Automated Tagging to Enable Fine-Grained Browsing of Lecture Videos

K.Vijaya Kumar  
(09305081)  

under the guidance of  
Prof. Sridhar Iyer  

June 28, 2011
Outline

1 Introduction
2 Motivation
3 Example Lecture Video Repositories
4 Problem Definition
5 Solution Approach
6 System Architecture
7 Implementation Details
8 Experiments and Evaluation Results
9 Conclusion and Future Work
**Outline**

1. Introduction
2. Motivation
3. Example Lecture Video Repositories
4. Problem Definition
5. Solution Approach
6. System Architecture
7. Implementation Details
8. Experiments and Evaluation Results
9. Conclusion and Future Work
Lecture video recordings are widely used in distance learning.

To make best use of the available videos a system called **Browsing System** is required.

Purpose of the browsing system is to provide search facility in the lecture video repository.

**Problem Statement**: To develop a browsing system which is useful for users to find their required video content easily.
Video Browsing System

- It takes keywords from users and gives them lecture videos matching their keywords
Outline

1. Introduction
2. Motivation
3. Example Lecture Video Repositories
4. Problem Definition
5. Solution Approach
6. System Architecture
7. Implementation Details
8. Experiments and Evaluation Results
9. Conclusion and Future Work
Text Search Example

(a) Query

(b) Results

(c) Finding Info

Figure: Google Search
Can we do the same in Lecture Videos?

Yes, We can provide the same type of search facility in lecture videos based on their contents

Example Scenarios

- Portion of video where *Matrix Multiplication* is discussed in a programming course lecture
- Searching for a video which discusses *Quick Sort* in a Data Structures course videos
- Finding video results containing *Double Hashing* in lecture video repository
Techniques for Searching in Lecture Videos

- **Meta data based**: Uses data such as video title, description or comments associated with the video.

- **Content based**: Based on data extracted from lecture videos, which represents contents present within it.
Motivation

How You Tube Searches Videos?

- Youtube video search is based on meta data associated with videos
- Meta data include video title, description and tags
Outline

1. Introduction
2. Motivation
3. Example Lecture Video Repositories
4. Problem Definition
5. Solution Approach
6. System Architecture
7. Implementation Details
8. Experiments and Evaluation Results
9. Conclusion and Future Work
Example Lecture Video Repositories

- CDEEP[5] : No search feature
- NPTEL[16] : No search feature
- freelecturevideos.com[8]
- videolectures.net[20]
- Lecture Browser, MIT[13]

Some more
- Academic Earth[1]
- Youtube Edu[23]

Link to list of available educational video repositories is at[15]
Slide Index feature in NPTEL

- Recently launched
- Through a video processing company called videopulp [21]
freevideolectures.com

- Provides Google custom search to index textual data
- Topic Looked for: **Double Hashing**
Keyword: double hashing

Result: Your search - double hashing - **did not match any documents.**
Keyword: hashing
Result: 6 video results

- Hashing Video Lecture IIT Delhi NPTEL Course, Videos, Video...
- Hashing NPTEL Video Online Course, Classes, video Tutorials, IIT Delhi Video Lectures, NeverenGap Audio Lectures, free Online Courses, Online Classes, ...
- freevideolectures.com/Course/2279/Data-Structures-And...

- Hashing, Hash Functions | MIT OCW Online videos, Erik Demaine...
- Hashing, Hash Functions Online Videos, Classes, Tutorials, MIT OCW Video Lectures, Erik Demaine Audio Lectures, free Online Courses, Online Classes, ...
- freevideolectures.com/Course/1641/Introduction-to-Algorithms/

- Universal Hashing, Perfect Hashing | MIT OCW Online videos, Erik...
- Universal Hashing, Perfect Hashing Online Videos, Classes, Tutorials, MIT OCW Video Lectures, Erik Demaine Audio Lectures, free Online Courses, ...
- freevideolectures.com/Course/1641/Introduction-to-Algorithms/

- Introduction to Algorithms Video Lectures by Prof. Erik Demaine...
  Topics covered include: sorting, search trees, heaps, and hashing, divide-and-conquer, dynamic programming; amortized analysis; graph algorithms; ...
  freevideolectures.com/Course/1641/Introduction-to-Algorithms
Example Lecture Video Repositories

freevideolectures.com

First video

- Duration - 61:22
- Found at - 42:32
Example Lecture Video Repositories

videolectures.net

- Provides free online access to lecture video recordings of various universities
- Has hyper links to slide change timings

