Extending Lecture Recording Systems

A simple proof of concept

Adam Reed
Division of Information
The Australian National University
Background to the Proof of Concept

What turned out to be an interesting research project
DLD
Digital Lecture Delivery
Lecture Recording System

- Podcast Producer based
- Mac Mini with a USB Epiphan Frame Grabber
  - Records what is sent to the projector
- All recordings are done on demand, not scheduled
- There is a mandate to record all lectures
Lecture Recording System

We generate a little bit of content...

- From 1st January 2012 to 3rd June 2012 (Summer and Semester 1)
- 7,704 recordings
- 365.5 days worth of content (8,772 hours)
Lecture Recording System
...that’s consumed by our community

• From 13th February 2012 to 3rd June 2012 (Semester 1)
  • 1,393,584 individual downloads, by
  • 9,784 unique students and staff, totalling
  • 89,241.64 GB of data transferred
Lecture Recording System

In any language

- Multiple Languages
- Content isn’t guaranteed to be in English
- Language both on slides and spoken can be intermixed
- Very popular to specialised like Sanskrit (14,113 native speakers as of 2001 Indian census)
- Highly domain specific language (chemistry, law, etc)

What drove the PoC?

Add value to binary blobs

- Recordings lectures is a solved problem!
- But what happens after the recording has been made?
- Can we add value to the users experience?
- Meetings about accessibility, and it’s associated requirements
WCAG 2.0
Web Content Accessibility Guidelines

• Wide range of recommendations about making web content more accessible for people with various disabilities, including but not limited to blindness or low vision and deafness or hearing loss

• Following these guidelines will also often make your content more usable to users in general

http://www.w3.org/TR/WCAG20/
WCAG 2.0
Web Content Accessibility Guidelines

• Content includes everything from the design, colours, layouts, alternative access mechanisms, etc

• This presentations focuses on audio visual content, referred to as time-based media within the guidelines

• Specifically pre recorded time-based media, vs live (streaming) media
WCAG 2.0
Web Content Accessibility Guidelines

• Guideline 1.2 - Provide alternatives to time-based media
  • Audio Only - Transcripts
  • Video Only - Audio equivalent, full text alternative
  • Audio - Video - Captions, Audio description, full text alternative, sign language, extended audio description

http://www.w3.org/TR/2008/REC-WCAG20-20081211/#media-equiv
WCAG 2.0

Levels

- The guidelines have 3 levels of compliance
  - A
  - AA
  - AAA
- Each level builds on the previous level
Quick Summary

<table>
<thead>
<tr>
<th>Alternative</th>
<th>WCAG 2.0 - A</th>
<th>WCAG 2.0 - AA</th>
<th>WCAG 2.0 - AAA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A Pre-recorded audio-only</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1 Transcript</td>
<td>Yes (1.2.1)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>B Prerecorded video-only</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B1 Audio equivalent</td>
<td>Yes (1.2.1)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>B2 Full text alternative</td>
<td>Yes (1.2.1)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>C Pre-recorded audio-video</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1 Captions</td>
<td>Yes (1.2.2)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>C2 Audio description</td>
<td>Yes (alt C3)* (1.2.3)</td>
<td>Yes (1.2.4)</td>
<td>Yes</td>
</tr>
<tr>
<td>C3 Full text alternative</td>
<td>Yes (alt C3)* (1.2.3)</td>
<td>No</td>
<td>Yes (1.3.8)</td>
</tr>
<tr>
<td>C4 Sign language</td>
<td>No</td>
<td>No</td>
<td>Yes (1.3.6)</td>
</tr>
<tr>
<td>C5 Extended audio description</td>
<td>No</td>
<td>No</td>
<td>Yes (1.3.7)</td>
</tr>
<tr>
<td><strong>D Live audio-only</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D1 Captions</td>
<td>No</td>
<td>No</td>
<td>Yes (1.3.9)</td>
</tr>
<tr>
<td><strong>E Live audio-video</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E1 Captions</td>
<td>No</td>
<td>Yes (1.2.4)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

WCAG 2.0 Driver
Mandated Federal Policy

- The Australian Federal Government has mandated compliance with WCAG 2.0 A by Dec 31st 2012, and AA by Dec 31st 2014
- For all Australian, State, and Territory government and agency websites
- Any website owned and/or operated by government under any domain for all internet, intranet, and extranet sites

What did I set out to test?

