

VideoDoc: Combining Videos and Lecture Notes for a Better Learning Experience

by

Rebecca P. Krosnick

S.B., Massachusetts Institute of Technology, 2014

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

September 2015

© 2015 Rebecca P. Krosnick. All rights reserved.

The author hereby grants to MIT permission to reproduce and to distribute publicly
paper and electronic copies of this thesis document in whole or in part in any
medium now known or hereafter created.

Author:

Department of Electrical Engineering and Computer Science
June 12, 2015

Certified by:

Robert C. Miller
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by:

Albert R. Meyer
Professor of Electrical Engineering and Computer Science
Chairman, Masters of Engineering Thesis Committee

VideoDoc: Combining Videos and Lecture Notes for a Better Learning Experience

by

Rebecca P. Krosnick

Submitted to the Department of Electrical Engineering and Computer Science on
June 12, 2015 in Partial Fulfillment of the Requirements for the Degree of Master of
Engineering in Electrical Engineering and Computer Science

Abstract

Videos provide learners an engaging way to learn material, but they are not easy to navigate. Electronic textbooks are easy to navigate and help learners review material they have already seen, but they are not very engaging. VideoDoc combines videos and textbooks to provide learners with a single resource that engages them and is easy to navigate. The interface can be played like a video or read like a textbook. Lecture videos and their corresponding transcripts are broken into sections by topic, and each section also has screenshots of representative video frames. A user can navigate the interface by scrolling through the sections or clicking on section titles in an interactive table of contents. A VideoDoc lecture is automatically generated from a time-annotated text transcript and a labeling of talking-head video frames, and an instructor can fine-tune section boundaries and add section titles using an editing interface. Through a user study we found that VideoDoc helped users more easily navigate lecture videos, but some users had trouble learning how to use features of the editing interface.

Thesis Supervisor: Robert C. Miller

Title: Professor of Electrical Engineering and Computer Science

Acknowledgements

I would like to thank:

- My outstanding advisor, Rob Miller, for his guidance, support, and interest. I appreciate the substantial amount of time he invested in me and this project even though I was a short-term member of his research group.
- Rob Miller and Elena Agapie for conceiving of the original idea of VideoDoc.
- Michele Pratusевич for working alongside me this year to build the computer vision parser that provides input to VideoDoc about people and lecture material. Thank you for your quick responses to my emails requesting JSON files.
- Juho Kim for offering advice and resources as I wrote my thesis proposal and designed my user study.
- My roommate and fellow UID member Elena Glassman for thoughtful conversation about research and life.
- The rest of the User Interface Design group, including Max Goldman, Carrie Cai, Amy Zhang, Abby Klein, and Lyla Fischer, for playtesting VideoDoc and offering suggestions and support.
- My friends and family for their support throughout this process. I would especially like to thank my parents, Lisa and Steven, and my sister, Sarah, for their endless love and support this year and always.

Table of Contents

Abstract	3
Acknowledgements	4
1 Introduction	7
2 Related Work	12
2.1 Making Videos Engaging	12
2.2 Navigating Videos	13
2.3 Combining Video and Text	16
2.4 Massive Open Online Courses	17
2.5 Video Digests	21
3 User Interface	23
3.1 Student Interface	23
3.1.1 Definitions.....	24
3.1.2 Layout	25
3.1.3 Getting an Overview	26
3.1.4 Watching.....	27
3.1.5 Navigating	31
3.1.6 Searching	32
3.1.7 Reviewing.....	33
3.2 Author Interface	34
3.2.1 Section Boundaries.....	35
3.2.2 Titles	41
3.2.3 Text Styling	41
3.2.4 Text Editing.....	44
3.2.5 Original Transcript With Edits	45
4 Implementation	47
4.1 Section JSON File Format	47
4.2 Generating a Section JSON File	54
4.3 Displaying the Interface	57
4.4 Playback	58
4.5 Author Interface	60
4.5.1 Text Editing and Styling	60
4.5.2 Indicating Text Styling in the Toolbar.....	61
4.5.3 Section Boundary Changes	62
4.5.4 Determining the Selected Sentence	62
4.5.5 Displaying Transcript Additions and Deletions	63
5 Evaluation	65
5.1 Research Questions	65
5.2 Participants	66
5.3 Student Interface	66
5.3.1 Procedure	66
5.3.2 Lecture Videos.....	69
5.3.3 VideoDoc Preparation	70
5.3.4 User Tasks	72
5.3.5 Satisfaction and Usability Questions	73

5.3.6	Results.....	74
5.3.6.1	First Time Watching Experience.....	74
5.3.6.2	Navigation.....	77
5.3.6.3	Other Observations.....	78
5.4	Author Interface	79
5.4.1	Procedure	79
5.4.2	Lecture Videos.....	80
5.4.3	VideoDoc Preparation.....	81
5.4.4	Satisfaction and Usability Questions	81
5.4.5	Results.....	82
5.4.5.1	General Editing Approaches	82
5.4.5.2	Changing Section Boundaries.....	83
5.4.5.3	Titles for Merged and Split Sections.....	84
5.4.5.4	Lecture Slides	84
5.4.5.5	Representative Frames	85
5.4.5.6	Styling Text.....	86
5.4.5.7	Editing Text.....	86
5.4.5.8	Other Observations.....	87
6	Discussion and Future Work	88
6.1	User Study	88
6.2	Other Future Work.....	89
7	Conclusion.....	91
8	References.....	92

1 Introduction

Videos and electronic textbooks are two important resources used in online education, and each has its advantages and disadvantages. Videos are highly engaging due to their dynamic nature and the presence of a human voice and often a human face, whereas textbooks are less engaging due to their static content and lack of human interaction. On the other hand, textbooks are easier to skim than videos. A user can see hundreds of words on a page of a textbook at a time, allowing them to get an overview of the content included, whereas in a video a user must scrub through to see multiple frames of content. Since no audio is played during scrubbing, a user is also missing out on these important spoken words, diminishing the quality of their skimming. The quality of skimming is closely related to the quality of navigation. A user can navigate a textbook relatively easily by using the table of contents and the index to identify a set of relevant pages and then using the Ctrl-f keyboard shortcut and skimming to narrow in on the desired content. In order to navigate a video, the user must scrub until they find the desired content, since they do not know ahead of time at what time which content occurs. They must rely on visual cues during scrubbing, and if visual cues are not informative enough, the user will use trial and error as they pause and play the video to determine if they have found the desired content.

Users take advantage of both videos and electronic textbooks in a single course [1] so they have an effective tool for each of their learning needs. They can use video the first time they learn the material so they are engaged and motivated. They can use the textbook for subsequent times they review the material because it is easier to

navigate. Users can also use the textbook for the very first time they encounter the material to get an overview of the material contained. However, if the video and textbook for a class do not appear on the same webpage, it is difficult for students to use them at the same time. Additionally, if the video and textbook content do not cover identical content, it is difficult for users to find the corresponding parts in the two resources, and it is also difficult for instructors to maintain the two resources. An example of non-identical content is the video and textbook using different example problems to explain the same concept.

To address these problems, we propose *VideoDoc* (Figure 1), an online lecture interface that combines videos and a textbook layout into one resource in order to improve online lecture delivery. VideoDoc can be played like a video or read like a textbook. These are VideoDoc's key features:

- **Sections:** A lecture video is broken up into sections by topic, shown with a title, the corresponding text transcript, and representative video frames.
- **Table of Contents:** A list of section titles appears on the left of the page, serving as a table of contents for the lecture.
- **Static Content:** A user can quickly skim the text, representative video frames, and table of contents to get an overview of the material on their first encounter. They can later read the text to review the material.
- **Playback:** A user can watch the engaging video to learn the material.
- **Navigation:** A user can easily navigate the lecture by scrolling through the page or by clicking on a table of contents title to be taken to the corresponding section.
- **Talking-head vs. Content:** VideoDoc extracts talking-head clips from a video and then displays content and talking-head video streams in parallel, allowing a user to view relevant content frames while watching the talking-head. The content video stream is played in the currently selected section's

content area, and the talking-head stream is always played in the bottom left corner of the viewport.

Combining videos and text, segmenting videos into sections and streams, showing representative video frames, and offering an interactive table of contents provides a user with greater control over the content they view.

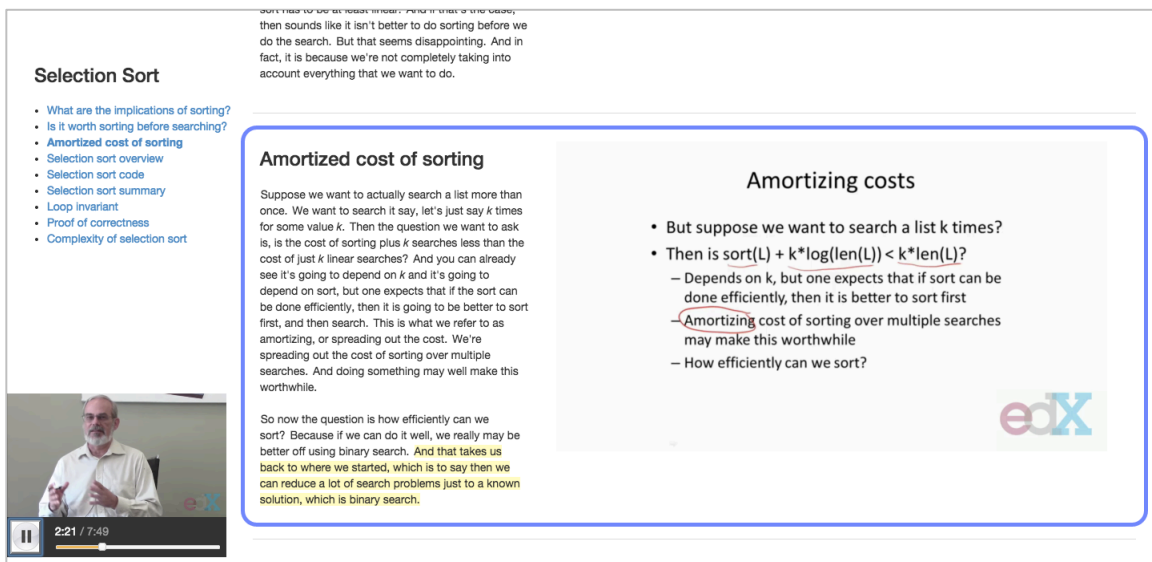


Figure 1: VideoDoc student interface, showing a 6.00.1x edX lecture on selection sort. Here the lecture is in play mode and currently at the last sentence of the section titled “Amortized cost of sorting”, as indicated by the sentence’s yellow highlighting and the section’s blue border.

Creating a VideoDoc lecture requires as input the lecture video, a time-annotated text transcript, and a labeling of which parts of the video display a talking-head and which parts do not. With this input we are able to create a basic VideoDoc with default section boundaries and no section titles. A course instructor can then edit their VideoDoc using the *author interface* (Figure 2), tailoring section boundaries, creating section titles, and adding text styling.

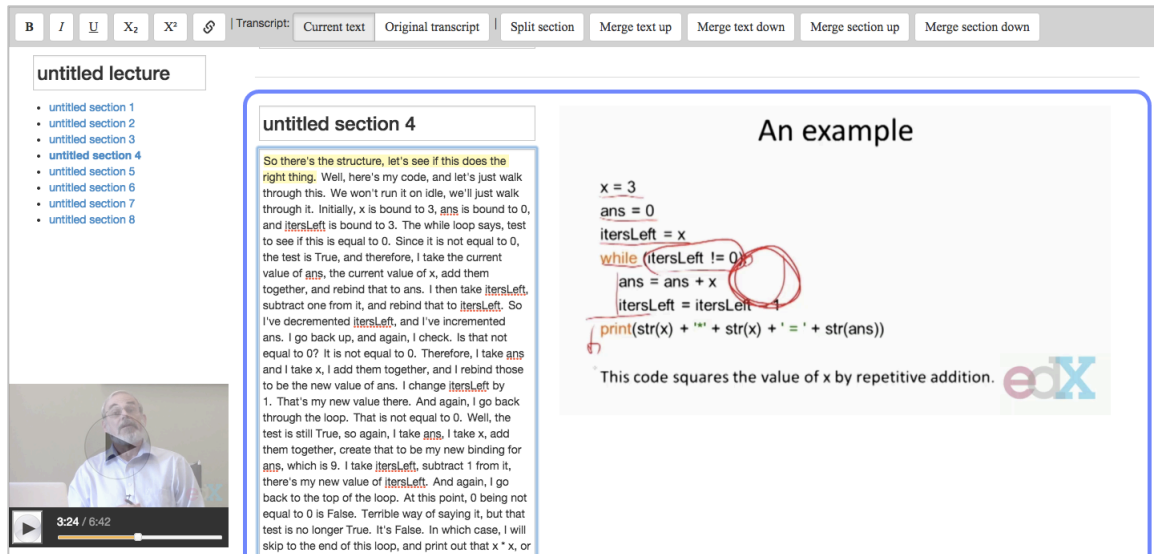


Figure 2: VideoDoc author interface, showing a 6.00.1x edX lecture on iteration [2]. Titles and the text transcript are editable text areas. The toolbar at the top of the screen allows for styling text, viewing text additions and deletions as compared to the input transcript, and changing section boundaries by splitting and merging sections.

We evaluated how the VideoDoc student interface affects learners' watching and reviewing experiences as well as how usable the interface is. We conducted a controlled study that compared the VideoDoc student interface with the edX video viewing interface. Participants watched and then answered questions about short 6.00.1x lectures in each interface. We found that while watching a lecture in VideoDoc, participants enjoyed being able to preview upcoming lecture material via the table of contents and scrolling through video content frames. When answering questions about the lecture material, participants found the overview provided by the table of contents to be helpful in guiding their search for material and found clicking on table of contents titles and scrolling through the page to improve navigation. Participants generally found the interface easy to learn.

We evaluated the usability of the VideoDoc author interface by presenting participants with a raw, unedited VideoDoc and asking them to edit it as if they were an instructor for the course. Participants were easily able to add and edit section

titles, but about half of the participants had trouble changing section boundaries, attempting to do so by copying and pasting text rather than by using the merge and split buttons in the toolbar. After coaching them on the features they had trouble learning, participants felt that the author interface allowed them to make all desired edits to the VideoDoc lecture.

The main contributions of this thesis are:

- An interface that presents a lecture video in a textbook-like layout to provide an overview of material, improve navigation, and reduce disruptions caused by media transitions
- A method for generating a bare-bones lecture provided the lecture video, time-annotated text transcript, and labeling of talking-head video frames
- An editing interface for modifying section boundaries, adding titles, editing text, and styling text

2 Related Work

In this chapter we present existing lecture delivery interfaces and how users learn with them. We explore properties that affect viewer engagement and navigation in videos, and we discuss the effect of combining videos and text.

2.1 Making Videos Engaging

Because videos are more engaging than text, we wish to make the video component of VideoDoc as engaging as possible. Engagement is defined as learner attention, motivation, and satisfaction. Previous work has shown three ways to make videos more engaging: include talking-heads, make videos short, and include tablet-writing.

Kizilcec et al. [3] study how picture-in-picture lecture videos that include a talking-head in the corner of the screen impact learners' information retention, visual attention, and affect as compared to lecture videos without a talking-head. They found that the presence of the talking-head did not have a significant impact on information retention, that "on average, learners spent 41% of time looking at the face when it was present and transitioned between looking at the face and slides every 3.7 seconds", and that learners greatly prefer the picture-in-picture lecture videos. Although there is no evidence that learners learn better with the presence of

a talking-head, it appears that learners are more engaged and motivated to learn when the talking-head is present. In VideoDoc we therefore include a talking-head when footage exists in order to maximize engagement.

Guo et al. [4] also explore properties of videos that are particularly engaging. They measure engagement “by how long students are watching each video, and whether they attempt to answer post-video assessment problems”. They analyze the user’s level of engagement with respect to video length, the instructor’s speaking rate, whether the video was a lecture or tutorial recording, and the form of delivery. The most significant findings are that the most engaging videos for learners are short videos of less than 6 minutes in length, videos that include talking-heads, and videos that are Khan-style. As a result, VideoDoc presents lecture videos as a series of shorter clips, and when there is appropriate footage, we include talking-heads and Khan-style drawings.

2.2 Navigating Videos

Kim et al. [5] study click data from edX MOOCs to analyze when users drop out of videos and which parts of videos users interact with the most. They find that users tend to drop out of videos that are longer, re-watched, or tutorial-style. They also find that high levels of user interaction often occur at visual transitions. One such example is users re-watching a video at the point where the slide has just changed and the instructor is starting to explain a new concept. Since this is a common place for users to want to start re-watching the video, we would like for it to be easy for them to navigate to. However, it currently is not easy for them to navigate to, since they must scrub. VideoDoc eases navigation to the beginning of distinct concepts, where interaction peaks are likely to occur, by segmenting videos by concept and displaying all video clips on the screen at the same time. In fact, Kim et al. [6] suggest “displaying a representative frame for each interaction peak [to] visually

summarize a video”, and this is what VideoDoc does, not only to summarize the video but also to make navigation easier. The representative frame for a video clip is the slide explained inside of it. As a result, the user is able to see, at the same time, all the slides contained inside the video and is able to easily skim the slides to determine which video clip contains their topic of interest.

Another example of high levels of user interaction occurring at a visual transition is users re-watching the slide that occurred right before the video transitioned to the talking-head. This may happen when the video transitions to the talking-head before the user finishes digesting the previous slide. VideoDoc tries to prevent these abrupt transitions by showing the user two video streams in parallel: a talking-head stream and a content stream. Instead of a slide disappearing when the video transitions to a talking-head, the slide remains visible in the content stream while the talking-head is shown in the talking-head stream, allowing the user to continue digesting the slide while the talking-head speaks.

