Redesigning the Online Video Lecture Player to Promote Active Learning

Ian Walk
Computer Science
University of Virginia
Charlottesville, Virginia USA
imw6jy@virginia.edu

Arnold Yim
Math and Computer Science
Bridgewater College
Bridgewater, Virginia USA
ayim@bridgewater.edu

Ed Novak
Computer Science
Franklin and Marshall College
Lancaster, Pennsylvania USA
enovak@fandm.edu

Charles Reiss, Daniel Graham
Computer Science
University of Virginia
Charlottesville, Virginia USA
{cr4bd, dgg6b}@virginia.edu

***Abstract*—This Research to Practice Work-In-Progress paper examines video lecture interactions and engagement boosting techniques. Posting video lectures online allows lecturers to extend their impact beyond the classroom. However, conventional video players may not be the best way to distribute lecture video content. When we looked at audience retention data for videos, we found that the watch patterns for lecture videos differed drastically from the watch patterns observed for non-lecture videos. In particular, we found that students were more likely to replay or skip sections of lecture videos than they were with non-lecture videos. Given these differences in viewing patterns, we created a new custom video player that is better suited for lectures to enhance student learning and to give instructors valuable feedback. Our custom video player has three main features: it logs student interactions to help instructors identify topics that students might struggle with, it allows students to search for key words in the video, and it has an integrated quiz tool to enhance active learning.**

***Index Terms*—data-sets, video player, pedagogy, online learning, teaching with technology**

I. Introduction

As the internet becomes ever more integral to all levels of education, many professors have began recording and posting their lectures online [9]. Unlike in-person lectures, students now have the option to control the pace of the presentation: they are able to pause, skip ahead, re-watch, play at increased speed, or simply stop watching. Since students engage and interact with the video content, we want to design a video player that promotes student learning and is easy to interact with.

To understand how students watch video lectures, we first analyzed audience retention on videos uploaded to the popular video sharing service YouTube. YouTube defines audience retention: "Audience retention helps you see how often each moment of your video is being watched as a percentage of total views. Rewinding and re-watching can result in values higher than 100%." For non-lecture videos, audience retention tends to steadily drop off over the course of the video. On the other hand, we observed that audience retention for lecture videos tends to have spikes in certain sections. Since students do not consume lecture content the same way that they consume other online videos, we concluded that distributing lecture videos using conventional players may not be the best approach. In this paper, we use a collection of YouTube analysis and instrumentation of a custom, browser-based video-player to analyze student watch behavior. We then compared data gathered from 24 lecture videos and 3 YouTube videos.

A survey of research on the use of video podcasts in education between 2002 to 2011 indicated that videos improved students' learning and perception [6]. Other studies have surveyed research on how to make video content more engaging [5]. However, in this work we focus on what aspects of the video player itself can encourage student engagement. Alqurashi proposed 5 strategies for improving video lectures [1]. Since two of the strategies are related to the actual content of the video, we focused on the remaining three strategies. These strategies are incorporated into our new custom video player:

- Assess student understanding
- Incorporate active-learning techniques
- Divide large lecture(s) into smaller segments

While YouTube's data on audience retention gives instructors a rough idea of which sections of a video are most watched, it does not paint a complete picture of how a student engages with the content. Our video player logs various user events such as play, pause, and seeking. This data can then be used by the instructor to identify topics that students spend the most time with. Our video player also allows instructors to incorporate quizzes and questions seamlessly into their video lectures. This feature not only promotes active learning, but it also helps the instructor assess student understanding of the material. Additionally, students have the ability to search for keywords in a lecture and jump directly to particular sections of the video. This feature allows students to actively seek out topics that they may be struggling with. While many of these features are also available on Panopto Video Platform [3], our video player offers many features that help instructors assess student learning.

In this paper, we analyze the difference between lecture and non-lecture videos, we investigate the data obtained from using our video player for twenty four lectures in a Computer Architecture class, we describe the implementation of the features of our video player, and we discuss the potential benefits of using our video player over conventional players.

