2011 IEEE International Conference on Technology for Education

AUTOMATED TAGGING TO ENABLE FINE-GRAINED BROWSING OF LECTURE VIDEOS

Vijaya Kumar Kamabathula
Department of Computer Science & Engg
Indian Institute of Technology
Email: kvijay@iitb.ac.in

Sridhar Iyer
Department of Computer Science & Engg
Indian Institute of Technology
Email: sri@iitb.ac.in

***Abstract*—Many universities offer distance learning by recording classroom lectures and making them accessible to remote students over the Internet. A university's repository usually contains hundreds of such lecture videos. Each lecture video is typically an hour's duration and is often monolithic. It is cumbersome for students to search through an entire video, or across many videos, in order to find portions of their immediate interest. It is desirable to have a system that takes user-given keywords as a query and provides a link to not only the corresponding lecture videos but also to the section within the video. In order to do this, lecture videos are sometimes tagged with meta-data to enable easy identification of the different sections. However, such tagging is often done manually and is a time-consuming process.**

In this paper, we propose a technique to automatically generate tags for lecture videos. This is based on generating speech transcripts automatically using a speech recognition engine and automatic indexing and search of the transcripts. We also describe our system implemented for easily browsing through a lecture video repository. Our system takes keywords from users as a query and returns a list of videos as the results. In each video of the retrieved list, the portion of the video that matches the query is highlighted so that users can easily navigate to that location within the video. Following the approach and using open source tools mentioned in the paper, a lecture video repository can provide features for users to access the content required by them easily.

We used open source libraries available for speech recognition and text search purposes. We have performed experiments to test the performance of our system, we have achieved a recall of 0.72 and an average precision of 0.84 as video retrieval results.

***Index Terms*—Speech Recognition, Text Search, Slide Synchronization, Lecture Videos, Tagging.**

I. Introduction

Many universities offer distance learning by recording classroom lectures, maintaining hundreds of lecture video recordings in a repository, and making them accessible to remote students over the Internet [1]. It is often cumbersome for students to search through an entire video, or across many videos, in order to find portions of their immediate interest. For example, one repository (NPTEL [6]) has a course on Data Structures, containing around 40 recorded lecture videos, each of approximately 60 minutes duration. Suppose a user wants to find the portion where Heap Sort is discussed, the user has to manually go through titles of all the lectures in the course and first decide which videos might contain the desired explanation. Then the user has to individually browse through each of the chosen videos to find the portion where Heap Sort is discussed. The problem is exacerbated if the user is not familiar with the area, or if the topic is very specific, since the user may not be able to decide the videos to be scrutinized.

Hence, it is desirable to have a system that takes user-given keywords as a query and provides a link to not only the corresponding lecture videos but also to the section within the video. A few such systems exist (discussed in Section 2). In some of them, meta-data is associated (manually) with the lecture videos, while others use proprietary software to generate an index from the video contents. In this paper, we describe the design and implementation of a system for automatic tagging and fine-grain browsing of lecture videos. We use available open source tools for speech recognition engine and text search engine to develop the overall system. In the experiments to test the performance of our system, we achieved a recall of 0.72 and an average precision of 0.84 as video retrieval results by searching in automatically generated tag file using speech recognition. We are trying to improve the accuracy of speech recognition engine to get better recognition results.

II. Background

In this section we provide a brief background on techniques for searching in lecture videos and a survey of search facilities currently supported in some well known courseware repositories.

A. Techniques for searching in lecture videos

Search facilities can be provided in lecture video repositories in two ways [2]. They are:

- ***Meta-data based***: Meta data is textual data that is applied to a piece of multimedia content in order to describe it. These methods use meta data, such as video title, video description and user comments, to identify video results matching given set of keywords. Such a meta data based approach may be able to identify videos that contain the keywords but they cannot locate where those keywords appear in the video time line.