**Probing strategies**

**Double hashing**

Given two ordinary hash functions $h_1(k)$ and $h_2(k)$, double hashing uses the hash function:

$$h(k,i) = (h_1(k) + i \cdot h_2(k)) \mod m.$$  

This method generally produces excellent results, but $h_2(k)$ must be relatively prime to $m$. It is to make $m$ a power of 2 and design $h$ to produce only odd numbers.
Example Lecture Video Repositories

Lecture Browser

- Provides free online access to lecture videos available in MIT Open Courseware
- Has Content-based Search feature and highlights relevant segments of each video
Going back look at the next student total marks but that's and that is valuable skilled statement is nothing going to be added this is a stack architecture exists on elements in this particular case we have x is ten first in the log one in the middle legal another one follow what i queries in it and then hit out i couldn't that column gordon also if i wanted but invalid intentional printing attention to matrix we call or and the difference because of that number example you all agree that such a die system be multiplied with so consequently if i then as a by n matrix a another matrix b which is n by p matrix multiplication of a might be matrix of entity this this simplest out a most specific every that every element of c is defined by the summation as indicated in the form binary with this that seek i j to system of constant eventually which it so for a given value of i and j which values given that it does some matrix the minute i mention is n so given that if someone to n so then i sum up all the matrix element on indefinitely that we will one that middle how do i element in a c plus plus program second element and element matrix if for example these n by an n then the number of rows and columns in that matrix that indexed by zero to n
# Features in Lecture Video Repositories

<table>
<thead>
<tr>
<th>Repository</th>
<th>Search</th>
<th>Navigation Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDEEP</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NPTEL</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>freelecturevideos.com</td>
<td>Meta data</td>
<td>No</td>
</tr>
<tr>
<td>videolectures.net</td>
<td>Meta data</td>
<td>Slide Index (Manual)</td>
</tr>
<tr>
<td>Lecture Browser, MIT</td>
<td>Content</td>
<td>Speech Transcript</td>
</tr>
<tr>
<td>Our System</td>
<td>Content</td>
<td>Speech Transcript Slide Index (Automated)</td>
</tr>
</tbody>
</table>

Table: Lecture Video Repositories Comparison
Problems with existing systems

**freevideolectures.com**
- No indication of where exactly searched keywords occur within the video
- Takes more time to find required information

**videolectuers.net**
- Uses manual process for Synchronization of the slides
Why can’t we use lecture browser?

- Can not be applied directly to our lecture videos.
- Requires speech recognition engine adaptation for non native english speakers
- Not an open source tool
- Their speech recognition engine is also not publicly available
How our system is different

- Provides automatic synchronization of slides.
- Improved user interface with more navigation features. It combines features in videolectures.net and lecture browser
- Open source application by integrating available speech recognition and text search engines
- Tune Sphinx speech recognition engine to recognize and transcribe Indian accents (English)
Outline

1. Introduction
2. Motivation
3. Example Lecture Video Repositories
4. Problem Definition
5. Solution Approach
6. System Architecture
7. Implementation Details
8. Experiments and Evaluation Results
9. Conclusion and Future Work
Problem Definition

- **Input:** keywords
- **Output:** List of videos matching the keywords

In each video portions where the keywords occur in the speech are highlighted
When user clicks on a particular portion video starts playing in the media player
Along with the media player user interface also shows slide index and speech transcript
• **Scope of the project**: Only deals with lecture videos which are in **English** and related **Computer Science** domain.

**Reason**: Speech Recognition Engine

**Figure**: Sphinx 4 Recognizer
Steps in Speech Recognition

1. Analog To Digital
2. Acoustic Modeling
3. Dictionary
4. Language Modeling

Problem Definition
Outline

1. Introduction
2. Motivation
3. Example Lecture Video Repositories
4. Problem Definition
5. Solution Approach
6. System Architecture
7. Implementation Details
8. Experiments and Evaluation Results
9. Conclusion and Future Work
Solution Approach

**Diagram Description:**

1. **Content Extraction**
   - Video File
   - Textual Data
   - Off-line Activities (1, 2)

2. **Indexing**
   - Index Tables
   - Repository
     - (Videos, Slides, Text Data, Index Tables, log files)

