WebIR Project: Lecture Video Info Retrieval Application

Yoke Kai Wen, 2020280598

June 1, 2021

Abstract

In this project, I created a lecture video information retrieval application. It allows users to quickly and conveniently locate the video segments within a long lecture video which are relevant to their search query. The application uses OCR to extract visual text from an uploaded PowerPoint slides based lecture video which are then used to index and score the video segments relative to a query. This can potentially improve the learning experience and efficiency of students studying online, which has become a dominant trend due to the ongoing pandemic and the advancement in video and communication technologies.

1 Background and Motivation

Lectures are increasingly delivered online, especially with improved technology such as recording equipment, video compression techniques and high-speed networks, and also with the ongoing global pandemic. However, lecture videos tend to be long, and sometimes the student is only interested in a particular video segment where a certain topic is being covered, for example while reviewing the lecture video. It can take quite a lot of trial and error to find the correct timing in the video, which is frustrating and inefficient for studying. Therefore, it is useful to create an application to allow students to conveniently locate the video segments in a long lecture video by entering a search query.

2 Related Work and Project Scope

In order to create a lecture video information retrieval application, I referred to related work in video retrieval from video databases, particularly work focusing on lecture webcasts [1] [2].

2.1 General video retrieval

In traditional video retrieval, typically video metadata such as title and brief description that are manually created by a human is relied on. However, manual annotation is time and cost consuming, particularly if annotation is required for segments within a video. Therefore, automatic annotation based on video contents is necessary, i.e. content-based video retrieval. Useful video content information in the modes of speech, video text and other visual features have been leveraged on to annotate and index videos.

2.2 Lecture video retrieval

Compared to videos in general, lecture videos tend to contain more textual and speech information, which are also the main information of interest to students. Hence, [1] focuses on retrieving lecture videos using speech and video text information with Automatic Speech Recognition

(ASR) and Optical Character Recognition (OCR). On the other hand, [2] only uses OCR to recognise text on PowerPoint slides since ASR is at the moment error-prone and highly dependent on speech clarity and production quality.

Lecture videos are also produced in various formats. For instance, it could be a PowerPoint slide screen-share with synchronous audio; a whiteboard with the lecturer writing and speaking on it; a multi-scene format with the speaker, presentation and whiteboard presented on one screen. Both [1] and [2] focuses on lecture videos with the PowerPoint slides displayed clearly, and develops algorithms to extract the screen before performing textual extraction.

2.3 Project Scope

I limit my project scope to (1) relying only on OCR to extract video text, and (2) only on lecture videos in the format of PowerPoint slide screen-shares. This is because highly accurate off-the-shelf deep learning based OCR models are easily available, while ASR frameworks tend to be less reliable. For effective information retrieval, it is vital for text and speech extraction to be highly accurate, hence I decided to abandon rich speech information and to rely solely on textual information. The reason why I restrict lecture videos to only PowerPoint screen-shares is because lecture videos come in a wide variety of formats, while I have limited time for this project, hence I chose to focus on the simplest and cleanest lecture format, so that only minimal preprocessing is required before I can perform OCR on the lecture video frames.

3 System Design Overview

The entire system design can be divided into three subsystems: (1) Video Analysis, (2) Indexing and Ranking, (3) GUI design, with details displayed in Figure 1 as seen below.