- ***Content based***: Lecture videos typically contain the following contents [3]: (i) Lecturer Speech: Portion of video that shows the instructor talking, (ii) Presentation Slides: Portion of video that shows the current slide of the presentation, and (iii) Notes: Portion of video that shows the board/paper on which instructor is writing. Content based approaches extract meta-data from appropriate portions of the video and create an index that can be used for searching within the video. Such techniques are difficult to automate and time-consuming to do manually.

B. Existing lecture video repositories

NPTEL [6], videolectures.net [7], freevideolectures.com [8] and MIT Lecture browser [9] are some of the existing lecture video repositories. We investigated support for search and browsing features available in those repositories. A list of some more video repositories providing educational video contents is given in [10]. Unfortunately, many of the repositories are not providing search functionality for their users. Some repositories have manual transcriptions (subtitles) for lecture videos but they are not making use of them to provide search features.

NPTEL currently does not have any overall video search facility. A user has to manually first pick a course and then go through titles of all the lectures in the course, to decide which videos might contain the desired explanation. Then the user has to manually browse through the videos to identify the portion of interest. videolectures.net, freevideolectures.com and the system described in [4], use meta-data based approach to provide search features in the video repository. They are not using contents present inside a lecture video for providing the search. A method for synchronization of lecture video timings with the lecture slides is presented in [5]. This method is based on using a plug-in in the slide presentation software that stores the slide transition timings during the lecture. So it has the limitation of not being applicable to lecture videos that have already been captured. MIT Lecture browser uses lecturer speech for searching and provides fine-grain browsing features [1] than the other repositories surveyed. However, the system is not publicly available (for use with other repositories) and its speech recognition engine is also not an open source tool.

The table I summarizes the comparison of features available in these video repositories. The last row of the table indicates the features provided in our system.

Repository	**Search**	**Navigation Features**
NPTEL	No	No
freelecturevideos.com	Meta data	No
videolectures.com	Meta data	Slide Synchronization (manual)
Lecture Browser, MIT	Content	Speech Transcript
Proposed System	Content	Speech Transcript & Slide Synchronization (automatic)

TABLE I
LECTURE VIDEO REPOSITORIES COMPARISON

III. Problem Definition

For any repository of lecture videos, our goal is to build a system that given a set of search keywords, (i) return a list of lecture videos that contain the keywords. (ii) highlight the portions of each video where the keyword occurs. (iii) allow the user to navigate directly to any highlighted portion of the video. (iv) provide direct links to starting locations of presentation slides for easily navigating through slide portions

Users enter keywords in search box provided in the user interface. For example, if a student is looking for video lectures in which a networking topic called TCP (Transmission Control Protocol) is discussed, he/she can enter “tcp” in the search box. The queries can be simple keywords, keywords with operators such as +/- and also titles of courses.

The output user interface is a results list of lecture videos matching the keywords. In each video, portions where the keywords occur (in the speech) are highlighted. When the user clicks on a particular portion, the video starts playing in the media player present in user interface. Along with the media player, the user interface also shows hyper-links to slide transitions, current slide and the speech transcript. The user can also click on any word present in *speech transcript* and directly go to the corresponding time in the currently playing video.

The scope of our system is currently limited to lecture videos which are in English. The reason for this is that one of the major components of our system is a automatic speech recognition engine, whose accuracy is influenced by *acoustic models* and *language models* [1]. We need large amount of text corpus for creating language models. The Language models are domain dependent so we need to collect text corpus related to the domain in which we want to apply speech recognition. We collected corpus for the Computer Science domain to recognize lectures related to that field. In order to apply our system in other domains, corresponding *corpus* has to be generated.

IV. Solution Overview

We follow a content based approach for providing search features in lecture video repositories. There are two technologies useful for extracting textual information from lecture video contents. One is Optical Character Recognition (OCR) and the other is Speech Recognition. We adopted speech recognition in our approach, as it is more suitable for lecture videos. More information can be extracted using speech recognition, as most of the portion of lecture videos contains lecturer speech. Following steps describe the overall process of our solution approach.

1) Take a video lecture and generate audio file containing corresponding speech of the lecture.