3. **Query Handling**
   - Query
   - Video Results
   - On-line Activities (3)
Solution Approach

Content Extraction

(a) Optical Character Recognition

(b) Speech Recognition
Speech Recognition Engines

- Sphinx 4 [18]
- Hmm Tool Kit (HTK) [9]

Reasons for choosing Sphinx

- Provides Java API (Application Programmable Interface)s, so it can be integrated easily into any application
- CMU Sphinx provides support for various tools useful in speech recognition
- Has easy configuration management where we need to set various parameters related to speech recognition
- Supporting tools are available for generation of acoustic and language models
- Completely written in java, it is highly modular and platform independent
Indexing & Query Handling

<table>
<thead>
<tr>
<th>ID</th>
<th>Word</th>
<th>Transcripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>control</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>internet</td>
<td>2, 3</td>
</tr>
<tr>
<td>3</td>
<td>protocol</td>
<td>1, 2</td>
</tr>
<tr>
<td>4</td>
<td>transmission</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>technology</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Transcript 1: It is about Transmission Control protocol
Transcript 2: It is on Internet protocol
Transcript 3: Deals with Internet technology

Discarded stop words
Text Search Engines

- Lucene[3], Indri[10]
- Xapian[22], Zettair[24]

Reasons for choosing Lucene

- It creates index of smaller size and search time is also very less[17]
- Supports ranked searching: best results returned first
- Can handle many powerful query types: phrase queries, wild card queries, range queries and more
- Mostly used text search engine. List of more than 150 applications and websites that are using Lucene to provide search facility[14]
Outline

1. Introduction
2. Motivation
3. Example Lecture Video Repositories
4. Problem Definition
5. Solution Approach
6. System Architecture
7. Implementation Details
8. Experiments and Evaluation Results
9. Conclusion and Future Work
System Architecture

System Components

Off-line Functional Units: 1, 2, 3, 4, 5

Online Functional Units: 6, 7, 8

Repository: Videos, Speech Transcripts, Slides, Tag files, Index Tables

User Interface: Query, Video Results

Query handling

Indexing

Speech Recognition

Extract Audio

Extract Frames

Slide Detection

Transcript

Audio File

Video Frames

Slide Index File

Index Tables
Outline

1. Introduction
2. Motivation
3. Example Lecture Video Repositories
4. Problem Definition
5. Solution Approach
6. System Architecture
7. **Implementation Details**
8. Experiments and Evaluation Results
9. Conclusion and Future Work
Audio Extraction

- Input: Video file
- Output: Audio file
- Command line tools provided by FFmpeg [7]
- Running `ffmpeg`:
  
  $ ffmpeg -i CS101_L10_Strings.mp4 -ar 16000 -ac 1 CS101_L10_Strings.wav
Speech Recognition

- Input: Audio file
- Output: Time aligned transcript in XML format
- Open source Java library for Sphinx-4 Speech Recognizer from CMU Sphinx [18]
- Requires language model, acoustic model and a pronunciation dictionary
Language model creation

- Large amount of text corpus related to the domain of speech recognition is required
- CMU SLM Toolkit [6] is useful for creating language model from the text corpus

Figure: Framework for creating large amount of text corpus
Language model creation

- Collected text corpus related to Computer Science domain
- Wiki Index: Randomly generated queries consisting of terms from CS and searched in Lucene Indexes
- Text books: Data structures, Algorithms, Computer Networks, DBMS and OS
- Converted PDF files to Text using Java library provided from PDFBox [11]
Acoustic model development

- Requires audio files and corresponding manual transcriptions
- Developing new acoustic modeling takes large amount of time
- Adaptation of acoustic model is an option which requires an existing model
- CMU Sphinx provides WSJ and HUB4 models useful for recognizing US English
- Sphinx Train and Sphinx Base are set of tools useful for development for acoustic model
Implementation Details

Acoustic model development

- We have to adapt an acoustic model to match our speakers to get better recognition accuracy.
- Time consuming, which requires small audio files each having a sentence and manual transcription of each of the audio file.
- Created 150 wav files for adaptation from CS101 lectures of Prof. Deepak Phatak.
- Each of the wav file duration is 2 to 5 seconds and gave manual transcriptions for them.
Speech Transcript Generation

