Extracting Notes From Youtube Videos

Project Report

Degree Of Bachelors of Technology
In
Computer Engineering(COE)

Submitted By:-

Sidhdharth Varshney(2K17/CO/340)
Suhail Akhtar(2K17/CO/345)
Sumit Gaurav(2K17/CO/349)

Under the supervision of:

Dr. Manoj Kumar
(Associate Professor, Computer Science and Engineering, Delhi Technological University)

Department of Computer Science and Engineering, Delhi Technological University

CANDIDATES' DECLARATION

Siddharth Varshney (2k17/CO/340), Suhail Akhtar (2k17/CO/345) and Sumit Gaurav (2k17/CO/349), students of B. Tech. (Computer Engineering), hereby certify that the work which is presented in the Major Project-II entitled "*Extracting Notes from Youtube Video*" in fulfilment of the requirement for the award of the Degree of Bachelor of Technology in Computer Engineering and submitted to the Department of Computer Science and Engineering, Delhi Technological University, Delhi is an authentic record of our own, carried out during a period from January to May 2021, under the supervision of Dr. Manoj Kumar. The matter presented in this report has not been submitted by us for the award of any other degree of this or any other Institute/University. The work has been accepted in peer reviewed Scopus indexed conference with the following details:

Title of the Paper:	**Extracting Notes From Youtube Video**
Author names :	**Manoj Kumar, Sumit Gaurav, Suhail Akhtar, Siddharth Varshney**
Name of Conference:	**The International Conference for Intelligent Technologies**
Conference Dates with venue:	**25-27 June, Hubli, Karnataka, India**
Have you registered for the conference?:	**Yes**
Status of paper Accepted:	**Accepted**
Date of paper communication:	**15-04-2021 (15 April 2021)**
Date of paper acceptance:	**22-04-2021 (22 April 2021)**
Date of paper publication:	**To be Announced**

[illegible]	Suhail	Sumit
Siddharth Varshney	Suhail Akhtar	Sumit Gaurav
(2K17/CO/340)	(2K17/CO/345)	(2K17/CO/349)

CERTIFICATE

We hereby certify that the Project Dissertation titled “Extracting Notes From Youtube Video” which is submitted by Siddharth Varshney (2k17/CO/340), Suhail Akhtar (2k17/CO/345) and Sumit Gaurav (2k17/CO/349), Department of Computer Science and Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

[illegible]

Place: New Delhi, India **(Dr. Manoj Kumar)**

Date: 21st May’2021 **(SUPERVISOR)**

ACKNOWLEDGEMENT

The foundation of any project is the combination of numerous building blocks. This project also is the result of sincere efforts of many individuals whom we would like to thank.

Firstly,we would like to express our gratitude to Sir Dr. Manoj Kumar, Associate Professor, CSE Department for her valuable inputs and the concept and idea of this project which made us learn about the realism of engineering.

Also, we are grateful to the department of Computer Science and Engineering, Delhi Technological University for the cooperation and guidance throughout the project.

Thank you!

Siddharth Varshney (2K17/CO/340)
Suhail Akhtar (2K17/CO/345)
Sumit Gaurav (2K17/CO/349)

ABSTRACT

In the world of technology, every B.tech student in our college uses Youtube to study the night before the exam, it will be of better use if there is an algorithm that can also provide notes for that video to look for last-minute revision. Hand-written notes are to be carried in a notebook, tedious to maintain for a long time. If a student wishes to pause and take notes, it will consume double the regular time of that streaming video. Those notes might or might not be used in the future or maybe not give proper context if the original content of the video is forgotten. In this project we are extracting frames from a presentation-based youtube video after every fixed time interval, using optical character recognition techniques on these images to make notes of the content written in a youtube video. Users will give the link of the desired youtube video and a pdf/doc file containing all the sentences shown in a video will be in the output. Students will just have to edit and delete some extra material that is not needed in notes. In the upcoming era education will bend more towards paperless, so to maintain the standards all the notes will be generated using this concept. Though it may sound simple it is a bit complex as to know when to extract data from a given frame or how to handle cases when some person comes in between the video. All these challenges will be major challenges in the future and will require a solution. Our idea is the first step towards this future .

Key words: Youtube, Key-Frame, Pytesseract, Similarity Index, String Redundancy Algorithm

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER-1
INTRODUCTION

1.1 Importance and Motivation

This is the process of creating a document based on the texts shown in the video. This technology can be used in a huge variety of segments, such as making notes for regular classes, instead of wasting time on writing down calculations of mathematical equations, just formulas can be remembered and the rest can be left for this program to write down all the text shown on the screen written by a teacher. Video-based Notes extraction or note creation is a new concept that is not performed by any known software in the author's knowledge.

The motivation for this project came from studying the night before an exam and wasting time copying notes from the study video for future use. Most of the content shown in a video is needed only to remember the context of a topic or to remember some keywords for an answer. Making these types of notes takes a lot of time. Automating this whole notes making process just providing a link to the video will give a boost to the amount of content that could be studied for a limited amount of time.

Not only will it help students it will also help teachers. Most of the time teachers have to pause the video to let the students copy the notes and that leads to waste of time. With the help of our project that time will not be wasted and teachers will be able to complete the syllabus before the end of the semester. The project can be extended to convert the speech lectures which are given in between the slides by the professor into text. In this way a complete note creation app can be created successfully. This will help students to have more focus on the class and reduce the time of writing down everything.

Techniques can be embedded to self focus on the slides and avoid any kind of obstacles in between. Obstacles can be anything like a teacher's finger or any other object. Our project, with the help of this technique can have vast use and can increase its horizon of working.

1.2 Concepts and Approach

Whole project is divided into several stages of processing. For the user to have the most comfortable experience, a website is made where there is a space in which the user will paste the link of the video and click submit. That URL will be sent to our background code through FLASK API which is the building framework of our project. An open library known as 'pafy' will retrieve the video of the URL at 30FPS in highest quality. This library will not download the video but just process the online video. Now we have the data for the video, further process is divided in three stages.

This project mainly uses three concepts:-
1. Key Frame Extraction from the given youtube video link
2. Optical Character Recognition
3. Algorithm to print only unique characters between two frames

Concept 1: Key Frame Extraction from a given youtube video link
Every video has variable length, different frame rate and pixel quality. Videos have different amounts of unique content in them, a fixed time for extracting a frame of a video, or extracting a frame after a fixed amount of frames is a resource and time consuming algorithm. In this project Key Frame Extraction is applied to extract the right amount of frames which contains unique content to avoid maximum repetition of characters.

What is the key Frame? Key frame in video refers to the beginning and end point for the transition of an object, i.e. the beginning key-frame is where the object is right now and the ending key-frame is where you want the object to be after transition, Key frames contain the necessary information about where a transition should begin and where it should end on a timeline. A sequence of key frames is what viewers see as motion.

Concept 2: Optical Character Recognition

Optical character recognition or optical character reader (OCR) is the mechanical conversion of text from different areas such as images, surroundings, scenes into human readable format. OCR helps in converting the text from improper format to proper format. There are too many examples of OCR such as google lens which extracts text from images. Similar to google lens we have a camscanner and so on. OCR is an evolving field and requires a good amount of data to train itself.

It learns from the surroundings then finds the accurate text next time. It is used as a method to enter data from printed paper data records where printed paper can be any sort of document like passport, school/college assignment, business card or any other suitable documentation. It helps in storing the text digitally so that they can be edited, stored and manipulated in any sort of way. OCR has applications in fields like cognitive computing, translation, text-to-speech, key data and text mining. Lot of work is left in OCR research. There are many areas in OCR which are not explored yet and have good amounts of research like pattern recognition, artificial intelligence and computer vision.

Concept 3: Unique Character Algorithm

Third phase of the project is the core structure of our note making process. First we extract frames of video which contain our notes to be written. Consecutive frames contain common sentences, paragraphs, symbols and diagrams. If we save the text of all the frames, notes will contain too much repetition of data and the whole purpose of making notes from video is wasted because notes should be short and neat. To avoid this data repetition, an algorithm has to be constructed which will eliminate common phrases and sentences of consecutive frames and notes will be unique and saved in proper structure given in the video. This algorithm should eliminate image captions, diagram definitions and labeling.

1.2 Tools Used

- Sublime
- Flask
- Cv2
- Pafy
- Python Imaging Library(PIL)
- OS
- CSV
- Time
- Numpy
- Pandas
- Matplotlib
- Skimage Matrics

1.4 Technologies Used

- Python Language
- String Dynamic Programming
- Machine Learning

1.5 Dataset Description

We screen recorded about 50 presentation videos with some data repetition in many slides in each video from the previous slide. Those 50 videos contain only text characters with no images. Each video contains at least 5 ppt slides. 20 videos were recorded which contain images in slides and images with and without text characters. All these videos were recorded as presentation videos.

1.6 Challenges

- ❖ First challenge was the selection of a dataset on which it could be possible for our system to work. There are hundreds of types of videos which contain texts. It was essential to choose video which could give us enough text material in it to make notes of the content of it. We decided to choose videos which contain no image with different coloured backgrounds.
- ❖ Due to the ongoing pandemic, we were forced to work from home. This resulted in limited resources that could be used for the project. As there were many calculations and processing to be done on images and videos, Graphical Processing Unit was a necessity. Due to lack of means we had to choose video video which would give us frames that were easier to process.
- ❖ Due to uneven output of text and noise from tesseract library, getting only useful text was a very daunting task. New algorithms had to be formed to process that.

Chapter II describes the background algorithms and literature review for frame extraction, Optical character Reader and String matching algorithm which are key processes and algorithms for the project. Chapter III describes the proposed approach in three phases in detail. Results and discussions followed by conclusion are presented in Chapter IV.

Chapter 2
BACKGROUND ALGORITHMS

There are three phases in which the video is processed to extract required notes from the video. All these three are vital steps and are sequential in order of their completion. To capture text lines from a moving image we need to stabilize it so we can extract text lines from it. But, with every motion some part of the text is kept on adding or some part is kept on removing, in essence in a presentation video, the person is writing or removing lines. Thus we need a particular frame of video to extract that text. Now the question arises which frame would be best suitable to extract, this problem leads to the use of the concept of key-frame extraction method. After having all the key-frames we need to convert the text embedded on an image to be converted to digital form. This is done with the help of Optical Character Reader (OCR). OCR outputs all the text present in the video in string format. These strings contain some random characters, duplicate lines from previous frames, and some other noise. These have to be removed, thus an algo to refine the notes was formed named as string matching algorithm.

