An Automated Analysis and Indexing Framework for Lecture Video Portal

Haojin Yang, Christoph Oehlke, and Christoph Meinel

Hasso Plattner Institute (HPI), University of Potsdam, Germany
{Haojin.Yang,Meinel}@hpi.uni-potsdam.de,
Christoph.Oehlke@student.hpi.uni-potsdam.de

Abstract. This paper presents an automated framework for lecture video indexing in the *tele-teaching* context. The major issues involved in our approach are content-based lecture video analysis and integration of proposed analysis engine into a lecture video portal. In video visual analysis, we apply automated video segmentation, video OCR (*Optical Character Recognition*) technologies for extracting lecture structural and textual metadata. Concerning ASR (*Automated Speech Recognition*) analysis, we have optimized the workflow for the creation of a German speech corpus from raw lecture audio data. This enables us to minimize the time and effort required for extending the speech corpus and thus improving the recognition rate. Both, OCR and ASR results have been applied for the further video indexing. In order to integrate the analysis engine into the lecture video portal, we have developed an architecture for the corresponding tasks such as, e.g., data transmission, analysis management, and result visualization etc. The accuracy of each individual analysis method has been evaluated by using publicly available test data sets.

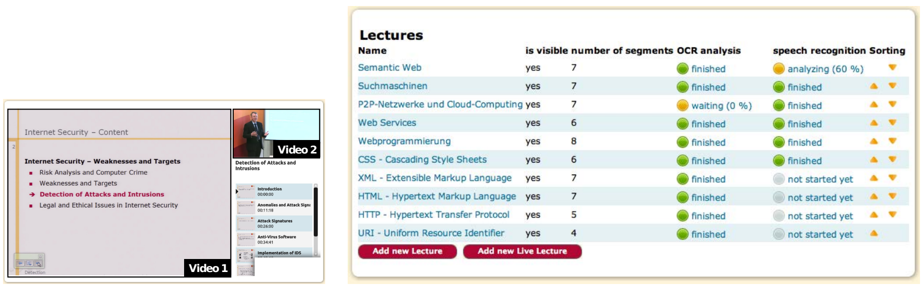
Keywords: Lecture videos, video indexing, ASR, video OCR, video segmentation, lecture video portal.

1 Introduction

Multimedia based education has become more and more popular in the past several years. Therefore, structuring, retrieval and indexing of multimedia lecture data has become an especially useful and challenging task.

Nowadays, the state-of-the-art lecture recording systems normally record two video streams synchronously: the main scene of lecturers which is recorded by using a video camera, and the second that captures the presentation slide frames projected onto the computer screen through a frame grabber card (as e.g., cf. Fig. 1(a)). Since two video streams can be synchronized automatically during the recording, the indexing task can be performed by using slide video only.

Text in lecture video is directly related to the lecture content. In our previous work [7], we have developed a novel video segmenter for key frames extraction from the slide video stream. Having adopted text detection and recognition algorithms on each slide shot in order to recognize the indexable texts. Furthermore,


Internet Security – Content
2
Internet Security – Weaknesses and Targets
Risk Analysis and Computer Crime
Weaknesses and Targets
Detection of Attacks and Intrusions
Legal and Ethical Issues in Internet Security
Detection
Video 1
Video 2
Detection of Attacks and Intrusions
Introduction 00:00:00
Anomalies and Attack Signa
00:11:18
Attack Signatures 00:26:00
Anti-Virus Software 00:34:41
Implementation of IDS
Add new Lecture
Add new Live Lecture

Lectures

Name	is visible	number of segments	OCR analysis	speech recognition	Sorting
Semantic Web	yes	7	finished	analyzing (60 %)	▼
Suchmaschinen	yes	7	finished	finished	▲ ▼
P2P-Netzwerke und Cloud-Computing	yes	7	waiting (0 %)	finished	▲ ▼
Web Services	yes	6	finished	finished	▲ ▼
Webprogrammierung	yes	8	finished	finished	▲ ▼
CSS - Cascading Style Sheets	yes	6	finished	finished	▲ ▼
XML - Extensible Markup Language	yes	7	finished	not started yet	▲ ▼
HTML - Hypertext Markup Language	yes	7	finished	not started yet	▲ ▼
HTTP - Hypertext Transfer Protocol	yes	5	finished	not started yet	▲ ▼
URI - Uniform Resource Identifier	yes	4	finished	not started yet	▲

