Automated Segmentation and Tagging of Lecture Videos

Submitted in fulfillment of the requirements
for the degree of

Master of Technology

by

Ravi Raipuria
Roll No: 113050077

Supervisor:

Prof. D. B. Phatak

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai

June, 2013

Abstract

Recorded video lectures are used more frequently by students for E-learning purpose. Generally, duration of a lecture video is about 90 to 120 min. So, it is cumbersome for student to search a particular area of interest in an entire video lecture due to lack of explicit annotation. However, Manual annotation is very time consuming process. It is desirable to have a system that take keywords as user-input and provide related section of video lecturer. A transcription of the lecture video can enable faster navigation and browsing. We have proposed an approach for automatic topic segmentation and tagging of video lectures based on speech transcription and slides.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Scope of the Project 2

1.3 Organization of Thesis 2

2 Existing Lecture Video Repository **3**

3 Indexing Technique for Lecture Videos **5**

3.1 Manual 5

3.2 Tagging 5

3.3 Slides 7

3.4 Speech Transcription 9

3.5 Chain Indexing of Speech Transcription 10

4 Problem Definition and Solution Approach **12**

4.1 Problem Definition 12

4.2 Solution Approach 12

4.2.1 Speech Transcript Extraction 13

4.2.2 Slide Extraction 19

5 Implementation Detail **20**

5.1 Speech Transcript Extraction 20

5.1.1 CMU Sphinx [7] 21

5.1.2 Sphinx4 Configuration 22

5.1.3 Adapting Default Acoustic Model 23

5.1.4 Building Language Model 25

5.1.5 Create Phonetic Dictionary 27

5.1.6 Generating Speech Transcript from Lecture Video 27

5.2 Slide Extraction 29

6 Experiments and Results **31**

6.1 Speech Recognition Evaluation 31

6.1.1 Word Error Rate 31

6.1.2 Word Accuracy 32

7 Conclusion and Future Work **33**

A Sphinx4 Configuration **34**

B Source Code **35**

List of Figures

3.1 MPEG-7 Basic Elements 6
3.2 Elements of VideoSegment Tag 7
3.3 Elements of MediaReview 8
3.4 Chain Index 11

4.1 Proposed System Architecture 13
4.2 Speech Transcript Extraction 14
4.3 Adaption of an acoustic model 15
4.4 Building a language model 16
4.5 Generating text corpus from WikiPedia 17
4.6 Crating Text Corpus for Topic Modeling 18
4.7 Extract Slide and Title from Lecture Video 19

5.1 Sphinx-4 Architecture and Main Components[7] 21
5.2 A sample slide 29
5.3 Title of a slide 29

List of Tables

2.1 Comparison of Lecture Video Repositories 4

4.1 Words with their Arpabet Representation 15

Chapter 1

Introduction

Now a days, Many universities provide free lecture videos for distance learning education. It is very difficult to browse within those videos for a particular topic of interest. Generally, video lecture duration ends up about 90 to 120 min. To make best use of this, a efficient content-retrieval mechanism is required for searching the keyword in lecture videos. However, The problem is not to find lecture in video archive rather than finding the proper position of desired keyword in video stream. The rapid growth of e-learning data required more efficient content-based retrieval mechanism for video lectures. The main challenge is to provide automatic indexing of lecture videos and find semantically appropriate part of the lecture video.

We have tried to develop a system for lecture browsing which facilitate the searching of particular topic in lecture video. It uses the speech transcript as main resource of segmenting lecture video into lexical coherent topics. Slides are also extracted from the lecture video to provide efficient browsing of topics.

1.1 Motivation

There is speedy growth in the amount of generating e-learning data for distance education in recent years. Content-based retrieval comes into picture, when a user asks to retrieve specific part of the video lecture. To retrieve a desired part of video is still a very difficult and time-consuming process. So, A browsing system is required based on content-based retrieval to provide desired part of the lecture video.

1.2 Scope of the Project

This project is restricted to lecture videos related to computer science domain. We have also used some unix tool for pre-processing for corpus data. Reasons for restricted scope are described below:

- We have used a open source automatic speech recognition tool to extract transcript from a lecture video. But, the tool requires a language model and acoustic model as a base to capture the sequence of words. The performance of this tool depends upon the language and acoustic models. Language model is domain dependent that is generated by collective large amount of corpus data related to the particular domain. So, Language model that we have used is particular for computer science domain.