Li et al. [7] built and tested a video delivery system that speeds up viewing and improves navigation. The system speeds up viewing by removing pauses in speech and by allowing the user to change the playback speed. The system improves navigation primarily with a table of contents and shot boundary frames. For a conference presentation or classroom video, the video is segmented into smaller clips based on slide boundaries, and each clip is represented in a ribbon of frames by a screenshot of its corresponding slide or talking-head. The user can then easily scroll through the ribbon to see the contents of the video and click on a particular frame to play the corresponding clip. Each clip has a corresponding line in the table of contents, which can also be clicked to play the clip. Li et al. found that users enjoyed the video delivery system because it saved them time and they felt they had more “control over what content they watched” as compared to a traditional system. Users likely felt they had more control because they could easily scan through the set of slides to get an overview of the content as well as to navigate to a desired topic. They could also watch the video continuously and anticipate when the next

visual transition would occur and what content the next slide would contain. VideoDoc uses a similar concept of slide screenshots in order to ease navigation and give users more control, but shows content and talking-heads in two parallel streams.

Li et al. tested their system on a variety of scenarios including classroom lectures, conference presentations, sports games, shows, news programs, and travel videos. For scenarios that did not include slides, clips were segmented based on significant boundaries as according to the system. Users particularly enjoyed the ribbon of screenshots in baseball games because they could choose to watch only clips where their team of choice was up to bat. They also enjoyed the screenshots in news programs because they could get a quick overview of the stories included and decide which ones they wanted to watch.

Monserrat et al. [8] built NoteVideo, a system that improves navigation and content-awareness in tablet-writing videos. In NoteVideo, each distinct blackboard character in the original video is linked to the corresponding video clip that starts at that character, and a user can click on a character to start playing the corresponding video clip. This is easier than scrubbing the video to the location where the letter of interest first appears, as is necessary in a traditional video interface. Additionally, at all times all characters from the original video appear on the screen, with future characters faded. This feature allows users to have a better idea of the overall content of the video.

Monserrat et al. next built an improved version of the software called NoteVideo+ that adds a hovering text transcript for each character and adds a scrubber. The transcript for each character allows a user to see more information about a particular part of the video before playing it. Scrubbing with the scrubber helps a user to understand the order in which certain characters appear in the video. In the first version the user could only easily determine which characters appeared before and after the character currently being written. The user could guess the order of

characters based on their location on the screen but when multiple columns of writing started appearing, this became more difficult. NoteVideo+ is an interesting solution for improving video navigation and content-awareness, and VideoDoc should allow similar video formats. However, we cannot rely solely on the NoteVideo+ video format to improve content delivery because not all lectures will contain tablet writing.

2.3 Combining Video and Text

Breslow et al. [1] conduct a broad study on edX's first MOOC, 6.002x. They analyze how and when students use each MOOC resource, discuss student demographics, and analyze student success. They found that students overall spend more time watching the lecture videos than they spend using any other single resource. They also found that textbook usage peaks around exam time and that during exams students spend more time using the textbook than any other single resource. These usage patterns suggest that students use video as their primary source of learning and that they use textbooks to study for exams and to search for particular content for answering exam questions. These observations indicate that it is important to have engaging videos for the first stage of learning and that it is important to have easily-skimmable text for later stages, like reference. Since videos and textbooks accommodate different and essential aspects of learning, it is important to include both of these resources in online course delivery.

An early, but high quality, video-text hybrid for education is eClass, built by Abowd et al. [9] in the late 1990s. The system provides a page of vertically scrollable slides and a continuous video of the professor and classroom from lecture. There is also a timeline of the lecture that includes slide and website references. The user can get an overview of the content by scrolling through the slides and can navigate to a particular part of the video by selecting the appropriate slide, similar to navigation

in the system created by Li et al. [7]. VideoDoc provides similar functionality to eClass with the addition of a text transcript of the audio. We believe a transcript will help the learner review explanations without having to re-watch the video. The transcript will also allow a learner to read along watching the video if they wish to do so. Adding in the transcripts requires us to modify the layout, but we keep the vertically scrollable slides.

In 2012, Miller et al. [10] built an interactive electronic textbook for teaching novice programmers. The textbook includes traditional text, short videos that cover the most important concepts, and a code visualizer. The visualizer allows students to run code written by the instructors, write and run their own code, and see the state of their variables and the program output at each time step. Miller et al. found that for students in a blended class using the textbook, the students found classroom lectures to be the most helpful to learning, followed by reading the textbook and then using the code visualizer. Students also enjoyed having all of the course material in one place, which supports the hypothesis that a hybrid video-textbook system like VideoDoc will improve the learning experience. VideoDoc does not, however, show short video clips that cover material non-identical to the text because we believe this is confusing for students and difficult for instructors to maintain. It is interesting that students did not mention the short videos in the textbook as one of the more helpful learning tools. Perhaps the fact that students ranked classroom lectures and textbook reading high indicates that students find audiovisual lectures given by humans to be helpful when introducing the material and they find text helpful when reviewing the material.

2.4 Massive Open Online Courses

We must of course also explore MOOCs because they are a large multimedia educational resource used today. We will see that current MOOCs do not provide the

quality of navigation and content-awareness that we would like. The major MOOC platforms are edX [11], Coursera [12], and Udacity [13]. Their MOOCs have videos as the centerpiece and they also have PDF course notes, homework assignments, and discussion forums. In Figures 3, 4, and 5 we show screenshots of the video-viewing interface on each MOOC platform.

The MOOCs show spoken words to varying degrees. edX MOOCs have a full text transcript next to the playing video, with the line currently being spoken highlighted in the transcript. Coursera MOOCs provide a transcript that can be downloaded as a text file from a different page on the site than where the video is played. A user could download the transcript to view side by side with the video, but this is not convenient. Udacity MOOCs provide no transcript at all. In Coursera and Udacity a user can turn on closed captioning to view spoken words as text on the screen, but closed captioning is not as informative as a full transcript.

Additionally these MOOCs are not as navigationally friendly as the systems built by Li et al. [7] and Abowd et al. [9]. Coursera videos are long, most longer than 10 minutes and many longer than 15 minutes. Yet Coursera gives no indication of the topics within a video besides one very general topic for the entire video, making it difficult for a user to navigate to a specific concept when re-watching the video. Udacity has short videos, most less than 5 minutes long, and each video is labeled with a topic on the ribbon of boxes at the top of the page. However, the user must hover over a particular box to see the topic for the corresponding video, so the user still cannot get an overview of all the short videos by just glancing at the screen. edX has a similar ribbon to Udacity and longer videos than Udacity, so it is also difficult to navigate and to get an overview of the lesson.

Udacity videos are always the tablet-writing style, but note that edX and Coursera videos can be a variety of production styles, not just the ones shown in the Figures 3 and 4. Possible production styles include slides, tablet-writing, picture-in-picture,

studio whiteboard, green-screen instructor, and traditional lecture hall, among others.

The screenshot displays the edX course interface for 'S8AV2: LEWIS STRUCTURES'. At the top, navigation links include Courseware, Course Info, Textbook: Averill, Textbook: Readings, Discussion, Periodic Table, Wiki, and Progress. A left sidebar contains a 'Help' button and a navigation menu with links for Overview, Entrance Survey, Week 1, Week 2, Week 3, Week 4, and Week 5. Under 'Week 4', there are sections for 'Covalent Bonding Learning Sequence', 'Periodic Trends and Bonding Learning Sequence', 'Molecular Orbitals Learning Sequence', and 'Additional Study Material' including a 'Week 4 Problem Set' due on Mar 29, 2015. The main content area features a video player with a ribbon of other lectures and exercises above it. The video shows a lecturer at a whiteboard with the title 'DRAWING LEWIS STRUCTURES' and three steps: 1) center the element with lowest AVEE, 2) count all valence e⁻, and 3) draw a single bond. To the right of the video is an interactive text transcript with the following text: 'Count all the valence electrons. So you take all the valence electrons off of all the atoms and then just count them all up. And then what you do, you draw a single bond-- that's, of course, two electrons. It counts as two electrons-- from each surrounding atom to the center. And of course you subtract two electrons from your tally each time you draw a bond. Because they consist of two electrons. And then you distribute the remaining electrons in pairs until each atom has an octet. We'll come up with some exceptions to this.'

Figure 3: An edX Introduction to Solid State Chemistry lecture [14]. To the right of the video appears an interactive text transcript. Directly above the video appears a ribbon of other lectures and exercises in the Covalent Bonding learning sequence. At the top of the screenshot are links to other class resources including the textbook and discussion forum.

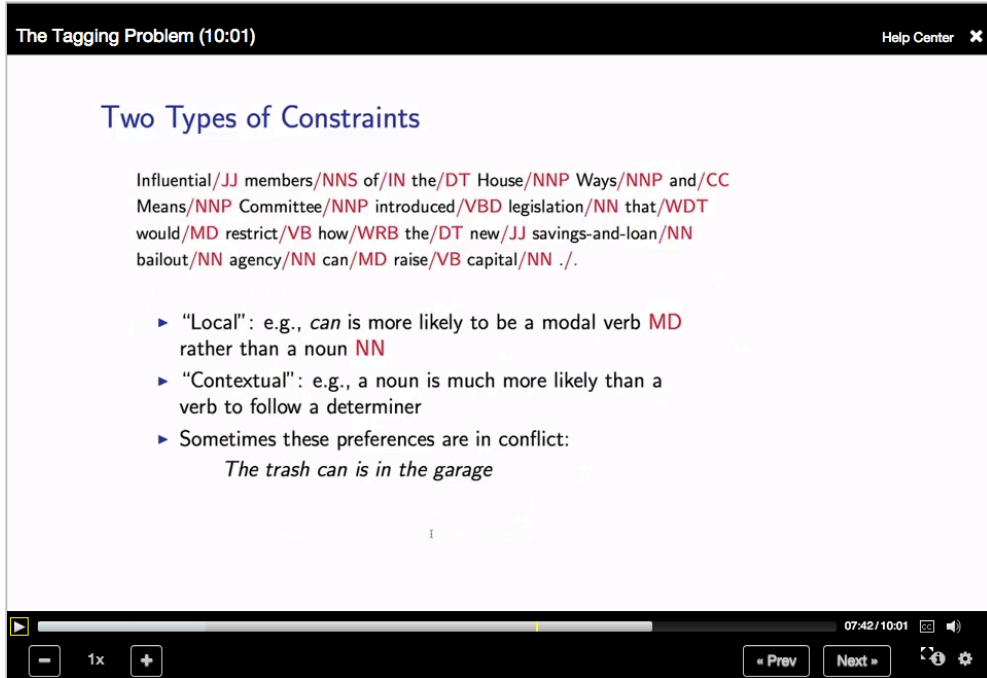


Figure 4: A Coursera Natural Language Processing lecture [15]. Coursera lectures do not provide a text transcript on the same page as the video.

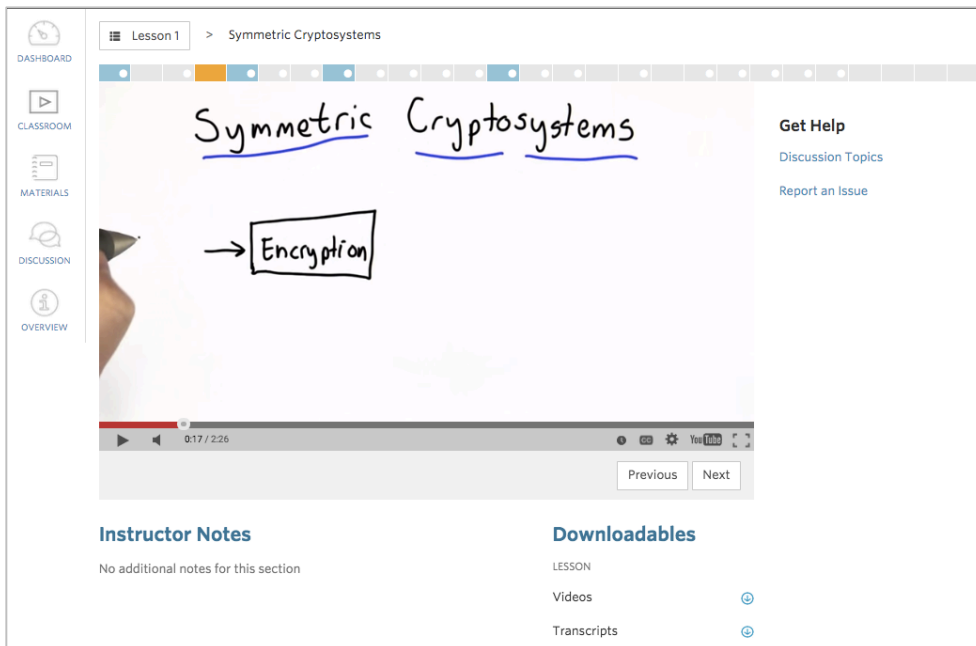


Figure 5: An Udacity Applied Cryptography lecture [16]. Above the video appears a ribbon of other lectures and exercises in Lesson 1. Below the video are downloadable videos and transcripts for Lesson 1. On the left of the page are links to other class resources including written materials and the discussion forum.

Another popular multimedia educational resource is Khan Academy [17]. Most lessons include tablet-writing videos, but no talking-head videos, and a transcript similar to that of edX lectures. Like the MOOC lectures described above, Khan Academy lessons do not allow for easy navigation or content overview. A screenshot of the Khan Academy video interface is shown in Figure 6.

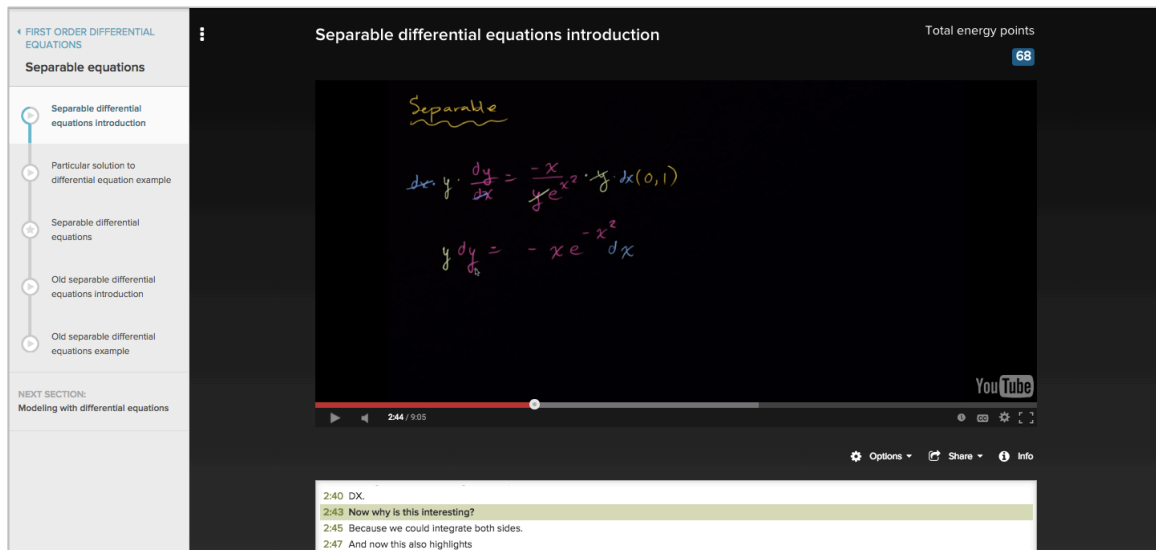


Figure 6: A Khan Academy Differential Equations lesson [18]. Below the video appears the interactive text transcript and to the left of the video appears the Separable Equations lesson sequence.

2.5 Video Digests

Bret Victor presents his lecture “Media for Thinking the Unthinkable” [19] online as a *video digest*, a webpage that presents a lecture video as a set of chapters and sections. Each chapter represents a major topic in the lecture and has a title, a viewport for playing this part of the video, and several sections. Each section represents a subtopic and is displayed as a video frame thumbnail and a short textual summary. A user can click on a chapter’s video viewport to start playing

from the beginning of the chapter. A user can click on a section to start playing its parent chapter from the beginning of the section.

Pavel et al. [20] built a system that creates video digest webpages for lecture videos. They found that users can more easily browse and skim video digests than traditional lecture videos. VideoDoc also presents a lecture video broken into sections by topic with representative video frames but offers the video's text transcript instead of section summaries. The video digest section summaries provide general information, can direct a user to the appropriate section during a search, and perhaps provide a better content overview than VideoDoc, but the user may need to re-watch part of the video if they cannot find desired information in the section summary. With a text transcript in VideoDoc, a user can find any desired information on the page without needing to re-watch the video.

The system built by Pavel et al. automatically generates video digests but also allows authors to edit video digests through an authoring interface. The authoring interface features a WYSIWYG video digest editor that allows authors to change chapter titles, section summaries, and section thumbnails. Next to the WYSIWYG editor is the video's text transcript where users can add and delete chapters and sections and shift their boundaries by placing and moving chapter and section start point markers within the text. VideoDoc's author interface also allows users to modify section boundaries through a text transcript but uses Split and Merge buttons to perform the appropriate operation at the user's selected section and cursor location rather than using start point markers.

3 User Interface

In this chapter we present the student and author interfaces of VideoDoc. We discuss features and associated design decisions and user tasks.

3.1 Student Interface

Traditional lecture video players have 3 main problems:

- It is difficult to search for material of interest.
- It is difficult to obtain an overview or review the lecture material without watching the video.
- If the video transitions from visual content to a talking-head during play mode, the learner cannot continue viewing that visual content.