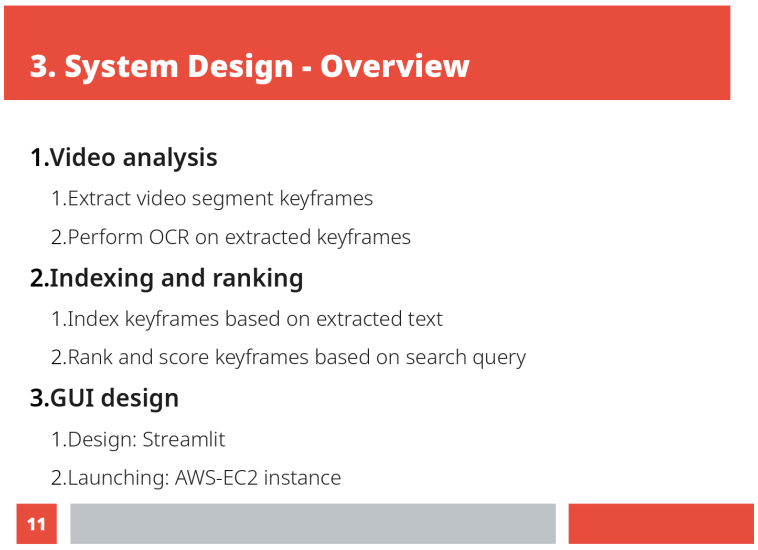

Figure 1: System design overview

4 Technical details

4.1 Video Analysis

Video analysis comprises two main steps: (1) Video slide keyframe extraction and (2) OCR.

Keyframe refers to a frame of a video segment that is most representative of that segment. It is necessary to extract keyframes before performing OCR, for a number of reasons. Firstly, OCR is an expensive process, and a lot of time will be required to OCR every single frame in lecture video (a 1.5h lecture video at 25fps will have 135k frames). Secondly, most of the consecutive frames will have little differences from each other, as lecturers tend to pause quite a long while at a certain slide, therefore selecting one representative keyframe avoids redundancy. Thirdly, students will be interested in finding the start of a video segment that covers a certain topic, hence it is more sensible to extract the first frame that changes as the keyframe of that segment.

After extracting the keyframes, OCR is then performed with the Tesseract v4 model which localises and recognises text automatically.

4.1.1 Video slide keyframe extraction

The key assumption behind keyframe extraction is that we assume each new slide represents a new topic i.e. a video segment of interest. I also adopt several rules of thumb to select keyframes. Since the PowerPoint slide is usually in the centre of the screen, with toolbars and speaker screens at the sides, I first extract the centre area of the frame which presumably contains only the PowerPoint slide, without containing speaker or other noisy movement. Then, I take the pixel difference between the current frame and the previous keyframe, to detect if the current frame is different from the previous keyframe. To determine if the frame difference is big enough, I further apply blurring, thresholding and contour detection on the frame difference, such that only when enough contours of large enough area are present, will the current frame be registered as a new keyframe. I also add another rule of thumb - the slide has to be constant for at least 3s for the keyframe to be registered, which makes sense as a video segment of interest has to last for a significant duration. Figure 2 illustrates how keyframes are selected based on frame differences.

Figure 2: Selecting keyframes from frame differences

4.1.2 OCR

To perform OCR, I used the Tesseract v4 pretrained deep learning model to detect and recognise text. Tesseract v4 allows configuration of settings, particularly the page segmentation mode (psm) that can dramatically influence OCR results. Figure 3 shows the OCR output with psm mode 6 which I set for this application.

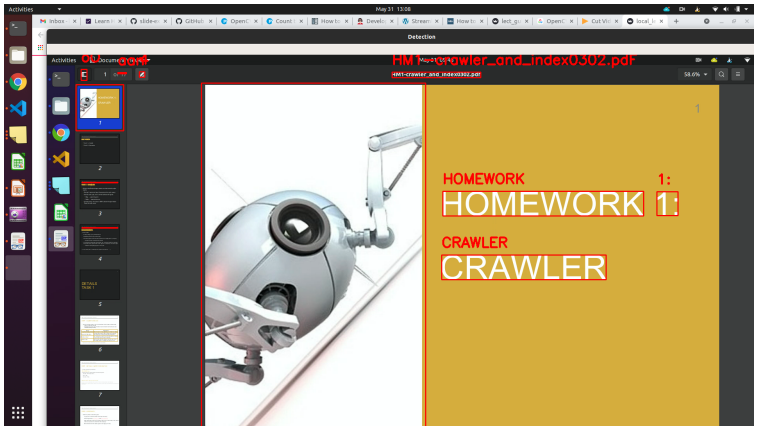

Figure 3: OCR sample output on lecture video frame

4.2 Indexing and Ranking

For indexing and ranking, I utilised the Whoosh python library that simplifies the indexing and ranking process.