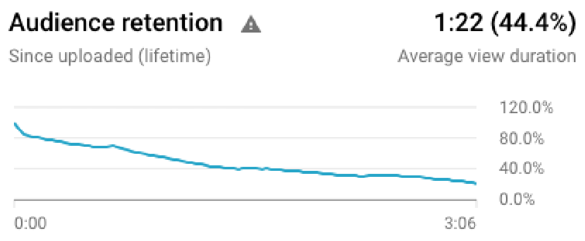


Fig. 1. Audience retention graph for drone flyover video

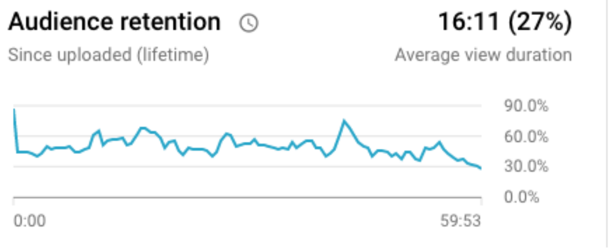


Fig. 2. Audience retention graph for computer architecture lecture

II. Analysis of Video Data

YouTube provides an analytic service that provides data on videos that you upload. Figure 1 shows an audience retention graph that was generated from video of a drone fly over of the William and Mary campus. The video was published in 2014 and currently has over 15 thousand views. The audience retention starts high, then declines over time.

The audience retention drastically differs for lecture videos. Figure 2 shows an audience retention graph that was generated for a computer architecture lecture. This video lecture was viewed over 387 times and produced a remarkably different audience retention graph. While there is a difference in video length between these two, longer videos (15min+) generally have lower audience retention. The graph shows that retention goes up and down throughout the video. This indicates that certain sections of the video were watched more than others. In an effort to ensure that difference in audience retention was not due to the number of views, we compared these results with a video that has only 900 views; Figure 3 shows the audience retention graph that was generated from a video demonstration of Sonar sensors for mobile devices. We noticed that even with a smaller view count, the audience retention graph resembled the one we saw in Figure 1.

The data shows us a difference in watch behavior and suggests that students are more active when consuming lecture content. While these audience retention graphs give instructors some understanding of which sections of the lectures are being watched, instructors could greatly benefit from more information like when the lecture is paused, which sections of the video students jump to, and when students stopped watching. With this in mind, we designed a JavaScript-based video player that not only encourages active learning for students, but also provides instructors with valuable, fine-

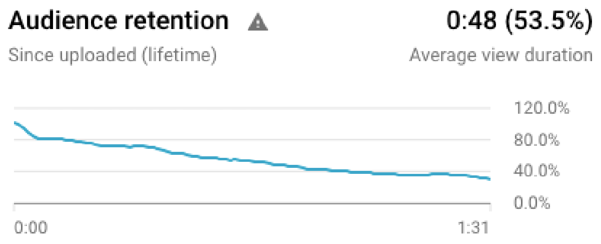


Fig. 3. Audience retention graph for video demonstration of Sonar sensor for mobile devices.

grained feedback.

III. Implementation of a Custom Video Player

Preliminary analysis of the video lectures in the previous section indicates a difference in watch behaviour depending on the type of video content. In order to understand the cause, we built a logging system that allows us to capture more data from the video player as it is used.

Our logging system is comprised of a AJAX JavaScript module that gets loaded on the same page as the video player. In our prototype implementation, the system requires that the professor hosts the video and JavaScript module on their own site, but different implementations are feasible. This allows us to attach the logging system to any video player that uses the HTML <video> tag. The logging script then captures the events summarized in Table I below. When any of these events are captured, a report is made containing all of the properties tracked in Table II.

Table I
Events tracked by the JavaScript logging system. The top events exist in HTML5 and are supported by any video player, while the bottom were written to trigger on user interactions specific to our player.

Event Tracked	Description
play	Played the video
pause	Paused the video
ended	Reached the end of the video
seek-ed	Selected a location in the video
volumeChanged	Changed the volume of the video
playbackRate	Selected a playback rate
showQuiz	Quiz was displayed to the user
exitQuiz	User exited the quiz
submitAnswer	User answered a quiz question
jumpToContent	User jumped our suggested location

Table II
Video properties tracked by the logging script.

current Time	Current location in video time
volume	Volume of the video player
playback Rate	Speed at which the video plays
paused	Track if the video is currently paused
event	Type of event that the user triggered
time	Time at which the event was triggered

The video player logging system is outlined in Figure 4. Our database is stored on MongoDB Atlas, while an API