- It also depend on the speaker. We have to adapt the acoustic model for each speaker to properly recognize the sentences.

1.3 Organization of Thesis

In chapter 2, we presented some existing lecture video repository and comparison between them based on various factor. In chapter 3, some existing indexing technique for lecture has been discussed along with their advantage and disadvantage. In chapter 4, we have described the problem definition. Solution approach and system architecture also presented in this chapter. This chapter also describe the various open source tool that are used in this project. In chapter 5, we have described implementation detail of each module speech transcript extraction , slide extraction and user interface design. In chapter 6, Results and analysis is presented. In chapter 7, conclusion and future work is presented.

Chapter 2

Existing Lecture Video Repository

In this chapter, we will talk about features of existing lecture video repository. These repository compared on the basis of following parameter:

1. **Video Retrieval:** It is a process of obtaining related lecture videos from the repository. There are many repository on the web, which provides free lecture videos. They have large number of videos in their repository. If somebody want to access a particular lecture video (e.g. Algorithm), they should have some retrieval mechanism to search in the video repository. It can be done in two ways:
   - **Meta-data Based:** Meta-data is the textual data associated with the video such as title, description, comments etc. This method use these meta-data to search the video in the repository.
   - **Content Based:** Content of a lecture video can be retrieved from speech transcript, slide etc. These content is used as index to search a video in the repository.
2. **Video Navigation:** Navigation within a video is important aspect, if you want to retrieve a particular topic from a large duration videos. It can also be done in two ways:
   - **Key-frames:** Key-frames are the screen-shot of slides from lecture video. A index is generated using these key-frames to navigate a video for a particular slides. Presentation slides can be additionally used to improve the performance of the system.
   - **Speech Transcription:** Most of the content in lecture video is a speech. Speech transcript can be used as main resource of indexing. Speech is converted into text and then stemming operation is performed to reduce the redundancy of keywords. Now, an index is generated based on these keywords for navigate a particular topic in lecture video.

Repository	**Retrieval**		**Navigation**	
	Meta-data	**Content**	**Key-frames**	**Speech Transcription**
CDEEP[2]	×	×	×	×
NPTEL[6]	✓	×	✓(manual)	×
MIT Open CourseWare [5]	✓	×	×	×
Coursera[3]	✓	×	×	×
Academic Earth[1]	✓	×	×	×
Free Video Lectures[4]	✓	×	×	×
VideoLectures.net[8]	✓	×	✓(manual)	×
MIT Lecture Browser[9]	✓	✓	×	✓

Table 2.1: Comparison of Lecture Video Repositories

Table 2.1 shows the comparison between existing lecture video repository based on the above parameter.

Chapter 3

Indexing Technique for Lecture Videos

Indexing is the process of creating access point to facilitate retrieval of information. Lecture video indexing comprises two steps: first, divide the video into coherent segments called topics and second, segments have to specify a topic with keyword or semantic description. There are four main indexing technique for video lecture indexing.

3.1 Manual

Here, topic segmentation and annotations are done manually. This process is done in very time-consuming process. A person who is familiar with topic can do the segmentation and then associate segments with its semantic description. Accuracy of this type of tagging is obviously high, and depends upon the knowledge of a person about the topic. It perform better for small lecture videos but not practically possible for large archive.

3.2 Tagging

Tagging is a process of annotating a document with an unstructured list of keywords. Collaboration tagging can be used for generation and enhancement of video meta-data to support content based retrieval[16]. It is very unlikely to tag or bookmark a document, if it's no use of reader. But, tagging has impact to all who will read this document. This technique needs lot of time to annotating a document and it depend on many learners who have ability to annotate.

