2021 International Conference on Intelligent Technologies (CONIT)
Karnataka, India. June 25-27, 2021

Extracting Notes From Youtube Video

Manoj Kumar
Department Of Computer Engineering
Delhi Technological University
New Delhi, India
mkumarg@dce.ac.in

Sumit Gaurav
Department of Computer Engineering
Delhi Technological University
New Delhi, India
sumitgaurav_2k17co349@dtu.ac.in

Suhail Akhtar
Department of Computer Engineering
Delhi Technological University
New Delhi, India
suhailakhtar_2k17co345@dtu.ac.in

Siddharth Varshney
Department of Computer Engineering
Delhi Technological University
New Delhi, India
siddharthvarshney_2k17co340@dtu.ac.in

2021 International Conference on Intelligent Technologies (CONIT) | 978-1-7281-8583-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CONIT51480.2021.9498392

***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 given frame or how to handle cases when some person come in between the video. All this challenges will be major challenges in the future and will require a solution our idea is the first step towards this future .**

***Index Terms*—YouTube, Key-Frame extraction, Optical Character Recognition, String matching, Image Structural Similarity**

I. Introduction

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.

Several steps have been performed in creating the program. The first step is extracting frames from a video that would be most refined or we say frames that contain the most useful content to be written in notes. We have calculated the Frame Rate per second.

This method is known as Key-Frame extraction. This method 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. Thus, we are extracting frames at a regular time interval.

The second step of the project is using Optical Character Reader technology. Optical Character Reader is also known as OCR in short is the conversion of graphical characters present in a sentence or in any form in an image such as scanned image, clicked the image, frame captured from a video to digital format of the same characters, and sentences. The need for the technique arose as to when archaeologist

were converting old manuscripts in digital format, it took a huge amount of time for such a task manually, there was a need to automate this task to save time. In our project, we have used a google python library named Tesseract as an OCR. Tesseract is free, modifiable, and gives us fairly good results on simple English text images. There are some defects this open-source library can face, such as it gives poor results in the colored background and stilted sentences. These defects are removed by binarization and image rotation during the image preprocessing phase before OCR. We have excluded these techniques.

This OCR-converted digital text is not totally unique in content between multiple frames. Redundancy has to be removed from the digital text that is going to be printed in the notes. The third step will contain an algorithm that is created which compares text from consecutive previous frames and removes the common data present in the previous frame from the current frame text.

II. METHODOLOGY

A. FRAME EXTRACTION

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 rotate 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]

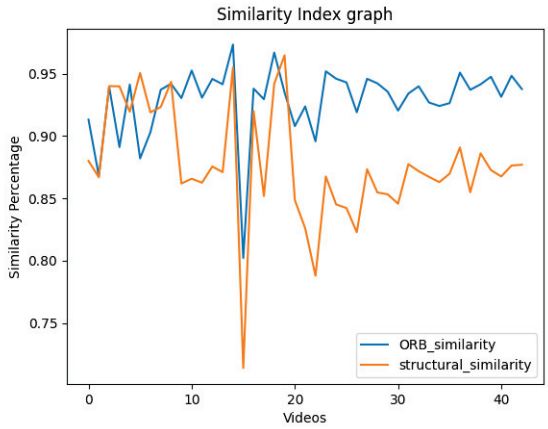

Fig. 1. Similarity Index Graph

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.

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.

TABLE I
FRAME SIMILARITY TABLE

Algorithms / video_types	SSIM	ORB_Sim
No_Image_videos	0.92	0.87
Image_Videos	0.52	0.44

B. Optical Character Reader

Tesseract is the library used for reading characters from image and converting into digital format.

Tesseract OCR:-
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.

Fig. 2. Figure shows how baseline are created.

As you can see it has extracted the text in red. Some of the methods used by tesseract library to detect texts are discussed below:-

Baseline fitting: It detects the lines which are straightly written by partitioning the line with the reasonably equidistant line for the original straight line.[13]

Fixed pitch detection: This is another method which is used by the tesseract library. In this it scans the entire line to check whether they are fixed pitch or not. Whenever it detects the fixed pitch texts, it slices the words into individual characters with the help of pitch. Following which it then disables other methods on it like associator and then chopper which helps it in finding given words correctly. Following figure will illustrate it perfectly.[13]

There are many more methods which are used by tesseract but these are the early ones and form the foundation of tesseract library.