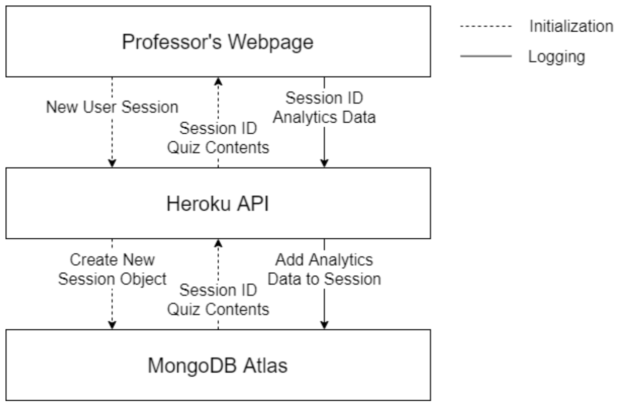


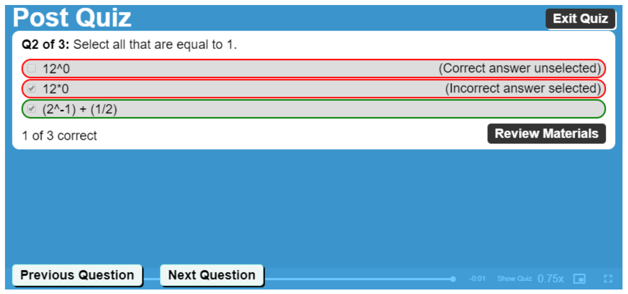
Fig. 4. Outline of the structure of the video logging system.

for limiting access to our database is run on Heroku. When a viewer requests a page with a video lecture a series of initialization steps are made. First, a request for a new session ID is made to our API. This request contains the source name (a URL) and length of the video in order to categorize the user sessions by lecture video. The Heroku API parses the request and instantiates a new Mongo object. The objects are stored in different collections according to the MD5 Hash of the video source name, in order to limit the size of the collection names while keeping them unique. The ID of the new object is returned to our JavaScript module and saved for use as a session ID. Subsequent interactions with the video player are captured by the JavaScript module and sent as logging requests to the Heroku API along with this session ID. Our Heroku API then appends these logs to the appropriate session object, building up a list of events triggered by that user session.

We implemented two additional features with the goal of increasing student engagement as advised by similar work [2] In the video player we inject: (1) integrated quizzes and (2) video script searching. When a user views a video, the database is searched for a corresponding quiz object. If a quiz is not found, the video player is unaffected, and no quiz can be accessed. Our quizzes consist of any number of multiple choice or multiple select questions. Users can access the quiz at any time via a dedicated button on the video player, and return to the same spot in the video when finished. The user can skip through or go back to questions without answering. When they make a selection and submit their answer to a question, instant feedback is provided showing which of their answers were correct. Additionally, a button is displayed which links to a time in the video that explains the question. Figure 5 shows a screenshot with the results for answering a multiple select question partially incorrect.

Quiz interactions are logged in the same system as described above, though the quiz content is stored locally and can be viewed and answered in its entirety without further connection required from the Heroku server.

From examining the data collected by our logging system, we believe that a significant use that students have for lecture videos is to find specific information. We believe that certain


Post Quiz

Q2 of 3: Select all that are equal to 1.

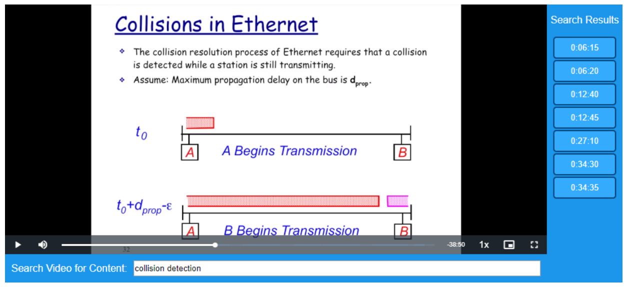
☐ 12^0 (Correct answer unselected)

✓ 12*0 (Incorrect answer selected)

✓ (2^-1) + (1/2)

1 of 3 correct

Fig. 5. Sample quiz question and answer feedback.


Collisions in Ethernet

- The collision resolution process of Ethernet requires that a collision is detected while a station is still transmitting.
- Assume: Maximum propagation delay on the bus is $\mathbf{d}_{prop}$.

Search Video for Content: collision detection

Fig. 6. Search bar and search results from a computer architecture lecture.

sections of the video are sought out by students in order for them to review an in class exercise, an important example, or a confusing topic. In order to make this process of finding information within a video easier for students, we created a mechanism for searching through *the audio* using text queries.