(a) (b)

Fig. 1. (a) An example lecture video. Video 2 shows the speaker giving his lecture, whereas his presentation is played in video 1. (b) Management page of the lecture video portal for the video analysis

unlike most of the existing OCR-based lecture video indexing approaches [2,1] we utilize the geometrical information and the stroke width value of detected text bounding boxes for extracting lecture structure from the slide video frames. In this paper, we have integrated and adapted our visual analysis engines in an automated framework of the lecture video portal.

Speech is the most natural way of communication and also the main carrier of information in nearly all lectures. Therefore, it is of distinct advantage that the speech information can be used for automated indexing of lecture videos. However, most of existing lecture speech recognition systems have only low recognition accuracy, the WERs (*Word Error Rates*) having been reported from [6,5,3,4] are approximately 40%–85%. The poor recognition results limit the quality of the later indexing process. In this paper, we propose a solution that enables a continued improvement of recognition rate by creating and refining new speech training data. It is important that the involved tasks can be performed efficiently and even fully automated, if possible. For this reason, we have implemented an automated procedure for generating a phonetic dictionary and a method for splitting raw audio lecture data into small pieces.

In order to make the analysis engine applicable for our lecture video portal, we have designed and implemented an architecture so that the analysis process could be easily handled and the efficient indexing functionalities could be provided to the users.

The rest of the paper is organized as follows: section 2 describes our automated analysis architecture. Section 3 and section 4 details the proposed analysis methods. Section 5 concludes the paper with an outlook on future work.

2 An Automated Analysis Framework

The major components of the framework are the analysis management engine, video analysis engine and result storage/visualization engine. The analysis management and result storage/visualization engines are associated with the video

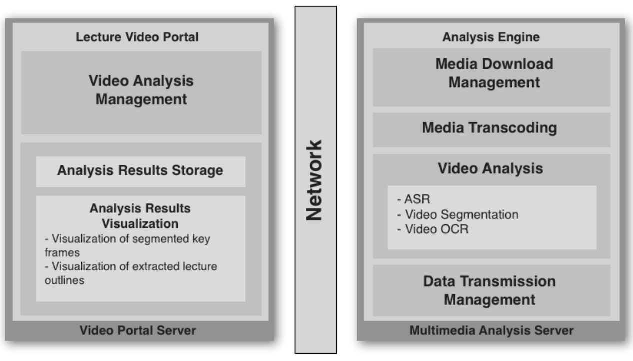

Lecture Video Portal
Video Analysis Management
Analysis Results Storage
Analysis Results Visualization
- Visualization of segmented key frames
- Visualization of extracted lecture outlines
Video Portal Server
Network
Analysis Engine
Media Download Management
Media Transcoding
Video Analysis
- ASR
- Video Segmentation
- Video OCR
Data Transmission Management
Multimedia Analysis Server

Fig. 2. Architecture of our multimedia analysis framework

portal server, while the video analysis engine is associated with the multimedia analysis server. Fig. 2 shows the framework architecture.

2.1 System Workflow

The analysis management engine handles the start of the analysis and reports the processing status. As shown in Fig. 1(b), we can start both, the OCR and ASR analysis for each video in the *staff* page of the lecture video portal. In order to manage each analysis request efficiently, we have defined a class "analysis job object" with the following properties: media id, date time, media URL, analysis type and language. Once the analysis for a lecture video is started, a job object will be created and encoded in the XML format. The XML request will then be sent to the analysis engine on the multimedia analysis server.