C. Identifying New Sentences Between Two Slides

When we see any lecture video online we often see text on board changes gradually that is we don't see a 100% switch in text and so if we just copy everything from slide to notes it will contain a very high amount of duplicate text so in order to obtain new text from newer slide we require an algorithm to do so.
So to understand our algorithm let us define few terms :-

String1 :- It represents the text we got from the last frame.
String2 :- It represents the text we got from the current frame.
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:-"ssumit"
Uni stores:- s-2,u-1,m-1,i-1 and t-1

Di :- It represents how many times a pair of characters has appeared in a string.
Example
string:-"siddd"
Di stores:- si-1,id-1 and dd-2

Tri :- It represents how many times a pair of three characters has appeared in the string.
Example
string:-"suhauha"
Tri stores:- suh-1, uha-2, hau-1 and auh-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.

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.

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

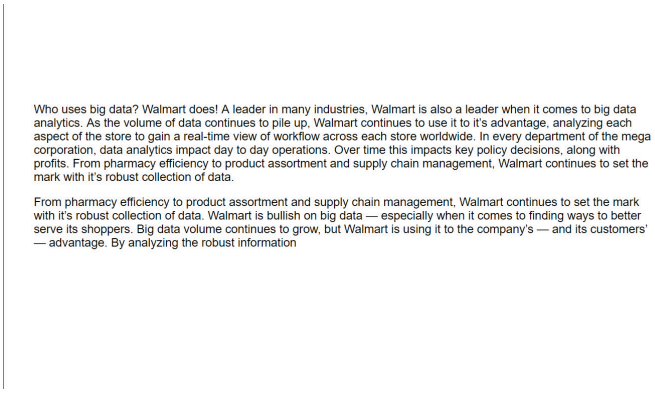

Fig. 3. Frame 1

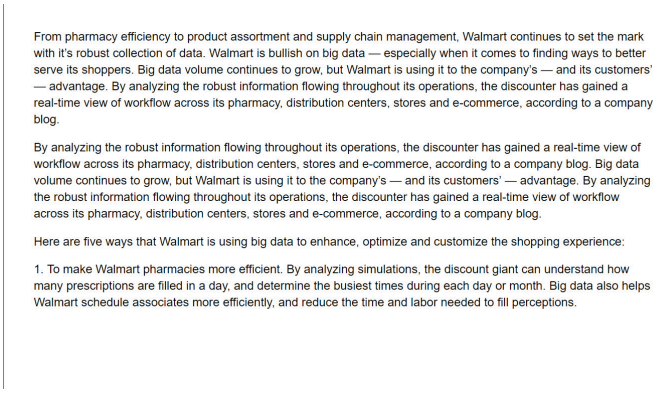

Fig. 4. Frame 2

Above consecutive frames give the following output:

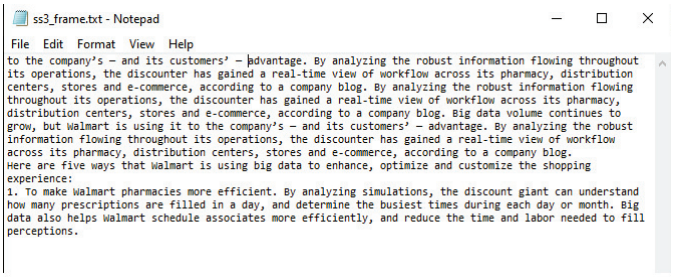

Fig. 5. Frame 3

In Fig.5 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.

III. RESULTS

A. *Comparing our output with original text:-*

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. Though this may sound simple but it is 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 but 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

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

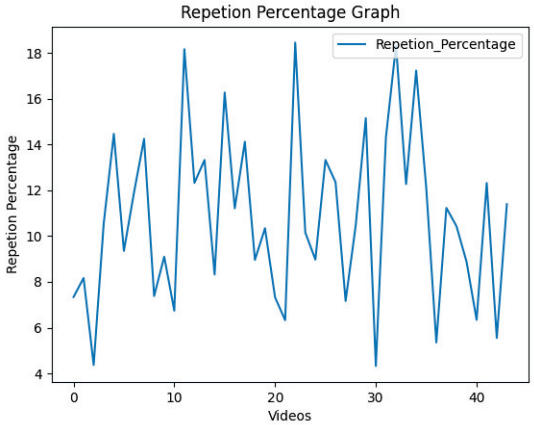

Fig. 6. Frame 1

For 45 videos, original text files and converted text files were compared according to the method above, giving us an average of about 10.8% repetition in text which frames originally had much more repetition.

Repetition varies from 4% to 19% across 45 videos. This means that all the necessary data is present in the text file present in the video with a little more due to repetition from consecutive frames.

IV. CONCLUSION

We started by extracting key-frames which contain most of the data related to the text part of 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. 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 containing graphical contents can not be processed due to OCR and phase III equation limitations.

V. 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 done through artificial intelligence with more advanced feature.

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