2) Input the audio file to automatic speech recognition engine to get a transcript which is time aligned textual representation of actual speech of the lecture.
3) Give speech transcripts of all the available lecture videos to a text search engine to produce index tables and store these index tables in database.
4) Generate a log file for each lecture which contains slide transition timings of the lecture.
5) Develop user interface of the system that takes queries from users and presents them with results from the database.

V. System Architecture and Implementation

The various components involved in development of our system (shown in figure 1) can be divided into two types: off-line and on-line components.

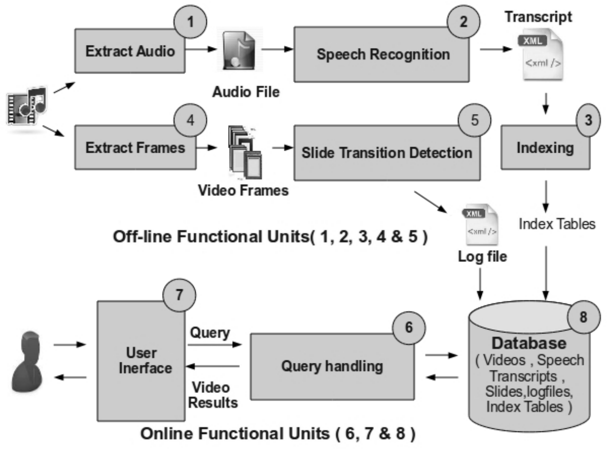

Extract Audio
1
Audio File
Speech Recognition
2
Transcript
XML
Extract Frames
4
Video Frames
Slide Transition Detection
5
Indexing
3
Log file
Index Tables
Off-line Functional Units(1, 2, 3, 4 & 5)
User Inerface
7
Query
Video Results
Query handling
6
Database (Videos , Speech Transcripts , Slides,logfiles, Index Tables)
8
Online Functional Units (6, 7 & 8)

Fig. 1. Our System

A. *Off-line Components*

These components perform activities at the server side. It involves processing of each lecture video in the repository and preparing an index for providing search facility based on contents of the lectures.

1) ***Audio Extraction***: This component of the system takes a lecture video as input and produces an audio file which contains lecturer speech. As speech recognition engine takes audio file as input, we need to extract audio track from the video.
FFmpeg [14] is an open source tool which provide libraries and programs for handling multimedia data. It provides a command line tool *ffmpeg*, which can extract audio track from a given video file. It just takes a given video file of one format as input and produces an audio file in specified format. There will not be any effect on the original video file
2) ***Speech Recognition***: This component takes audio file generated in the audio extraction unit as input and creates a speech transcript. The speech transcript contains individual spoken words in textual form along with their time of occurrence in the lecture video.
A sample speech transcript looks as shown in the following paragraph. The **word** tags represent each individual word in the speech transcript. The text between **text** tags the word present in the lecture speech. Value between **start** and **end** tags denotes starting time and ending time (in seconds) of the corresponding word in the speech.

```
<?xml version="1.0" ?>
<transcript>
  <word>
    <text>deals with</text>
    <start>5</start>
    <end>7</end>
  </word>
  <word>
    <text>internet related</text>
    <start>8</start>
    <end>11</end>
  </word>
  <word>
    <text>technologies</text>
    <start>11</start>
    <end>13</end>
  </word>
</transcript>
```

Example Speech Transcript

CMU Sphinx [13] provides open source tools and libraries for speech processing. Sphinx 4 [15] is a speech recognition engine from CMU Sphinx and is developed in Java. Sphinx 4 provides easy configuration management where we need to set various parameters related to speech recognition such as acoustic model, language model etc. It also supports tools for creating acoustic and language models. As it is completely written in Java, it is highly modular and platform independent. So we have chosen Sphinx 4 as speech recognizer in our system. A speech recognition engine uses *acoustic model* and *language model* for automatically converting spoken words into text. It also requires a *pronunciation dictionary* that

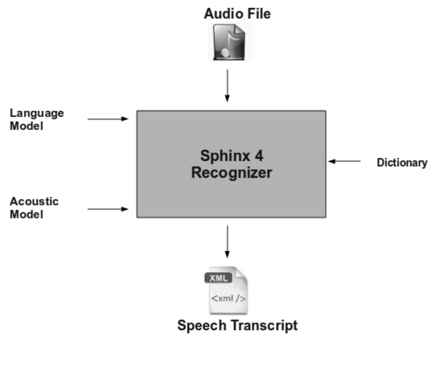


Fig. 2. Sphinx 4 Recognizer

maps each word to its phoneme representation.