We created VideoDoc in an attempt to solve these problems. Here we discuss the features offered by VideoDoc's student interface, first by describing the general page layout and then by explaining how users can perform typical video-related tasks using the interface.

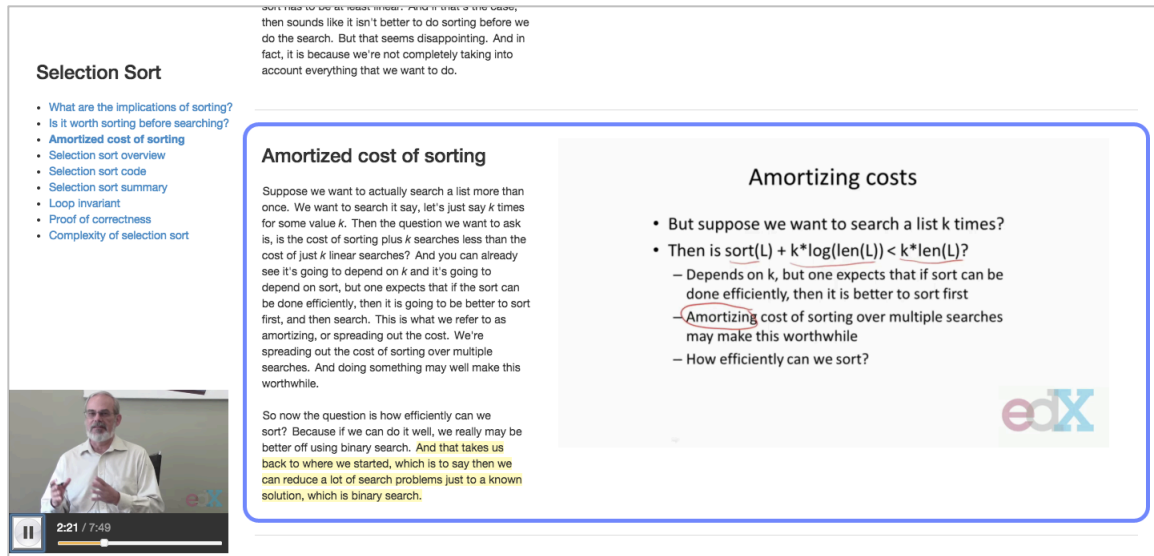


Figure 1 (reproduced for convenience): VideoDoc student interface, showing a 6.00.1x edX lecture on selection sort [2]. Here the lecture is in play mode and currently at the last sentence of the section titled “Amortized cost of sorting”, as indicated by the sentence’s yellow highlighting and the section’s blue border.

3.1.1 Definitions

We define a *talking-head* to be video footage of the instructor’s face or body. We define *content* to be video footage that visually shows lecture material. In technical lectures content is usually lecture slides, tablet notes, or chalkboard notes but can also be a code console, science demos, or pictures.

We choose to make a distinction between talking-head and content footage because we want to help the user focus on content while watching a lecture and allow a user to see screenshots of important content frames while reviewing a lecture. A screenshot of a talking-head usually does not provide the user any information about the lecture material. Note that in interview settings where people are the focus, footage of people may count as content since this content alerts users of the special guest speaker. In these cases there often are not slides or other concrete visual material to show anyway.

In VideoDoc we show the talking-head and content in different locations in the interface, as explained below.

3.1.2 Layout

VideoDoc provides in its main pane a vertically scrollable list of sections, each containing a small topic of the lecture. In the left part of a section is the section title and below it an interactive text transcript. The text transcript within a section may also be broken into paragraphs to improve readability. In the right part of the section is the content area, with representative frames from this part of the video. For technical lecture videos with slides or handwritten notes, we intend for there to be just one representative frame per section, since likely one topic is contained within one slide or small set of notes. This representative frame would most often be the final frame of the section, showing the completed built-up slide or notes. For lectures that show pictures or interview people in quick succession, these pictures and people can be shown together in the same section as thumbnails (Figure 7). A thumbnail can be a full video frame or a cropped video frame and often is shrunk so multiple thumbnails fit in the content area. When a section is being played, the content area shows the playing content video instead of the representative frames.

Wolfpack Runners

- Introductions
- History of the Wolfpack
- Keeps us healthy
- Stretching
- Running
- Injuries
- Many benefits

Introductions


Andrew: Hi. My name's Andrew Toeman. I'm 72.

Jack: I'm Jack Rosenthal. I'm almost 77.

Evelyn: My name is Evelyn Van Hille Sousana and I am 62 years old.


Frank: My name is Frank Elekes. I'm 68 years old.

Armand: I'm Armand Cymbalista. I'm 81, and I've been with the Wolfpack since 1979.



History of the Wolfpack

Andrew: Wolf Burnett formed it about 50 years ago, and at that time there were very few runners. But over the years, I could say several thousand people have gone through the wolf pack. We run regularly Tuesday night, Thursday night, and Sunday morning all season, all year, even in this weather outside tonight. Oh, it's a beautiful evening. Look at this.



0:00 / 3:03

Figure 7: An edX Body101x lecture shown in the VideoDoc student interface. Multiple representative video frames are shown in each section. The first section shows interviewed individuals, and these thumbnails are cropped video frames. The second section shows pictures describing the history of the Wolfpack, and these thumbnails are full video frames.

In the left pane of the page are the lecture title, a table of contents of the section titles, the talking-head video viewport, and a video control bar with a scrubber to indicate progress, the time elapsed, and a play/pause button. The left pane is pinned to the page, with all components visible even when the user scrolls the main pane.

3.1.3 Getting an Overview

A user can get an overview of the lecture video by viewing the table of contents titles or by scrolling through the page and viewing section text and representative frames. VideoDoc allows users to get an overview of the lecture material without having to watch the video or scrub the scrubber to view content, such as slides. Watching a video takes time and using the scrubber is tedious, so VideoDoc offers users faster and more pleasant options for obtaining an overview of a lecture video.

3.1.4 Watching

A user can play or pause the lecture using functions similar to those in other video players. They can press the play/pause button in the control bar, press the spacebar, double click on a transcript sentence, click on a representative frame or the content video, or click on the talking-head video if it is visible. We'll talk more about clicking on representative frames and videos later.

When the user plays the lecture, the lecture starts playing from the current playhead position and progresses downward through the sections automatically. The section currently playing has a blue border around it, its title in the table of contents is bolded, and the currently spoken sentence is highlighted yellow in the text transcript. We emphasize the current section and sentence to help the user focus on and keep track of their current place in the lecture. In particular, if a user spends most of their time watching the content video but misses a word and wants to find it in the text transcript, they are still able to quickly find their place using the yellow highlighted sentence. The blue border allows a user to scroll around the lecture to briefly view other sections and then quickly find and return to the currently playing section. In a traditional video player a user has no way of keeping their spot when they want to view a different part of the video. They scrub the scrubber and leave no bookmark behind.

Additionally, when a section is playing, its representative frames disappear from its content area and the content area now shows the current content at that point in the lecture (Figure 8). The content video takes up the entire content area.

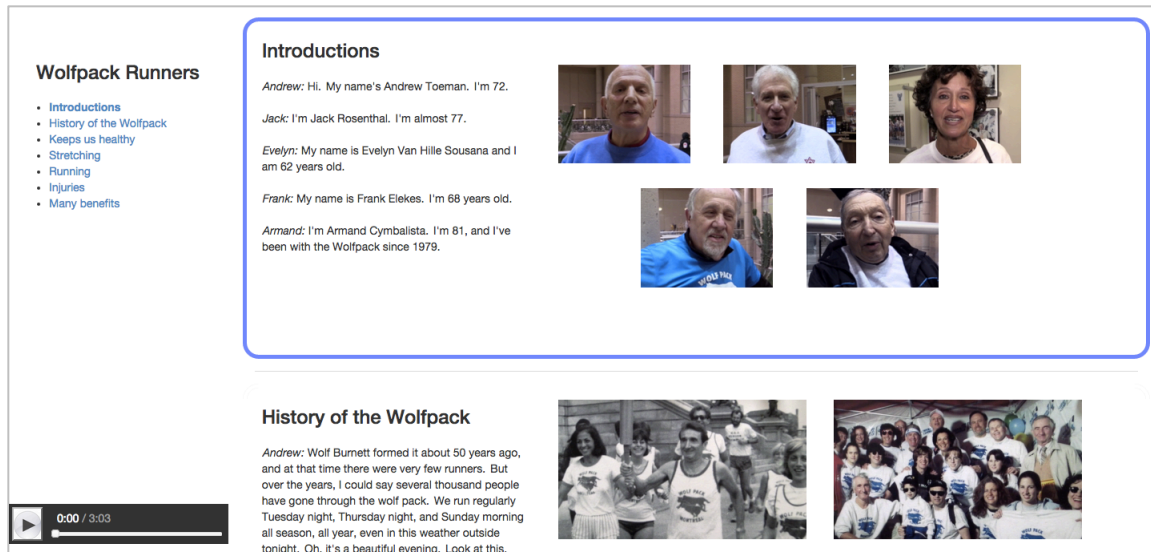


Figure 7 (reproduced for convenience): An edX Body101x lecture shown in the VideoDoc student interface. Multiple representative video frames are shown in each section. The first section shows interviewed individuals, and these thumbnails are cropped video frames. The second section shows pictures describing the history of the Wolfpack, and these thumbnails are full video frames.

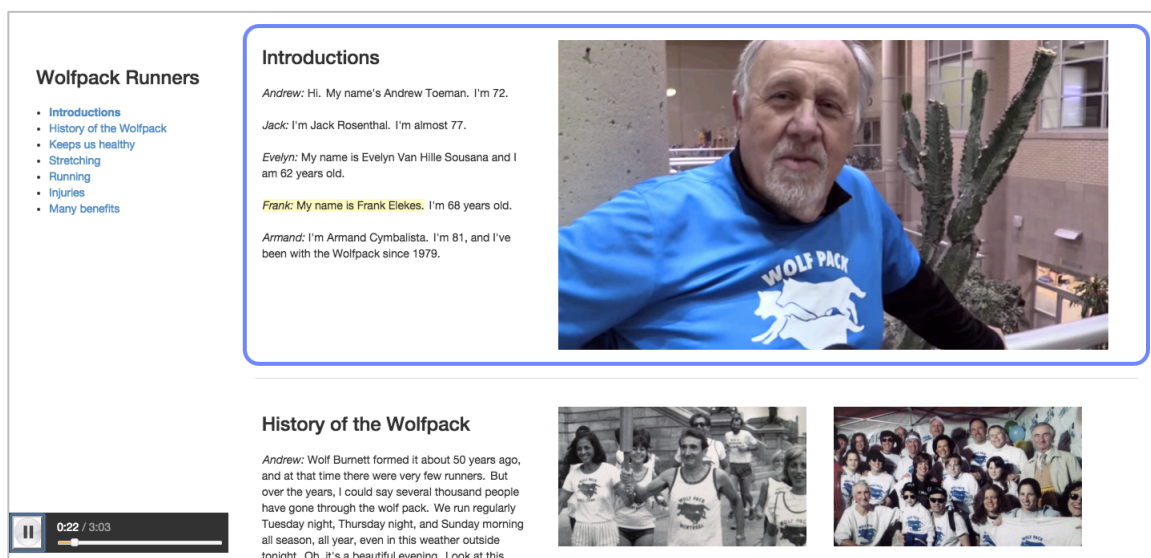


Figure 8: The VideoDoc in Figure 7 in play mode. The representative frames of the current section disappear and the video is played in the section's content area. Here Frank is being interviewed as is indicated by the yellow-highlighted sentence and the footage in the content area.

We considered not hiding the current section's representative frames during play mode. The video would play through one representative frame viewport at a time, jumping through the frames to play through the relevant frame viewport. For example, when Frank is being interviewed in the Body101x lecture shown in Figures 7 and 8, rather than footage of Frank occupying the entire content area as in Figure 8, the footage would occupy only Frank's representative frame in Figure 7, and the video would jump to the next representative frame when the next person is being interviewed. However, we decided against playing through representative frames for a few reasons. Currently we allow cropped rectangular thumbnails of any aspect ratio and we also allow multiple thumbnails per a given video frame (e.g., there are 2 pictures in one video frame). We thought playing the video through an odd shaped thumbnail would be strange, and if there were multiple thumbnails it might be strange to choose one thumbnail over another. In addition to odd thumbnail shapes perhaps the thumbnails may be too small to play the video and adequately show detail. Another reason we decided not to play through one representative frame and still show the others is the concern that seeing the other representative frames could be distracting to the viewer.

If the current playhead is at the talking-head, the content area displays the most recent video content. This behavior allows users to continue viewing content such as slides or handwritten notes after the video transitions from content to the talking-head. After this kind of transition often the instructor is discussing the content just shown, so it is useful for the user to be able to view the content the instructor is discussing. Also, the user may not have fully digested the content, so continuing to display the content while the talking-head speaks gives the user more time to understand the material. In a traditional video player the user would no longer be able to view the content after the transition and would need to scrub backward to see it. Scrubbing disrupts the watching experience and still does not allow the user to view the content while the talking-head is later discussing it. Allowing the user to view relevant content while the talking-head is shown is the reason we separate talking-head and content video. Note that when a new section

begins, if the section begins with a talking-head then the content area will show the last content frame of the previous section. This is the case even if the talking-head is no longer discussing this content because it allows the viewer to continue to digest the most recent content until the video transitions to new content.

Note that during a transition from the talking-head to content the talking-head video in the bottom-left corner of the screen fades away over one second and then the content video starts playing in the content area. We fade away the talking-head rather than abruptly removing it to prepare the user for the transition.

When a section finishes, it loses focus. The blue border and bolded title move to the next section and the page automatically scrolls downward to put the new section in view. In addition to losing its selection, the old section's content area now displays its representative frames, and the new section's content area shows the content at the current playhead.

We include the ability to click on a video or representative frame to play or pause the lecture in order to mimic the similar behavior of clicking on the video viewport to play or pause in a typical video player. In VideoDoc, when the video is currently playing and the talking-head is speaking, if the user clicks on the talking-head the lecture will be paused and a gray overlay with a play button icon will appear over the talking-head. Whenever the lecture is paused and the talking-head is visible at the current playhead, the talking-head will have this play button overlay. The user can click on the overlay to play the video. When a user clicks on a representative frame, the lecture will begin playing from the first point in the containing section that is content and is not the talking-head. We chose to begin playing from the first content time rather than the beginning of the section because we thought users may be confused if upon clicking on the representative frame the content area was not playing and instead the talking-head was playing. Note that clicking on representative frames has not yet been fully explored for when there are multiple representative frames per section. Currently, clicking on any representative frame

will start playing the section from the first instance of content. In the future perhaps we may want clicking on a representative frame to start playing the lecture from the time where the screenshot was taken. Also note that when the lecture is playing and the user clicks on a representative frame in a different section, the current section will stop playing, its representative frames will be restored, and then the new section will start playing. When a section has already been playing, clicking on the content video will pause the video, but no overlay will appear over the video. We chose not to put an overlay over the content video since the content area may contain useful information for a user, such as equations, definitions, or diagrams. When a user pauses the video, it is likely because they want more time to understand the material, possibly by looking at the current content frame. Adding an overlay would make it harder for a user to see the content.

3.1.5 Navigating

A user can navigate the VideoDoc interface by scrolling through the page or clicking on a table of contents section title to automatically be scrolled to that section. Clicking on a table of contents section title also selects that section, giving it the blue border and bolding the table of contents title. Note, however, that only the section is selected and not a particular sentence. The section's content area continues to show the representative frames and no sentence is highlighted yellow. We chose this section representation rather than immediately selecting the section's first sentence and showing the corresponding content frame because it allows users to get an overview of a section upon selecting it. If we instead selected the first sentence upon clicking a table of contents title, the content area would show the beginning state of the section's content, perhaps a blank page or slide, which would not be representative of the section's material. Even though the first sentence is not highlighted, the playhead is moved to the beginning of the section, so when the lecture is played it will play from the beginning of the section. Besides clicking on a

table of contents title, another way to select a section is to click on any whitespace in the section.

A user can also move the video playhead by clicking on a sentence in the text transcript. Clicking on a sentence highlights it yellow, moves the video playhead to the beginning of the sentence, and shows the appropriate talking-head and content frames. Hovering over a sentence in the text transcript shows a dotted gray border around the sentence to help users discover the ability to click on sentences and to help users see the start and end points of the sentence. The mouse cursor also changes to a pointing finger as another affordance for clicking on sentences.

A user can also use the scrubber to move the playhead and update the selected section and sentence, but this is a less likely choice since it is often hard to predict which direction and how far to scrub on the scrubber to find the desired material.

3.1.6 Searching

With these options for navigating a lecture, it should be easier for a user to search for desired material in VideoDoc than in many other lecture video players. For many searches, the table of contents titles should be able to help a user determine which part of a lecture contains the desired material and then quickly navigate to that part by clicking on the appropriate title. If a user is searching for a particular diagram or equation, they can scroll through the VideoDoc page and find the diagram or equation in a representative frame. Users can also use the Ctrl-f “find” keyboard shortcut to search for spoken words.

In other video players, a user needs to scrub through the video to find a frame containing the desired diagram or equation. The user could also search through an interactive text transcript, if one is available, but this could be challenging if the transcript is not well formatted and not broken into sections like in VideoDoc. The

worst case is if the user is searching for spoken words and no interactive transcript is available. The user would need to scrub, using trial and error, and stop occasionally to listen to for the desired words. This kind of scrubbing truly would be tedious. VideoDoc's table of contents, text transcript, and representative frames should ease searching for desired material in a lecture.