2.1 Frame Extraction

First we tried to use the Key-Frame extraction method, which has been used in advertisement industries[1] for many years such as in Google and Amazon advertisements. The key-Frame extraction method studies consecutive frames of a video and analyses them against the previously selected frame of the video stored in the database. As soon as it calculates a particular frame is distinct enough from the last one, it stores that frame for further processing. The key-Frame method generally is applied in videos that have high feature differences, the challenge we are facing in applying this algorithm in our work is that the videos are presentation based in which theme is the same throughout the video and only the basic content of the video i.e. the English sentences are changing without a single diagram, throughout the video. In these types of videos, there is not much of a feature difference so after the first selected frame, the key-frame extraction method will not select any frame for further processing. If we change the parameter of frame selection so that a minute difference in frames would make it be selected, then a video that has a high feature difference in consecutive frames will give us an unnecessarily high number of frames which will decrease the processing speed in later stages of work. Thus, we had to drop this method.

To solve the problem that arises while extracting frames from video is to determine the time i.e. at what time one should extract the particular frame from the video. The basic intuition is that a particular machine learning model should be used to detect sudden change in the background and take action corresponding to that background. Another machine learning concept that we used for this purpose was 'image similarity detection'. Image similarity detection allows the detection of similar images. Similar images are a set of images which falls under the same class. Images taken from different angles or even images which are edited will be considered similar to the original image. For example in Fig-1 all images in row 1 will fall into the same class, row 2 in the same class and similarly in row 3. In Fig-2 all images will fall into the same class despite their size.

Fig-2.1 Example Same background

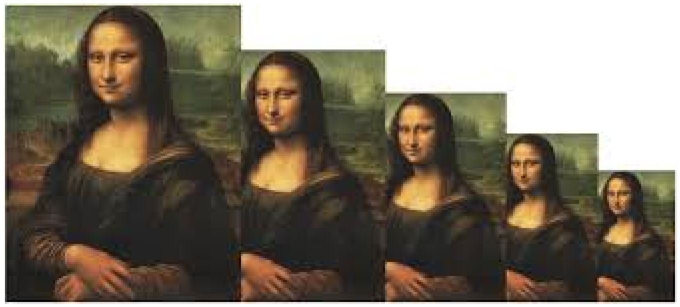


Fig-2.2: Example size different images

In this procedure, we extracted frames from different videos and compared each extracted frame to the consecutive extracted frame, and obtained a percentage of similarity among them which is qualified as a similarity index. We used open libraries of python such as “skimage.metrics.structural_similarity” and “cv2.ORB_create()”, these libraries compared images based on their features and gave the result as how much percentage of the two images are exactly equal.

ORB Algorithm:-

ORB stands for Oriented FAST and BRIEF. Fusion of FAST and BRIEF descriptor along with some modification, mainly optimization, led to the formation of ORB. FAST is a Feature from the Accelerated segment test which is used to detect features from provided images. Fast doesn’t compute orientation and descriptors so for this BRIEF(Binary Robust Independent Elementary Features) is used. ORB rotates BRIEF according to the key points. ORB is the best alternative for SURF(Speed Up Robust Features) and SIFT(Scale Invariant Feature Transform). ORB has better performance in feature extraction, computation cost, and matching performance as compared to the former two.[14]

Structural_similarity:-

Structural_similarity often written as ssim is used to calculate the Structural Similarity Index. Structural Similarity Index is a numeric value calculated between two images. Its value ranges from -1 to +1. A value of +1 indicates that both the images are similar with little to no difference and a value of -1 indicates that both the images are different. For most of the time, the range is normalized from 0 to 1 where the end carries the same meaning. Before the structural similarity index, the mean squared error was used to calculate the difference between two images because it is easy to implement but the accuracy is not up to the mark, that’s why the structural similarity index is used nowadays.[15]

About 45 videos were analyzed in this procedure. Videos were divided into two parts, which contain graphical images and another which contained only text. The average similarity index of frames which contain only text was very high as compared to the set of videos that contained images in some parts. We are taking an average of all the similarity indices of all comparisons between frames of a single video, then taking an average of all the indices computed for each video in a set. The Average Similarity Index of non graphical videos was approx 87% and the set of videos with graphical content was approx 44%. This difference occurs as frames that contain images differ highly from frames with no image which gives the two frames a very low similarity index value thereby decreasing the overall similarity in different frames of that video and giving a false indication that the Key-frame extraction method can work, therefore we excluded graphical videos for computing result in this part.

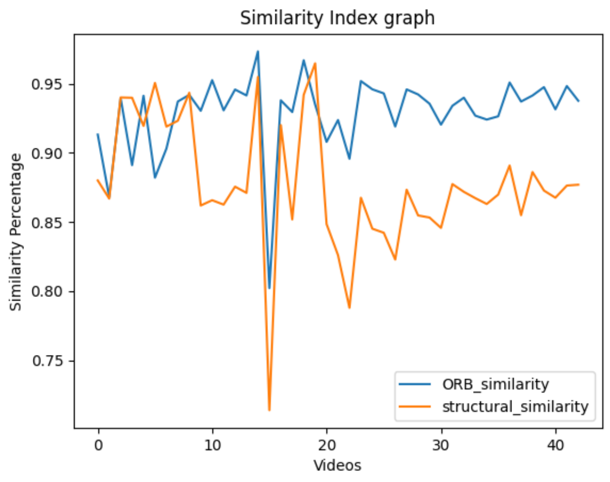


Fig 2.3:Similarity Index Graph

By this result, we came to conclude that the Key-Frame Extraction method would not be a good option to extract the computable frames. Thus we are extracting frames at a constant time of 5 sec. This time difference to extract consecutive frames is based solely on a random value, it can be changed as per the usage of the system.

Algorithms / video_types	**SSIM**	**ORB_Sim**
No_Image_videos	0.92	0.87
Images_Videos	0.52	0.44

Table 1: Frame Similarity Table

2.2 Optical Character Recognition

Optical Character Recognition (OCR) is the procedure of reconstruction, transformation of different varieties of text or text-based documents like written language, written or scanned language pictograms, to an electronically erasable configuration for complex and higher computation. Optical character recognition machinery allows us to work on text based documents instantaneously which would have taken a life-long time to be written again.

Since the OCR analysis is currently a vigorous and vital area in general Natural language processing, due to its exponential advancement, exhaustive study of the field on a daily basis is required to keep a log on the new developments. This paper gives us information by delivering us a thorough literature analysis of the Optical character Recognition technology. This paper discusses principal problems one faces and key stages of optical character recognition such as preprocessing, segmentation, normalisation, feature extraction, classification and post processing in detail which needs to be considered throughout building any application associated with the OCR.

OCR Challenges

1. Scene Complexity

In a regular image, there may be structures, symbols or edges of objects which resemble certain characters. K-NN based OCR can easily misidentify symbols or edges of objects as characters.

2. Conditions of Uneven Lighting

Natural environment images result in uneven lightning in images. It diminishes the boundary of edges of characters and background colour which can result in inaccurate classification.

3. Blurring and Degradation

Images captured at moving cameras will result in shady characters. Where shade rate is high those characters will not be recognised by OCR.

4. Aspect Ratios

Sentences on images have different sizes and fonts. Text may be in short forms such as hand written signs e.g. “and” is written as “&”, while other lines might be very large, such as Headings and captions. Text depth like dark blue ink, black pen ink, ball and gel pen ink differences, size and length ratios are needed to be taken into account with training strategy to classify text, which might introduce high computational resources.

5. Tilting

Pictures might be clicked while the camera is not aligned with the level of the plane of the paper,e.g. pictures clicked and sent as pdf for submitting as an online answer sheet in exams. Accordingly, text lines that are farther from the line of sight seem littler than those that are nearer to the line of sight which seems bigger. This condition causes an absurdly tilted document. Here line of sight means able of camera image.

6. Multilingual Environments

More than one language in an image especially which language has connected characters changes shape of characters e.g. Arabic.

OCR Phases

1. Pre-processing Phase

The objective of the pre-processing phase is to reduce unwanted features or noise present in a picture while not missing any vital data. Preprocessing techniques are applied or we say essential on coloured, black-n-white, scanned or duplicate document pictures containing text with some graphical representations. Coloured pictures have 3 layers of pixels instead of 1 as compared to grey pictures, thus processing is computationally more expensive. Taking computational complexity into account, most OCR is done on scanned images.This phase reduces the inconsistency and noise. Non vital area is recognised and pixel data is zeroed at that place where amplification of the image occurs and compose it for succeeding OCR phases.

Some important preprocessing operations are:

1. **Binarization** - Classifies image pixels as a part of character or noise.
2. **Noise Reduction** - Removes excessive variation in image pixel data produced by varying light while capturing image. Just like varying noise signals enter in communication signals.
3. **Skew Correction** - There is possibility of rotation of the input image because of un-parallel image capture, document skew can be amended.
4. **Morphological Operations** - Addition or subtraction of pixels in a multi layered character picture that have depleted or extra pixels.
5. **Thresholding** - In image data analysis, separating data from its non-textual background.
6. **Thinning and Skeletonisation** -Thin and skeleton forms a structure of lines which run along the centre pixel of thick text reducing the number of pixels for that text.

2. Segmentation Phase

One of the major challenges of Optical character recognition learning phase is segmentation of text data from pictorial background. Generally, text segmentation from a document picture integrates character segmentation after word segmentation after line segmentation. Segmentation is the technique of dividing text objects from background images. Image is formed by different pixel values, sudden change in pixel values denote text to image or image to text contained in consecutive pixels forming a smooth transition from dark to light or light to dark. This transition is recognised and segmentation works here.

In a list of most required phases comes the document segmentation phase. It is an important phase in working in menu-script-like works. Homogeneous zones are formed in this phase and every zone constitutes similar kinds of data, like word texts, an image,important symbol, a table, or a figure. In several instances, the accuracy of the OCR algos depends heavily on the reliability of the image segmentation algorithm program used.

There are three structural type of algorithms present in document segmentation given below:

- Top-down methods,
- Bottom-up methods,
- Hybrid methods.

The top-down approach divides big regions in smaller subregions of text. Once this recursive process stops, satisfactory criterion is met and the number ranges obtained represent the lengths of final segmentation. In the bottom-up approach, the consecutive pixels which contain similar data are grouped to form a character. Then many words are formed, by combining the characters formed with pixels. These words are in turn combined like connected components to form lines,text blocks or paragraph blocks. The integration of strategies in a single program is termed hybrid approaches.

3. Normalization Phase

In this phase the obtained image from the segmentation phase is resized. Different segments obtained have different pixel density which sometimes increases the amount of data from initial data. Matrix of size m*n are obtained after the segmentation phase, this matrix dimensions are then reduced without losing any major data from the image.