Author of [16] proposed an approach for automated tagging of synchronized multimedia contents. Firstly, Lecture video is synchronized with corresponding desktop presentation, which is use for generating MPEG-7 metadata. MPEG-7 is an XML based markup language which is used for annotation and description of multimedia documents. Textual information of presentation slides is used to annotate each part of the video. After that you can search a video by querying a keyword used in annotating process. In this process, MPEG-7 is used to maintain the collaborative tagging information.

```
<Mpeg7 xmlns=“.....”>
    <Description xsi:type=“ContentEntityType”>
        ...
        <MultimediaContent xsi:type=“VideoType“>
            <Video>
                <MediaInformation>
                ...
                <TemporalDecomposition>
                    <VideoSegment> . . . </VideoSegment>
                    <VideoSegment> . . . </VideoSegment>
                    ...
                </TemporalDecomposition>
            </Video>
        </MultimediaContent>
    </Description>
</Mpeg7>
```

Figure 3.1: MPEG-7 Basic Elements

Figure 3.1 represents the basic element of MPEG-7. <TemporalDecomposition> element of MPEG-7 description schema can be used to annotate the video by storing various information. <TemporalDecomposition> facilitate to annotate a specific part of video rather than whole video stream. A video segment can be identified by <VideoSegment> tag. <VideoSegment> associate with <MediaReview> element to provide collaborative tagging information.

To integrate collaborative tagging information into MPEG-7 description schema, we need to add an extra element <MediaReview> with each video segment. We need to store the tagging information for each user. So, the collaborative tagging information can be stored in a tupel

$$(\{\text{tagset}\}, \text{username}, \text{date}, [\text{rating}])$$

where a set of tags is submitted by the user with rating of a specific video segment. Each tag has date and time associated with it. tagset denote all tags submitted by distinctive user for a specific video segment. It is encoded in <FreeTextReview> element with coma separated value. Media Review element are shown in Figure 3.3.

MediaReview element provides all necessary information needed for collaborative tagging. Date stores the last modified date of tags for each user and Reviewer element stores all user

```
<VideoSegment>
    <CreationInformation>....</CreationInformation>
    ...
    <TextAnnotation>
        <KeywordAnnotation>
            <Keyword>Quick Sort</Keyword>
            <Keyword>MergeSort</Keyword>
        </KeywordAnnotation>
        <FreeTextAnnotation>
            This video explains all sorting techniques
        </FreeTextAnnotation>
    </TextAnnotation>
    <MediaTime>
        <MediaTimePoint>T00:05:05:0F25</MediaTimePoint>
        <MediaDuration>PT00H00M31S0N25F</MediaDuration>
    </MediaTime>
</VideoSegment>
```

Figure 3.2: Elements of VideoSegment Tag

related information with user type. MediaReview element is embedded inside the <Creation-Information> and <classification> elements of video segments.

A system is implemented [16] for collaborative tagging in video lectures. A user interface is provided where a user can add a tag for a particular time duration of lecture videos. If you are retrieving a part of lecture video, all tags associated with this part are displayed. It facilitate similarity feature, so you can easily access similar video segment. This technique is also extended to text-document, where you can perform all the operation provided by this system.

The accuracy of this approach is high, but manual annotating of lecture videos requires lot of time. It also depend upon users, who have the knowledge of a particular domain.

3.3 Slides

Slides represent most of the information in video segments. Slides can also be use for indexing of related video lectures. For synchronizing between slides and videos, a log file is maintained during the live lecture recording which store timestamps(start and finish) of each slide[17]. The task of synchronizing desktop presentation slides with lecture video is done manually. The textual content of presentation slide is used to generate MPEG-7 meta-data information.

```
<CreationInformation>
    <Classification>
        <MediaReview>
            <Rating>
                <RatingValue>9.1</RatingValue>
                <RatingScheme style=”HigherIsBetter“/>
            </Rating>
            <FreeTextReview>
                tag1,tag2,tag3
            </FreeTextReview>
            <ReviewReference>
                <CreationInformation>
                    <Date>...</Date>
                </CreationInformation>
            </ReviewReference>
            <Reviewer xsi:type=”PersonType“>
                <Name>Ravi Raipuria</Name>
            </Reviewer>
        </MediaReview>
        <MediaReview>...</MediaReview>
    </Classification>
</CreationInformation>
```

Figure 3.3: Elements of MediaReview

To integrate MPEG-7 with lecture videos meta-data, slides converted into MHTML and textual information is extracted from the slides. After eliminating all stop word that doesn't have semantic meaning, all words are stemmed. Porter algorithm is used for word stemming. The timestamps stored in a log file are used to partition the video into smaller segments, So you have one video segment corresponding to particular slide. Now, you can extract outline and headings from HTML file and integrate it with MPEG-7 <FreeTextAnnotation> and <KeywordAnnotation> element of a video segment.

