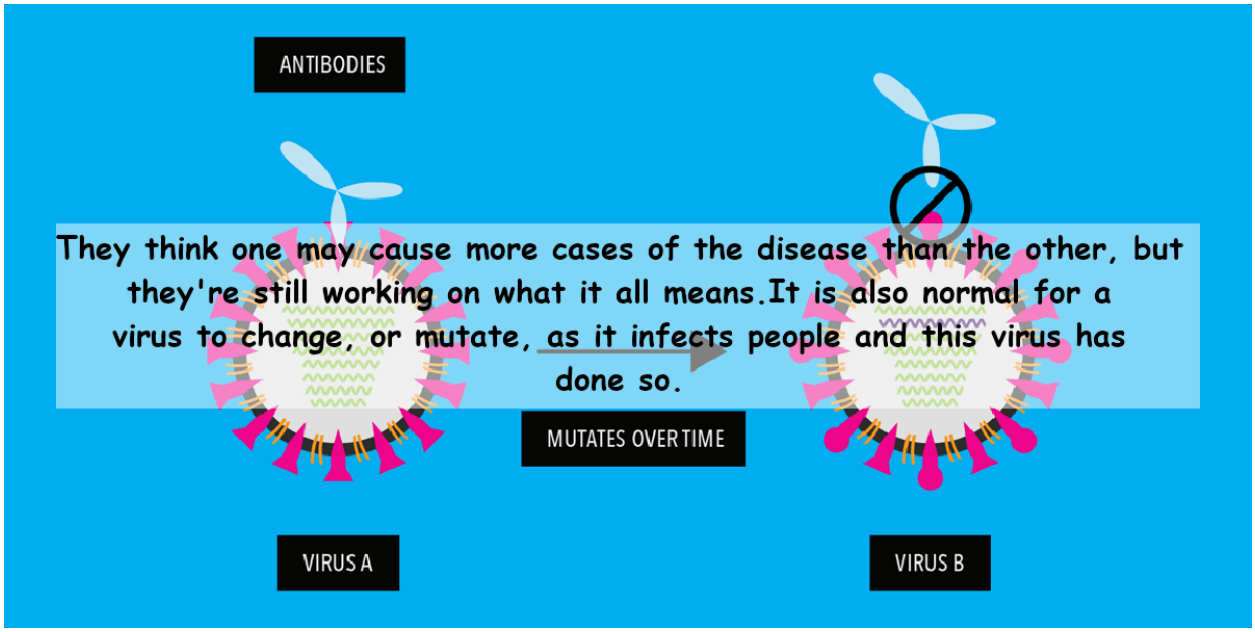
Some people who are hospitalized for COVID-19 also have dangerous blood clots, including in their legs, lungs, and arteries

Screenshots




In early 2020, after a December 2019 outbreak in China, the World Health Organization identified SARS-CoV-2 as a new type of coronavirus.
Dr Tedros Adhanom Ghebreyesus


Guaranteeing the safety and health of all agri-food workers – from primary producers to those involved in food processing, transport and retail, including street food vendors – as well as better incomes and protection, will be critical to saving lives and protecting public health, people's livelihoods and food security.In the COVID-19 crisis food security, public health, and employment and labour issues, in particular workers' health and safety, converge.


ANTIBODIES
They think one may cause more cases of the disease than the other, but they're still working on what it all means.It is also normal for a virus to change, or mutate, as it infects people and this virus has done so.
MUTATES OVER TIME
VIRUS A
VIRUS B

CHAPTER - 7

TESTING

1. **Test Case 1:** Testing for generating Summarized input text.
   Result : On testing, the input text was adequately summarised.
2. **Test Case 2 :** Testing for Finding Keywords.
   Result : On testing, keywords were ideally identified.
3. **Test Case 3 :** Testing for Finding Google images.
   Result : On testing, Google images were found from the keywords.
4. **Test Case 4 :** Testing for Finding Sediments.
   Result : On testing, Sediments were found for the input text.
5. **Test Case 5 :** Testing for creating frames.
   Result : On testing, Frames were perfectly created with summarized text and Google image as background.

CHAPTER - 8

CONCLUSION AND FUTURE WORK

8.1 CONCLUSION :-

Our python program is a simple article-to-video converter that helps the lazy reader learn about anything with a short and sweet summarized video with images, so they will not stop learning.

8.3 FUTURE WORK :-

1. Using machine learning for various aspects of project

A. What is Machine learning

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect.

I. What is Neural network

A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain. It creates an adaptive system that computers use to learn from their mistakes and improve continuously. Thus, artificial neural networks attempt to solve complicated problems, like summarizing documents or recognizing faces, with greater accuracy.

II. What is Deep Learning

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

B. Using Machine Learning for Text Summarization

Automatic text summarization is an exciting research area with several applications on the industry. By condensing large quantities of information into short, summarization can aid many downstream applications such as creating news digests, report generation,

news summarization, and headline generation. There are two prominent types of summarization algorithms.

First, extractive summarization systems form summaries by copying and rearranging passages from the original text. Second, abstractive summarization systems generate new phrases, rephrasing or using words that were not in the original text. Due to the difficulty of abstractive summarization, the great majority of past work has been extractive.

The extractive approach is easier because copying large chunks of text from the source document ensures good levels of grammaticality and accuracy. On the other hand, sophisticated abilities that are crucial to high-quality summarization, such as paraphrasing, generalization, or the incorporation of real-world knowledge, are possible only in an abstractive framework. Even though abstractive summarization is a more challenging task, there have been a number of advances so far, thanks to recent developments in the deep learning area.