Acoustic Model: The *acoustic model* represents how individual characters are pronounced, also known as phonemes. It gives statistical representation of distinct sounds that make up each word in a language.

Acoustic model creation needs large amount of speech corpus and their corresponding accurate speech transcriptions. As it is a time consuming process to generate new acoustic models, we configured sphinx 4 system with open source acoustic models provided by CMU Sphinx. It provides WSJ and HUB4 acoustic models which are trained for the recognition of American English accents. So recognition accuracy is very low (less than 6%) for our lectures which are in Indian English accent.

In order to improve the recognition accuracy of Sphinx 4 decoder, we have adapted the HUB4 models for recognizing our lecturer speech. Sphinx base and Sphinx Train provide tools useful for the acoustic model adaptation. After the adaptation the recognition accuracy increased to 45%.

The adaptation requires manually transcribing 3 to 4 minutes of speech of lecturer. This process takes the segmented audio files and manually transcribed sentences as input and generates acoustic models adapted for the lecturer speech.

Language Model: A *language model* represents grammar of a language and is a file containing the probabilities of sequences of words. It gives the probability of word (X) being followed by word (Y) so that the word sequence which is having higher probability will be chosen during decoding process. Language modeling requires supplemental text files, such as journal articles, book chapters, etc., to adapt the vocabulary of the system.

CMU Sphinx provides statistical language modeling toolkit to create language models. Using the tools provided by toolkit we can generate language model from available text corpus. We have collected text corpus related to computer science courses to create language model for Sphinx 4 decoder. The text corpus includes electronic versions of text books, presentation slides and available manual transcriptions.

3) ***Index Generation***: Index Generation constructs index tables from speech transcripts and stores them in the repository. It takes the XML files generated in speech recognition and slide detection phase for constructing indexes.

   We have chosen Lucene text search engine for indexing purpose. Lucene Java library contains classes for creating indices of a set of documents. IndexWriter class is one of them and useful to create indexes and store them in a given directory.

4) ***Slide Transition Detection***: This component identifies slide transition timings in a lecture video and creates a file containing slide title and time at which slide occurred in the video. This takes frames extracted from video and examines each frame to identify slide transition.

   It is required to extract individual frames of the video, we extracted one frame of the lecture video per second using *ffmpeg*. Our algorithm to detect slides in a lecture video is based on matching titles of individual frames of the video. It uses Optical Character Recognition (OCR) technology for extracting frame text. Tesseract OCR [17] is one of the open source tool which gives better accuracy. It takes a binary, gray scale or color image and save text present in the image into a separate file.

   A sample log file generated for slide synchronization looks as shown in the following paragraph. The **slide** tags represent each slide of the lecture. Each of the slide is an image and its location is specified in **url** tags. The time in seconds at which slide transition occurs is indicated in **time** tags.

```
 <?xml version="1.0" ?>
<slideshow>
  <slides>
    <slide>
      <url>slide1.jpg</url>
      <time>10</time>
    </slide>
    <slide>
      <url>slide2.jpg</url>
      <time>20</time>
    </slide>
  </slides>
</slideshow>
```

Example log file for slide synchronization

B. On-line Components

These components work at the server side through the Internet. These include requests made by users for particular content and the responses given by the system.