MPEG-7 is XML based markup language, so searching in MPEG-7 is just required to parse the XML file, which follows document object model. Their are mainly two problems, one is parsing the XML file is slow downs the system and second, ranking of user query result. To overcome the first issue, an index technique is used for MPEG-7 document. During search process index must contain all the information provided by MPEG-7 document. For second one, the are taking word frequency for as a ranking parameter. If a video segment has a word that has higher word frequencies, it will be put at higher level.

This technique also require lot of manual work, so not reliable. And, nobody cares about maintaining the log file at the time of presentation.

3.4 Speech Transcription

Speech can also be used as resource of index for lecture videos. State-of-art SRE is used to transforming a speech into text[11]. Now the task is to divide the text into smaller parts called topics. A transition(for e.g., pause) in a vocabulary could mark a segment boundary, based on we can divide the video into multiple segments. For each video segment an index is generated, for searching the specific video segment. The development of text segment is a very challenging task of natural language process.

After generating the text stream, some standard text algorithm are implemented to check whether it is possible to recognize the topic segment based on speech transcription[14].

- **Linear:** The linear distribution of slide transition during the presentation time is implemented. It assumes that number of topic is given for a particular lecture video.
- **Pause:** The duration of silence(pause in speech) is used as feature for detecting the segment boundary. The longest pause in a lecture video used as a segment boundaries.
- **C99, LCseg and MinCut:** The number of segment boundary is given to this algorithm and sentence length is defined as 10 words. A slide transition is assumed to be segment boundary, and detected by this algorithm.
- **SlidingWIndow:** It is based on TextTiling algorithm[10]. In this, a sliding window (120 words) moves across the text stream over a certain interval(20 words) and neighboring window is compared by cosine measure. After post-processing, lowest similarity points will become the segment boundaries.
- **VoiceSeg:** In this algorithm, first text stream is passed to stemmer, removed all stop words and similar words are grouped into clusters. Now vector element is filled by the value of cluster data and build the vectors. Similarity between two Adjacent vector are calculated by cosine measure and segment boundaries are detected based on these similarity.

The result of these algorithm is shows that determining the segment boundary is nearly impossible(**First Test**). If the segment count is not given to algorithm, the result could be much worse. It needs to count the number of segment boundary to work it properly.

If the transcript is error-free, it will improve the performance of this algorithm(**Second Test**). Further the segmentation result could be better if we have the real segment boundary, rather than assuming pause as segment boundary. The overall result shows that detecting the segment boundary is very complex task.

3.5 Chain Indexing of Speech Transcription

Chaining is a process of detecting cohesive areas in lecture video. A linguistic research shows that word repetition in a speech transcription of lecture video is hint for creating thematic cohesion[15]. Text stream can contain all parts of speech. First, all stop words are removed and then it is passed to stemmer. A stemmed word in a text stream is called term[12]. All distinctive term are stored in a list L. If we have n terms the list will have n distinct terms[13].

$$L = \{T_1, T_2, T_3, ..., T_n\}$$

A chain is constructed for each term in list L. Chain contains each appearance of the term from beginning to end in text stream with timestamps. Now, the chain is divided into subparts if there is large distance **d** between two successive appearance. Store each segment in database with term count in each segment. The whole process will work as follows:

1. Take a term T_n from list L.
2. build segments: divide the chain in subparts, count the occurrence in text stream, store start and end time for each segment.
3. store the data in database as chain index for the lecture video.
4. take the next term T_{n+1} from list L and repeat go to step 2.

There may be chances of overlap of chains because they are generated for each distinct term separately. Segments have attached weight(no of occurrence) called TN, to reduce ambiguity. If a segment have higher score, it will be more relevant.

Figure 3.4 represents an example of the chain index. If you query for sorting in chain index database. It has two appearance in the table. First appearance has 10 occurrences in speech transcription and second appearance has 4 occurrences. If a term has higher occurrences, it will be given higher priority. In this case, First appearance will be given higher priority.