The analysis engine manages four major processes: media download, video transcoding, video analysis for each request, and analysis result transmission to the video portal server. Since the analysis engine is designed fully configurable, it is therefore suitable to be extended and work with a multi-process workflow or a distributed computing system. Once a video is downloaded, a transcoder will convert it to a predefined format with an uniform resolution which is most appropriate for the video segmentation and OCR analysis. For ASR, we extract the audio track with a format of 16KHz and 16 Bit which meets the requirements of the speech recognition software.

After the analysis, the results will be automatically sent to the destination place on the portal server. Subsequently, we send HTTP-POST requests for saving the corresponding results to the portal database. Thanks to the plugin architecture of our video portal, once the analysis result data is saved, the corresponding GUI elements will be created automatically by refreshing the web page.

For our purpose, we apply this framework to analyze the lecture videos from our institute. However, using it for the videos from other sources (as e.g., Youtube[1],

[1] `http://www.youtube.com`

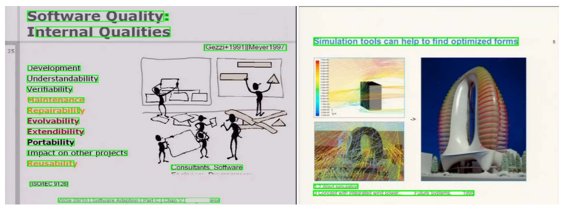
Berkeley[2] etc.), we would only need to adapt the configuration file of the analysis engine.

3 Visual Analysis for Lecture Videos

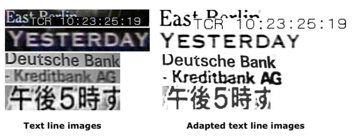
Our video visual analysis system consists of two parts: unique slide shots detection and video OCR.

In [7], we have proposed a novel CC-based (*Connected Component*) video segmenter to capture the real slide transitions in the lecture video. The video OCR system is applied on each slide shot. It consists of the following steps:

- Text detection: this process determines whether a single frame of a video file contains text lines, for which a tight bounding box is returned (cf. Fig. 3(a)). We have developed a two-stages approach that consists of a fast edge based detector for coarse detection and a *Stroke Width Transform* (SWT) based verification procedure to remove the false alarms.
- Text segmentation: in this step, the text pixels are separated from their background. This work is normally done by using a binarization algorithm. Fig. 3(b) shows the text binarization results of our dynamic contrast and brightness adaption method. The adapted text line images are converted to an acceptable format for a standard OCR engine.
- Text recognition: we applied a multi-hypotheses framework to recognize texts from extracted text line images. The subsequent spell-checking process will further filter out incorrect words from the recognition results.



(a)



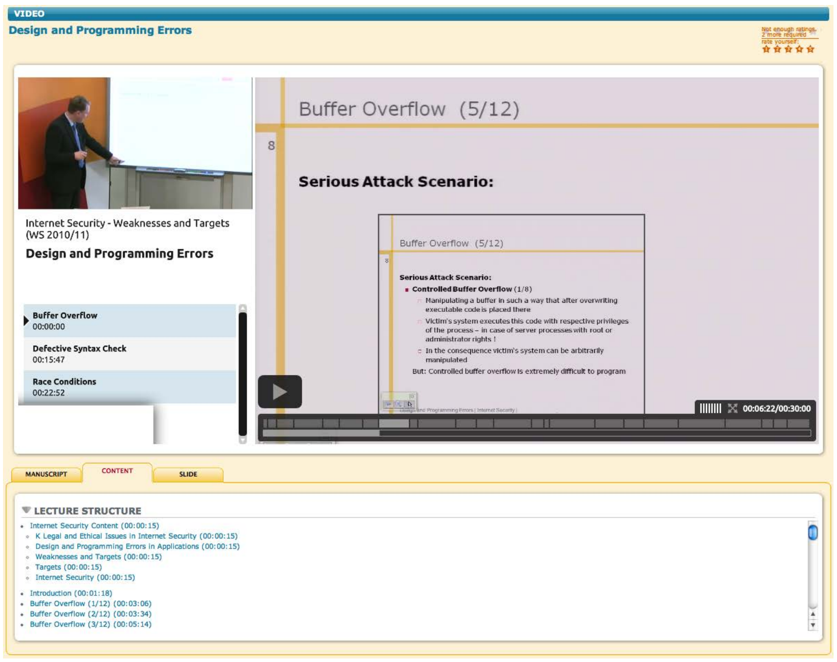
(b)