4. Feature Extraction Phase

Word vectors or feature vectors are created by extracting relevant features from objects or alphabets, this process is known as feature extraction. Classifiers utilize these feature vectors to match the input feature with an output unit. Dissimilar categories are classified differently very easily by the classifier by comparing these output features vector units.

5. Classification Phase

- **Template matching**- The operation of cross-matching decides the amount of resemblance between 2 vectors by gathering information on features like pixel density, combined pixel curvature. The classification rate of this algorithm is highly dependent on noise and segmentation phase.
- **Statistical Techniques** - The main statistical methods that are performed are Nearest Neighbor (NN), naive Bayes classifier, Clustering Analysis, Hidden Markov Modelling (HMM), Fuzzy Set Reasoning, and Quadratic classifier.
- **Neural-Network** - A neural network is a parallel web-like architecture which has different weights on its web strings and nodes are the feature values which are computed using weights on the web strings. Every column of nodes is connected to a similar column of nodes using parallel strings. Output is computed using collaboration of all the nodes and weights and the final answer is computed and tested against given output, if not correct back propagation is started to correct weights on strings. This process continues until it starts to give correct output to given input.

2.3 String Matching algorithm

Before understanding our algorithm we must understand what is the need for it in our problem. To make notes what we are doing is taking screenshot and retrieving text from it but the issue is we can not make sure we have completely different text all the time so it is bound to have repetition in texts given that we take screenshot frequent enough and this is also necessary because otherwise we can lose on data and data is most important so we have to come up with some solution for repetition .

There can be three different cases in handling texts from different cases one of them is hundred percent overlap that is texts match hundred percent so we don’t need to copy any data in notes second type is partial overlap so we have to copy to certain part of the new string in our notes and third part is zero overlap and in this case we have to copy entire data in our notes.

All this idea is usable because we observe that in an average tutorial video entire text don't change instantly so we can say that we will observe gradual change in text and hence we can take snaps at different moment and there is a very high probability of overlap in texts and hence need for an solution to handle this repetition .

Before seeing our algorithm let us see what is wrong with normal KMP or any other string matching algorithm that already exists .

When we try to retrieve text from any frame it is very probable that we might not get text from the slide with hundred percent accuracy or it can be the case when we have a hand in front of the slide and hence we will not have everything written on slide which creates a serious issue for any existing string matching algorithm.
For ex :-
Old text :- i am sumit
New text :- i sumit gaurav

So is we use any standard algorithm for string matching the output we will get is "sumit gaurav" is new the text or only "i" matches but we will be getting many cases like these in real life and this absence of "am" can be because of presence of hand in front of it or it might be the fault of OCR .

So we need an algorithm which can take these two texts and output "gaurav" as new string or text that we need to add.

Now as we know what is already present and what we require we present our idea for the problem :-

So to understand our algorithm let us define few terms :-

1. String1 -> it represents the text we got from the last frame.
2. String2 -> it represents the text we got from the current frame.
3. match_points :- match point is given to a pair of strings and high match_point represents a pair of strings that are very similar and low match_point shows less matching strings.

Method used for calculating match_points :-

To find match_points we are taking three different type of parameters :-

Uni :- It represents how many times a character has appeared in a string .

Example : -

String - "aabcddd"

Equivalent Uni :- [a]-2 , [b]-1 ,[c]-1,[d]-3

Di :- It represents how many times a pair of characters has appeared in a string.

Example:-

String - "aabcddd"

Equivalent Di :- [a,a] - 1 , [a,b]-1 , [b,c]-1 , [c,d] -1, [d,d]-2

Tri :- It represents how many times a pair of three characters has appeared in the string.

Example:-

String - "aabcddd"

Equivalent Tri - [a,a,b]-1 , [a,b,c]-1 , [b,c,d]-1 , [c,d,d]-1 , [d,d,d]-1 match_points = number of uni matched between two strings + 2* number of di matched between two strings + 3* number of Tri matched between two strings

Example:-

String1 :- “sid”	String2:- “suh”
Uni- [s]-1,[i]-1,[d]-1	Uni - [s]-1 , [u]-1 , [h]-1
Di -[s,i]-1, [i,d]-1	Di - [s,u]-1 , [u,h]-1
Tri- [s,i,d]-1	Tri - [s,u,h]-1

match_points(“sid”, “suh”) = 1*1 + 0*2 + 0*3 = 1

Now we will take a different starting position from string1 and compare the first 20 letters from that position to the first 20 letters of string2 after finding all the possible values of match_points .

Now if we have a highest match_point value greater than equal to 30 we will say that yes there is some overlap otherwise we will consider it as 0 overlap and copy the entire string2 to our notes. Now in case of overlap the highest match_point value giving position is considered as the point from where overlap is starting .

Now from that position we will keep taking a block of 10 size and match it with corresponding block in string2 and if at any point mismatch in two blocks is greater than or equal to thirty percentage we will break our process at that point and copy entire string2 from that point and if this never takes place we will copy text from that position where we are left in string2.

The validity of our algorithm can be easily verified by comparing the length of our output with original output length . Since we are trying to add text from output generated by OCR the only concern left for us is , if we are able to copy only unique texts from generated output . To do so we are using a very simple algorithm that is to check by what percentage we are writing extra words or by what percentage we are writing less words.

Chapter 3

Proposed Approach

We are starting with hosting a basic web page which contains a form where users can submit a link to a video to be processed to extract notes from it. Currently only youtube links are supported. This link will be stored in a python variable through FLASK framework with 'GET' request.

The video without being downloaded will be processed and frames every five seconds will be extracted from the video. These frames will then be feeded into OCR, and its output will be compared in string matching algorithm. These are the three stages of code that the video will go so that notes can be extracted from it.

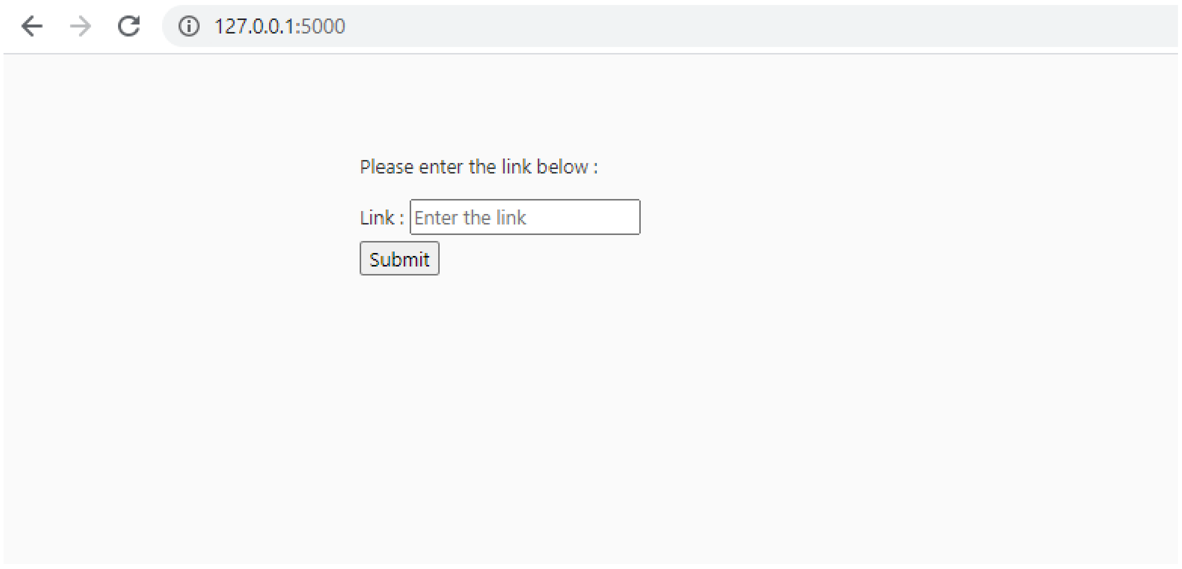


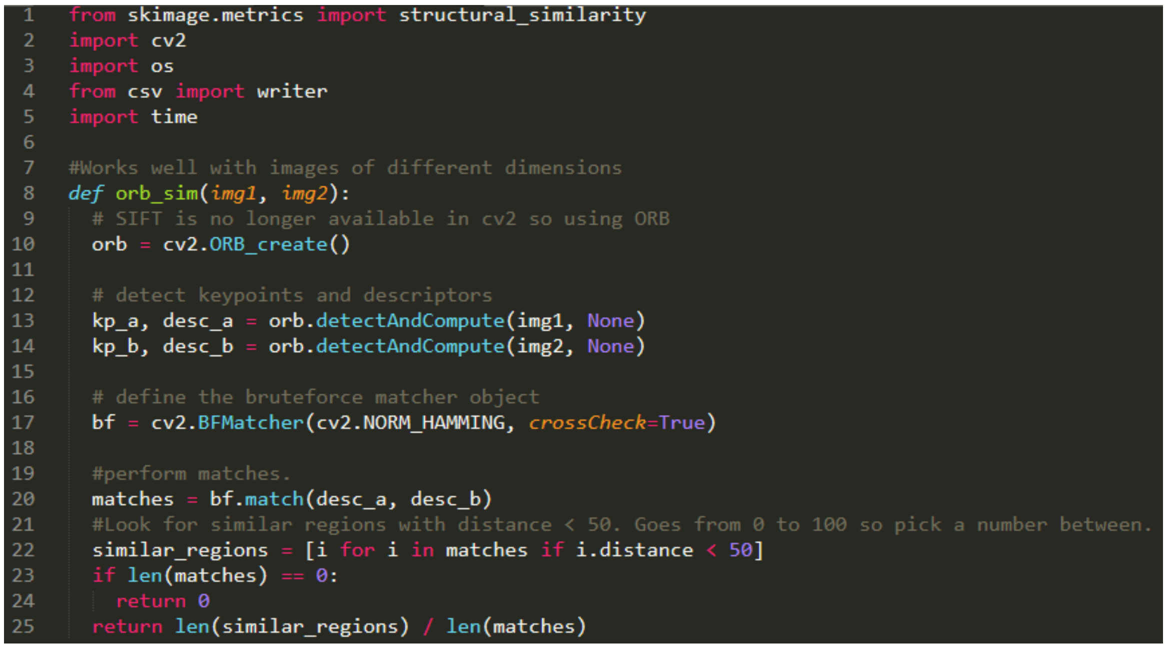
Fig 3.1: Webpage to paste link of video

Web page is written in html, designed by CSS and Javascript, and hosted by the Flask framework in python. Web page is on a local network which can only be accessed for testing purposes only, not from a public network.