3.1.7 Reviewing

After a user locates their desired material in VideoDoc, they can review that material without needing to watch video. They can read the text transcript and view representative frames. In a traditional video player, a user who wants to review material likely needs to watch at least part of the video. They could scrub through the video to view slides or notes, but they would need to watch the video if they want to know lecturer's spoken words. Watching a video takes time. If a text transcript is provided then the user might not need to watch the video, but scrubbing to view slides or notes is still inconvenient.

The way the text transcript is broken into sections and presented as paragraphs in VideoDoc should also help in reading the transcript. The presentation is similar to that of a textbook. The text transcript in edX lecture videos, on the other hand, is displayed as a list of phrases of variable length. One phrase may take one and a half lines, and then the next phrase would begin at the start of the next line. Reading text in this format may be challenging since it requires the reader's eyes to shift around more.

3.2 Author Interface

The VideoDoc author interface allows course instructors to edit existing VideoDocs. An editing interface for VideoDoc is necessary because the automatic VideoDoc generation does not include section titles and likely the sectioning is not ideal. The VideoDoc author interface allows a course instructor to modify section boundaries, add section titles, style text, and edit the text transcript. The author interface is the same as the student interface except that the titles and the text transcript are editable textboxes and there is an editing toolbar at the top of the screen. A user can play and navigate the VideoDoc author interface as they would the student interface. Note that a user can make edits to the VideoDoc while playing the lecture. Also note that all edits are saved immediately. The user does not need to press a “save” button. Here we discuss the editing operations in more detail.

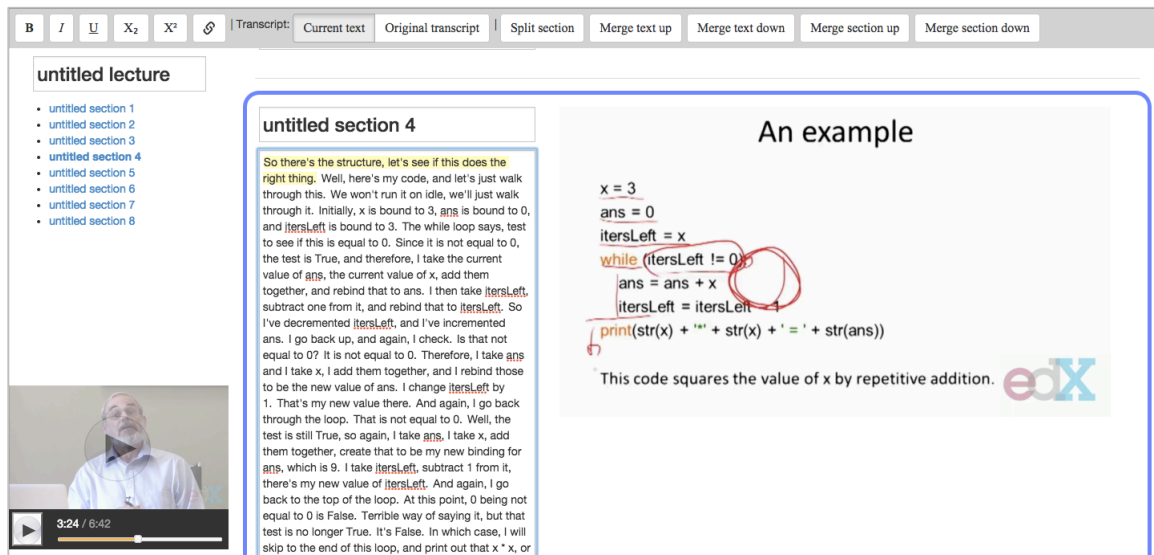


Figure 2 (reproduced for convenience): VideoDoc author interface, showing a 6.00.1x edX lecture on iteration. Titles and the text transcript are editable text areas. The toolbar at the top of the screen allows for styling text, viewing text additions and deletions as compared to the input transcript, and changing section boundaries by splitting and merging sections.

3.2.1 Section Boundaries

If an instructor is not satisfied with the default sections produced for a VideoDoc, they can use the split and merge buttons in the editing toolbar (Figure 9).



Figure 9: Buttons for changing section boundaries in the editing toolbar

Some operations require the user to put their cursor down in the text transcript to indicate a new boundary point or to range select the text to be moved. Below we describe how each button functions.

- **Split Section:** The sentences before the cursor or range selection remain in the current section, and the sentences after the cursor or in the range selection move to a new section created immediately below the current section. The current section keeps its title and the new section is named “untitled”.
- **Merge Text Up:** The sentences before the cursor or in the range selection move to the end of the previous section. Both sections keep their current titles.
- **Merge Text Down:** The sentences after the cursor or in the range selection move down to the beginning of the next section. Both sections keep their current titles.

- **Merge Section Up:** All sentences in the current section move up to the end of the previous section, and the current section is deleted. The merged section's new title is a concatenation of the two sections' titles: "[top section title]/[bottom section title]".
- **Merge Section Down:** All sentences in the current section move down to the beginning of the next section, and the current section is deleted. The merged section's new title is a concatenation of the two sections' titles: "[top section title]/[bottom section title]".

Split Section, Merge Text Up, and Merge Text Down require that the user make a selection in the transcript textbox. Otherwise the buttons are disabled and cannot be clicked. The user can either place a single cursor to indicate the new boundary point or range select the sentences to be moved. The range selection made should make sense in the context of the button pressed, however. For example, for Merge Text Up, the range selection should start from the beginning of the section's text. It would not make sense for the range selection to select only the last two sentences of the section, for instance. If a user were to select the last two sentences of the section, VideoDoc would handle this as if all of the section's text were range selected, moving all of the text up to the previous section. Note that when a user places their cursor in the middle of a sentence, the entire sentence is included in the set of sentences moved.

If all of the text in a section is determined to be moved for Merge Text Up or Merge Text Down, the operation is considered to be a degenerate case and is handled as a Merge Section Up or Merge Section Down, respectively. The two sections are just merged and no section is left behind.

To use the Merge Section Up and Merge Section Down buttons, the user does not need to make a text selection. The operations are performed with regard to the currently selected section (i.e., the section with a blue border around it).

When merges and splits occur, a section's representative frame is updated to be the final video frame of the section. Note that the interface currently does not support merging or splitting user-defined thumbnails. A user needs to manually edit a JSON file to merge or split thumbnails.

To help users learn how the merge and split buttons work, upon hovering over a button a tooltip explanation is shown and the text that would be moved upon clicking the button is range selected (Figure 10a). The range selection may be most helpful when a user places only a single cursor in the textbox or if they incorrectly range select text, for example selecting the last two sentences of a section for Merge Text Up. If the user correctly range selects sentences then the text to be moved is already range selected before hovering over the merge and split buttons. We also use range selection after Merge Text Up or Merge Text Down is pressed to show the user which sentences were moved (Figure 10b). This is another way of teaching users how these buttons work and also reminds them of which text they just moved.

After Merge Section Up or Merge Section Down is pressed we range select the merged title to allow the user to quickly change the title (Figure 11b), since likely the merged title is not the final title they want for the section. Range selecting the title also teaches the user that the two sections' titles have been merged and suggests that they should change the title. Similarly, the title is range selected after Split Section is pressed.

test is no longer True. It's False. In which case, I will skip to the end of this loop, and print out that $x * x$, or if you like, $3 * 3$, is equal to 9.

untitled lecture

- untitled section 1
- untitled section 2
- untitled section 3
- untitled section 4
- untitled section 5
- untitled section 6
- untitled section 7
- untitled section 8

untitled section 5

Cool. A little slow, but it does what I want. Notice I have reused this code an arbitrary number times. And in fact, if I were to change x to be something else, I will reuse that piece of code a different number of times. There's my iteration that I really want. You can also see some properties of an iterative loop. First of all, we need to set an iteration variable outside of the loop. In this case, it's x and $itersLeft$. Actually, the one I really care about here is $itersLeft$. I also need to test that variable to determine when I'm done. There's the use of $itersLeft$ inside of the test. Now, it could be a simple test, it could be a more compound test, but that's basically what I need to test. And finally, I need to be changing that variable inside of the loop, right there. If I didn't, then that test value would never change, which means I would never stop the loop. But there's a property I need. When I set up an iterative loop, I need to say what's the variable I'm setting outside, how am I testing it, and am I making sure to change it somehow inside of the loop in order to ensure that I get the pieces that I want.

Stepping through this code

```

x = 3
ans = 0
itersLeft = x
while (itersLeft != 0):
    ans = ans + x
    itersLeft = itersLeft - 1
print(str(x) + " * " + str(x) + " = " + str(ans))

```

x	ans	itersLeft
3	0	3
	3	2
	6	1
	9	0

5:02 / 6:42

Figure 10a: The cursor was placed immediately after the sentence “There’s my iteration that I really want” in section 5. Upon hovering over the Merge Text Up button in the toolbar, the text that would be moved by clicking this button is shown range selected.

together, and rebind that to ans . I then take $itersLeft$, subtract one from it, and rebind that to $itersLeft$. So I've decremented $itersLeft$, and I've incremented ans . I go back up, and again, I check. Is that not equal to 0? It is not equal to 0. Therefore, I take ans and I take x , I add them together, and I rebind those to be the new value of ans . I change $itersLeft$ by 1. That's my new value there. And again, I go back through the loop. That is not equal to 0. Well, the test is still True, so again, I take ans , I take x , add them together, create that to be my new binding for ans , which is 9. I take $itersLeft$, subtract 1 from it, there's my new value of $itersLeft$. And again, I go back to the top of the loop. At this point, 0 being not equal to 0 is False. Terrible way of saying it, but that test is no longer True. It's False. In which case, I will skip to the end of this loop, and print out that $x * x$, or if you like, $3 * 3$, is equal to 9.

Cool. A little slow, but it does what I want. Notice I have reused this code an arbitrary number times. And in fact, if I were to change x to be something else, I will reuse that piece of code a different number of times. There's my iteration that I really want.

untitled lecture

- untitled section 1
- untitled section 2
- untitled section 3
- untitled section 4
- untitled section 5
- untitled section 6
- untitled section 7
- untitled section 8

untitled section 5

Iteration

5:02 / 6:42

Figure 10b: After clicking Merge Text Up in Figure 10a, the range selected text in section 5 is moved up to the end of the previous section and remains range selected to indicate which text moved.

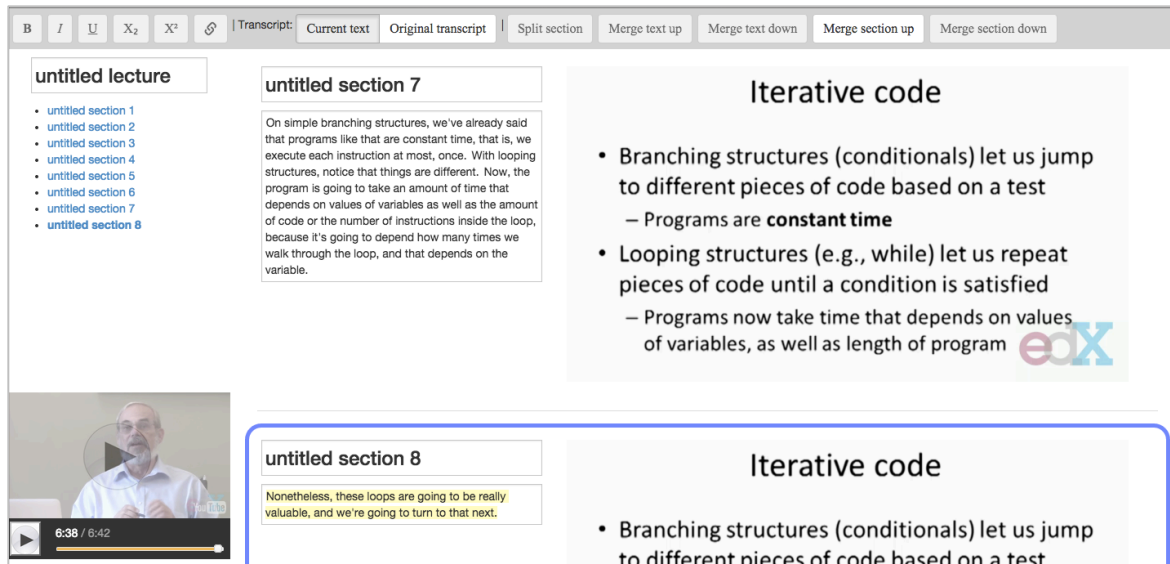


Figure 11a: Sections 7 and 8, where section 8 is currently selected.

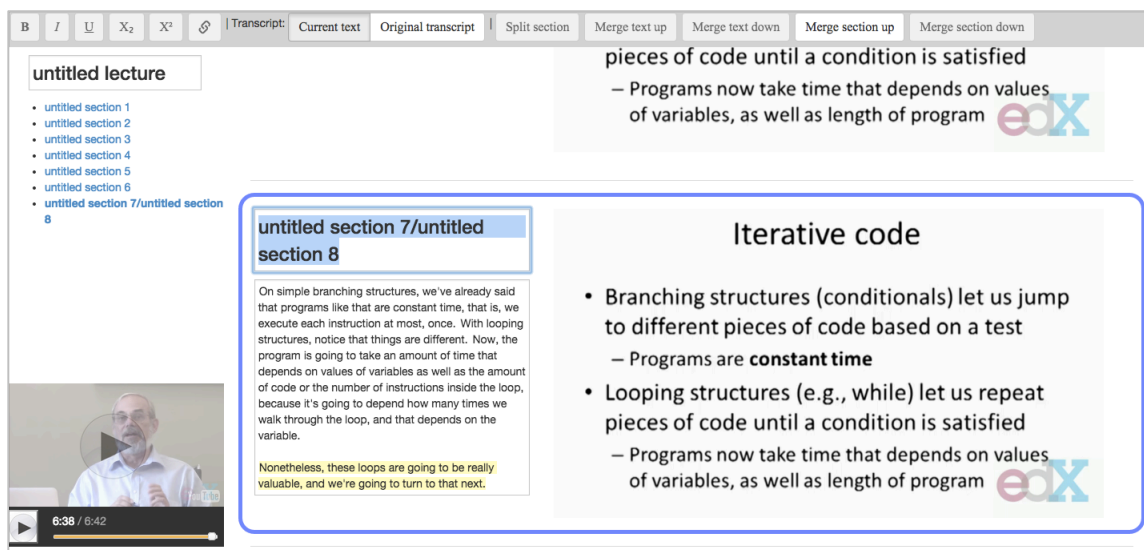


Figure 11b: After clicking Merge Section Up in Figure 11a, sections 7 and 8 are merged, and the merged title is range selected to allow quick editing.

As mentioned earlier, the user can edit a VideoDoc while playing it, and this includes changing section boundaries. After the user clicks one of the merge or split buttons, the lecture pauses for only a split second while the sections are modified and then begins playing again.

We considered other options for changing section boundaries besides the merge and split buttons in the editing toolbar. We list these options and reasons for not pursuing them:

- *Put the Merge Section Up and Merge Section Down buttons at the top and bottom of each section, respectively. We could not think of a way to similarly place the Split Section, Merge Text Up, and Merge Text Down buttons inline with the sections that made sense, so we decided to put all buttons in the toolbar.*
- *Remove the Merge Section Up and Merge Section Down buttons since they are degenerate cases of Merge Text Up and Merge Text Down, respectively, and can be executed by selecting all text and pressing a Merge Text button. We decided to still include Merge Section Up and Merge Section Down in case merging full sections was a common use case. It may become tedious to select all of a section's text to use the Merge Text buttons.*
- *Change section boundaries by cutting sentences from one section and pasting them into the adjacent section rather than using merge and split buttons. We decided to use buttons rather than copying and pasting text because we thought using buttons would be less work for the user. Also, we were concerned a user would cut text from one section and then forget to paste it or forget the correct place to paste it. However, as we note in the Evaluation chapter of this thesis, a few participants in the user study actually did try copying and pasting text to change section boundaries.*

3.2.2 Titles

A user can easily edit the lecture title or a section title by clicking in the appropriate title textbox and typing the new title. After a section title is typed and the textbox loses focus the corresponding title in the table of contents is updated. The user can make the textbox lose focus by pressing the Enter or Tab button on the keyboard or clicking elsewhere on the screen. By not updating the table of contents title until the textbox loses focus, we allow the user to see the old section title while typing a new title, providing some element of safety in case they wish to revert to the old title.

3.2.3 Text Styling

A user can bold, italicize, and underline text as they would in a typical text editor. They can range select text and then press the appropriate button in the toolbar (Figure 12). Alternatively they can use native keyboard shortcuts, such as Ctrl-b for bold. Also as in typical text editors a user can put down a single cursor, select desired styles, and then start typing new text to automatically be typed with the selected styles. A user may want to add bold, italics, or underline to emphasize certain text in the transcript, such as keywords or key points, or to differentiate math symbols or programming variables.