Fig. 3. (a) Text detection results. All detected text regions are identified by bounding boxes. (b)Text binarization results of our dynamic contrast-brightness adaption method.

After the text recognition process, the detected text line objects and their texts are further utilized in the lecture outline extraction method. Unlike other approaches, we observe the geometrical characteristics of detected text line objects, and apply size, position and average stroke width information of them to classify the recognized OCR text resources. In this way, the lecture outline can be extracted based on the classified OCR texts. Which can provide an overview about the lecture content, and help the user with a navigation within the lecture video as well (cf. Fig. 4).

More technical details as well as the evaluation results of proposed visual analysis methods can be found in [7].

[2] `http://webcast.berkeley.edu`


VIDEO
Design and Programming Errors
Not enough ratings. 2 more required rate yourself:
MANUSCRIPT
CONTENT
SLIDE
00:06:22/00:30:00

Internet Security - Weaknesses and Targets (WS 2010/11)

Design and Programming Errors

Buffer Overflow 00:00:00

Defective Syntax Check 00:15:47

Race Conditions 00:22:52

Buffer Overflow (5/12)

8

Serious Attack Scenario:

Serious Attack Scenario:
- **Controlled Buffer Overflow** (1/8)
  - Manipulating a buffer in such a way that after overwriting executable code is placed there
  - Victim's system executes this code with respective privileges of the process – in case of server processes with root or administrator rights !
  - In the consequence victim's system can be arbitrarily manipulated

But: Controlled buffer overflow is extremely difficult to program

Design and Programming Errors | Internet Security |

LECTURE STRUCTURE

- Internet Security Content (00:00:15)
  - K Legal and Ethical Issues in Internet Security (00:00:15)
  - Design and Programming Errors in Applications (00:00:15)
  - Weaknesses and Targets (00:00:15)
  - Targets (00:00:15)
  - Internet Security (00:00:15)
- Introduction (00:01:18)
- Buffer Overflow (1/12) (00:03:06)
- Buffer Overflow (2/12) (00:03:34)
- Buffer Overflow (3/12) (00:05:14)

Fig. 4. Visualization of the segmenter slide shots and extracted outline (lecture structure) of the lecture video underneath the video player

4 Speech Recognition for Lecture Videos

In addition to OCR, ASR is an effective method for gathering semantic information about lecture videos. Combining both OCR and ASR offers the chance to improve the quality of automatically generated metadata dramatically. For that reason, we decided to build a speech model with the help of the CMU Sphinx Toolkit [3] and the German Speech Corpus by Voxforge[4] as a baseline. We collected hours of speech data from our lecturers and the corresponding transcripts in order to improve speech recognition rates for our special use case. Using a random assortment of speech segments from our lecturers as a test corpus, Table. 1 shows how the WER decreases when adding 7.2 hours of speech data from our lecturers to the training set.

However, manually collecting appropriate speech segments and transcripts is rather time consuming and costly. There is a number of steps to be performed to acquire high quality input data for *SphinxTrain*, as shown in Fig. 5(a). This entire process is described more detailed in the following subsections.

[3] http://cmusphinx.sourceforge.net/
[4] http://www.voxforge.org/

4.1 Creating the Speech Training Data

A recorded lecture audio stream yields approximately 90 minutes of speech data, which is far too long to be processed by *SphinxTrain* or the decoder at once. Several shorter speech segments are not only easier to manage by ASR tools, they also enable us to perform tasks like speech model training and decoding on highly parallelized hardware.