- Configured the Sphinx-4 recognizer with the created language model and acoustic model
- Transcribed audio files of CS101 lectures and generated time aligned transcripts
- Transcribing of an audio file took approximately double the duration of the file
- The transcription speed can be increased but gives low recognition accuracy
Example Speech Transcript

<transcript>
  <tt>
    <text> deals with </text>
    <time> 7 </time>
  </tt>
  <tt>
    <text> searching </text>
    <time> 11 </time>
  </tt>
  <tt>
    <text> of lectures </text>
    <time> 14 </time>
  </tt>
</transcript>
Video Frames Extraction

- **Input**: Video file
- **Output**: Frames extracted from the video at specified intervals
- `ffmpeg` can be used for the frame extraction
- $ ffmpeg -i CS101_L10_Strings.mp4 -r 1 -f image2 image_%4d.jpeg$
Slide Detection

- Input: Video frames of a lecture
- Output: Slides of the lectures along with their title and time of occurrences
- Designed an algorithm based on slide title matching which uses OCR for slide text extraction
- Found an OCR tool called tesseract-ocr [19] which gives better recognition accuracy among available the Open Source tools
Example frame from a video lecture

Overview
- Engineering Education
- Research and Critical thinking
- Introduction to the course
- Quiz
After applying OCR

Overview
Engineering Education
He$searchar1&iUrilmu| lhirknng
lnirucluctivn tc the course
Oui;
# Title Matching algorithm for Slide Detection

<table>
<thead>
<tr>
<th>Title</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>overview</td>
<td>0104</td>
</tr>
<tr>
<td>overview</td>
<td>0105</td>
</tr>
<tr>
<td>overview</td>
<td>0106</td>
</tr>
<tr>
<td>overview</td>
<td>0107</td>
</tr>
<tr>
<td>overview</td>
<td>0108</td>
</tr>
<tr>
<td>overview</td>
<td>0109</td>
</tr>
<tr>
<td>overview</td>
<td>0110</td>
</tr>
</tbody>
</table>
| engineering | 0135  | → Will be identified as starting of a slide
| engineering | 0136  |
| engineering | 0137  |
| engineering | 0138  |
| engineering | 0139  |
| engineering | 0140  | → Will be identified as starting of next slide
while i < titles.length-1
begin
    if !titles[i].equals(prev) && matchesNextTwo(titles,i)
        indices.add(i);
        i = findNextSlide(titles,title[i],i+3)
    if i == -1
        return;
    endif
    prev = titles[i];
    indices.add(i);
    i = i + 2;
endif
i = i + 1;
end
Example Slide Index

<slides>
  <slide>
    <title> Overview </title>
    <time> 13 </time>
  </slide>
  <slide>
    <title> Introduction </title>
    <time> 79 </time>
  </slide>
</slides>
Indexing

- Input: Transcript file and Slide index file
- Output: Creates an Index or adds to existing indexes
- Apache Lucene [3] provides Java library for indexing text documents
- Parsed the transcript and slide index file which are in XML format
- Indexed CS101 lectures of Autumn 2009 and created indexes are of size 2.5MB
Query Handling

- Input: User given queries
- Output: List of lectures matching the query
- Apache Lucene [3] is also include Java classes for searching the indexes
- Technologies: Java Server Pages (JSPs) and Java Servlets
- Operating System: Ubuntu Lucid Lynx 10.04 LTS
Implementation Details

User Interface

- Created web pages using HTML and Java Script
- Using a freely available version of JW Player [12] for playing videos in the interface

![User Interface of our System](image)

**Figure:** User Interface of our System
Figure: Search Results for query **binary search**
User Interface

Figure: playing selected video with the navigation
Implementation Details

Content Repository

- Recorded videos of lectures
- Speech transcripts
- Slide Index files
- Lucene indices
Experiments and Evaluation Results

Outline

1. Introduction
2. Motivation
3. Example Lecture Video Repositories
4. Problem Definition
5. Solution Approach
6. System Architecture
7. Implementation Details
8. Experiments and Evaluation Results
9. Conclusion and Future Work
### Slide Detection Results