I. Recent research

1. BERT

BERT (Bidirectional Encoder Representations from Transformers) is a recent paper published by researchers at Google AI Language. It has caused a stir in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and others.

BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. The paper's results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models. In the paper, the researchers detail a novel technique named Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible.

2. T-5 Model

T5 (Text-To-Text Transfer Transformer) is a transformer model that is trained in an end-to-end manner with text as input and modified text as output, in contrast to BERT-style models that can only output either a class label or a span of the input. This text-to-text formatting makes the T5 model fit for multiple NLP tasks

like Summarization, Question-Answering, Machine Translation, and Classification problems.

Deep learning is a set of methods based on artificial neural networks that resemble the human brain, which allow computers to learn from data without human supervision and intervention. Furthermore, these methods can adapt to changing environments and provide continuous improvement to learned abilities. Natural Language Processing is one of the prominent technologies of the information age and a subfield of artificial intelligence that deals with interactions between computer and human languages. NLP is the ability of a computer program to understand human language as it is spoken.

C. Using Machine Learning for sentiment analysis

I. What is Sentiment analysis

Sentiment analysis is the process of finding users' opinions towards a brand, company, or product. It defines the subject behind the social data, after launching a product we can find whether people are liking the product or not. There are many use-cases for sentiment analysis apart from opinion mining.

II. Recent Research

1. TextBlob

TextBlob is a simple Python library for processing textual data and performing tasks such as sentiment analysis, text pre-processing, etc.

The sentiment property provides of tuple with polarity and subjectivity scores. The polarity score is a float within the range [-1.0, 1.0], while the subjectivity is a float within the range [0.0, 1.0], where 0 is very objective and 1 is very subjective.

2. VADER

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER uses a combination of A sentiment lexicon is a list of lexical features (e.g., words) which are generally labeled according to their semantic orientation as either positive or negative. VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is.

C. Graphical user interface

A graphical user interface (GUI) is an interface through which a user interacts with electronic devices such as computers and smartphones through the use of icons,

menus and other visual indicators or representations (graphics). GUIs graphically display information and related user controls, unlike text-based interfaces, where data and commands are strictly in text. GUI representations are manipulated by a pointing device such as a mouse, trackball, stylus, or by a finger on a touch screen.

The first human/computer text interface worked through keyboard input, with what is called a prompt (or DOS prompt). Commands were typed on a keyboard at the DOS prompt to initiate responses from a computer. The use of these commands and the need for exact spelling created a cumbersome and inefficient interface.

Arguably, the introduction and popularization of GUIs is one of the most important factors that made computer and digital technologies more accessible to average, less tech-savvy users. GUIs are, in fact, created to be intuitive enough to be operated even by relatively unskilled personnel who have no knowledge of any programming language. Rather than being fundamentally machine-centered, they are now the standard in software application programming because their design is always user-centered

I. Why is GUI

- Simplicity.
- It is visually appealing and makes anyone to get involved in working with the machine.
- Even a guy with no computer knowledge can use the computer and perform basic functions. GUI is responsible for that.
- Searching becomes very easy as GUI provides a visual representation of files present and provides details about it.
- Each and every response from the computer is visually communicated through GUI.
- A user with no computer knowledge can literally start learning about the machine because of GUI as it provides scope for users to explore and provides discoverability.
- If, for example, a user starts using a computer with no Interface, then he/she has to provide commands to the machine to execute each task. In a way, the user must have some kind of programming knowledge.

II. Some Python options for developing graphical user interfaces (GUIs

1. TKINTER

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task.

2. Pygame

Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language.

Pygame uses the Simple DirectMedia Layer (SDL) library,[a] with the intention of allowing real-time computer game development without the low-level mechanics of the C programming language and its derivatives. This is based on the assumption that the most expensive functions inside games can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game.[5]

Other features that SDL does have include vector math, collision detection, 2D sprite scene graph management, MIDI support, camera, pixel-array manipulation, transformations, filtering, advanced freetype font support, and drawing.[12]

Applications using Pygame can run on Android phones and tablets with the use of Pygame Subset for Android (pgs4a).[13] Sound, vibration, keyboard, and accelerometer are supported on Android.[14]

3. Flet

Flet is a framework that allows building interactive multi-user web, desktop and mobile applications in your favorite language without prior experience in frontend development.

You build a UI for your program with Flet controls which are based on Flutter by Google. Flet does not just "wrap" Flutter widgets, but adds its own "opinion" by combining smaller widgets, hiding complexities, implementing UI best-practices,

applying reasonable defaults - all to ensure your apps look cool and professional without extra efforts.

Flet is a rich User Interface (UI) framework to quickly build interactive web, desktop and mobile apps in Python without prior knowledge of web technologies like HTTP, HTML, CSS or JavaSscript. You build UI with controls based on Flutter widgets to ensure your programs look cool and professional.

CHAPTER - 9

Reference

1. https://en.wikipedia.org/wiki/Python_(programming_language)
2. https://en.wikipedia.org/wiki/Requests_(software)
3. https://en.wikipedia.org/wiki/Requests_(software)
4. https://en.wikipedia.org/wiki/Python_Imaging_Library
5. https://www.crowdwisdomlms.com/blog/5-benefits-of-video-based-learning/
6. https://en.wikipedia.org/wiki/Natural_Language_Toolkit
7. https://pypi.org/project/moviepy/
8. https://www.who.int/health-topics/coronavirus#tab=tab_1
9. https://www.tandfonline.com/doi/full/10.1080/10408363.2020.1783198