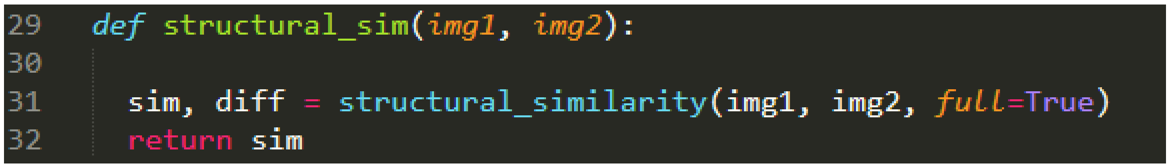
3.1 Stage-1

First stage is about extracting frames from videos for further processing. In this stage we tried to extract only key-frames which are different from previously extracted frames, so that not many frames are extracted to reduce redundancy and reduce computing resources used for one particular noting making process.

Key-frame was tried to be implemented by using image similarity method. We tried to compare consecutive frames at different constant time intervals for 5 seconds and 10 seconds, for similarity index. For calculating similarity index between two images we compare feature vector similarization between those two images, and similarity index is the percentage of similarity those two images have. If two frames have a high similarity index, the second frame will be discarded and a new frame after 5 seconds will be selected for similarity detection between frames.



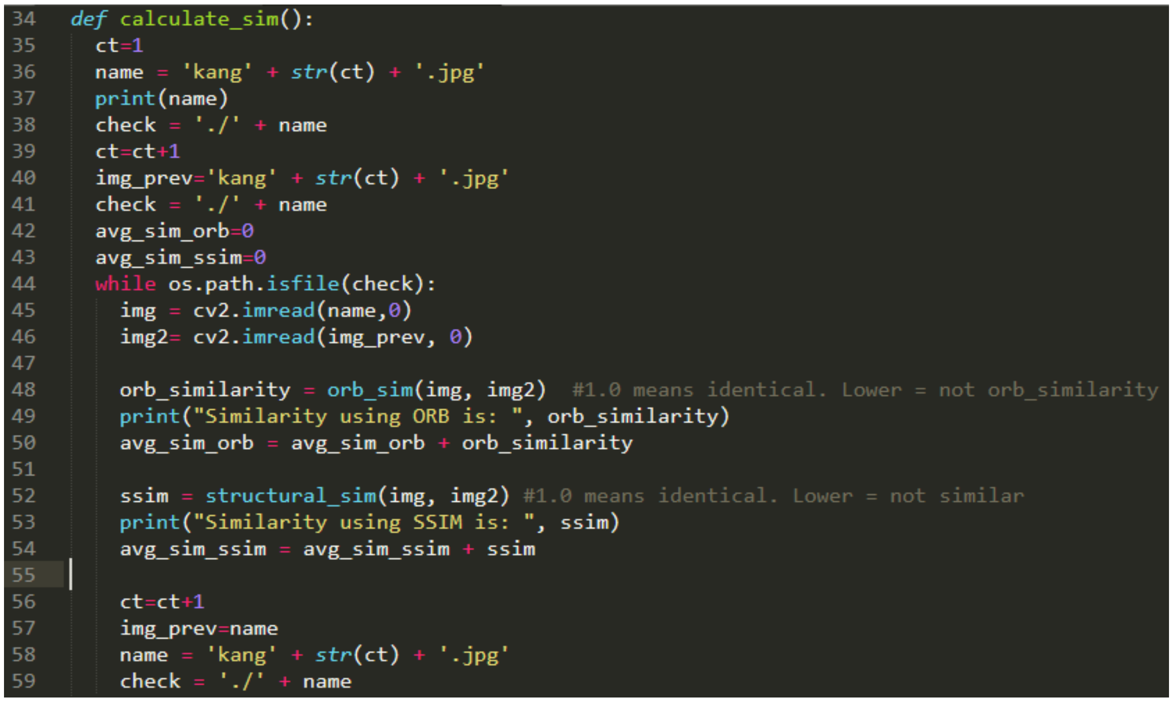
```python
from skimage.metrics import structural_similarity
import cv2
import os
from csv import writer
import time

#Works well with images of different dimensions
def orb_sim(img1, img2):
  # SIFT is no longer available in cv2 so using ORB
  orb = cv2.ORB_create()

  # detect keypoints and descriptors
  kp_a, desc_a = orb.detectAndCompute(img1, None)
  kp_b, desc_b = orb.detectAndCompute(img2, None)

  # define the bruteforce matcher object
  bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

  #perform matches.
  matches = bf.match(desc_a, desc_b)
  #Look for similar regions with distance < 50. Goes from 0 to 100 so pick a number between.
  similar_regions = [i for i in matches if i.distance < 50]
  if len(matches) == 0:
    return 0
  return len(similar_regions) / len(matches)
```



```python
def structural_sim(img1, img2):

  sim, diff = structural_similarity(img1, img2, full=True)
  return sim
```

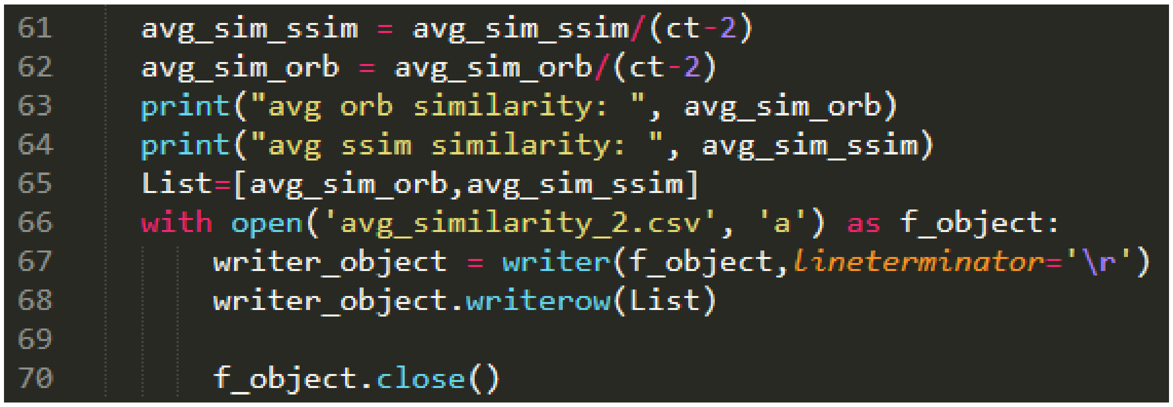
The above images contain the code for the function to calculate image similarity through two libraries, those are ‘ORB_Similarity’ and ‘Structural Similarity’. Function named ‘calculate_sim()’ is called after all the frames are captured from the video after every 5 seconds. Consecutive frames are compared in the while loop of this function and these frames are passed as an object to the ‘orb_sim’ and ‘structural_sim’ function.

These two functions return an index as a percentage to which these frames are similar to each other. If the index is higher than a fixed value of our choice, the frame will be discarded as we will consider it to contain similar text as the previous frame, and skipping the text of that frame won’t affect the content of our notes thereby reducing the resource used in processing.



```python
def calculate_sim():
  ct=1
  name = 'kang' + str(ct) + '.jpg'
  print(name)
  check = './' + name
  ct=ct+1
  img_prev='kang' + str(ct) + '.jpg'
  check = './' + name
  avg_sim_orb=0
  avg_sim_ssim=0
  while os.path.isfile(check):
    img = cv2.imread(name,0)
    img2= cv2.imread(img_prev, 0)

    orb_similarity = orb_sim(img, img2)  #1.0 means identical. Lower = not orb_similarity
    print("Similarity using ORB is: ", orb_similarity)
    avg_sim_orb = avg_sim_orb + orb_similarity

    ssim = structural_sim(img, img2) #1.0 means identical. Lower = not similar
    print("Similarity using SSIM is: ", ssim)
    avg_sim_ssim = avg_sim_ssim + ssim

    ct=ct+1
    img_prev=name
    name = 'kang' + str(ct) + '.jpg'
    check = './' + name
```

The name kang.jpg is a random name given to all the frames extracted from videos, it does not depict anything in general.



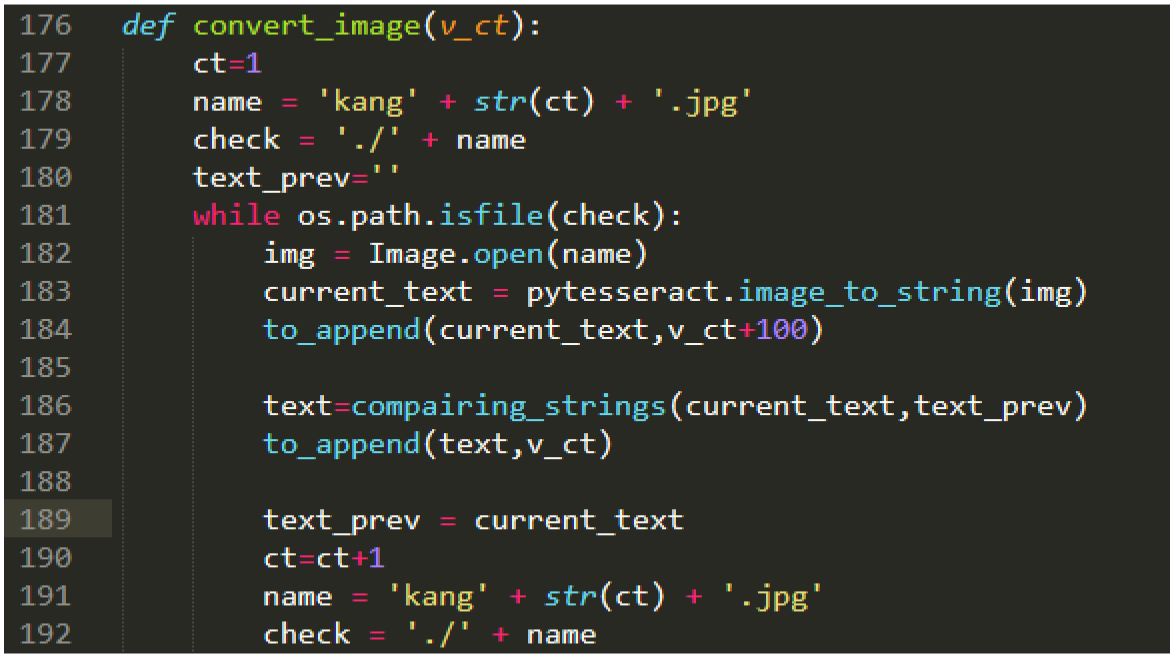
```
    avg_sim_ssim = avg_sim_ssim/(ct-2)
    avg_sim_orb = avg_sim_orb/(ct-2)
    print("avg orb similarity: ", avg_sim_orb)
    print("avg ssim similarity: ", avg_sim_ssim)
    List=[avg_sim_orb,avg_sim_ssim]
    with open('avg_similarity_2.csv', 'a') as f_object:
        writer_object = writer(f_object,lineterminator='\r')
        writer_object.writerow(List)

        f_object.close()
```

Due to high similarity even when the text was totally different in compared frames due to the same background theme and no general image feature vector difference, all compared images had a high similarity index, which meant that this process can not be implemented to the dataset we have chosen but will be very useful once we consider the videos which contain tutor teaching in the video, moving around making every frame different enough to give low similarity index. We have omitted this code file to be included in our present working code, as if we include videos with high variation in theme and with high contrast, OCR will fail to provide good text digitalisation.