4.2.1 Indexing

Using Whoosh, I was able to easily index the documents (extracted keyframes) based on the configured schema - I could choose to weight text fields (title, content, captions etc) differently. For this application, I did not differentiate between the OCRed text, although I could potentially separate the extracted text into different fields based on their position and font size on the lecture slide. This would probably improve search results.

4.2.2 Ranking

Whoosh supports several ranking algorithms, including frequency, TF-IDF, BM25 and cosine scoring. I used BM25 as the default for this project.

4.3 GUI Design

To design the GUI, I used the Streamlit python library, which converts python code into a beautiful and interactive UI with very few lines of code. I used merely 70 lines to design the GUI, showing that Streamlit is very quick for prototyping a GUI. Subsequently, I launched my application on an AWS-EC2 instance. In that server, I stored a preprocessed lecture video `0510_-week12.mp4` (on Big Data Stream Processing, 2.5hrs) and a short demo video `demo.mp4` (10s). The GUI is able to demonstrate the search function - when the user enters a search query and the desired number of results, the top N keyframes will be shown, and the lecture video will also

be started at the top keyframe timestamp. The GUI also demonstrates the preprocessing function on `demo.mp4` and outputs the first OCRed keyframe upon successful preprocessing. The demo application can be accessed from http://54.169.99.59:8501/, and Figure 4 shows a screenshot of the GUI at work.

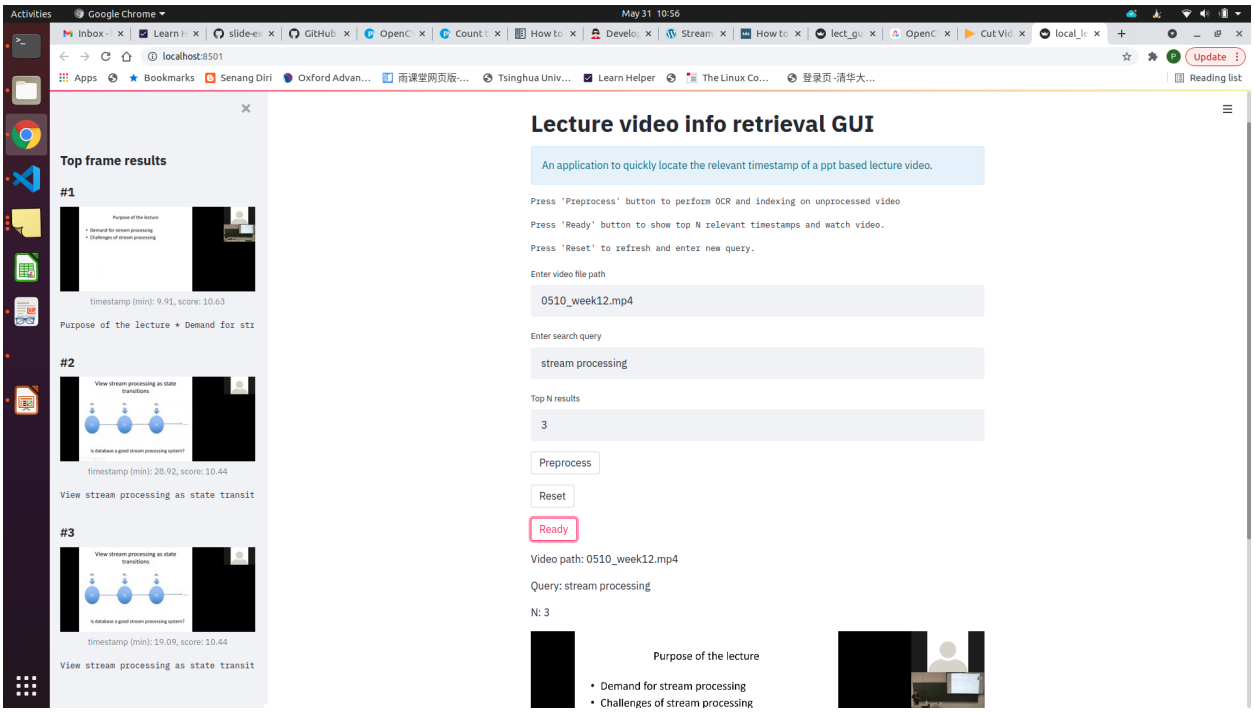

Figure 4: Lecture video info retrieval application GUI

5 Evaluation and limitations

Due to time constraints, I only managed to create a very rudimentary lecture video info retrieval application, that demonstrates basic functions such as search and preprocessing. As a result, there are many areas that can be further improved.