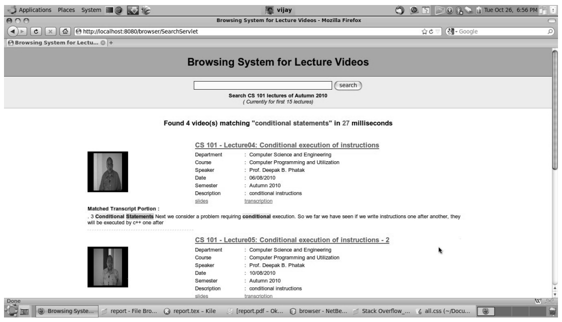
1) ***Query Handling***: It takes user input keywords and searches in database for matching video lectures and gives the video results as output.

   This component development involved server side programming which has been done using Java Servlets. Lucene also provides Java classes for searching in the created indexes.

   The figure 3 shows interface which shows search results for a user entered queries.

2) ***Database***: It stores all the information related to lecture videos. It contains actual video recordings of the lectures, presentation slides, speech transcriptions generated during content extraction, log files for slide synchronization and index tables created during indexing process.

3) ***User Interface***: User interface contains a search box in which user can enter keywords to search for videos in


Applications Places System
vijay
Tue Oct 26, 6:56 PM
Browsing System for Lecture Videos - Mozilla Firefox
http://localhost:8080/browser/SearchServlet
Google
Browsing System for Lectu...
search
Done
Browsing Syste... report - File Bro... report.tex - Kile [report.pdf - Ok... browser - NetBe... Stack Overflow_... all.css (~/Docu...

Browsing System for Lecture Videos

Search CS 101 lectures of Autumn 2010
(Currently for first 15 lectures)

Found 4 video(s) matching "conditional statements" in 27 milliseconds

CS 101 - Lecture04: Conditional execution of instructions

Department	: Computer Science and Engineering
Course	: Computer Programming and Utilization
Speaker	: Prof. Deepak B. Phatak
Date	: 06/08/2010
Semester	: Autumn 2010
Description	: conditional instructions
slides	transcription

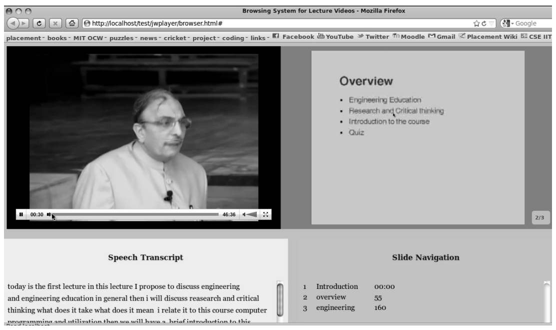
Matched Transcript Portion :
. 3 Conditional Statements Next we consider a problem requiring conditional execution. So we far we have seen if we write instructions one after another, they will be executed by c++ one after

CS 101 - Lecture05: Conditional execution of instructions - 2

Department	: Computer Science and Engineering
Course	: Computer Programming and Utilization
Speaker	: Prof. Deepak B. Phatak
Date	: 10/08/2010
Semester	: Autumn 2010
Description	: conditional instructions
slides	transcription

Fig. 3. *Search Results Page*

which the particular topic is discussed. After retrieving results from database, list of lecture videos are displayed in the interface.
Our system is a web based tool and runs on standard browsers. We are using Apache Tomcat, an open source web server at the server side. Web pages are developed using HTML, CSS, Java Script and Java Server Pages (JSPs). The figure 4 shows user interface of our system. As shown in the figure, the interface contains the following four sections.


Browsing System for Lecture Videos - Mozilla Firefox
http://localhost/test/jwplayer/browser.html#
Google
placement books MIT OCW puzzles news cricket project coding links Facebook YouTube Twitter Moodle Gmail Placement Wiki CSE IIT
00:30
46:36
2/3

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

Speech Transcript

today is the first lecture in this lecture I propose to discuss engineering and engineering education in general then i will discuss reasearch and critical thinking what does it take what does it mean i relate it to this course computer programming and utilization then we will have a brief introduction to this