3.2 Stage-2

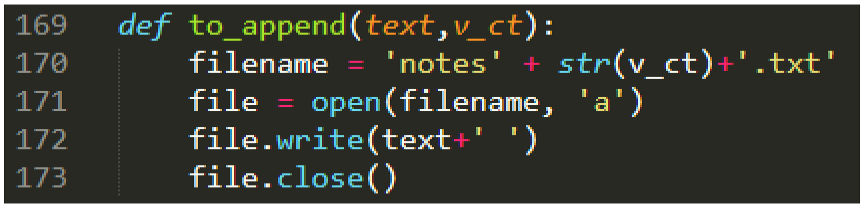
Now frames are selected from which text has to be digitised and be written for notes. Test digitalisation is done with the help of Optical Character Reader. We have used the library ‘Pytesseract’ in python language for this purpose.

Tesseract is an open source OCR engine. In recent times its popularity among developers has increased significantly. It was originally developed by HP in the 1980s, and was finally made open source in 2006. Its performance is far better than similar libraries because it supports more than 100 languages and can be trained to learn new languages with higher accuracy. Tesseract is used for reading text from binarized images. It is used for text detection in mobile devices, in video as we are using and in detecting spam mails in gmail. In python this library is present with the name pytesseract. A simple example can be given with the following image.



```python
def convert_image(v_ct):
    ct=1
    name = 'kang' + str(ct) + '.jpg'
    check = './' + name
    text_prev=''
    while os.path.isfile(check):
        img = Image.open(name)
        current_text = pytesseract.image_to_string(img)
        to_append(current_text,v_ct+100)

        text=compairing_strings(current_text,text_prev)
        to_append(text,v_ct)

        text_prev = current_text
        ct=ct+1
        name = 'kang' + str(ct) + '.jpg'
        check = './' + name
```

This piece of code converts text present in the frames to digital format and stores them in variable ’current_text’ and ‘text_prev’. ‘current_text’ contains text from the current frame and ‘text_prev’ contains text from the previous frame. Both these are passed as variables onto the 3rd stage which is the string matching function. After we get the unique text present in the current frame, in essence the text which is not present in the previous frame is then appended to the notes file.

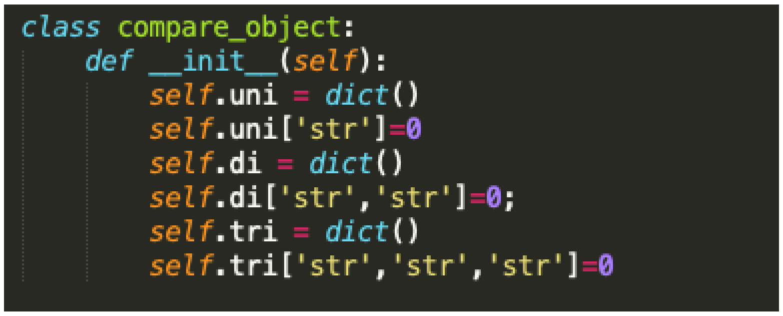


```python
def to_append(text,v_ct):
    filename = 'notes' + str(v_ct)+'.txt'
    file = open(filename, 'a')
    file.write(text+' ')
    file.close()
```

For the current dataset, tesseract is converting 100% of the text into digital format thus there is no need for image processing before using the library.

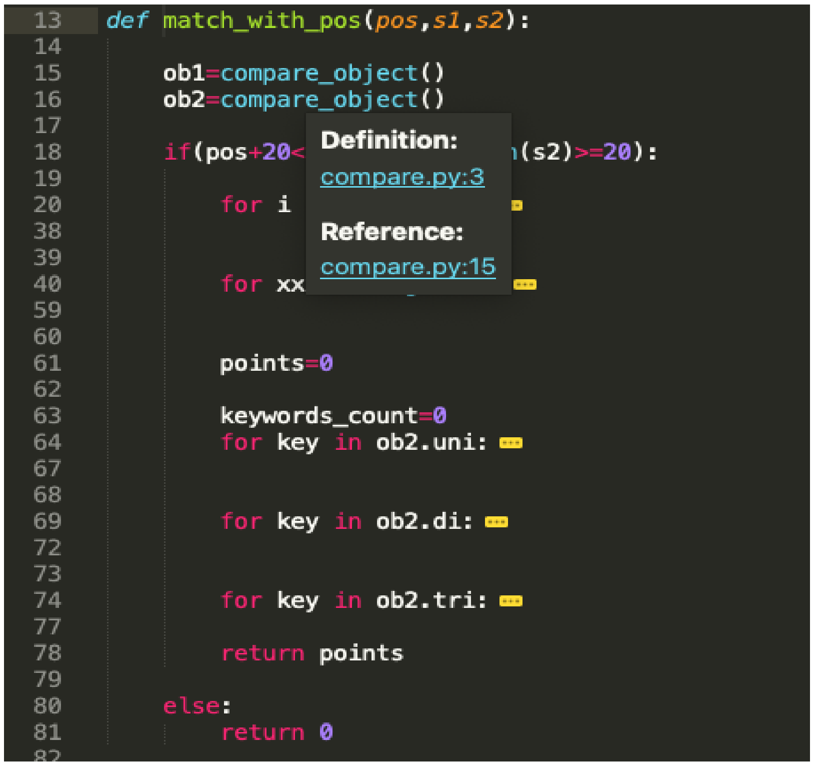
3.3 stage-3

To compare different strings we are defining a class compare object and compare object has three members uni di and tri as explained in approach.



```python
class compare_object:
    def __init__(self):
        self.uni = dict()
        self.uni['str']=0
        self.di = dict()
        self.di['str','str']=0;
        self.tri = dict()
        self.tri['str','str','str']=0
```

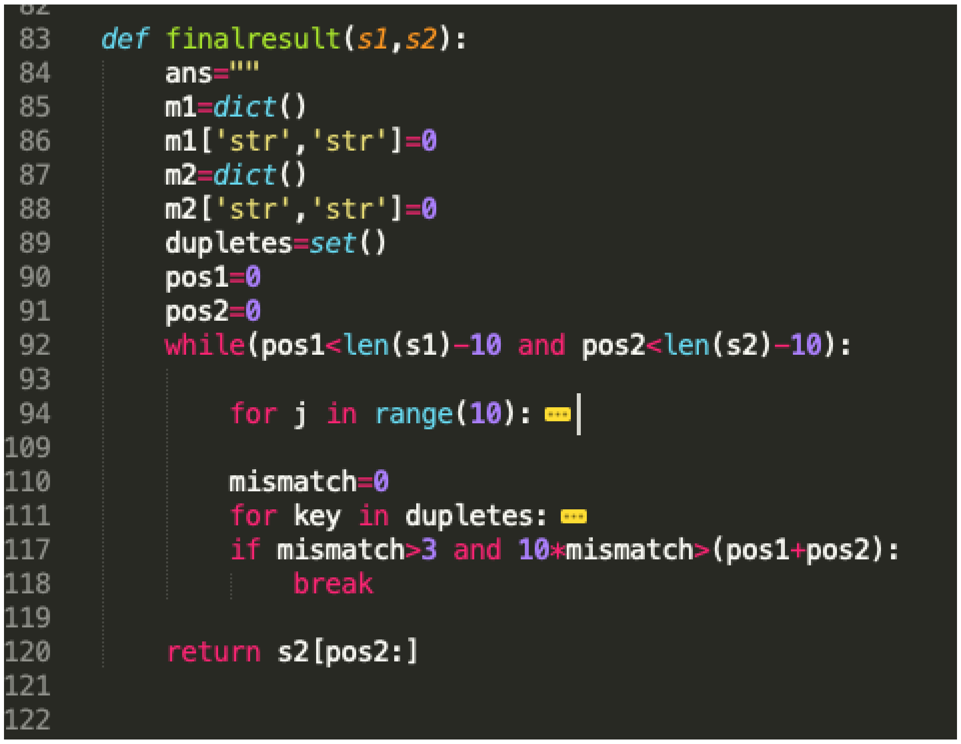
Function match with pos is used to match 2 section of length 20 in both the string and hence calculate matchpoint with respect to different starting position.



```python
def match_with_pos(pos,s1,s2):

    ob1=compare_object()
    ob2=compare_object()

    if(pos+20<                  n(s2)>=20):

        for i

        for xx

        points=0

        keywords_count=0
        for key in ob2.uni:

        for key in ob2.di:

        for key in ob2.tri:

        return points

    else:
        return 0
```

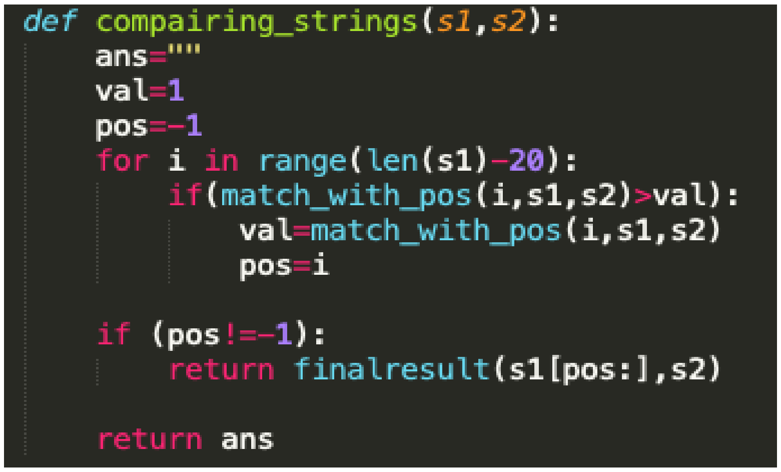
Then starting point with maximum match_point is selected as correct starting point.