Table 1. Current progress of our speech model training. The WER results have been measured through a random set of 708 short speech segments from 7 lecturers.

Training Corpus	WER
Voxforge (23.3h)	82.5%
Voxforge (23.3h) + Transcribed Lectures (7.2h)	62.6%

Therefore, we looked for efficient methods to split the long audio stream into manageable pieces. The commonly used methods are described as follows:

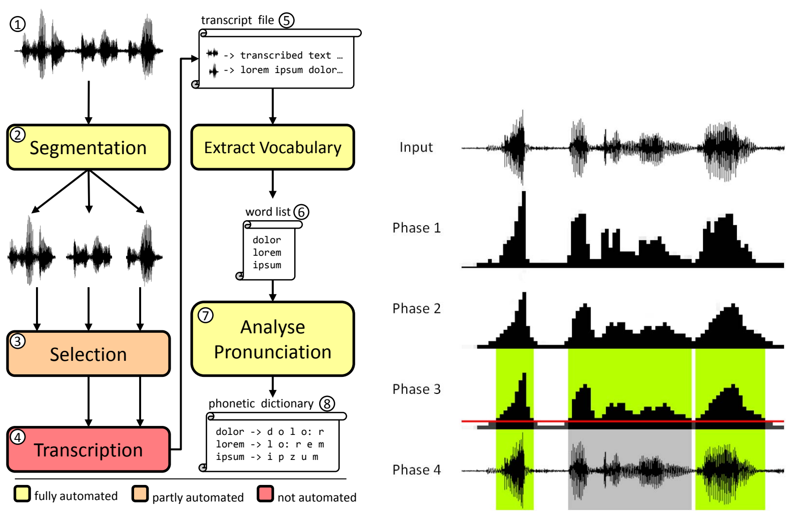
– Combine segmentation and selection processes by using an audio file editor like Audacity[5]. Both steps are performed manually by selecting parts with a length of 2–5 seconds from the audio stream and save them as separate audio files. This approach results in high quality segments without any cropped words, but the required time and effort (50+ minutes real time/1 minute transcription) is not acceptable.
– Split the input audio stream uniformly every n seconds, where n is a constant between 5–30. This method is fast, but it yields a significant disadvantage: a regular subdivision of speech often results in spoken words being partly cropped, which makes a lot of segments useless in the ASR context. Selection of bad segments has to be performed manually in order to achieve adequate database quality. For 1 minute of transcribed speech data, we need approximately 35 minutes in real time.

Compared to these rather inefficient methods, our current approach is to fully automate segmentation and partly automate selection without suffering from quality drawbacks like dropped word endings. Fig. 5(b) describes the fundamental steps of the segmentation algorithm. The audio curve is first down-sampled, then blurred and split according to a loudness threshold. Too short or too long sentences are sorted out automatically. The effort to transcribe 1 minute of speech decreases to 15–20 minutes in real time.

In order to obtain the best recognition rates in the long run, we have to ensure that all speech segments used for speech model training meet certain quality requirements. Experience has shown that approximately 50% of the generated audio segments have to be sorted out due to one of the following reasons:

– The segment contains acoustical noise created by objects and humans in the environment around the speaker, as e.g., doors closing, chairs moving, students talking.

[5] `http://audacity.sourceforge.net`



(a) (b)

Fig. 5. (a) Workflow for extending our speech corpus. First, the recorded audio file (1) is segmented into smaller pieces (2) and improper segments are sorted out (3). For each remaining segment, the spoken text is transcribed manually (4), and in doing so, a line of text is added to the transcript file (5). As an intermediate step, a list of all used words in the transcript file is created (6). In order to obtain the phonetic dictionary (7), the pronunciation of each word has to be represented as phonetics (8).
(b) General idea of the algorithm used in our automatic audio stream segmenter. Phase 1: compute the absolute values of the input samples to get a loudness curve, which is then down-sampled (e.g. by factor 100). Phase 2: a blur filter (e.g. radius=3) is applied to eliminate outliers. Phase 3: a threshold (red horizontal line) determines which areas are considered *quiet*. Every connected area which is not quiet is a potential utterance (highlighted with green). Phase 4: all those green areas that are longer than a specified maximum length (e.g. 5 seconds) are dropped. Finally, the computed borders for each remaining block are remapped to the original WAVE samples and form a segment.