The Disatance d

The distance d is varied from 0.5 to 10 minutes, for finding the proper position of breaking the chain. and measure the accuracy for 10 keywords.These keyword are randomly selected

Term(T)	Start Time	End Time	TN(No Of Occurrences)
Sorting	1600s	2400s	10
Sorting	2600s	3300s	4
QuickSort	2100s	2800s	8
MergeSort	1500s	2200s	5
Searching	100s	1500s	20

Figure 3.4: Chain Index

from the domain lexicon. Result of this analysis is shown in Figure **??**. it may be vary because analysis is done for only 10 keywords. It is still debatable whether this interval(2 to 8 minutes) will always provide accurate result.

Chapter 4

Problem Definition and Solution Approach

4.1 Problem Definition

Now a days, E-Learning is the main resource of distance education. Many repository provides free lecture videos on the Internet for distance education purposes. Duration of these videos around 90-120 minutes, so it is very difficult to browse a specific topic within the lecture video. To efficient browsing within lecture videos, an tagging mechanism is required for content based retrieval. Our aim is to develop a content-based retrieval tool for providing efficient browsing within lecture videos.

4.2 Solution Approach

Our proposed solution provide search and navigation feature based on the content of lecture video. A lecture video contains audio part and visual part i.e. slide frames. Figure 4.1 represents the proposed system architecture of lecture browsing system. The whole process is divided into two parts.

- Speech Transcript Extraction
- Slide Extraction

Once we have speech transcript, it is segmented into lexical coherent topics and stored in database. At the other side, titles are also extracted from each slide and stored in the database. User can query for any topic present in the database using a interface. System will return the desired result to the user. The task of extracting speech transcript and slide is done by open source tools.