After this we use the function final result to obtain the text that is new and also to increase the accuracy of starting point.



```
def finalresult(s1,s2):
    ans=""
    m1=dict()
    m1['str','str']=0
    m2=dict()
    m2['str','str']=0
    dupletes=set()
    pos1=0
    pos2=0
    while(pos1<len(s1)-10 and pos2<len(s2)-10):

        for j in range(10): ...

        mismatch=0
        for key in dupletes: ...
        if mismatch>3 and 10*mismatch>(pos1+pos2):
            break

    return s2[pos2:]
```

We use above two function and run them with the help of compaing string function that run the function match_the_pos for all the possible values of pos and it also ensures that if the two texts are completely new then it will not be able to cross a certain threshold which will indicate that the text is totally new.



```
def compairing_strings(s1,s2):
    ans=""
    val=1
    pos=-1
    for i in range(len(s1)-20):
        if(match_with_pos(i,s1,s2)>val):
            val=match_with_pos(i,s1,s2)
            pos=i

    if (pos!=-1):
        return finalresult(s1[pos:],s2)

    return ans
```

Chapter 4

RESULTS AND DISCUSSIONS

The validity of our algorithm can be easily verified by comparing the length of our output file with the original file. Since we are trying to add text from output generated by OCR the only concern left for us is , if we are able to copy only unique texts from generated output . To do so we are using a very simple algorithm that is to check by what percentage we are writing extra words or by what percentage we are writing less words. Though this may sound simple, it is an effective and easy way to check the accuracy.

Algorithm:-

Percentage of mismatch = abs(length of generated output/length of original text - 1)*100

Example :-

Original text :- My name is Sumit Gaurav

Generated text :- My name is Sumit mit Gaurav

% mismatch = |27/23 - 1 |*100 = 17.39

Although this percentage seems so high in this sample, for larger texts this % will drop significantly as the words reused remain more or less the same but new words added generally increase significantly.

Example:-

Original text length :- 1200

Generated text length : - 1250

Percentage of mismatch = |1250/1200 -1|*100 = 4.16%

We had already made a dataset of around 50 videos. We had run our code on those 50 videos and compared the original text file with our generated output file. After running the code we found out that in those 50 videos the percentage of repetition is around 10.8 percent. What it means is that an average of about 10.8 percent repetition in text which frames originally had much more repetition. We plotted the graph of the percentage of repetition of output file versus original file as below.

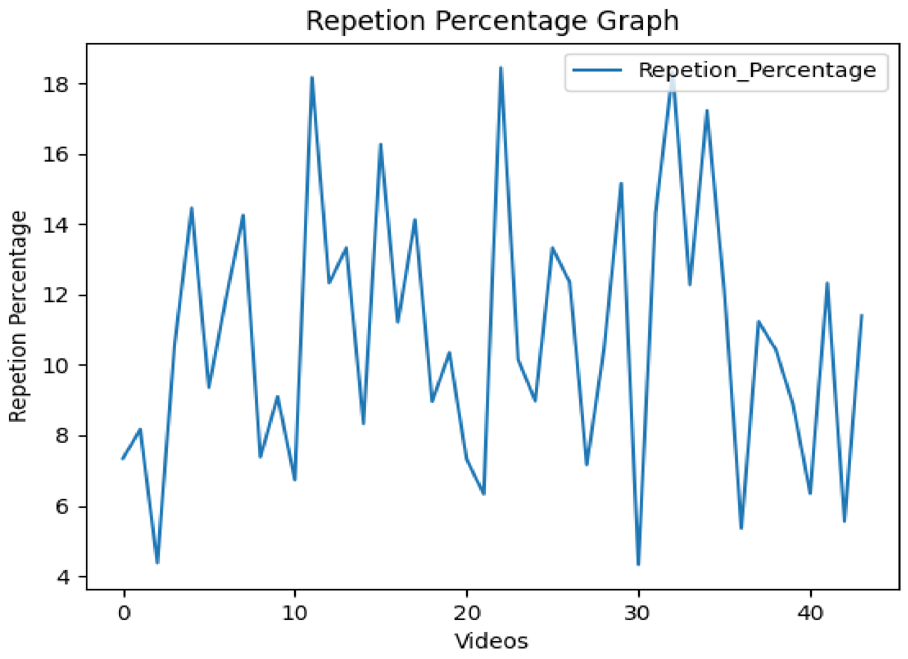


Fig 4.1: Repetition Percentage Graph

What other astonishing feature we noticed was that the percentage of repetition varies from 4 percent to 19 percent across all the 50 videos. This indicates that all the generated output files from 50 videos contained text from original text along with extra text which is generated due to consecutive frames. For some videos which contained less symbols and figures their percentage of repetition was less than 10 percent indicating that they are equivalent to original text files. Our primary goal was to have the percentage of repetition less than 20 percent and we were able to achieve it successfully.

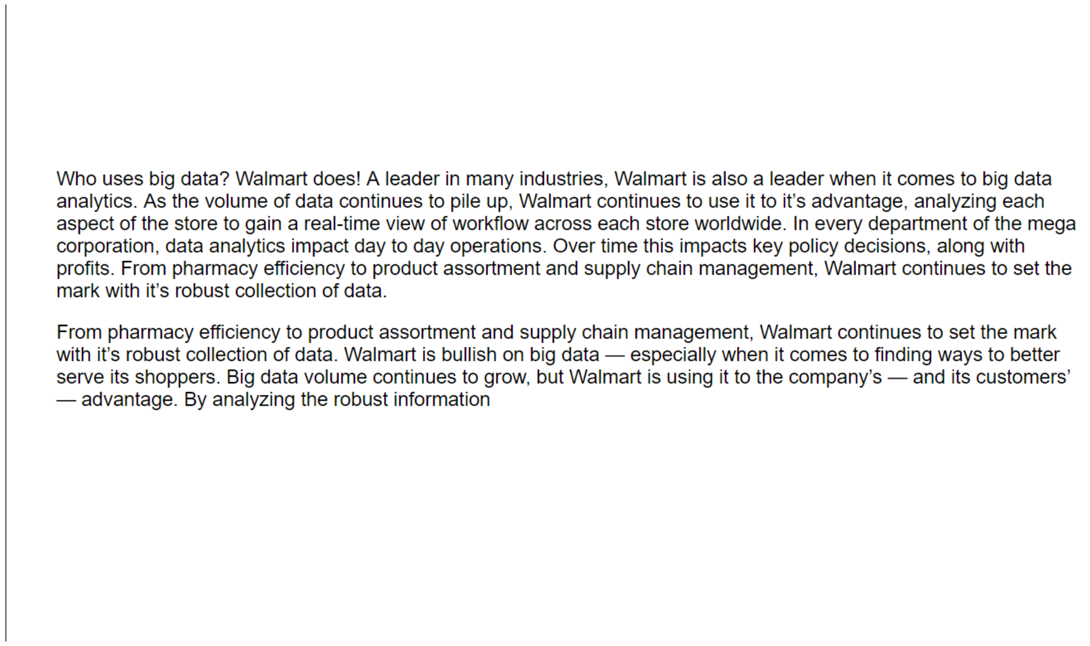
It means that the notes which are generated from the videos will definitely contain some extra text as compared to the notes that should have been produced. One thing which should be taken care of is that all those extra text are irrelevant and will be present in the form of symbols which are not understandable and can be easily ignored like '?','*' etc.

Another very interesting feature that we have noticed is that higher the content in the video will lead to less percentage of repetition. For example if we have two videos say A and B. A has around 500 words in whole and B has 200 words. After producing the output files and comparing it to their original file we will notice that the percentage of repetition of B will be greater than the percentage of repetition of A.

So if we list major factors affecting percentage of repetition, they are the volume of video and presence of symbols and figures. If a video has high volume or content and less figure or symbol its percentage of repetition will be very less as compared to any other video.

In the nutshell our model will produce the text file and if there is any presence of symbol or figure and if volume is very high then the produced text file will have less percentage of repetition compared to any other file. In this case students will have to work a lot less manually as compared to other files. If a video contains symbols as well as volume is very low it will lead to presence of unreadable characters in the produced text file in which students will have to omit the extra characters manually. At max the percentage of repetition in our case reached 20 percent, which means that 20 words were extra as compared to the original text file.

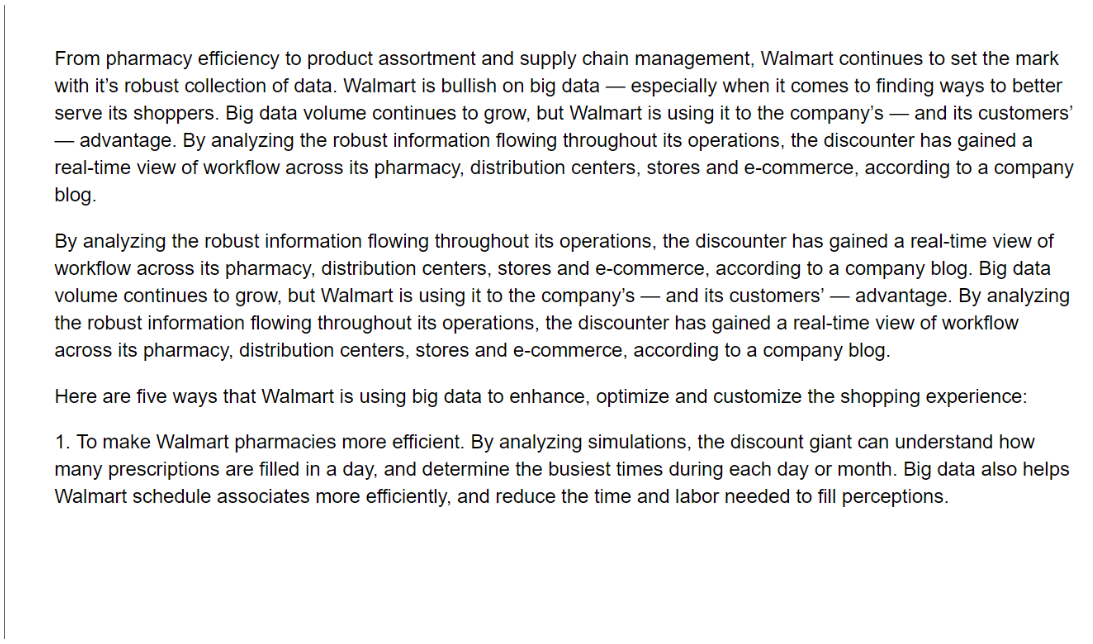
Below is the example of processing of 2 frames and it's output:



Who uses big data? Walmart does! A leader in many industries, Walmart is also a leader when it comes to big data analytics. As the volume of data continues to pile up, Walmart continues to use it to it's advantage, analyzing each aspect of the store to gain a real-time view of workflow across each store worldwide. In every department of the mega corporation, data analytics impact day to day operations. Over time this impacts key policy decisions, along with profits. From pharmacy efficiency to product assortment and supply chain management, Walmart continues to set the mark with it's robust collection of data.