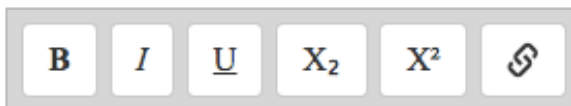
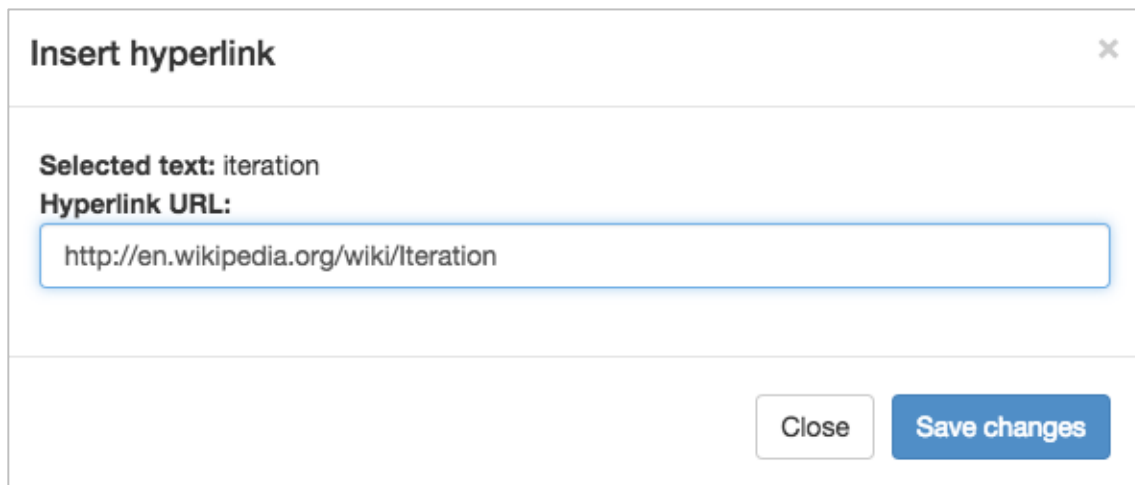


Figure 12: Styling buttons in the editing toolbar

A user can make text subscript or superscript by pressing the appropriate button in the toolbar. This feature would be helpful when fixing math formatting in the transcript.

A user can add hyperlinks to text by range selecting the text and clicking the link button in the toolbar. This brings up a modal (Figure 13) that reminds the user of which text they selected and allows them to enter a URL. Once the user clicks the Save Changes button the hyperlink will appear on the selected text. The user can then click on the hyperlink to view a popover with a few options (Figure 14). Clicking on the Access Link will open the destination in a new tab. Clicking on Edit Link will bring up the hyperlink modal again, and clicking on Remove Link will immediately remove the hyperlink. Note that clicking on the text hyperlink in the VideoDoc student interface shows no popover and immediately opens the destination. An instructor may want to use hyperlinks in their lecture to direct students to the definition of a word or to direct students to other relevant course material.



The image shows a modal window titled "Insert hyperlink" with a close button (X) in the top right corner. Inside the modal, it displays "Selected text: iteration" and "Hyperlink URL:" followed by a text input field containing the URL "http://en.wikipedia.org/wiki/Iteration". At the bottom right of the modal, there are two buttons: "Close" and "Save changes".

Figure 13: Modal for inserting a hyperlink

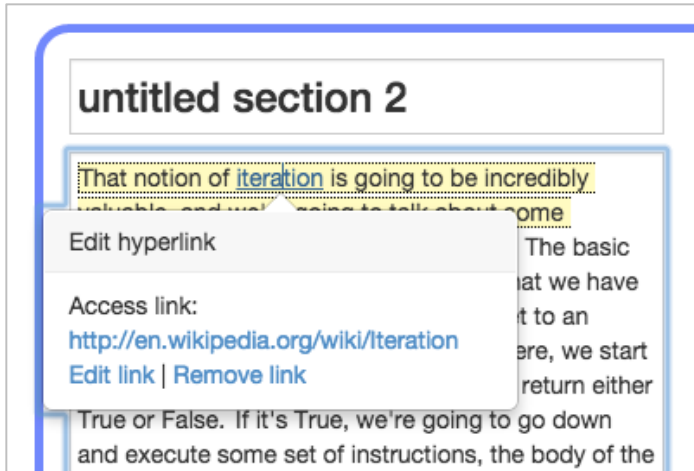


Figure 14: Clicking on the hyperlinked text reveals a popover with options for accessing, editing, and removing the hyperlink.

Also as in most text editors, the toolbar styling buttons indicate whether the current text selection has a given style. The bold, italics, underline, subscript, and superscript buttons appear pressed in when the current text selection has those styles (Figure 15). For most text selections all the text has the same styling, so the buttons to be pressed in is obvious. However, when there is only a single cursor or when different parts of the selected text have different styles, it is not obvious which buttons should be pressed in. It also is not obvious in other text editors, as there appears to be no standard across text editors for choosing which buttons are pressed in. In VideoDoc whether a given text selection has a given style is dictated by whether the DOM interprets the text selection as having that style, and we explain this more in the Implementation chapter. Practically, if the selection is only a single cursor, then the character before the cursor determines the style, and if the selection is a range selection, then the first character of the selection determines the style. Showing pressed in and non-pressed in buttons helps users realize whether pressing the button would toggle the style on or off. Pressing a non-pressed in button adds the style to the entire selection and pressing a pressed in button removes the style from the entire selection.

That notion of **iteration** is going to be incredibly valuable, and we're going to talk about some constructs to help us make that happen. The basic

Figure 15a: Bold text is selected.

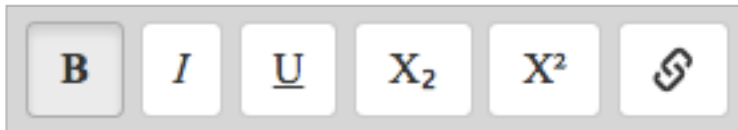


Figure 15b: The text selection in Figure 15a causes the bold button in the editing toolbar to be pressed in.

3.2.4 Text Editing

The VideoDoc author interface also allows for editing the raw text of the transcript. Likely an instructor will not want to make many changes to the raw text since the text displayed should follow along with the instructor's spoken words. However, an instructor may want to fix transcript typos, remove filler words, correct equation typing (e.g., change "5 squared" to 5^2), or add parentheses. The instructor can edit text as they would in a typical text editor. To navigate to different lines and characters, the user can either click with their mouse or use the arrow keys of the keyboard. The yellow sentence highlighting will follow the cursor. Note that the space between any two sentences is always white, not yellow. The space is not considered part of either sentence. Putting the cursor immediately after a sentence's punctuation and before the white space will make that sentence yellow. Moving the cursor one character to the right to be after the white space and immediately before the next sentence will make that next sentence yellow. When a user types, the text will be added to the sentence that is currently yellow.

3.2.5 Original Transcript With Edits

An instructor can view the specific edits they have made to the original text transcript by clicking the Original Transcript button in the toolbar (Figure 16). The original transcript will be shown with added text highlighted green and deleted text highlighted red with a strikethrough (Figure 17).

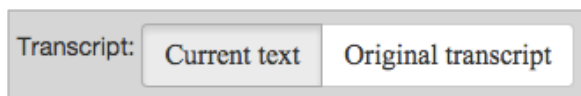


Figure 16: The author interface initially shows the current state of the text, but the user can click on the Original Transcript button to show the original transcript with edits highlighted.

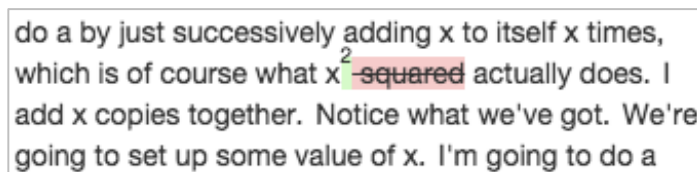


Figure 17: Text additions and deletions as compared to the original transcript. Additions are shown in green and deletions are shown in red with a strikethrough.

The transcript is still shown in textboxes, so a user can continue making edits in this Original Transcript mode. Naturally, new text added is highlighted green and original transcript text deleted is highlighted red. A user can undo changes to the original transcript by “deleting” the appropriate text. Pressing backspace or forward delete on a green text selection permanently removes that text from the transcript since it was not part of the original transcript. Pressing backspace or forward delete on a red text selection removes the red highlighting and adds that text back to the current transcript since the text was part of the original transcript. Undo-ing text

added to the original transcript by deleting the green text seems highly intuitive, but there does not seem to be a highly intuitive way of undo-ing text removed from the original transcript. We considered right-clicking on the red text to select Undo Removal from a local floating menu but were concerned this would be difficult to discover. We considered showing an Undo Removal popover option upon hovering over red text but thought users might find it annoying. We also considered having an Undo Removal button in the toolbar that only is enabled when the current text selection is red text. However, after informally testing this Undo Removal toolbar button with a couple users we found that they did not discover this button. Instead they pressed backspace in order to remove the red highlighting and add the text back to the current transcript. Therefore ultimately we chose to offer only backspacing or forward deleting red text.

Note that in the Original Transcript mode users can still see styling they added in the Current Text mode and they can also add new styling.

Also note that the Original Transcript mode only shows text changes and not changes in a section's text resulting from a change in section boundaries. We do not need to show changes in section boundaries compared to the original non-edited VideoDoc because the instructor did not choose those original boundaries and they have no significance to the instructor. It is important to show text changes because the original text is the lecturer's spoken words.

Clicking the Current Text button will bring the user back to the current text state with new changes included.

4 Implementation

The VideoDoc interface is displayed in an HTML5 web document and is run with a Node, Handlebars, and JSON file infrastructure. First a default section JSON file that represents the contents of each section is created from a time-annotated text transcript and a labeling of talking-head and content parts of the lecture video. This default section JSON file, or one that has already been edited, can then be fed in to either the student or author interface where appropriate data is displayed via Handlebars templates. User events are handled with JavaScript event handlers, videos are played using YouTube video players [21], and playback transitions, such as moving to the next sentence or section, are monitored with JavaScript timers. In the author interface edits are sent to the Node backend where they are written to the section JSON file and then relevant data is sent back to the frontend for appropriate updates to be made. Below we describe each of these aspects of the implementation in more detail.

4.1 Section JSON File Format

The section JSON file contains the lecture's title, total length, a list of section objects, and a list of sentence objects. An example section JSON file is shown in Figure 18. Note that for the entire section JSON file times are expressed in milliseconds rather than seconds.

```

{
  "lecture_title": "Selection Sort",
  "total_length": 469623,
  "sections": [..., {
    "id": 33,
    "thumbnails": [],
    "first_frame_time": 174440,
    "last_frame_time": 250920,
    "end_time": 250920,
    "name": "Selection sort code"
    "chunks": [{
      "index": 29,
      "first_frame_time": 174440,
      "last_frame_time": 243995,
      "end_time": 244015,
      "video_id": "01Is56hu4EU",
      "person": false,
      "content": true,
      "prevContentId": 9
    }, {
      "index": 11,
      "first_frame_time": 244015,
      "last_frame_time": 250920,
      "end_time": 250920,
      "video_id": "01Is56hu4EU",
      "person": true,
      "content": false,
      "prevContentId": 29
    }],
    "paragraphs": [{
      "id": 306,
      "paragraph": true,
      "items": [{
        "index": 37,
        "start_time": 174440,
        "end_time": 175840,
        "current_text": [{
          "content": "So you can see that here.",
          "bold": false,
          "italics": false,
          "underline": false,
          "subscript": false,
          "superscript": false,
          "hyperlink": null
        }], ...]
      }, ... ]
    }, ... ]
  }, ...],
  "sentences": [..., {
    "index": 37,
    "start_time": 174440,
    "end_time": 175840,
    "original_text": "So you can see that here."
  }, ...]
}

```

Figure 18: Excerpt of a section JSON file

Each section object contains the section's title, text with styling and formatting information, video talking-head and content chunks, and thumbnail information. Each of these attributes is explained in more detail below. Note that each section contains a discrete number of sentences. A sentence cannot straddle adjacent sections. Section start and end times are strictly defined by the sentences it contains.

Text is organized by paragraph and stored in the `paragraphs` list. Each object in the `paragraphs` list is either a line break (`"paragraph": false`) or a true paragraph (`"paragraph": true`). Each paragraph object contains a list of sentences in the `items` attribute. Each sentence has a start time, an end time, and a list of sentence pieces in the `current_text` attribute. Each sentence piece contains its text and whether the text is bolded, italicized, underlined, subscript, superscript, or contains a hyperlink. Generally `current_text` contains one sentence piece if the entire sentence has the same styling and contains multiple sentence pieces to indicate different styling through the sentence. For example, a sentence with one word in the middle bolded and the rest not styled would contain 3 sentence pieces, as shown in Figure 19. All style attributes have either `true` or `false` values except for `hyperlink` which is `null` or the URL string.

right thing? And I can do that by identifying what we'll call a **loop invariant**, something we've seen before. And the loop invariant basically says here's a

Figure 19a: Only “loop invariant” is bolded and the rest of its containing sentence is not styled.

```
"current_text": [{
  "content": "And I can do that by identifying what we'll call a ",
  "bold": false,
  "italics": false,
  "underline": false,
  "subscript": false,
  "superscript": false,
  "hyperlink": null
}, {
  "content": "loop invariant",
  "bold": true,
  "italics": false,
  "underline": false,
  "subscript": false,
  "superscript": false,
  "hyperlink": null
}, {
  "content": ", something we've seen before.",
  "bold": false,
  "italics": false,
  "underline": false,
  "subscript": false,
  "superscript": false,
  "hyperlink": null
}]
```

Figure 19b: JSON representing the sentence in Figure 19a. Note that only the second object, representing “loop invariant”, is bold. The text before and after “loop invariant” has no styling.

A section’s time is divided into chunks that are labeled as either talking-head or content, as can be seen in a section’s `chunks` list (Figure 20). These labels are used to indicate whether video footage should be shown in the talking-head viewport or the section’s content viewport at a certain time. Moreover, chunks are used to indicate transitions between talking-head and content footage, and adjacent chunks

in a given section will have different labels. A chunk has 3 time attributes.

`first_frame_time` is the time at which a chunk begins and is the time at which the chunk's first frame is first shown. `last_frame_time` is the time at which the chunk's last frame is first shown. `end_time` is the time at which the chunk ends and is equivalent to the `first_frame_time` of the next chunk. A particular frame lasts a certain number of milliseconds and will vary between videos and possibly within a video. The frame that begins at `last_frame_time` will be shown until immediately before `end_time`. We want to keep track of the `last_frame_time` for a chunk because this is when we take a screenshot of the final frame of a chunk, as discussed later. We cannot take a screenshot at `end_time` because this will be the next chunk.

```
"chunks": [{
  "index": 29,
  "first_frame_time": 174440,
  "last_frame_time": 243995,
  "end_time": 244015,
  "video_id": "01Is56hu4EU",
  "person": false,
  "content": true,
  "prevContentId": 9
}, {
  "index": 11,
  "first_frame_time": 244015,
  "last_frame_time": 250920,
  "end_time": 250920,
  "video_id": "01Is56hu4EU",
  "person": true,
  "content": false,
  "prevContentId": 29
}]
```

Figure 20: Chunks from Figure 18, reproduced for convenience

A section object may also contain representative frame thumbnail information in the `thumbnails` attribute, but it is not required. If no thumbnails are specified then

the chosen representative frame for a section will be its final frame, and this is often the case for technical lectures with slides or handwritten notes. The format for thumbnail information is shown in Figure 21a. A thumbnail is specified by the video frame `time` at which the screenshot is taken, the selected window of the video frame, the location of the thumbnail in the section content area, and a thumbnail shrink factor. The video frame window is defined by the window's width, height, and top-left corner offset with respect to the frame's dimensions (Figure 21b). The thumbnail location in the section content area is defined by the thumbnail's top-left corner offset with respect to the top-left corner of the content area (Figure 21c). The thumbnail shrink factor is defined with respect to the thumbnail's original size in the content area during play mode. For example, a full video frame thumbnail with a shrink factor 1.0 would occupy the entire section content area, and a full video frame thumbnail with a shrink factor of 0.5 would occupy half the height and half the width of the section content area.

```
"thumbnails": [..., {
  "index": 3,
  "time": 25000,
  "frameXOffset": 0.1,
  "frameYOffset": 0.0,
  "frameWidthFraction": 0.6,
  "frameHeightFraction": 1.0,
  "thumbXLocation": 0.15,
  "thumbYLocation": 0.5,
  "shrinkFactor": 0.4
}, ...]
```

Figure 21a: JSON representing the bottom-left thumbnail of the Introductions section in Figure 7 (reproduced on page 54).

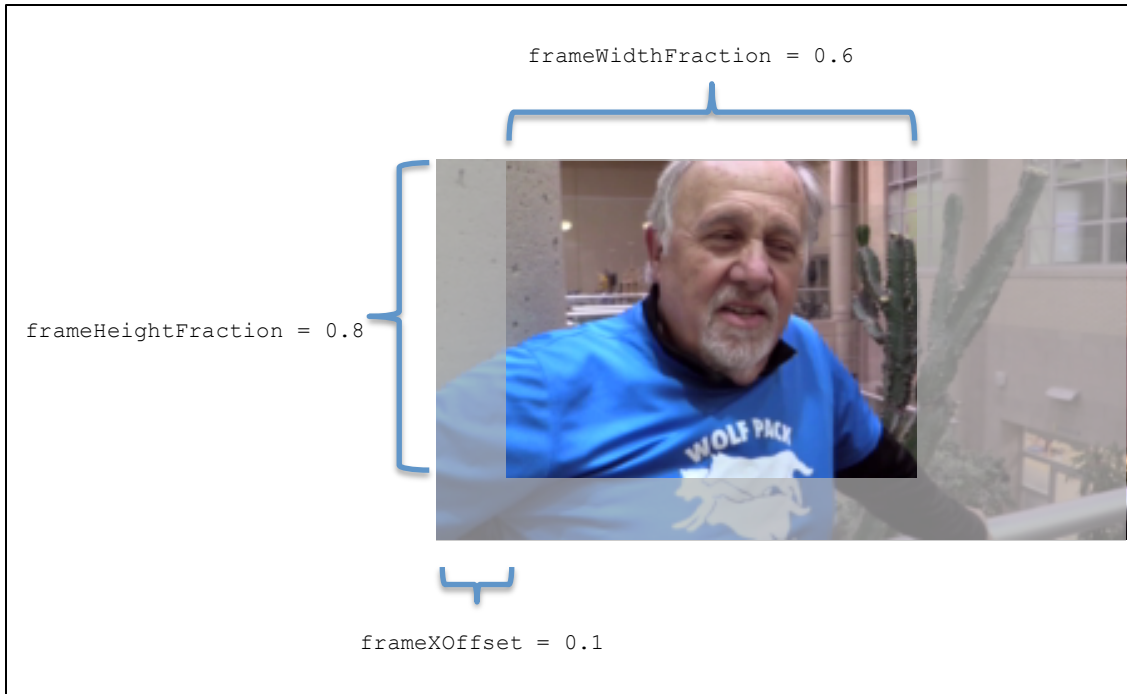


Figure 21b: Here the thumbnail specified in Figure 21a and shown in Figure 7 is seen clearly through the window and the rest of the video frame is shown faded. Horizontally, the window begins 10% from the left of the video frame and is 60% of the width of the frame. Vertically, the window begins at the top of the video frame and is 80% of the height of the frame.