<table>
<thead>
<tr>
<th>Video</th>
<th>Actual slides</th>
<th>Detected slides</th>
<th>Correctly detected</th>
<th>Duplicates</th>
<th>Recall (%)</th>
<th>Prec. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>L_01</td>
<td>14</td>
<td>14</td>
<td>12</td>
<td>0</td>
<td>100</td>
<td>85</td>
</tr>
<tr>
<td>L_02</td>
<td>20</td>
<td>20</td>
<td>16</td>
<td>6</td>
<td>100</td>
<td>80</td>
</tr>
<tr>
<td>L_03</td>
<td>12</td>
<td>11</td>
<td>11</td>
<td>2</td>
<td>91.6</td>
<td>100</td>
</tr>
<tr>
<td>L_04</td>
<td>32</td>
<td>30</td>
<td>26</td>
<td>9</td>
<td>93.7</td>
<td>86.6</td>
</tr>
<tr>
<td>L_05</td>
<td>32</td>
<td>30</td>
<td>28</td>
<td>5</td>
<td>93.6</td>
<td>93.3</td>
</tr>
<tr>
<td>Total</td>
<td>110</td>
<td>105</td>
<td>93</td>
<td>18</td>
<td>95.4</td>
<td>88.5</td>
</tr>
</tbody>
</table>

**Table:** Slide Detection results
## Speech Recognition Results

<table>
<thead>
<tr>
<th>Adaptation files</th>
<th>Words in test files</th>
<th>Matches</th>
<th>Accuracy(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>127</td>
<td>22</td>
<td>13</td>
</tr>
<tr>
<td>30</td>
<td>119</td>
<td>43</td>
<td>31</td>
</tr>
<tr>
<td>60</td>
<td>124</td>
<td>70</td>
<td>52</td>
</tr>
<tr>
<td>90</td>
<td>120</td>
<td>76</td>
<td>59</td>
</tr>
<tr>
<td>120</td>
<td>110</td>
<td>69</td>
<td>61</td>
</tr>
<tr>
<td>150</td>
<td>123</td>
<td>82</td>
<td>62</td>
</tr>
</tbody>
</table>

*Table: Speech Recognition results*
Video Retrieval Results

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No.of queries tested</td>
<td>30</td>
</tr>
<tr>
<td>Avg Search seconds</td>
<td>0.004</td>
</tr>
<tr>
<td>Recall</td>
<td>0.72</td>
</tr>
<tr>
<td>Avg Precision</td>
<td>0.91</td>
</tr>
</tbody>
</table>

**Table:** Search Quality Results
Conclusion and Future Work

Outline

1. Introduction
2. Motivation
3. Example Lecture Video Repositories
4. Problem Definition
5. Solution Approach
6. System Architecture
7. Implementation Details
8. Experiments and Evaluation Results
9. Conclusion and Future Work
Conclusion and Future Work

- Built a system for providing search facility in CS101 Autumn 2009 lectures
- Speech recognition accuracy can be improved through more adaptation
- Slide Detection method can be improved to reduce duplicate slides
- More lectures can be added to the repository
Conclusion and Future Work

- **Academic Earth.**
  http://academicearth.org/.

- **Apache : An Open Source Web Server.**
  http://tomcat.apache.org/.

- **Apache Lucene.**

- **Audio/Video Lectures from MIT OCW.**

- **CDEEP, IIT Bombay.**
  http://www.cdeep.iitb.ac.in/.

- **CMU Statistical Language Modeling Toolkit Documentation.**
  http://www.speech.cs.cmu.edu/SLM/toolkit_documentation.html/.
Conclusion and Future Work

- **FFmpeg.**
  http://www.ffmpeg.org/

- **freevideolectures.com.**
  http://www.freevideolectures.com/

- **HTK.**
  http://htk.eng.cam.ac.uk/

- **Indri.**
  http://www.lemurproject.org/indri/

- **Java PDF Library.**
  http://pdfbox.apache.org/

- **JW Player.**
  http://www.longtailvideo.com/players/jw-flv-player/

- **Lecture Browser, MIT.**
http://web.sls.csail.mit.edu/lectures/.

*List of Applications that are using Lucene.*

*List of educational video websites.*

*nptel.*
http://www.nptel.iitm.ac.in/.

*Open Source Text Search Engines Evaluation Results.*

*sphinx.*
http://www.speech.cs.cmu.edu/.

*tesseract-ocr.*
Conclusion and Future Work

- videolectures.net.
  http://www.videolectures.net/.

- VideoPulp: Official Partners for Slide Index feature in NPTEL.
  http://www.videopulp.in/.

- xapian.
  http://xapian.org/.

- Youtube Edu.
  http://www.youtube.com/education?b=400.

- zettair.
Thank You