The search feature is implemented using a vector space model [12]. We first convert the lecture video into an MP3 file and then use the open source Python 3 module "SpeechRecognition" [14] and the open source speech recognition toolkit "CMUSphinx" [4] to generate a script from the audio file [10]. The script was generated in ten second chunks, which formed a list of phrases that had been said during lecture. A second pass of ten second chunks was made with a five second offset in order to catch words that were broken up by the first pass. The phrases generated by this process are treated as unique documents and stored as text in a hidden element on the site with the video.

In order to search this script, we convert the hidden lines of the script into a list of documents, which are each stored as a vector. When the user inserts a query, the query is also converted to vector form, before ranking our list of documents by cosine similarity with the user query. The times associated with the ten most similar documents are then returned to the user, though documents with no similarity will not be returned, so some queries will return less than ten results. Repeatedly entering the same query (by simply pressing enter again) will jump the position of the video to the time associated with the next best match, and a list of all times is displayed on the side of the video so that the user can skip directly to a time of their choosing. Figure 6 shows our user interface for searching through videos.

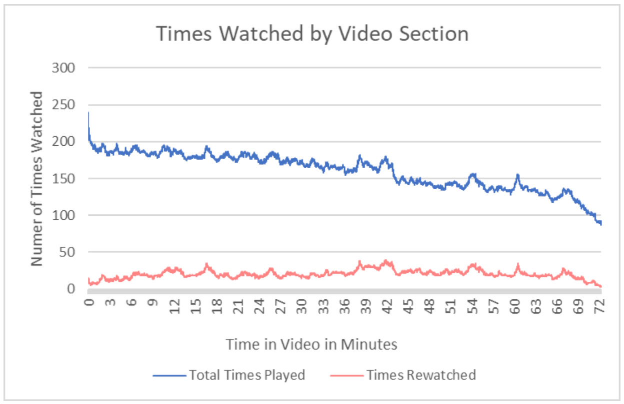


Fig. 7. Graph of video views by time in the video. We chart both the total number of views each segment of the video received, as well as the amount of times that each part of the video was re-watched.

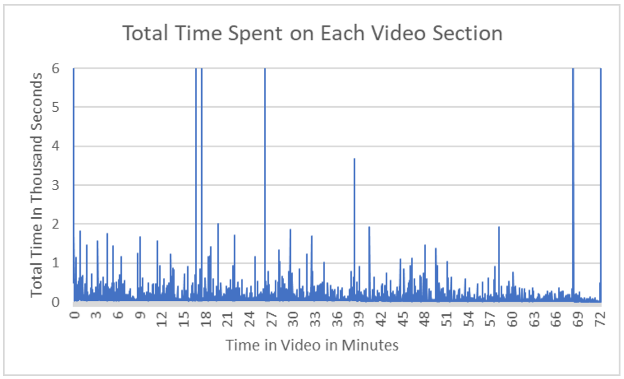


Fig. 8. Total time that was spent on each moment of the video. The spikes represent users who left the video paused for long periods of time.

IV. Analysis

When using conventional video players like YouTube, the viewer can only interact with the video content in limited ways. The features that we have implemented gives our students the ability to take control of their learning. Additionally, our video player provides instructors with valuable feedback.

One of the main features of our video player is its log of interactions. We piloted our video player for a computer architecture course taught at the University of Virginia during the 2019 fall semester. In our trial, twenty four lecture videos were made available to the students. Looking at the logs produced by our video player, we were able to confirm the watch behavior we noticed from YouTube's audience retention graph. For example, Figure 7 shows the number of times each section of a video was watched for one of the more popular lectures from the course. Like the audience retention graphs, the number of times students watch each segment of a lecture rises and falls. We also graphed the number of times segments of the video we re-watched. We found that this emphasized the spikes from our preliminary analysis. After examining the sections in the video corresponding to these spikes, we found that they typically corresponded to (ungraded) content quizzes or other interactive portions of the lecture.

Figure 8 shows the amount of time spent on each segment of the lecture video. This graph not only indicates which portions of the video were commonly re-watched, but also allows us to identify when students pause the video. We found that students often paused on slides with a lot of information and on slides with examples.