From pharmacy efficiency to product assortment and supply chain management, Walmart continues to set the mark with it's robust collection of data. Walmart is bullish on big data — especially when it comes to finding ways to better serve its shoppers. Big data volume continues to grow, but Walmart is using it to the company's — and its customers' — advantage. By analyzing the robust information

Fig 4.2: Frame 1



From pharmacy efficiency to product assortment and supply chain management, Walmart continues to set the mark with it's robust collection of data. Walmart is bullish on big data — especially when it comes to finding ways to better serve its shoppers. Big data volume continues to grow, but Walmart is using it to the company's — and its customers' — advantage. By analyzing the robust information flowing throughout its operations, the discounter has gained a real-time view of workflow across its pharmacy, distribution centers, stores and e-commerce, according to a company blog.

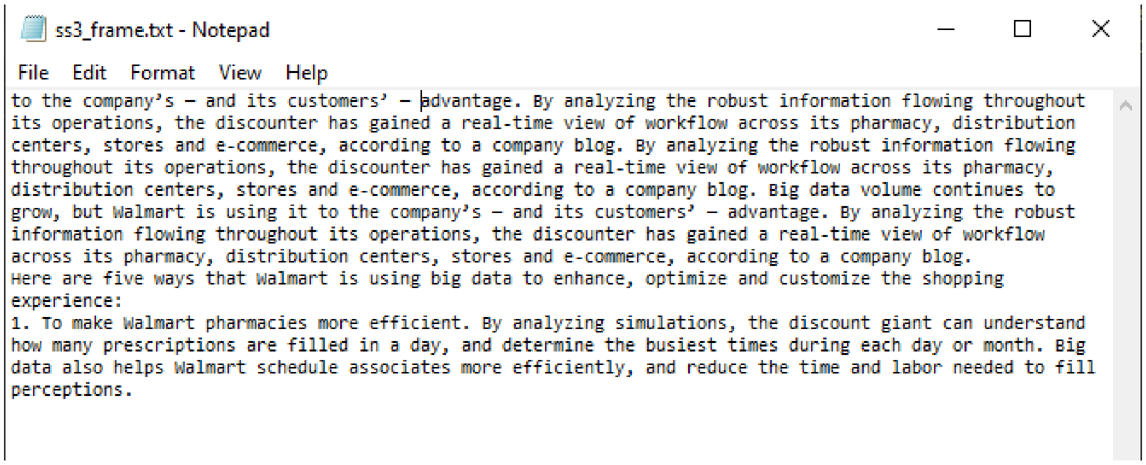
By analyzing the robust information flowing throughout its operations, the discounter has gained a real-time view of workflow across its pharmacy, distribution centers, stores and e-commerce, according to a company blog. Big data volume continues to grow, but Walmart is using it to the company's — and its customers' — advantage. By analyzing the robust information flowing throughout its operations, the discounter has gained a real-time view of workflow across its pharmacy, distribution centers, stores and e-commerce, according to a company blog.

Here are five ways that Walmart is using big data to enhance, optimize and customize the shopping experience:

1. To make Walmart pharmacies more efficient. By analyzing simulations, the discount giant can understand how many prescriptions are filled in a day, and determine the busiest times during each day or month. Big data also helps Walmart schedule associates more efficiently, and reduce the time and labor needed to fill perceptions.

Fig 4.3: Frame 2

Above consecutive frames give the following output:



```
to the company's – and its customers' – advantage. By analyzing the robust information flowing throughout
its operations, the discounter has gained a real-time view of workflow across its pharmacy, distribution
centers, stores and e-commerce, according to a company blog. By analyzing the robust information flowing
throughout its operations, the discounter has gained a real-time view of workflow across its pharmacy,
distribution centers, stores and e-commerce, according to a company blog. Big data volume continues to
grow, but Walmart is using it to the company's – and its customers' – advantage. By analyzing the robust
information flowing throughout its operations, the discounter has gained a real-time view of workflow
across its pharmacy, distribution centers, stores and e-commerce, according to a company blog.
Here are five ways that Walmart is using big data to enhance, optimize and customize the shopping
experience:
1. To make Walmart pharmacies more efficient. By analyzing simulations, the discount giant can understand
how many prescriptions are filled in a day, and determine the busiest times during each day or month. Big
data also helps Walmart schedule associates more efficiently, and reduce the time and labor needed to fill
perceptions.
```

Fig 4.4: Output

In Fig.4.4 the output of the above frames comes with 7.3\% repetition and 92.7\% original text in the second frame. All the text which was present in frame 2 from frame 1 has been omitted, thus the final text file which will be generated will have maximum unique and minimum redundancy needed by the user.

Chapter-5

5.1 Conclusion

We started by extracting key-frames which contain most of the data related to the text part of the video, but, due to high similarity in consecutive frames, we drop the idea for future scope.We extracted frames every 5 seconds. OCR was applied to the given frames to convert text from digital form into string format for further processing.

The text was compared among consecutive frames to remove redundant text which is common in both frames so that maximum unique content could be delivered into the final text file for the user. Redundancy was removed using a special algorithm which compares how much of the text in consecutive frames are similar. After calculating the overlapping rate which is metric the text in the next frame is removed.

Higher the overlapping rate means that both the frames are too similar so there is no need to extract the text from the current frame. Lower the optimality rate means that text in two consecutive frames is not too much similar so text from the current frames will be extracted. This will help the students to remove the redundancy in their note and further optimise it on their own.

Applying the given model on the videos we were able to successfully extract text with 10% overlap which can be removed by the user manually. Videos having objects in between the text for example teacher's finger or any other obstacle can not be processed due to limitations of OCR and phase III equation limitations. All the videos which need to be processed should be free of any kind of obstacle to be processed smoothly. If any frame contains a complex figure then that figure will not be able to convert into text and students will have to draw the figure on their own. Only the caption of the figure will be generated through OCR.

5.2 Future Scope

Key-Frame extraction method could be improved and will work with videos containing images to extract frames which are most data. With better OCR technology we can also include videos which contain images and videos in which a person is teaching on board with repetition aim of less than 5%. In coming era as we are relying more on technology, students will less likely make handwritten notes. Note making will be done through artificial intelligence with more advanced features.

Future scope can be broadly divided into three categories:

1. At the moment our project does not handle presentation with human body involvement that is if a human appears in front of presentation it will be a huge issue for us as we will not be able to get proper text but in future we would like to change it such that we can choose frames such that it involves minimum presence of human body and hence improving quality of notes.

2. There can be cases where instead of teaching through presentations, blackboard is being used. In this case our priority is to identify the handwriting of the teacher and then convert each note provided by the teacher into separate notes.

3. Along with the frame extraction our project can be expanded to also convert the speech of the instructor into text. For example consider that the teacher takes a pause and is giving a speech now that speech can be converted into text. From converted speech only the optimal part will be taken and can be merged with the optimal notes generated from the frames of the video. This will give the complete note of the entire lecture.

REFERENCES

[1] Karez Hamad,Mehmet Kaya."**A Detailed Analysis of Optical Character Recognition Technology**". International Journal of Applied Mathematics Electronics and Computers 4(Special Issue-1):244-244.

[2] Sheena, N.K.Narayanan. "**Key-frame Extraction by Analysis of Histograms of Video Frames Using Statistical Methods.**"

[3] Yunyu Shi, Haisheng Yang, Ming Gong, Xiang Liu, and Yongxiang Xia."**A Fast and Robust Key Frame Extraction Method for Video Copyright Protection.**"

[4] Gianluigi Ciocca Raimondo Schettini. " **Dynamic key-frame extraction for video summarization**". The International Society for Optical Engineering.

[5] "**KEY FRAME EXTRACTION METHODS**" By Israa Hadi Ali and Talib T Al-Fatlawi

[6] Z. Lu and Y. Shi, —**Fast video shot boundary detection based on SVD and pattern matching,**|| IEEE Trans. Image Processing, Vol. 22, No. 12, 2013, pp. 5136-5145.

[7] Rachida Hannane, Abdessamad Elboushaki, Karim Abdel 1, P. Naghabhushan
and Mohammed Javed, —**An efficient method for video shot boundary detection and keyframe extraction using SIFT-point distribution histogram**||, International Journal of Multimedia Information Retrieval, Vol. 5, No. 2, 2016,pp. 89-104

[8] W. Hu, N. Xie, L. Li, X. Zeng, and S. Maybank, —**A survey on visual content-based video indexing and retrieval**||, IEEE Transactions on Systems, Man, and Cybernetics—part c: Applications and Reviews, Vol. 41, No. 6, 2011, pp. 797 819.

[9] T. He, W. Huang, Y. Qiao, and J. Yao, "**Text-attentional convolutional neural network for scene text detection,**" IEEE Transactions on Image Processing, vol. 25, no. 6, pp. 2529–2541, 2016.

[10] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "**Reading text in the wild with convolutional neural networks,**" International Journal of Computer Vision.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "**Deep residual learning for image recognition,**" in CVPR, 2016.

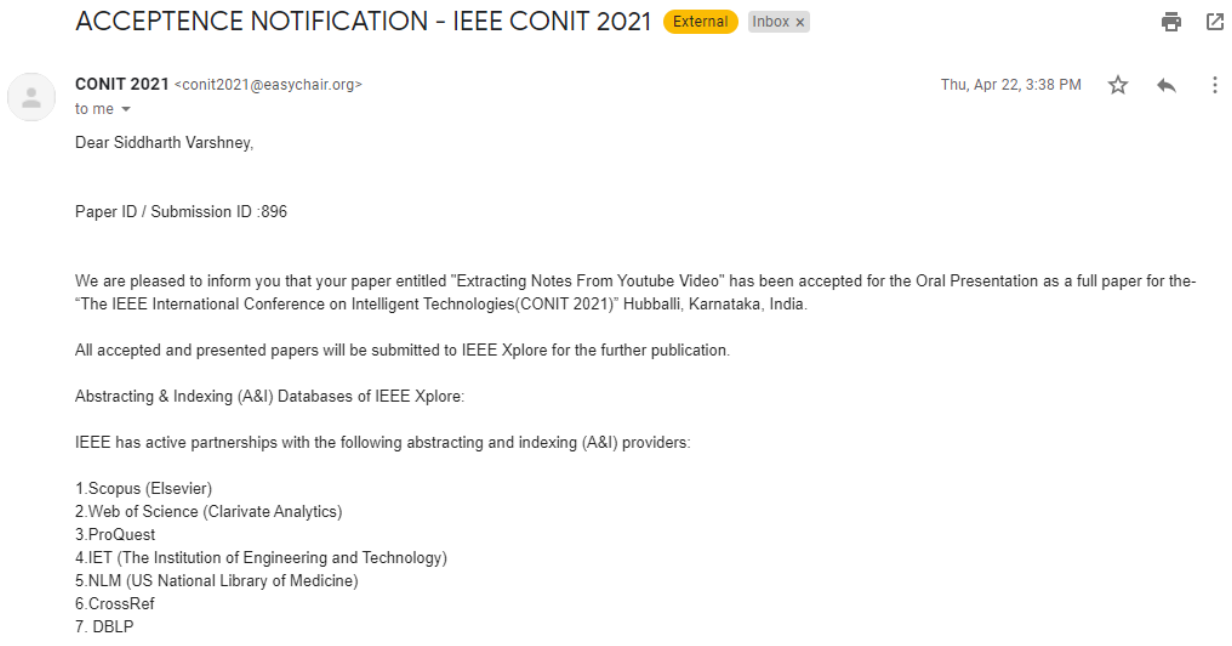
[12] I. Sutskever, O. Vinyals, and Q. V. Le, "**Sequence to sequence learning with neural networks**," in Advances in neural information processing systems.