Slide Navigation

1	Introduction	00:00
2	overview	55
3	engineering	160

Fig. 4. *User Interface*

a) *Video player*: As shown in the top left part of the figure 4, it plays video of the selected lecture. Users can navigate to any particular location by clicking on video time line of the player.
We are using a JW player [16] for playing videos in the interface.It supports various plugins for controlling the player and handling events. Navigation based on speech transcript has been done using Java Script and JW Player APIs.
b) *Current slide*: It is the top right part of the figure 4. It shows presentation slide of the corresponding video portion.
c) *Speech transcript*: It is the bottom left part of the figure 4. It displays textual representation of the lecturer speech. User can click on any particular word of the speech transcript and can jump directly to that location.
d) *Slide Navigation*: It is the bottom right part of the figure 4. It gives hyper links to slides of the video being played. Each link represents title of the slide and time at which it starts. So users can click on any required slide and go directly to that location of the video.

VI. Evaluation and Experiments

This section presents evaluation metrics and experimental results we have got during the implementation so far.

A. Evaluating Speech Recognition

In order to evaluate the Sphinx-4 system, we have to compare the output text called hypothesis with the actual transcription named reference. Three error types are distinguished in this process.

- *Substitution*: deals with words that are wrongly recognized (including singular and plural differentiation).
- *Insertion*: additional word in the hypothesis that is absent in the reference.
- *Deletion*: word present in the reference but not in the hypothesis.

Word Error Rate (WER) and Word Accuracy (WA) are useful for measuring the performance of speech recognition [11]. WER is defined as

$$WER = \frac{S + D + I}{N}$$

where

- N is the total number of words in the reference text.
- S is the number of substitutions.
- D is the number of deletions.
- I is the number of insertions.

WA is defined as

$$WA = 1 - WER = \frac{N - S - D - I}{N}$$

B. Evaluating Video Retrieval

An information retrieval system can be evaluated based on two parameters, called recall and precision [12]. Here document means video lecture in our case. As our system also deals with information retrieval, we are going to use those metrics for evaluation of the system. Recall and precision values describe the search quality of an information retrieval system.
Recall :
Performance of the search system in finding relevant documents can be measured using recall. It is a measure of the ability of a retrieval system to present all relevant documents.

$$Recall = \frac{Number\ of\ relevant\ videos\ retrieved}{Total\ number\ of\ existing\ relevant\ videos}$$

Precision :
Precision measures how well the system filters out the irrelevant. It is a measure of the ability of a system to present

only relevant documents.

$$Precision = \frac{Number\ of\ relevant\ videos\ retrieved}{Total\ number\ of\ videos\ retrieved}$$

The goal of an information retrieval system is to maximize both recall and precision.

C. Experiments

To build basic functionality of the system, we took the lectures of CS 101: Computer Programming and Utilization by Prof. Deepak B Phatak. We collected text corpus related to programming and data structures to create language model using CMU language modeling toolkit. Then we adapted HUB 4 acoustic models to voice of the professor using tools provided by sphinx train. We configured sphinx 4 recognizer with the created language and acoustic models to provide speech transcriptions. We have provided the search facility in those 20 lectures by indexing the speech transcripts using lucene text search engine. User can enter keywords and our system searches in the indexed transcripts, and then retrieves videos that match user input.

*1) **Audio Extraction**:* As mentioned in the previous sections we used Java libraries provided by *ffmpeg* for extracting audio track from a video file. We tested the performance of this unit on video lectures of CS 101. The table II shows results of audio extraction on some of the lecture videos.

Input	Length	Output (Audio File)	Extraction Time
Lec_1.mp4	46 min 36 sec	Lec_1.wav	21 sec
Lec_2.mp4	59 min 38 sec	Lec_2.wav	27 sec

TABLE II
Audio Extraction Results

*2) **Video Retrieval**:* We have built the system to provide search facility in the videos lectures of CS 101 course. The table III shows search quality of Lucene based on the metrics mentioned in the above section. Quality package under benchmark of Lucene allows us to measure recall and precision of the Lucene indexing mechanism.

The process of measuring recall and precision requires following three files as input.