With this additional data, instructors can assess their student's understanding of the material better. The instructor can pinpoint the portions of the lecture that the students spend the most time consuming. This may indicate that a particular topic is challenging, and so the instructor can tailor in-class meetings or create more videos to clarify topics that the students are struggling with.

Our video player also allows instructors to incorporate quizzes into their lectures. In addition to providing feedback to the instructors, these quizzes help students actively engage with the material. The students receive instant feedback after answering the questions and can re-watch portions of the videos if they are missing key ideas.

Lastly, the search feature in our video player will be a useful tool for both students and instructors. This tool promotes active-learning in students since it encourages answer-seeking. With the search tool, students can easily jump to topics they are interested in or are struggling with. In a way, this feature also helps divide a large lecture into smaller segments since students can jump around to different topics any time they want. Instructors also benefit because they will be able to see what their students commonly search for, allowing them to adjust their their lectures to address topics in more detail.

V. State of the Art

The work most closely related to ours is that of Juho Kim et al. [8], in which they explored redesigning the video player for Massively Open Online Courses (MOOCs). These MOOCs are normally offered exclusively online. However, the video player presented in this paper was designed with data collected in a hybrid environment in which lectures where offered both in person and online. For the reader looking for a list of other video players that promote active learning, we have compiled a list here [7], [11], [13], [15].

VI. Future Work and Conclusion

To the best of the authors' knowledge, our system is the first to provide detailed logging of student interactions and automatic speech searching for lecture-content videos. The current video player system could see a variety of improvements and new applications. We provide viewer engagement metrics in a form that builds on those of YouTube. Further evaluation is needed to measure impacts on student engagement. Better speech to text models or training models on individual professors could be performed in order to improve the accuracy of script generation and, by extension, video searching. In general, a more thorough evaluation of the system is required, but is outside the scope of this work.

References

[1] Emtinan Alqurashi. Technology tools for teaching and learning in real time. In *Educational Technology and Resources for Synchronous Learning in Higher Education*, pages 255–278. IGI Global, 2019.

[2] Cynthia J. Brame. Effective educational videos: Principles and guidelines for maximizing student learning from video content. *CBE—Life Sciences Education*, 15(4):es6, 2016. PMID: 27789532.

[3] Eric Burns. Systems and methods for generation of composite video, February 2 2016. US Patent 9,251,852.

[4] Sphinx group at Carnegie Mellon University. Open source speech recognition toolkit: Cmusphinx, 2020. Available at: https://cmusphinx.github.io/.

[5] Philip Guo, Juho Kim, and Rob Rubin. How video production affects student engagement: An empirical study of mooc videos. pages 41–50, 03 2014.

[6] Robin H Kay. Exploring the use of video podcasts in education: A comprehensive review of the literature. *Computers in Human Behavior*, 28(3):820–831, 2012.

[7] Juho Kim, Elena L Glassman, Andrés Monroy-Hernández, and Meredith Ringel Morris. Rimes: Embedding interactive multimedia exercises in lecture videos. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 1535–1544, 2015.

[8] Juho Kim, Philip J. Guo, Carrie J. Cai, Shang-Wen (Daniel) Li, Krzysztof Z. Gajos, and Robert C. Miller. Data-driven interaction techniques for improving navigation of educational videos. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, page 563–572, New York, NY, USA, 2014. Association for Computing Machinery.

[9] Kyong-Jee Kim and Curtis J Bonk. The future of online teaching and learning in higher education. *Educause quarterly*, 29(4):22–30, 2006.

[10] Paul Lamere, Philip Kwok, William Walker, Evandro Gouvea, Rita Singh, Bhiksha Raj, and Peter Wolf. Design of the cmu sphinx-4 decoder. In *Eighth European Conference on Speech Communication and Technology*, 2003.

[11] Toni-Jan Keith Palma Monserrat, Yawen Li, Shengdong Zhao, and Xiang Cao. L. ive: an integrated interactive video-based learning environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3399–3402, 2014.

[12] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[13] Xiang Xiao and Jingtao Wang. Towards attentive, bi-directional mooc learning on mobile devices. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pages 163–170, 2015.

[14] Anthony Zhang. Speech recognition 3.8.1 python 3 module, 2020. Available at: https://github.com/Uberi/speech_recognition.

[15] B. Zhao, S. Xu, S. Lin, R. Wang, and X. Luo. A new visual interface for searching and navigating slide-based lecture videos. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 928–933, 2019.