5.1 Limitations of video analysis

5.1.1 Slow preprocessing

Firstly, preprocessing of the video is very slow, roughly 60 frames are processed per second, meaning that an 1.5hr video would require 30min for processing. This can be attributed to a few reasons: (1) the OCR is currently running on CPU instead of GPU; (2) video segment keyframe extraction generates too many false positives, resulting in many wrongly identified keyframes that are unnecessarily OCRed, thus consuming extra time.

5.1.2 Video segmentation algorithm not robust

Secondly, video segmentation requires manual parameter tuning (minimum contour area, minimum number of contours etc) which is time consuming. This also means that video segmentation will fail when the lecture video format is slightly changed, for instance if the PowerPoint

slide takes up half the screen while the speaker takes up the other half. Furthermore, the current method of video segmentation that is based on global pixel differences is very crude and might not be the most accurate way to identify distinct slides. Right now, I also have not considered slide animation, so when the slide is animated, the algorithm detects a new slide even though the slide is still the same one. Hence, the robustness of the video segmentation algorithm needs to be improved.

5.1.3 OCR inaccuracies

Another significant issue is that I did not constrain the slide area on which OCR is applied to, leading to extraneous text outside the PowerPoint slide being identified. Hence, I should write an accurate slide detection algorithm and extract the slide out before applying OCR for more accurate text recognition.

5.2 Limitations of indexing and ranking

I did not put in much effort into indexing and ranking and merely used default methods, because it is far more important to ensure accurate OCR, otherwise the indexing would not work. However, this part is still important for effective search results.

5.2.1 No differentiation between extracted text

In the current algorithm, I treat all the extracted text the same. However, we know that text that are bigger, or located at the top of the slide etc tend to be more important and hence should hold a higher weight during indexing. Therefore, extracted text should be differentiated based on their location and sizes, in order to return more relevant keyframe search results.

5.2.2 Query function not robust

The current algorithm only works if the query text matches exactly with the text extracted from the keyframes. It fails if there is a spelling error in the query or in the OCRed text; if the query is a synonym but does not appear exactly in the OCRed text the query also fails. To make the query function could be made more robust by understanding query and text semantics through NLP word vectors, and with spelling autocorrect functions.

6 Conclusion and future work

In conclusion, I implemented a simple application to search for relevant segments of a lecture video. Other than the limitations discussed in the previous sections that could be improved on, there are also some ways this work could be extended.

6.1 Leveraging other data modes for keyframe indexing

Speech is also an important source of lecture information that could be exploited for frame indexing, if it can be automatically recognised accurately. Other than PowerPoint slide text, handwriting on whiteboards could also be exploited.

6.2 Generalising to other lecture video formats

The algorithm can be expanded to work also on non-PowerPoint based lectures, such as white-board based lectures and multi-scene lectures comprising PowerPoint, whiteboard and the speaker.

6.3 Retrieval of lecture video from a video database

Instead of only searching within a lecture video for the relevant video segment, it is much more useful to find the desired video segment from a large lecture video database.

References

[1] H. Yang and C. Meinel, "Content based lecture video retrieval using speech and video text information," *IEEE Transactions on Learning Technologies*, vol. 7, no. 2, pp. 142–154, 2014.

[2] J. Adcock, M. Cooper, L. Denoue, H. Pirsiavash, and L. A. Rowe, "Talkminer: a lecture webcast search engine," in *Proceedings of the 18th International Conference on Multimedia 2010, Firenze, Italy, October 25-29, 2010*, A. D. Bimbo, S. Chang, and A. W. M. Smeulders, Eds. ACM, 2010, pp. 241–250. [Online]. Available: https://doi.org/10.1145/1873951.1873986