– The lecturer misspeaks some words, so that they are completely invalid from an objective point of view.
– The speaker's language is clear, but the segmentation algorithm cut off parts of a spoken word so that it turns invalid.

The classification of audio segments in good/bad quality is not yet solvable automatically, as the term 'good quality' is very subjective and strongly bound to the human perception.

Subsequently, the manual transcription process is performed. By definition, this step has to be done by a human, which is the reason why it is not perfectly optimizable. However, we speed up this process with the help of our transcription software. It provides a simple GUI where the user can play the audio segments and type the transcription with very few key presses and without doing anything unnecessary. File playing, naming, and saving is completely managed by the software. Furthermore, it provides a Sphinx compatible output format.

4.2 Creating the Phonetic Dictionary

The phonetic dictionary is an essential part of every ASR software. For every word occurring in the transcripts, it defines one or more phonetic representations. To build them, we use a customized phonetic alphabet which contains 45 phonemes used in German pronunciation.

As our speech corpus is still growing continuously, the extension and maintenance of the dictionary is a common task. Experience has shown that for each transcribed lecture, there are about 100 new words which have to be added to the vocabulary. The Voxforge community proposes to use eSpeak[6] to create the phonetic representation for each word. This method works fine, but yields some striking disadvantages: first, the output generated by eSpeak is only partly usable. For almost every word, corrections have to be made manually. eSpeak uses its own format for phonetic spelling. The conversion into a representation accepted by Sphinx is complex and error-prone. In addition, dictionary management is inflexible. Discarding the dictionary and regenerating it with new parameters or changed transcripts is expensive, because all manual corrections will be discarded. For these reasons we have built an own phonetics generator. In order to maximize the output quality, our approach operates on three different layers of abstraction while successively generating the phonetic representation string P.

On *word level*, our so-called *exemption dictionary* is checked whether it contains the input word as a whole. If so, the phonetic representation P is completely read from this dictionary and no further steps have to be performed. Otherwise, we have to scale down to *syllable level* by applying an extern hyphenation algorithm[7]. The exemption dictionary particularly contains foreign words or names whose pronunciation cannot be generalized by German rules, as e.g., `byte`, `phänomen`, and `ipv6` etc.

On *syllable level*, we examine the result of the hyphenation algorithm, as e.g., `com-pu-ter` for the input word `computer`. For a lot of single syllables (and also pairs of syllables), the *syllable mapping* describes the corresponding phonetic representation, e.g. `au f` for the German prefix `auf` and `n i: d er` for the disyllabic German prefix `nieder`. If there is such a representation for our input syllable, it is added to the phonetic result P immediately and we succeed with the next part. Otherwise, we have to split the syllable even further into its characters and proceed on *character level*.

On *character level*, a lot of German pronunciation rules are utilized to determine how the current single character is pronounced. Typical metrics are:

- Character type (consonant/vowel).
- Neighboring characters.

[6] An open source speech synthesizer (cf. `http://espeak.sourceforge.net`)
[7] `http://swolter.sdf1.org/software/libhyphenate.html`

– Relative position inside the containing syllable.
– Absolute position inside the whole word.

First and foremost, a heuristic checks if the current and the next 1–2 characters can be pronounced natively. If not, or the word is only one character long, the character(s) is/are pronounced as if they were spelled letter by letter such as, e.g., the abbreviations `abc` (for alphabet) and `zdf` (a German TV channel).