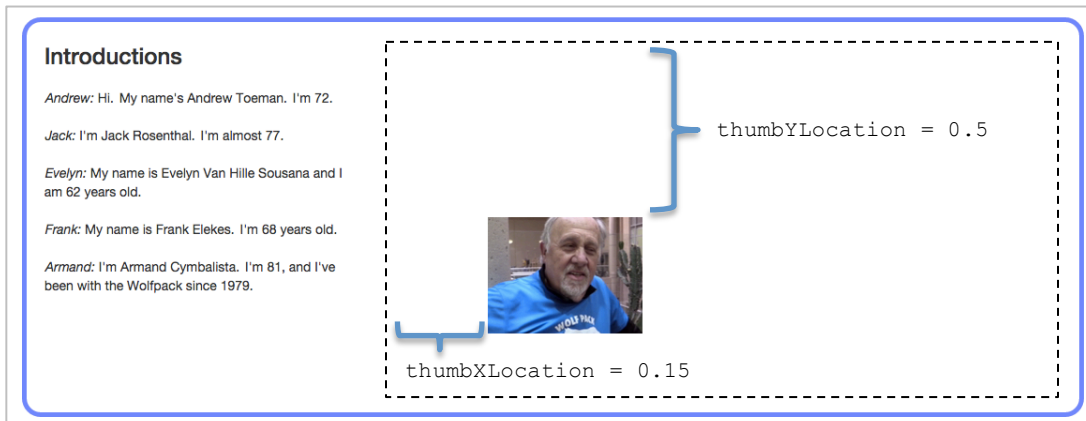


Figure 21c: Here the thumbnail specified in Figure 21a is shown in its native VideoDoc lecture (Figure 7). The other thumbnails for the section are not shown, in order to reduce clutter. The dotted rectangle shows the content area's border. The thumbnail appears 15% from the left border and 50% from the top border.

Wolfpack Runners

- Introductions
- History of the Wolfpack
- Keeps us healthy
- Stretching
- Running
- Injuries
- Many benefits

Introductions


Andrew: Hi. My name's Andrew Toeman. I'm 72.

Jack: I'm Jack Rosenthal. I'm almost 77.

Evelyn: My name is Evelyn Van Hille Sousana and I am 62 years old.


Frank: My name is Frank Elekes. I'm 68 years old.

Armand: I'm Armand Cymbalista. I'm 81, and I've been with the Wolfpack since 1979.



History of the Wolfpack

Andrew: Wolf Burnett formed it about 50 years ago, and at that time there were very few runners. But over the years, I could say several thousand people have gone through the wolf pack. We run regularly Tuesday night, Thursday night, and Sunday morning all season, all year, even in this weather outside tonight. Oh, it's a beautiful evening. Look at this.



0:00 / 3:03

Figure 7 (reproduced for convenience): An edX Body101x lecture shown in the VideoDoc student interface. Multiple representative video frames are shown in each section. The first section shows interviewed individuals, and these thumbnails are cropped video frames. The second section shows pictures describing the history of the Wolfpack, and these thumbnails are full video frames.

In addition to a list of section objects, the section JSON file also contains a separate list of sentence objects (see bottom of Figure 18). The contained sentences are the sentences from the original text transcript and are only used in the author interface for showing the differences between the current text and the original transcript with green and red text highlighting.

4.2 Generating a Section JSON File

We generate a default section JSON file using a time-annotated text transcript and a talking-head/content labeling. During implementation we tested VideoDoc with edX lecture videos and used the text transcript JSON files provided on the edX lecture pages. Therefore, we built the section JSON generator with that text transcript format in mind. Figure 22 gives an example of the format.

```
{
  "start": [
    1450,
    4890,
    9250,
    10900,
    ...
  ],
  "end": [
    4890,
    9250,
    10900,
    12130,
    ...
  ],
  "text": [
    "Eric Grimson: We've just shown that if we have a sorted list,
    we can search",
    "it much more efficiently than just linear searching, and
    that's great.",
    "But wait a minute.",
    "How do I get a sorted list?",
    ...
  ]
}
```

Figure 22: An excerpt from edX's transcript for the 6.00.1x selection sort lecture.

The talking-head/content labels are also provided as a JSON file (Figure 23). The file features the video duration, the video YouTube ID, and a list of segments, where each segment is a talking-head or content label for a time range of the video. Segments are meant to represent distinct visual pieces of a lecture. In addition to adjacent talking-head and content footage being two different segments, adjacent slides might also be different segments. Note that segment start and end times are given in milliseconds and only the overall video duration "length" attribute is given in seconds. We primarily obtained labeling JSON files from a computer vision parser that specifically labels people and content in educational lecture videos [22]. The labeling JSON files can also be created manually, and this is what we did for VideoDocs used in the user study in order to ensure labels were 100% correct.

```
{
  "length": 469,
  "segments": [{
    "id": 0,
    "content": 0,
    "person": 1,
    "start_time": 0,
    "end_time": 27500
  }, {
    "id": 1,
    "content": 1,
    "person": 0,
    "start_time": 28150,
    "end_time": 65580
  }, {
    "id": 2,
    "content": 0,
    "person": 1,
    "start_time": 65600,
    "end_time": 89700
  }, ...]
}
```

Figure 23: Talking-head and content labels for a lecture video

As we create a default section JSON file, we use 2 heuristics:

1. Try to put each segment in its own section.
2. Make sure each section has at least 3 sentences.

Also note that default sections are given the title “untitled[section id number]”.

We decided to ensure each section has at least 3 sentences because with some lecture styles, in particular lecture halls where the lecturer or camera moves frequently, the computer vision parser created many short segments, each a sentence or shorter. Since a VideoDoc section is supposed to be a topic or an example in a lecture, rarely if ever would a section contain only one sentence. We do not want an instructor using the author interface to be overwhelmed by a large number of default sections and have to tediously merge many of these sections together because they are impractically short.

We chose to put approximately one segment in each section since segments represent distinct visual pieces in a lecture, for instance different slides. One segment per section seemed like a likely way to create sections of different material and reasonable durations. Techniques such as machine learning, natural language processing, and crowdsourcing for creating better topically distinct sections and titling them were out of scope of this thesis project but are potential areas of future work.

Running a Node script generates the section JSON file.

4.3 Displaying the Interface

VideoDoc lectures are displayed in the browser by running a Node application that takes a section JSON file and an mp4 video as input. The application first reads the section JSON file and writes the data to memory. It then uses the fluent-ffmpeg Node module [23] to take screenshots of the mp4 video for section thumbnails and for the final frame of each section chunk. As we later explain in more detail, we use screenshots of the final frame of section chunks to aid in smooth transitions between chunks.

When the user navigates the browser to the correct VideoDoc URL, the page is rendered with Handlebars templates. We use Bootstrap and jQuery UI to aid in styling and widgets. Although we use the video mp4 file for taking screenshots, we present the video in the browser using a YouTube video player. This allows VideoDoc lectures to easily be viewed remotely and we do not have to worry about serving the video ourselves.

After the page loads, the client-side JavaScript extracts sentence start times, chunk start times, and a time-ordered list of chunk ID numbers from the DOM. This information aids in handling user events and playback, as discussed below.

4.4 Playback

When the lecture is playing, the client-side keeps a timer, using JavaScript's `setInterval`, to check every 20 milliseconds for potential status changes in video attributes, such as reaching the end of a sentence or section. Some state is kept about each attribute in order to determine if a status change has occurred since the last check. The table in Figure 24 lists status changes and the corresponding DOM updates to be made.

Attribute	Status change	DOM update
Clock	1 second has passed since the clock display has been updated	Increment the clock by 1 second
Sentence	The current time has passed the end time of the current sentence	Update to the next sentence and highlight it
Chunk	The current time has passed the end time of the current chunk	Update to the next chunk, appropriately transitioning between talking-head and content
Section	The current time has passed the end time of the current section	Update to the next section, select it with the blue border and bolded table of contents title, and scroll to it

Figure 24: Playback status changes and DOM updates

The clock, sentence, and section DOM updates are straightforward, but the chunk DOM update is less obvious. To play a video in VideoDoc, we use only one YouTube

video player. We move and resize the player element as the lecture transitions between talking-head and content chunks and progresses through sections so that the video plays through the correct viewport. Originally we used one YouTube player per chunk, with player start and end times set to those of the chunk, and we listened for player end events to determine when to play the next chunk's player. However, the delay between the end event and telling the next player to play was too great and resulted in noticeably discontinuous audio. We then tried using a separate, single, invisible YouTube player only for audio, muting the individual chunk players, in order to ensure continuous audio. However, we found that this audio was not always synchronized with the chunk player videos. Therefore we chose to use one YouTube player for the entire lecture in order to ensure continuous audio that is synchronized with the video. With this one player we do not need to pause the lecture during chunk transitions. We simply use the frequent `setInterval` timer to determine when to move the player.

We now discuss how we perform smooth transitions between talking-head and content chunks. When transitioning from talking-head to content, we fade the talking-head video over the course of a second before moving the player to the content viewport, in order to help prepare the user for the transition. When transitioning from content to talking-head, we want to make sure the content viewport does not display any talking-head, even for a split second, so that users are not distracted or confused. In order to avoid showing talking-head in the content viewport, once the current time is within 500 milliseconds of the end of the content chunk, we cover up the video player with a screenshot of the final content frame. Now when the transition from content to talking-head occurs, the final content frame will be covering the video so no talking-head will be seen. We experimented with smaller thresholds than 500 milliseconds but found that the final content frame was not displayed quickly enough and that the talking-head was still seen for a split second. 500 milliseconds seemed to work well during testing. Note that we do not need to perform a similar cover-up over the talking-head video during a

transition from talking-head to content because by the time the transition occurs, the video will have almost completely faded and the transition should not be visible.

4.5 Author Interface

Here we describe implementation details specific to the author interface.

4.5.1 Text Editing and Styling

The transcript textboxes are HTML5 `contentEditable` elements. When a user uses keyboard shortcuts, such as Ctrl-b for bold, for styling transcript text, the styling is automatically applied because of this `contentEditable` property. When a user uses the toolbar buttons for styling text, we manually add the styling to the DOM by using the JavaScript `document.execCommand` function, which performs common styling operations and functions the same as using keyboard shortcuts. Specifically, for bold, italics, underline, subscript, and superscript, `document.execCommand` simply toggles whether the current text selection has the given style.

Whenever the user types in a transcript textbox or edits text styling, we send updates to the backend Node application to be written to the section JSON file. We parse the current section's transcript textbox DOM to determine the new text content and styling, and we create JSON of a similar format to a section object in the section JSON file. To parse the transcript, for each sentence we perform a depth-first traversal of the DOM. The leaf nodes will be text, so when we reach a leaf node, we check the node's ancestors' attributes to determine the text's styling. Since we rely on the `contentEditable` text editor and `document.execCommand` to perform styling, we do not know exactly how the DOM will be structured after styling changes are made. There could be arbitrary layers of complexity that we cannot

predict, so we found that using a depth-first traversal approach and checking ancestors' attributes worked well. Note that after writing updates to the section JSON file we do not recreate the transcript DOM using Handlebars templates because it may be difficult to correctly restore the cursor location in the new DOM structure. It is necessary for the cursor to be in the correct location because the user may be making quick and constant edits. The potential advantage of updating the DOM with Handlebars templates after each edit is that we would then know the DOM structure and a limited space of potential restructuring changes after the next edit, which in turn could allow extraction of this single edit. However, our depth-first traversal parsing algorithm works and we do not need to restore the cursor. Note that currently we parse the entire section's text after any text or styling change because styling could be applied to multiple sentences at a time, though perhaps we could specifically determine the modified sentences and only parse those.

4.5.2 Indicating Text Styling in the Toolbar

As mentioned in the User Interface chapter, toolbar styling buttons are pressed in when the current text selection has the given styling. For most text selections the styling is obvious, but for single cursors on styling boundaries or range selections of mixed-styled text, it is not always obvious which toolbar buttons should be pressed in. Since toggling toolbar buttons toggles the text selection's style, we need to make sure the representation in the toolbar is consistent with the `contentEditable` element's interpretation of the DOM, which toggles style through `document.execCommand` and keyboard shortcuts. `contentEditable` determines whether a given text selection has a particular style by looking at its containing node, so this is also how we determine whether or not toolbar style buttons should be pressed in.

4.5.3 Section Boundary Changes

When the user presses a merge or split button for changing section boundaries, we send the affected section ID and sentence ID to the backend. Specifically, the sentence ID is only sent for the Split Section and Merge Text buttons and represents the first sentence below the new boundary. For instance, for Split Section the sentence ID would be the ID of the first sentence in the new section. Using the affected section and sentence IDs, the backend updates its section representation, partitioning sentences and chunks and creating or deleting sections as necessary. Next the backend takes final frame screenshots for the affected chunks and then it sends JSON for the updated sections to the client-side. The client-side then adds, replaces, and removes appropriate section DOM elements, using Handlebars templates to create new HTML. The client-side also updates the time-ordered list of chunk ID numbers that it uses for determining playback order since likely chunks have been added or deleted.

4.5.4 Determining the Selected Sentence

When the user makes a text selection, we determine which sentence was selected and should be highlighted. In addition to keeping track of which sentence was selected, we also determine whether the text selection was entirely before or entirely after the selected sentence so that we can correctly partition a section's sentences when a Split Section or Merge Text button is pressed. For example, imagine sentence 5 is currently yellow and then Split Section is pressed. Before the button press, if the cursor was before sentence 5, then sentence 5 would move down into the new section. If the cursor were instead after sentence 5, then sentence 5 would remain in its current section. Note that we need to determine this information about the text selection when the selection is made rather than when the Split Section or Merge Text button is pressed. Once the button is pressed, the text selection no longer exists.

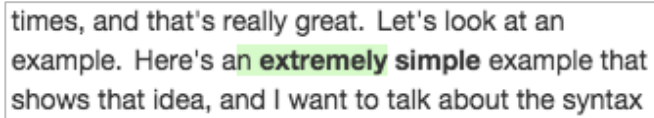
4.5.5 Displaying Transcript Additions and Deletions

In the Original Transcript editor mode, we use Google’s diff-match-patch library [24] to help determine and then display text additions and deletions. We create JSON in the backend representing the additions and deletions and then send them to the client-side. For each transcript sentence we create a plain text string from the backend’s section JSON by concatenating the sentence pieces in `current_text`. We then compute the difference between the original sentence text, which we stored in a separate list of original transcript sentences, and the `current_text` plain text string using diff-match-patch. The difference algorithm returns a list of strings, for each indicating whether the string is an addition, a deletion, or unchanged text. From now on we will call this list the *difference strings list*.

Since diff-match-patch only indicates plain text differences, we must now determine ourselves the author-added styling (e.g., bold, italics, hyperlink) for each addition or unchanged string. Note that deleted text has no author-added styling because the original transcript had no styling. We do not keep track of styling for original transcript text that was at one point styled and then later deleted.

We step through the `current_text` list and difference strings list to match text and create a new text representation that documents both styling and the addition/deletion/unchanged status for a given piece of plain text. To match text, we alternate moving pointers through the `current_text` and difference strings lists as we build up our new sentence representation. For the first difference string we step through the `current_text` list until we have built up and reproduced the difference string. We document the styling for this difference string, which could include multiple `current_text` pieces of varying styles, and we keep a pointer to the `current_text` piece where we should begin our exploration for the next difference string. Text of varying styles and difference statuses could possibly only partially overlap (Figure 25). We account for this in our matching algorithm by keeping a pointer not just to the appropriate `current_text` piece but also to the

next character index that should be explored within the `current_text` piece's plain text.



times, and that's really great. Let's look at an example. Here's an **extremely simple** example that shows that idea, and I want to talk about the syntax

Figure 25: An example of only partially overlapping styles and difference statuses. The original sentence started “Here’s a simple example”. Here we’ve added “n extremely” and we’ve bolded only “extremely simple”. “extremely” is both bolded and green, but it is not the entire bold text or the entire green text.

After creating the new sentence representation documenting both style and difference status we send this new representation to the client-side where it is rendered via Handlebars templates and placed into the DOM.

5 Evaluation

5.1 Research Questions

We conducted a user study to assess how VideoDoc affects a learner's watching and reviewing experience of a lecture video as compared with using a more traditional video-viewing interface. We also evaluated the usability of the student and author interfaces. Below are our primary research questions:

RQ1. How does VideoDoc affect a learner's experience the first time they watch a particular lecture video?