[13] **An Overview of the Tesseract OCR Engine**. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/33418.pdf

[14] https://opencv-python-tutroals.readthedocs.io/en/latest/py tutorials/py feature2d/py orb/py orb.html

[15] **Structural Similarity** https://en.wikipedia.org/wiki/Structural similarity

Paper Acceptance Proofs



ACCEPTENCE NOTIFICATION - IEEE CONIT 2021 External Inbox ×

CONIT 2021 <conit2021@easychair.org>
to me
Thu, Apr 22, 3:38 PM

Dear Siddharth Varshney,

Paper ID / Submission ID :896

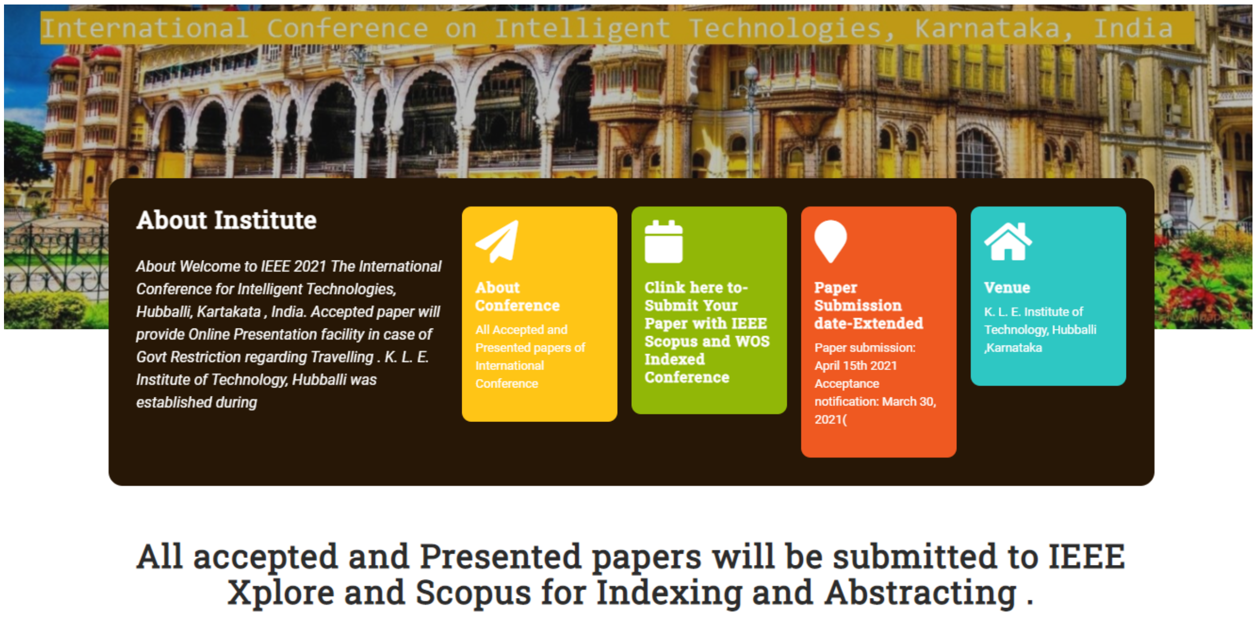
We are pleased to inform you that your paper entitled "Extracting Notes From Youtube Video" has been accepted for the Oral Presentation as a full paper for the- "The IEEE International Conference on Intelligent Technologies(CONIT 2021)" Hubballi, Karnataka, India.

All accepted and presented papers will be submitted to IEEE Xplore for the further publication.

Abstracting & Indexing (A&I) Databases of IEEE Xplore:

IEEE has active partnerships with the following abstracting and indexing (A&I) providers:

1.Scopus (Elsevier)
2.Web of Science (Clarivate Analytics)
3.ProQuest
4.IET (The Institution of Engineering and Technology)
5.NLM (US National Library of Medicine)
6.CrossRef
7. DBLP



International Conference on Intelligent Technologies, Karnataka, India

About Institute

About Welcome to IEEE 2021 The International Conference for Intelligent Technologies, Hubballi, Kartakata , India. Accepted paper will provide Online Presentation facility in case of Govt Restriction regarding Travelling . K. L. E. Institute of Technology, Hubballi was established during

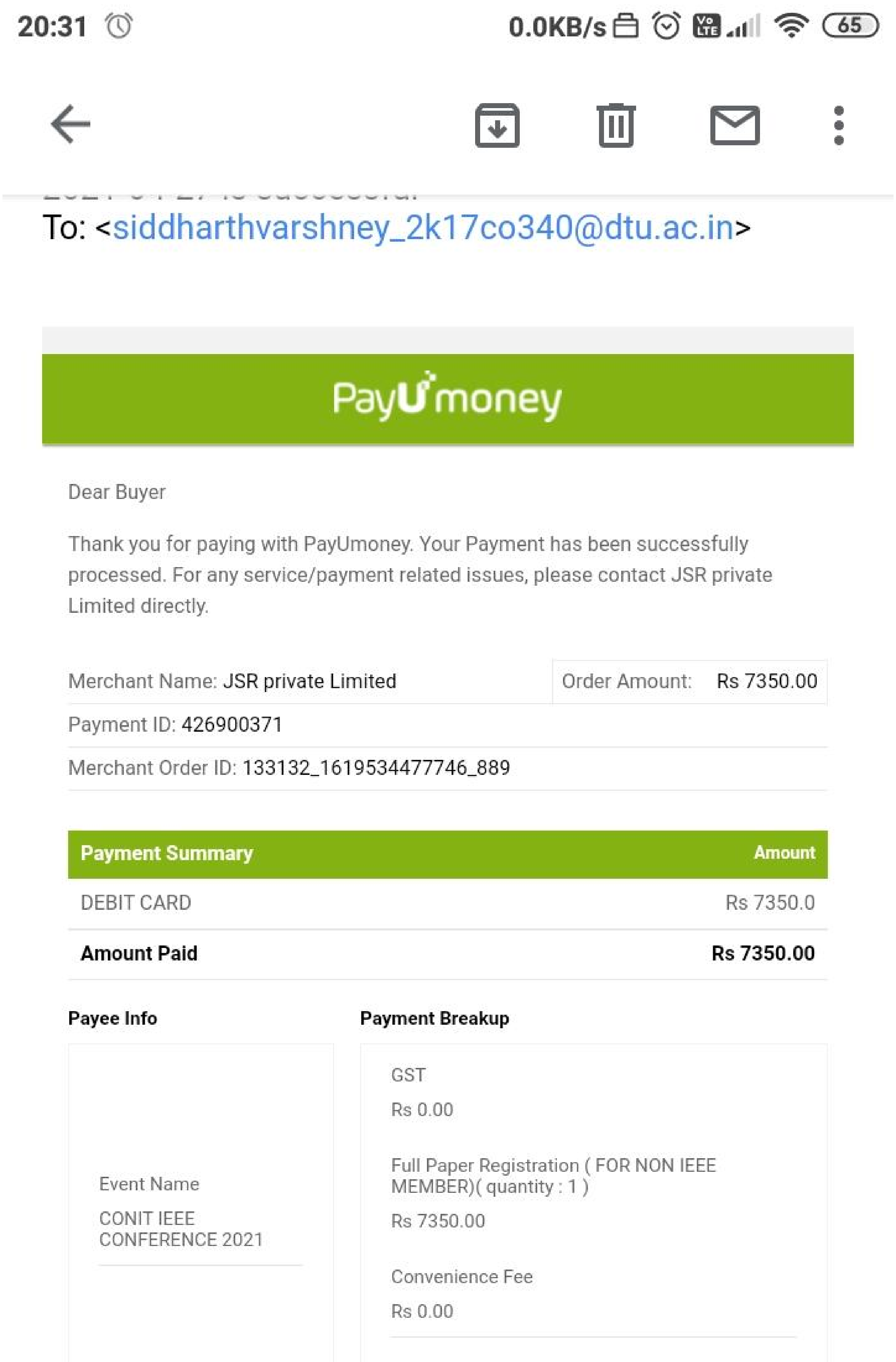
About Conference
All Accepted and Presented papers of International Conference

Clink here to- Submit Your Paper with IEEE Scopus and WOS Indexed Conference

Paper Submission date-Extended
Paper submission: April 15th 2021
Acceptance notification: March 30, 2021(

Venue
K. L. E. Institute of Technology, Hubballi ,Karnataka

All accepted and Presented papers will be submitted to IEEE Xplore and Scopus for Indexing and Abstracting .



20:31 0.0KB/s 65

To: <siddharthvarshney_2k17co340@dtu.ac.in>

PayUmoney

Dear Buyer

Thank you for paying with PayUmoney. Your Payment has been successfully processed. For any service/payment related issues, please contact JSR private Limited directly.

Merchant Name: JSR private Limited	Order Amount: Rs 7350.00
Payment ID: 426900371	
Merchant Order ID: 133132_1619534477746_889	

Payment Summary	Amount
DEBIT CARD	Rs 7350.0
Amount Paid	**Rs 7350.00**

Payee Info

Event Name

CONIT IEEE CONFERENCE 2021

Payment Breakup

GST

Rs 0.00

Full Paper Registration (FOR NON IEEE MEMBER)(quantity : 1)

Rs 7350.00

Convenience Fee

Rs 0.00