Whether we could add value to a lecture recording...
Simple Goals

How hard can it be?

• How could I take a potentially multi hour "blob" and enhance it, so that students could “find” content

• Chapter markers to enable jumping to the relevant spot in a recording

• Allowing searching within the video, and the ability to jump to the relevant spot

• With no budget
Tools and steps used in my workflow

Everything including the kitchen sink...
Tools

- All tools were either free, or open source (with one optional exception)

- Utilised Homebrew ([http://mxcl.github.com/homebrew/](http://mxcl.github.com/homebrew/)) to install a lot of the tools, which made my life far easier

- Glued together using Perl

- Based on H.264 encoded MP4’s
Step 1
Find the chapters

• Compared 3 tools
  • Podcast Producer - Chapterize
  • ImageMagick - Compare
  • Scene Detector - Scene Detector Pro
  • Commercial product, with a command line designed for Final Cut projects

Step 2
Massage the chapter data

- The tools all produced different data about the scenes
- Extract this data to get the following
  - Chapter #
  - Start time in SMPTE timecode
  - End time in SMPTE timecode
Step 3

Create chapter metadata

• From the massaged chapter data, create a csv file with
  • Start time of chapter in SMTPE
  • Chapter name
    (I used “Detected Chapter ###”)

XW12
Step 4
Add chapter markers to file

• MP4Box
  • Adds chapters from a CSV in Nero format
  • Good - we now have chapter markers in the file
  • Bad - nothing really can read or use these markers

http://gpac.wp.mines-telecom.fr/
Step 5
Convert chapter markers to Quicktime format

• mp4chaps (From MP4v2 Library)
  • Converts chapter markers from Nero to Quicktime format
  • Works on iOS devices, iTunes, Quicktime, VLC, and potentially others

http://code.google.com/p/mp4v2/
Achievement Unlocked

Students can now jump to the automatically detected scenes instead of needing to scrub through all of the video
Step 6
Capture a still frame at the chapter marker

- FFmpeg
  - Generate a jpg at each chapter marker, and save all of the resulting files

http://ffmpeg.org/
Step 7
Preform OCR on each of the still frames

- Tesseract-ocr
- Scan each jpg, and run optical character recognition over it
- Save the results

http://code.google.com/p/tesseract-ocr/
Step 8  
Create HTML 5 Player

• popcorn.js

• Use HTML5’s video element and associated javascript to create a player

• Show a table of the still frames and OCR text

• Give options to jump forward or back chapter

• Use browsers find feature to find the text and jump to the appropriate place

http://popcornjs.org/
Second Achievement

Students can now search for content (as long as it was displayed), and jump to the appropriate part of the lecture.
Results

How did it actually turn out...
Demo
Promising...

But there is a lot of room for improvement

- Scene detection isn’t too bad, but needs tweaking
- The tools have thresholds that can be modified - with a large sample set you could find some good defaults
- Design of slides greatly impacts ability to preform OCR, with results from spot on, to absolute gibberish
CPU Intensive

Required a lot of processing power

- Complete processing time was between 1/3 and 1/2 of the running time of the video
- This takes longer than it takes to compress the original file for distribution
- Could be optimised, but will add significant time to existing processing, requiring either more compute time, or longer wait for content
Where to from here?

Watch this space...
How do you do it?

Man vs Machine

- The automated tools aren’t really “there” yet
- Do you use people power to do the transcription and scene detection, or attempt the machine solution?
- Machine is far cheaper, but less accurate
- Lecture recording systems generate too much content for human based services to be cost effective
How do you correct it?

• Crowd Sourcing

• If using automated processes, how can you leverage students to

• Flag bad detection (so that the thresholds can be reviewed and tweaked) and the systems performance reviewed

• Make corrections (think Wikipedia for lecture content)
Discussion & Questions

Are you tackling similar issues, or do you have any insights that could shed some light on the topic?