RQ2. How do learners use VideoDoc to search for an answer to a question?

RQ3. Are the features of VideoDoc's student interface learnable and useful?

RQ4. Are the features of VideoDoc's author interface learnable and useful?

We used a single user study to explore the four research questions, by having participants first use the student interface and second use the author interface.

5.2 Participants

We recruited 8 participants by posting a call for participants in the Course 6 Facebook group. Since the lecture videos used in the study contain Python, we wished to ensure baseline familiarity with Python to avoid difficulty in task completion due to insufficient background. Therefore we asked interested persons to indicate their level of experience with Python on a scale from 1 (no experience) to 5 (highly proficient) and accepted individuals who ranked at least 3. Participants ranged in age from 19 to 23. Four of the participants were female and 4 were male. Each participant was paid \$20 at the end of the hour-long study.

5.3 Student Interface

Through the student interface part of the study we aimed to answer research questions **RQ1**, **RQ2**, and **RQ3**.

5.3.1 Procedure

To evaluate the watching and reviewing experience with VideoDoc, we conducted a controlled study, comparing the VideoDoc student interface with the edX video-viewing interface (Figure 3, reproduced below). The edX interface offers users a basic video-viewing interface with a scrubbable progress bar and an interactive text transcript, both features that VideoDoc offers.

Figure 3 (reproduced for convenience): An edX Introduction to Solid State Chemistry lecture [14]. To the right of the video appears an interactive text transcript. Directly above the video appears a ribbon of other lectures and exercises in the Covalent Bonding learning sequence. At the top of the screenshot are links to other class resources including the textbook and discussion forum.

Participants viewed two lecture videos, one in edX and one in VideoDoc. We counterbalanced the video viewing order and the video interface assignment.

For each lecture video, the participant was asked to watch the entire video and was told that they would be asked a few questions afterward. For each lecture video we asked the participant to perform 5 tasks: 1 visual search task, 3 problem search tasks, and 1 play task.

- A *visual search task* asks a user to search for a particular visual feature in the video, such as a slide with a given heading, and emulates a learner searching for a slide or diagram they remember from a previous watch of the video.
- A *problem search task* asks a user to navigate to a part of the lecture that would answer a particular recall question and emulates a learner searching

in the video for the answer to a problem set question. Note that for the problem search task we specifically asked participants to navigate to a relevant part of the lecture, since some participants may be able to answer the question from memory or previous knowledge without using the interface.

- A *play task* asks a user to play the lecture starting from a particular topic and emulates a learner wanting to watch a specific part of a lecture. The visual and problem search tasks do not necessarily require the participant to play the video. The participant may simply point their mouse or finger to the relevant point on the computer screen. Therefore the play task specifically allows us to observe how participants play the lecture at a desired point.

For each lecture the tasks are given in the same order for every participant. This order was determined such that completing any task should not be trivial, specifically that completing the task should require some navigation through the lecture. The first task cannot be completed using the state of the video interface upon finishing watching the lecture, and each following task cannot be completed using the final state after finishing the previous task.

We modeled parts of our user study after the user study presented by Kim et al. [25] for evaluating LectureScape, another lecture delivery system. Specifically we borrowed the idea of giving participants visual search and problem search tasks, and we borrowed the definitions for these two types of tasks. We also modeled specific task questions and post-study questions after those used in the LectureScape user study.

5.3.2 Lecture Videos

In this study we used lecture videos from the Fall 2013 6.00.1x Introduction to Computer Science and Programming edX course [2]. We wanted to use a course whose videos were abundant with slides because we believe VideoDoc is currently best suited for content-heavy videos. We also wanted videos that flip between entirely content and entirely talking-head frames so that we could evaluate the talking-head viewport in the bottom-left corner and the ability to view content while the talking-head is shown on screen. Additionally, we wanted to use course material that participants would be reasonably comfortable with. We did not want a potential unfamiliarity with course material to make completing tasks difficult. Since our source of participants was the Course 6 Facebook group, we expected that most potential participants would have computer science and Python experience equivalent to 6.00.1x.

We chose 2 videos of similar length, difficulty, and number of slides so that corresponding tasks in the 2 videos would have similar difficulty and require similar solutions when solved in the same interface. The lecture videos we chose for the student interface study were *Bisection Search* from Lecture 3 – Simple Algorithms and *Selection Sort* from Lecture 10 – Memory and Search. Below are details regarding these lecture videos:

Bisection Search

- Length: 7 minutes, 18 seconds
- Number of slides: 5
- YouTube Video ID: hX1aUXnDwgA
- edX lecture: Lecture 3 – Simple Algorithms

Selection Sort

- Length: 7 minutes, 49 seconds
- Number of slides: 5

- YouTube Video ID: 01Is56hu4EU
- edX lecture: Lecture 10 – Memory and Search

5.3.3 VideoDoc Preparation

We used the author interface to create well-sectioned and well-titled VideoDocs for the Bisection Search and Selection Sort lectures. We chose the section boundaries and section titles ourselves, using the same authoring style for the two lectures. Since VideoDoc currently does not handle well sections with text exceeding the browser viewport height, we made sections short enough such that each section's text fit entirely within the viewport. The Bisection Search and Selection Sort VideoDocs are shown in Figures 26 and 27, respectively.

We only minimally styled the text, with italics for variable names, because we primarily wanted to test how VideoDoc's navigation and layout features affect a user's watching and reviewing experiences. We were afraid that if we used bold styling, in particular on key phrases, that would be the reason a user answered a question more easily in VideoDoc than in edX and that we would not be able to attribute improved performance to navigation and layout features.

Note that for the talking-head/content partitioning input, we manually created the partitioning, rather than using the computer vision partitioning algorithm, to ensure that the video was labeled correctly.

Bisection Search


- Exhaustive search too slow
- Square root search setup**
- Zero-ing in on square root
- Square root code - setup
- Square root code - body
- Code in IDLE
- Larger examples
- Observations

Square root search setup

And it turns out, for a lot of problems, we can do that using a wonderful idea called bisection search. So what do we know? Let's go back to the idea of trying to find the square root. We know that \sqrt{x} lies somewhere between 0 and x . We're assuming x is positive just to make life a little easier for us. That's a mathematical fact. Now, what we did was we said let's start with 0. Then, 0 plus a little bit. Then, 0 plus 2 times a little bit, and then 0 times 3 times a little bit, trying all of those examples, until we got to something that was close enough to the answer we wanted. That's exhaustive. Rather than doing that, suppose instead we say, look, we know that \sqrt{x} is somewhere between 0 and x . So let's just pick a guess right here in the middle. Let's call that g . Let's just pick the midpoint between 0 and x and try it. Now, if we're lucky, the answer is close enough and then we're done. That's unlikely.

Bisection search

- We know that the square root of x lies between 0 and x , from mathematics
- Rather than exhaustively trying things starting at 0, suppose instead we pick a number in the middle of this range



- If we are lucky, this answer is close enough

Zero-ing in on square root

But even if that is not the case, we have a good situation. Even if we're not close enough, we can now ask was that guess g too big or too small? Well, if g^2 is bigger than x , then we know that it's too big. We know that the square root has to lie

Bisection search

- If not close enough, is guess too big or too small?
- If $g^2 > x$, then know g is too big; but now search

Figure 26: The Bisection Search lecture shown in VideoDoc's student interface.

Selection Sort

- What are the implications of sorting?
- Is it worth sorting before searching?
- Amortized cost of sorting**
- Selection sort overview
- Selection sort code
- Selection sort summary
- Loop invariant
- Proof of correctness
- Complexity of selection sort

Amortized cost of sorting

Suppose we want to actually search a list more than once. We want to search it say, let's just say k times for some value k . Then the question we want to ask is, is the cost of sorting plus k searches less than the cost of just k linear searches? And you can already see it's going to depend on k and it's going to depend on sort, but one expects that if the sort can be done efficiently, then it is going to be better to sort first, and then search. This is what we refer to as amortizing, or spreading out the cost. We're spreading out the cost of sorting over multiple searches. And doing something may well make this worthwhile.

So now the question is how efficiently can we sort? Because if we can do it well, we really may be better off using binary search. And that takes us back to where we started, which is to say then we can reduce a lot of search problems just to a known solution, which is binary search.

Amortizing costs

- But suppose we want to search a list k times?
- Then is $\text{sort}(L) + k \cdot \log(\text{len}(L)) < k \cdot \text{len}(L)$?
 - Depends on k , but one expects that if sort can be done efficiently, then it is better to sort first
 - Amortizing cost of sorting over multiple searches may make this worthwhile
 - How efficiently can we sort?

Selection sort overview

OK, so let's look at sorting. And we're going to look at two examples. The first example is shown with a piece of code. It's called selection sort. And the idea

Selection sort

```
def selSort(L):
```

Figure 27: The Selection Sort lecture shown in VideoDoc's student interface.

5.3.4 User Tasks

Below are the tasks we asked participants to complete for each lecture video. They are annotated here with their task type.

Bisection Search

1. Can you show me a slide that has the title “Example of square root”? (*visual search task*)
2. Please take me to a point in the lecture that would answer the following question: At each stage of bisection search by how much do we reduce the range of values we have to search through? (*problem search task*)
3. Please take me to a point in the lecture that would answer the following question: When does the idea of bisection search work well? (*problem search task*)
4. Please take me to a point in the lecture that would answer the following question: How many steps of the bisection search method does it take in order to find the square root of 25 within $\epsilon = 0.01$? (*problem search task*)
5. Can you play the lecture starting from the point where they first introduce bisection search? (*play task*)

Selection Sort

1. Please take me to a point in the lecture that would answer the following question: Can you explain a procedure for switching the values of 2 variables? (*problem search task*)
2. Please take me to a point in the lecture that would answer the following question: What is the loop invariant in selection sort? (*problem search task*)
3. Can you show me a slide that has the title “Sorting Algorithms”? (*visual search task*)

4. Please take me to a point in the lecture that would answer the following question: What is the overall complexity of selection sort? (*problem search task*)
5. Can you play the lecture starting from the point where they first discuss amortizing the cost of sorting? (*play task*)

5.3.5 Satisfaction and Usability Questions

After a participant watched both lecture videos and completed the tasks, we asked the participant qualitative questions to gather their thoughts on VideoDoc and to assess usability. These are the questions we asked:

1. Overall, how was your experience using VideoDoc?
2. Overall, how was your experience using edX?
3. Which features of VideoDoc did you like, and why?
4. Were there any features of VideoDoc that you disliked, found distracting, or were confused by?
5. Are there any features you feel were missing from VideoDoc?
6. Which of the following did you find yourself using the most while first watching the lecture?
 - a. The text transcript
 - b. The slide and code video on the right
 - c. The instructor video in the bottom left corner
7. Do you have any other questions or comments?

5.3.6 Results

5.3.6.1 *First Time Watching Experience*

Overall users had a positive experience watching videos using VideoDoc.

Users liked the text transcript in VideoDoc more than the one in edX. Four out of 8 users said that they found the constantly scrolling text transcript in edX to be distracting, and they appreciated the less frequent scrolling in VideoDoc. They liked that in VideoDoc the text remained still and only the yellow highlighting moved to show the currently spoken text. Some users appreciated that text was highlighted by sentence rather than by partial sentence as in edX. Users also liked the transcript's wider text area as compared to edX, and they liked how text was broken up into paragraphs. These features helped some users read the text the more easily in VideoDoc than in edX.

However, a couple of users found the text transcript to be distracting while they were watching the video. They often found themselves reading the transcript while the video was playing, sometimes reading ahead, and then had trouble processing the video slides at the same time. One user also said that they found themselves somewhat discouraged by seeing the large amount of text ahead of them in the lecture and that as a result they felt less interested in continuing to view this lecture. In edX this user hid the text transcript while watching the video to avoid distraction, and said they would have liked to do something similar in VideoDoc.

Users liked that VideoDoc allowed them to preview a lecture before watching all of it. They liked that the table of contents gave an overview of the lecture content and structure. While watching the lecture for the first time, curious users could get an overview of the upcoming content in the lecture by viewing the table of contents or by scrolling down the page to view upcoming slides.

In general users either said they liked the overall page layout or they made no comment on the layout. However one user said that the placement of the text transcript between the talking-head and the content made it hard to switch their focus back and forth between the talking-head and content while watching the video. They said they might prefer the text transcript be to the right of the content.

When the lecture first begins playing, users are unaware that the video is split into talking-head and content pieces and that these pieces are displayed in separate places on the page. Note that both 6.00.1x videos begin with a talking-head. One user asked if they could enlarge the video in the bottom-left corner, likely because they thought the entire duration of the video would be displayed small and in the corner. During the first transition from talking-head to content, many users are confused as to why the talking-head has disappeared. Additionally, in the Bisection Search lecture, before the lecture is played, the first section displays a slide in the content area, and then when the video starts playing users are confused about why the slide disappears. The reason the slide disappears is because the lecture video starts with the talking-head, and while this talking-head is talking, there is no previous slide to show. The end of the section is a slide, so this is why the section has a representative slide frame before the lecture is played. After seeing a couple transitions between talking-head and content users better understand the page layout. However we should continue to explore other interface behaviors that will reduce confusion and distraction regarding separate talking-head and content viewports.

Users did not spend much time looking at the talking-head. Some users said they enjoyed the feeling of a human presence, even though they did not look at the talking-head much. Some users were distracted by the talking-head appearing and then disappearing in the bottom-left corner of the page during transitions between talking-head and content frames. Other users felt the talking-head was not necessary but were not distracted by it.

One user said they appreciated that even when the talking-head was visible they could still see the current slide. In the 6.00.1x edX videos if the talking-head is shown then only the talking-head is shown, so in the edX interface the user is not able to see the slide the instructor is currently discussing.

One user said that the movement of the blue border from one section to the next alerted them that a new topic is about to be discussed and helped them stay focused while watching the lecture.

We asked users whether they used the text transcript, the content video, or the talking-head video in the bottom-left corner the most while watching the video. About half of the users said they looked at the content video the most, and about half said they looked at the text transcript the most. A couple users said they used the content video and the text transcript equally. Users only looked at the talking-head minimally. One user who mostly looked at the content video said they looked at the text transcript if they thought they missed something the instructor said. One user who mostly looked at the text transcript said they glanced at the content video when the instructor was writing on the slide. Another user said they “approached it as a textbook”, where they read the text transcript, and then only used the content video for viewing diagrams and pictures, since the text in the slides is already in the transcript.

Five out of 8 users said they wished VideoDoc had the ability to speedup the lecture pace, a feature that they often use in YouTube and edX.

5.3.6.2 Navigation

Users much preferred VideoDoc over edX for answering questions about a lecture.

While completing tasks for a VideoDoc lecture, users navigated the lecture by clicking on table of contents titles and scrolling through the page. For the visual search task, half of the users found the desired slide by clicking on table of contents titles and the other half scrolled through the page. For problem search and play tasks 6 out of 8 users clicked on table of contents titles for at least one task, and 4 out of 8 users scrolled through the page for at least one task.

If after clicking on a table of contents title the user realized they still had not reached the desired section, they would usually click on another table of contents title. It is interesting to note that for some users, when they wanted to navigate to an adjacent section, they chose to click on the adjacent table of contents title rather than scroll the page. We suspect that since these users were already looking at the table of contents to guide their search, it was easier to click on the appropriate title rather than make the correct scroll gesture. However, there were some users who chose to scroll the page when they realized the desired content was in an adjacent section.

Only one user tried scrubbing the slider progress bar to navigate the lecture, and only did so for one task.

Somewhat surprisingly only one user used the Ctrl-f “find” keyboard shortcut for searching for a particular word. This user used the keyboard shortcut for each task in edX, but they did not use the keyboard shortcut for any tasks in VideoDoc and instead scrolled through the page. Note that the user interacted with edX first and VideoDoc second. This user commented that they found the text layout and scrolling in VideoDoc to be more pleasant than in edX, so it seems they used the “find”

keyboard shortcut in edX because reading the text transcript and scrolling were too unpleasant.

Upon interviewing users, it was clear that they found the table of contents very helpful for completing tasks. When answering questions about the lecture, users could use the table of contents to remind themselves of the order of topics in the lecture and then subsequently could determine where in the VideoDoc to navigate to. The edX video interface provides no summary of the video or additional information of topic locations, so users must use their memory of the lecture structure and tediously scrub the slider or scroll through the text transcript to find a desired topic. In VideoDoc users also liked that they could click on table of contents titles to easily jump around the lecture.

Most users realized they could click on a text transcript sentence to navigate to that sentence in the lecture.

One user would have like the blue section border and bold table of contents title to follow them while they were scrolling through the page, such that the section currently in view would be the one with a blue border and a bold table of contents title.

5.3.6.3 Other Observations

Here we discuss user comments that are not directly related to the first time watching experience or navigation.

One user said they liked how they could see a slide and then could see next to it the spoken text that discussed that slide.

One user said they liked that the VideoDoc page provides a summary of the lecture and that they could use it to study for a test instead of re-watching the lecture video or reading the text transcript.

One user was confused about why the discussion of a particular slide was spread out over 2 sections. When we created the Bisection Search VideoDoc lecture, we broke the square root code slide into 2 sections to reduce the amount of text per section and also to break the discussion down into 2 main components: the variable initialization and the square root procedure. The user expected the square root code slide to be contained within one section.

5.4 Author Interface

Through the author interface part of the user study we aimed to answer research question **RQ4**: Are the features of VideoDoc's author interface learnable and useful?