- *Index file*: Lucene generated file during indexing process.
- *Topics file*: This contains list of search queries for which we want to measure the recall and precision.
- *Qrel file*: This is filled with file names of expected video lectures for each of the query mentioned in topics file.

No.of queries tested	10
Avg Search seconds	0.007
Recall	0.72
Avg Precision	0.84

TABLE III
Search Quality Results

*3) **Speech Recognition**:* As mentioned in the previous sections, Sphinx 4 decoder requires acoustic and language models for recognition. Language model can be generated by collecting text corpus related to the domain. Generation of these acoustic models is a time consuming process which would require producing manual transcripts for several hours of audio. For example, the pre-trained HUB 4 acoustic models available from the CMU Sphinx group were trained on 140 hours of English Broadcast News Speech from the 1996 and 1997 hub 4 corpora.

We have adapted the HUB 4 models for recognizing our lecture videos voice and tested the recognition performance. It gave word accuracy (WA) of 45% which is not good enough for our purpose. So we are collecting more data for adaptation for improving the results.

VII. Conclusion and Future Work

We have proposed design and implementation of a web based browsing system for lecture videos. It facilitates users to easily find required video contents at a fine-level of granularity. We are using open source speech recognition engine and text search engine to build overall system. By following the approach and set of tools mentioned in our paper, lecture video repositories can be made easily accessible for their users.

We have built a basic prototype of the system by providing search facility in lecture videos on CS 101: Computer Programming and Utilization course. We have performed experiments to test the performance of our system. We achieved a recall of 0.72 and an average precision of 0.84. We have achieved an average word recognition accuracy of 45% in the speech transcription process. The low value of recognition accuracy affected the decrease in recall and precision results. We are trying to improve the accuracy of speech recognition engine by collecting more data for adaptation to get better overall results.

Acknowledgment

We would like to thank the people involved in development of open source tools which are being used in our system. We would also like to thank Prof. Deepak B Phatak for providing us with CS 101 lecture videos and presentation slides of the course.

References

[1] Glass, J., Hazen, T., Cyphers, S., Malioutov, I., Huynh, D., Barzilay, R.: Recent Progress in the MIT Spoken Lecture Processing Project. In Proc. Interspeech 2007, Antwerp (2007), pp. 2553 - 2556.

[2] Alan F.Smeaton, *Techniques used and open challenges to the analysis,indexing and retrieval of digital video*, Adaptive Information Cluster and Center for Digital Video Processing , Dublin City University, 2006.

[3] Ganesh Narayana Murthy and Sridhar Iyer, *Study Element Based Adaptation of Lecture Videos to Mobile Devices*, National Conferece on Communications,Chennai, 2010.

[4] Jesse Jin and Ruiyi Wang, *The Development of an Online Video Browsing System*, ACM International Conference Proceeding series; Vol. 147, 2001.

[5] H. Sack and J. Waitelonis, *Automated Annotations of Synchronized Multimedia Presentations*, Proc. Workshop Mastering the Gap: From Information Extraction to Semantic Representation (ESWC 06), 2006.

[6] NPTEL
http://www.nptel.iitm.ac.in/

[7] videolectures.net
http://www.videolectures.com/
[8] freevideolectures.com
http://www.freevideolectures.com/
[9] Lecture Browser, MIT
http://web.sls.csail.mit.edu/lectures/
[10] List of educational video repositories
http://en.wikipedia.org/wiki/List_of_educational_video_websites
[11] Word Error Rate
http://en.wikipedia.org/wiki/Word_error_rate
[12] Recall and Precision
http://en.wikipedia.org/wiki/Precision_and_recall
[13] CMU Sphinx - An Open Source Toolkit for Speech Recognition
http://cmusphinx.sourceforge.net/
[14] FFmpeg
http://www.ffmpeg.org/
[15] CMU Sphinx 4 Speech Recognition Engine
http://cmusphinx.sourceforge.net/sphinx4/
[16] JW Player
http://www.longtailvideo.com/players/jw-flv-player/
[17] Tesseract Open Source OCR tool
http://code.google.com/p/tesseract-ocr/