Next, we determine the character type (consonant/vowel) in order to apply the correct pronunciation rules. The conditions of each of these rules are checked, until one is *true* or all conditions are checked and turned out to be *false*. If the latter is the case, the standard pronunciation is applied, which assumes *closed* vowels. This is an example for the rules we use in our heuristic:

– A vowel is always *open* if followed by a double consonant, e.g. `affe`, `wetter`, `bitte`.
– A vowel is always *closed* if followed by the German consonant ß, e.g. `straße`, `größe`, `fußball`.
– If the consonant `b` is the last character of a syllable and the next syllable does not begin with another `b`, then it is pronounced like the consonant `p`. Examples: `hubschrauber`, `lebkuchen`, `grab`.

An evaluation with 20000 words from transcripts shows that 98.1% of all input words where processed correctly, without the need of any manual amendments. The 1.9% of words where application of the pronunciation rules failed mostly have an English pronunciation. They are corrected manually and added to a special 'exemption dictionary'. Besides the striking advantage of saving a lot of time for dictionary maintenance, Table. 2 shows that our automatic dictionary generation algorithm does not result in worse WER. The recognized ASR texts are further used by search and key-word extraction functions.

Table 2. Comparison of the Voxforge dictionary and our automatically generated dictionary with an optimized phone-set for the German language. The used speech corpus is a smaller version of the German Voxforge Corpus which contains 4.5 hours of audio from 13 different speakers. It is directly available from the Voxforge website including a ready-to-use dictionary. Replacing this with our automatically generated dictionary results in a slightly better recognition rate.

Phonetic Dictionary	WER
Voxforge German dictionary	22.2%
Our dictionary with optimized phoneset	21.4%

5 Conclusion and Future Work

In this paper, we have presented an automated framework for the analysis and indexing of lecture videos. The proposed video analysis methods consist of video segmentation, video OCR, and automated speech recognition. In order to integrate the analysis engine into our lecture video portal, we have designed and implemented an architecture which enables an efficient processing of analysis management, multimedia analysis, data transmission, and result visualization.

As the upcoming improvement, we plan to implement an automated method for the extraction of indexable key words from ASR transcripts. In [4], the course-related text books have been used as the context reference for refining ASR results. In our case, the high accurate OCR texts from each video segment can be considered as a hint for the correct speech context. In this way, a more accurate refinement and key word extraction from ASR transcript could take place. It may solve the issue of building search indices for imperfect ASR transcripts. Furthermore, our research will also address the following issues: search-index ranking for hybrid information resources (as e.g., OCR, ASR, user-tagging etc.); usability and utility evaluation for our indexing and video navigation functionalities in the lecture video portal.

References

1. Adcock, J., Cooper, M., Denoue, L., Pirsiavash, H.: Talkminer: A lecture webcast search engine. In: Proc. of the ACM International Conference on Multimedia, MM 2010, Firenze, Italy, pp. 241–250. ACM (2010)
2. Wang, T.-C.P.F., Ngo, C.-W.: Structuring low-quality videotaped lectures for cross-reference browsing by video text analysis. Journal of Pattern Recognition 41(10), 3257–3269 (2008)
3. Glass, J., Hazen, T.J., Hetherington, L., Wang, C.: Analysis and processing of lecture audio data: Preliminary investigations. In: Proc. of the HLT-NAACL Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval (2004)
4. Haubold, A., Kender, J.R.: Augmented segmentation and visualization for presentation videos. In: Proc. of the 13th Annual ACM International Conference on Multimedia, pp. 51–60. ACM (2005)
5. Lee, D., Lee, G.G.: A korean spoken document retrieval system for lecture search. In: Proc. of the SSCS Speech Search Workshop at SIGIR (2008)
6. Leeuwis, E., Federico, M., Cettolo, M.: Language modeling and transcription of the ted corpus lectures. In: Proc. of the IEEE ICASSP, pp. 232–235. IEEE (2003)
7. Yang, H., Siebert, M., Lühne, P., Sack, H., Meinel, C.: Lecture video indexing and analysis using video ocr technology. In: Proc. of 7th International Conference on Signal Image Technology and Internet Based Systems (SITIS 2011), Dijon, France (2011)