5.4.1 Procedure

We conducted the author interface study after introducing participants to VideoDoc through the student interface study. To evaluate the author interface of VideoDoc, we presented each participant with a raw, non-edited VideoDoc and asked them to edit it as if they were an instructor for the class. We explained that creating a VideoDoc is a semi-automated process, requiring as input the lecture video, a time-annotated text transcript, and a talking-head/content partitioning. We explained that with this input, a basic VideoDoc with likely non-ideal section breaks, no section titles, no text styling, and potential transcript typos, can be created. We explained that in order to create a nicely titled and sectioned VideoDoc like we showed them earlier in the student interface study, an instructor must manually edit the VideoDoc, and this was their task.

We gave participants about 15 minutes to edit a 5-8 minute long video, coaching them when they got stuck and prompting them to share their thoughts.

5.4.2 Lecture Videos

We again used lecture videos from the 6.00.1x course since users were already familiar with this video style from the student interface study, and they now should have a good idea of what a completed 6.00.1x VideoDoc should look like. We used 4 different lecture videos, with 2 users per video. We are not performing a controlled study for author interfaces, so there are no requirements on which videos we show participants. We wanted some variety in the course material presented in VideoDoc in order to increase the number of potential interaction scenarios in which we could observe a user.

We chose some of the shorter 6.00.1x lecture videos since 15 minutes is not much time for users to edit a VideoDoc, and longer lecture videos may overwhelm or discourage users. These are lecture videos we chose:

Programming Languages

- Length: 5 minutes, 13 seconds
- YouTube Video ID: BvooljkNJ24
- edX lecture: Lecture 2 – Core Elements of Programs

Iteration

- Length: 6 minutes, 43 seconds
- YouTube Video ID: waIE0L9vfII
- edX lecture: Lecture 3 – Iteration

Functions

- Length: 7 minutes, 49 seconds
- YouTube Video ID: zhKN60gDjk8
- edX lecture: Lecture 4 – Functions

Fibonacci

- Length: 5 minutes, 18 seconds
- YouTube Video ID: e71ErqC25nU
- edX lecture: Lecture 5 – Fibonacci

5.4.3 VideoDoc Preparation

We created raw, non-edited VideoDocs using only the lecture video, a time-annotated text transcript, and a talking-head/content partitioning. As we did for the VideoDocs for the student interface study, we manually created the talking-head/content partitioning to ensure that the video was labeled correctly.

5.4.4 Satisfaction and Usability Questions

After a participant spent about 15 minutes editing in the author interface, we asked the participant qualitative questions to gather their thoughts on the author interface of VideoDoc and to assess usability. These are the questions we asked:

1. Overall, how was your experience using the author interface of VideoDoc?
2. Which features of the author interface did you like, and why?
3. Were there any features of the author interface that you disliked, found distracting, or were confused by?
4. Are there any features you feel were missing from the author interface?
5. Do you have any other questions or comments?

5.4.5 Results

All users were able to make meaningful edits to their VideoDoc lecture. Users had an easy time creating and changing section titles but had more trouble changing section boundaries. Some users learned how to change section boundaries themselves whereas other required coaching. Below are more details on how users interacted with the author interface.

5.4.5.1 *General Editing Approaches*

Some users began by getting an overview of the lecture through clicking on table of contents titles, scrolling through the page, and skimming text. These users said they wanted to familiarize themselves with the material first, since they personally had not given the lecture and initially did not know the material it contained.

Users then generally edited the lecture from top to bottom. Some users titled several sections first before making any section boundary changes. These users tended to not understand how to make section boundary changes, requiring our guidance to make splits and merges, and presumably they titled sections first because this was a task they felt comfortable with. Other users created titles and performed section boundary changes together as they walked through the lecture.

Seven out of 8 users edited the VideoDoc lecture in pause mode, and 6 of them never played the video at all. After finishing editing their lectures, we asked a few participants why they read through the text and did not watch the video. Most of these participants said they thought reading the text would be faster than watching the video but then realized watching the video would not have taken that long and perhaps would have helped them create more accurate section boundaries and section titles. One user said they preferred reading the text and if they were to edit another VideoDoc they would not replace reading text with watching the video. This user said that if they had more time, perhaps they would watch the lecture only

after a first editing attempt in order to evaluate the editing. The one user who edited while playing the lecture still did read the text while watching.

5.4.5.2 Changing Section Boundaries

Four out of 8 users discovered the section merge and split buttons in the toolbar and used them for changing section boundaries. Three out of 8 users copied text from one section and pasted it into an adjacent section as they attempted to change section boundaries. One user tried selecting adjacent sections to be merged as one might select multiple files in a file directory, by making mouse clicks and pressing the shift button.

One user said they did not discover the merge and split buttons because they did not even see the toolbar at the top of the screen. This user said they would have been more likely to see buttons had they been sitting between or inside the sections. Some other users said they saw the merge and split buttons but were not confident in how they worked so decided to execute their alternate plan instead, thinking it would work.

For these users who did not discover the section merge and split buttons, we allowed them some time to attempt section boundary changes using their own approach, and then we guided them toward the merge and split buttons.

In general, once users found the merge and split buttons they were able to use them correctly. Users had no problem with the Split Section button. However, some users were confused about the roles of the Merge Text and Merge Section buttons. A few users used only one of the two button types because they did not understand the difference between them. To move only some text from a given section to an adjacent section, a couple of users first clicked Split Section and then clicked Merge Section because they did not realize Merge Text could perform this operation in one

step. A different user tried to use Merge Text for merging entire sections but failed to do so correctly because they put the cursor down in the middle of the section and made no range selection, resulting in only some of the text being moved.

After they experimented with the merge and split buttons for some time, we asked the users who originally employed other methods for changing section boundaries how they felt about the buttons. In general these users said they now felt reasonably comfortable with the buttons but thought their initial approaches for changing section boundaries were more intuitive.

5.4.5.3 Titles for Merged and Split Sections

In general users thought that merging section titles upon merging adjacent sections made sense. One user commented that merging titles reminded them of the content of the two original sections and as a result helped them assign a new title covering the content of both sections. However, users said that if one of the original sections had not yet been titled by the user (i.e., had the default title “untitled#”), then they would prefer the merged section’s title omit this default title. As a result users would have less text to delete when re-titling the merged section or they could simply keep the single title.

Overall users seemed to think titling a new section, created with the Split Section button, “untitled” made sense.

5.4.5.4 Lecture Slides

When editing the lecture, some users specifically adjusted the section boundaries such that each slide in the lecture video appeared in exactly one VideoDoc section and each VideoDoc section contained exactly one slide. They did this even when the section text became very long.

A couple of users said they wanted to get an overview of all the slides in a given section, perhaps as a banner of screenshots at the bottom of the section, so that they could more quickly learn about the lecture content and determine places to split the section.

5.4.5.5 Representative Frames

One user was confused about how the representative frames for each section were chosen, and another user was confused about why the first section was the only one to not have a representative frame. As a reminder, the first section would not have a representative frame if the first section were entirely talking-head.

A couple of users were unsure of what would happen to representative frames upon merging two sections. They were not sure if the merged section would contain both representative frames or if one would be deleted. As a result, these users tried moving just text in an effort to not accidentally lose a representative frame. One user decided to use the Merge Text button rather than the Merge Section button, thinking this button would move just text and not representative frames and was therefore a safer option. The other user tried copying and pasting text manually.

A couple of users pointed out that sometimes a section's representative frame could be misleading. A section's representative frame is by default chosen to be the final frame of the section. If the slide shown in the final frame only appears for the last fraction of a second of the section, a user could perceive this representative frame as misleading because it does not truly represent the overall content of the section. We choose section boundaries based on sentence start and end times, and it is possible visual transitions will not perfectly align with section start and end times. This leads to potentially misleading representative frames. Misleading representative frames

made it challenging for users to quickly choose smart places for merging and splitting sections.

5.4.5.6 Styling Text

A couple of users tried using the styling buttons, such as bold and italics, because they were curious, but in general users did not try styling the text. We asked some users why they did not add styling, and they said they did not think it was necessary. There was only minimal styling in the student interface VideoDocs they saw previously (i.e., italics for variable names), so perhaps users did not see that italicized text or they thought styling was not a major feature of VideoDocs.

One user who did try styling selected a sentence with a single cursor click, turning the sentence yellow, and then clicked on the bold button in the toolbar. They were surprised to see the sentence not turn bold because they had thought clicking a styling button would appropriately style the currently yellow-highlighted text. Instead the styling buttons operate as in a traditional text editor, ignoring the yellow highlighting.

5.4.5.7 Editing Text

Users did not edit the text transcript, except for one user who deleted the “Eric Grimson:” speaker identifier at the beginning of the transcript. After they finished editing their VideoDoc, we asked two users if they thought any text should be changed and why they did not change any text. One user said they did not realize they could make text changes. Both users said they did not think it would make sense to edit the text because then the text would not perfectly correspond to the lecturer’s spoken words. The users said this lack of correspondence could potentially confuse users when they are watching the lecture and seeing different words as they read along with the transcript.

5.4.5.8 Other Observations

Three users tried to undo their section merge and split operations using the Ctrl-z “undo” keyboard shortcut, and said this is a feature they would like to have.

One user tried pressing the spacebar to toggle the play mode, a feature that is offered in most video player interfaces. However, this action did not toggle the play mode and instead added a space at the current cursor location. The user quickly realized that having this feature would not work well with the current text editor interface, but perhaps we should modify the interface to allow for this.

After they finished editing a section title, most users clicked out of the title textbox to commit the new title. These users clicked on the area to the right of the textbox, which is the content, and upon clicking the video began playing. This surprised all of these users and nearly all found it annoying. One user suggested having a small play icon over the content area and that only clicking on the small icon would play the video. Clicking elsewhere on the content area would do nothing, so likely clicking out of the section title textbox would not start playing the video.

6 Discussion and Future Work

We first discuss key user study findings and potential changes that should be made as a result. Second we discuss other limitations of VideoDoc and ways to address them.

6.1 User Study

Through the user study we found that users overall enjoyed using the VideoDoc student interface. Users said the ability to click on table of contents titles and to scroll through lecture slides made navigating VideoDoc lectures easier than navigating edX lectures.

Users commented on some features that they found distracting and other features they thought were missing. Some users were distracted by the talking-head appearing and disappearing in the corner of the page. A partial solution is to offer users the ability to hide the talking-head, but this does not help users who do want to see the talking-head and currently find its transitions distracting. We should explore other options for presenting the talking-head in VideoDoc that are less-distracting. Additionally a couple of users found the text transcript to be distracting while they were watching the lecture because they felt tempted to read along or

ahead. We might want to consider an option to hide the text transcript as well. Users also asked for common video player features including speed-up and full-screen display.

In the author interface some users did not discover the merge and split buttons for changing section boundaries without coaching. Instead they tried to copy text from one section and paste it into another to modify the section boundaries. Some users also did not understand the difference between the Merge Text and Merge Section buttons. We should work to improve the discoverability and learnability of changing section boundaries. We may also consider including multiple ways to change section boundaries, such as offering the current buttons option and adding the copy/paste text option. Based on user feedback we should also add the ability to undo and redo section boundary changes.

6.2 Other Future Work

One limitation of VideoDoc is that lectures cannot be created completely automatically. Lecturers must fine-tune section boundaries and add section titles, which could take much time over many lectures. Ideally VideoDocs should be created with better default section boundaries and default titles to reduce the lecturer's workload. We could employ techniques such as machine learning, natural language processing, and crowdsourcing.

The current implementation only takes one video as input, often a studio-edited lecture. Ideally VideoDoc should be able to take multiple videos as input, each one containing only talking-head, only content, or both and each one lasting arbitrary duration. This would allow authors to input separate talking-head and content videos and eliminate the need to join them together in one video.

We designed VideoDoc primarily for lectures whose focus is slides or tablet notes. Each section is meant to hold one slide screenshot in the content area. We did consider other lecture styles but realize that the current interface might not suit them well. For example, it is challenging to show a traditional lecture hall lecture video in VideoDoc. The lecturer might block the chalkboard when walking around and sometimes the camera will change perspective to show the audience. This kind of footage is not visual content and should not replace chalkboard notes in the VideoDoc content area, but it might not make sense to put this footage in the talking-head area either. We also considered green-screen lecture videos where the instructor appears in front of a green-screen displaying slides next to him or her. We prototyped separating the instructor and the slides, showing the instructor in the talking-head viewport and showing only the slides in the content viewport, but we did not explore this in depth. VideoDoc should also be developed more for humanities lectures where multiple pictures are shown in quick succession or where people are interviewed. Earlier we discussed using section thumbnails instead of one representative frame in these situations, but we should continue exploring designs and perform user testing. We also found that the current VideoDoc interface does not work well for videos that frequently flip between two important pieces of content, for example slides and the IDLE console in the 6.00.1x course. There is currently only one content viewport in VideoDoc and perhaps we should have a second content viewport for code or science demos.

Finally, there are two more editing widgets the author interface should offer. The first is a widget for choosing section thumbnails. Currently thumbnails can only be specified by manually editing the section JSON file. The second is a widget for editing the talking-head/content labels of a video in case the input labels are incorrect.

7 Conclusion

We have presented VideoDoc, a lecture video interface that improves a user's watching and reviewing experience by breaking a lecture into sections, displaying a static representation of each section, and providing a clickable table of contents for these sections. A VideoDoc lecture can be generated automatically from a lecture video, a time-annotated text transcript, and a labeling of talking-head video frames. We have also presented an editing interface that allows course instructors to modify a VideoDoc lecture's section boundaries, titles, and text. The VideoDoc interface appears promising for lecture videos with slides or handwritten notes but should be developed more for lectures of other styles.

8 References

- [1] Lori Breslow, David E. Pritchard, Jennifer DeBoer, Glenda S. Stump, Andrew D. Ho, and Daniel T. Seaton. "Studying learning in the worldwide classroom research into edX's first MOOC". *Research and Practice in Assessment*, 2013, pp. 13-25.
- [2] MITx. "6.00.1x Introduction to Computer Science and Programming". edX, Fall 2013. <https://courses.edx.org/courses/MITx/6.00.1x/3T2013/info/>.
- [3] René F. Kizilcec, Kathryn Papadopoulos, and Lalida Sritanyaratana. "Showing Face in Video Instruction: Effects on Information Retention, Visual Attention, and Affect". *CHI*, 2014, pp. 2095-2102.
- [4] Philip J. Guo, Juho Kim, and Rob Rubin. "How Video Production Affects Student Engagement: An Empirical Study of MOOC Videos". *Learning at Scale*, 2014, pp. 41-50.
- [5] Juho Kim, Philip J. Guo, Daniel T. Seaton, Piotr Mitros, Krzysztof Z. Gajos, and Robert C. Miller. "Understanding In-Video Dropouts and Interaction Peaks in Online Lecture Videos". *Learning at Scale*, 2014, pp. 31-40.
- [6] Juho Kim, Shang-Wen (Daniel) Li, Carrie J. Cai, Krzysztof Z. Gajos, and Robert C. Miller. "Leveraging Video Interaction Data and Content Analysis to Improve Video Learning". *CHI Workshop on Learning Innovations at Scale*, 2014.
- [7] Francis C. Li, Anoop Gupta, Elizabeth Sanocki, Li-wei He, and Yong Rui. "Browsing digital video". *CHI*, 2000, pp. 169-176.
- [8] Toni-Jan K. P. Monserrat, Shengdong Zhao, Kevin McGee, and Anshul V. Pandey. "NoteVideo: Facilitating Navigation of Blackboard-style Lecture Videos". *CHI*, 2013, pp. 1139-1148.
- [9] Gregory D. Abowd, Lonnie D. Harvel, Jason A. Brotherton. "Building a Digital Library of Captured Educational Experiences". *Kyoto International Conference on Digital Libraries*, 2000, pp. 395-402.

- [10] Bradley N. Miller and David L. Ranum. "Beyond PDF and ePub: toward an interactive textbook". *ITiCSE*, 2012, pp. 150-155.
- [11] edX. <http://www.edx.org/>.
- [12] Coursera. <http://www.coursera.org/>.
- [13] Udacity. <http://www.udacity.com/>.
- [14] MITx. "3.091x Introduction to Solid State Chemistry". edX, Spring 2015. https://courses.edx.org/courses/course-v1:MITx+3.091x_4+1T2015/info.
- [15] Columbia University in the City of New York. "Natural Language Processing". Coursera, Spring 2013. <https://www.coursera.org/course/nlangp/>.
- [16] "Applied Cryptography". Udacity. <https://www.udacity.com/course/applied-cryptography--cs387/>.
- [17] Khan Academy. <http://www.khanacademy.org/>.
- [18] "Separable equations". Khan Academy. <https://www.khanacademy.org/math/differential-equations/first-order-differential-equations/separable-equations/v/separable-differential-equations-introduction/>.
- [19] Bret Victor. "Media for Thinking the Unthinkable". <http://worrydream.com/MediaForThinkingTheUnthinkable/>.
- [20] Amy Pavel, Colorado Reed, Björn Hartmann, Maneesh Agrawala. "Video digests: a browsable, skimmable format for informational lecture videos". *CHI*, 2014, pp. 573-582.
- [21] "YouTube IFrame Player API". https://developers.google.com/youtube/iframe_api_reference/.
- [22] Michele Pratusевич. "EdVidParse: Detecting People and Content". M.Eng. Thesis, Massachusetts Institute of Technology, 2015.
- [23] node-fluent-ffmpeg. <https://github.com/fluent-ffmpeg/node-fluent-ffmpeg/>
- [24] Google Diff Match and Patch libraries. <https://code.google.com/p/google-diff-match-patch/>.
- [25] 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". *UIST*, 2014, pp. 563-572.