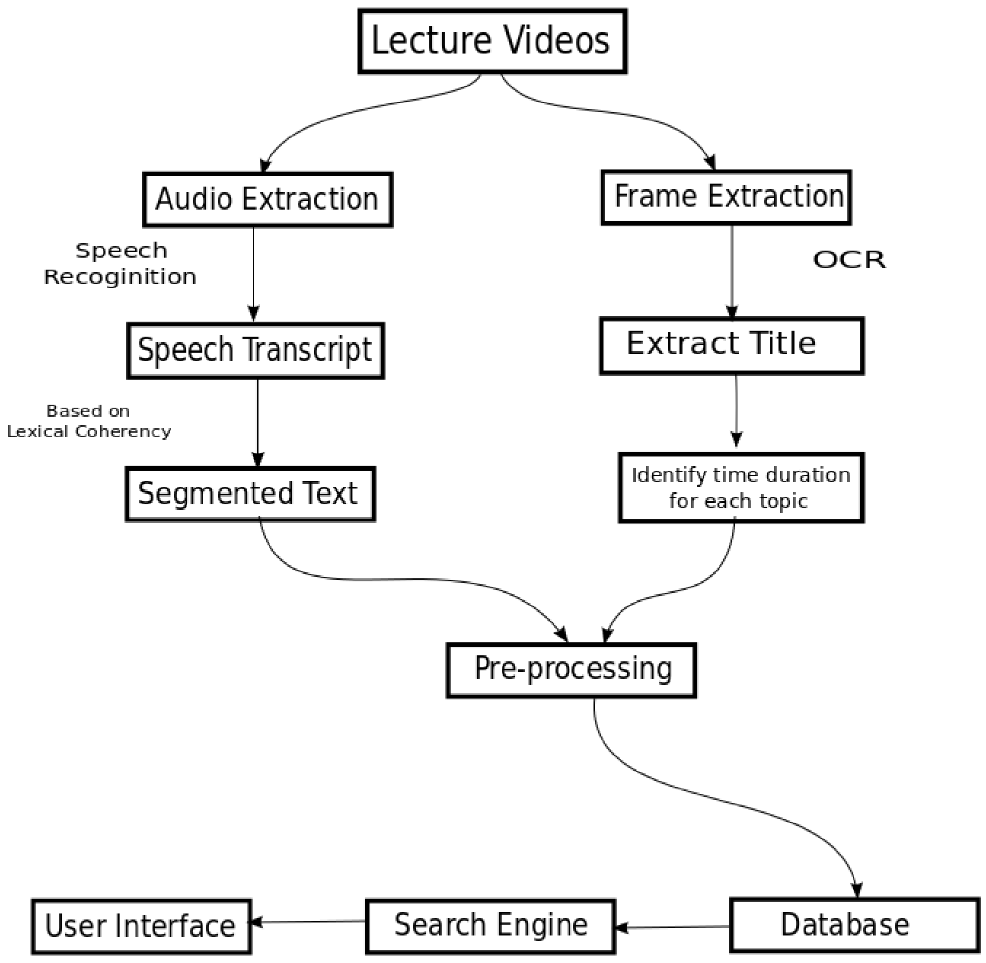


Figure 4.1: Proposed System Architecture

4.2.1 Speech Transcript Extraction

Speech transcription is done by automatic speech recognition system. Automatic speech recognition are imperfect and may introduce some errors in transcript. Accuracy of an automatic speech recognition engine depends on the following factor:

- Audio quality of recording

- Acoustic model, whether it is trained for a particular speaker or not
- Match between the vocabulary and pronunciation in the lecture with the language model and pronunciation dictionary

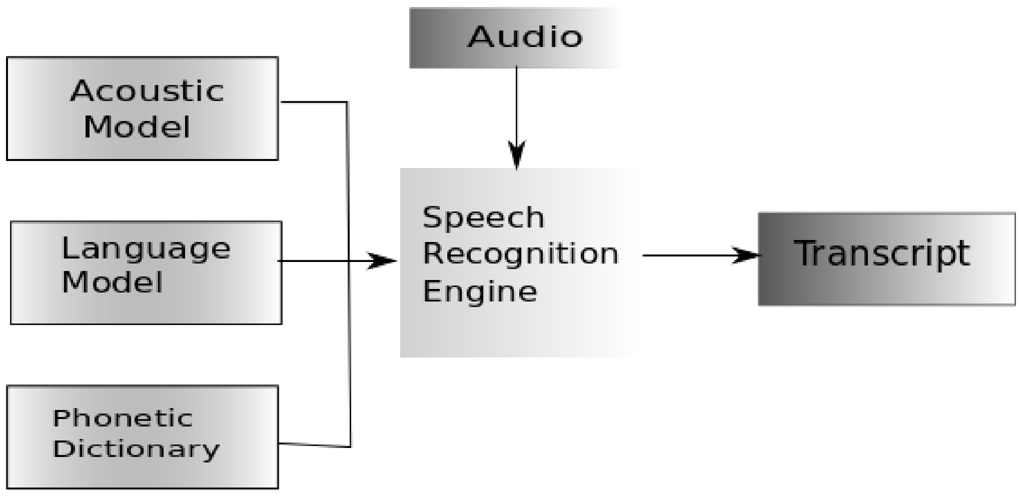


Figure 4.2: Speech Transcript Extraction

Figure 4.2 represents the process of extracting speech from audio file. A speech recognition system involves use of four main resources for a given lecture video:

1. A set of phonemes
2. A phonetic dictionary
3. An acoustic model
4. A language model

A phoneme is a basic unit of sound making system. In transcription, a phoneme is placed between slash marks: /c/. The term phoneme is usually restricted to vowels and consonants. For speech recognition system, ASCII representation of phonemes are more practical. A most widely used set is Arpabet, created by ARPA (Advanced Research Projects Agency) to represent sound in general American English. The Arpabet comprises 39 phonemes each represented by one or two letters with optional stress markers represented by 0, 1 or 2.

Phonetic dictionary represents the relationship between a word and it's phoneme. A word comprises one or more number of phonemes. Phonetic dictionary allows you to locate a word by it's sound. Table 4.1 represents word with their arpabet representation based on The CMU Pronouncing Dictionary.

Word	Arpabet Representation
SPEECH	S P IY1 CH
HELLO	HH AH0 L OW1
WORLD	W ER1 L D
SLIDE	S L AY1 D

Table 4.1: Words with their Arpabet Representation

Acoustic Model

An acoustic model contains statistical representation of each sound unit. It associates the acoustic feature of a sound signal with phoneme. The exact pronunciation of a phoneme may vary widely, even for a single speaker, acoustic model gives the probability of acoustic feature that represent a particular phoneme.

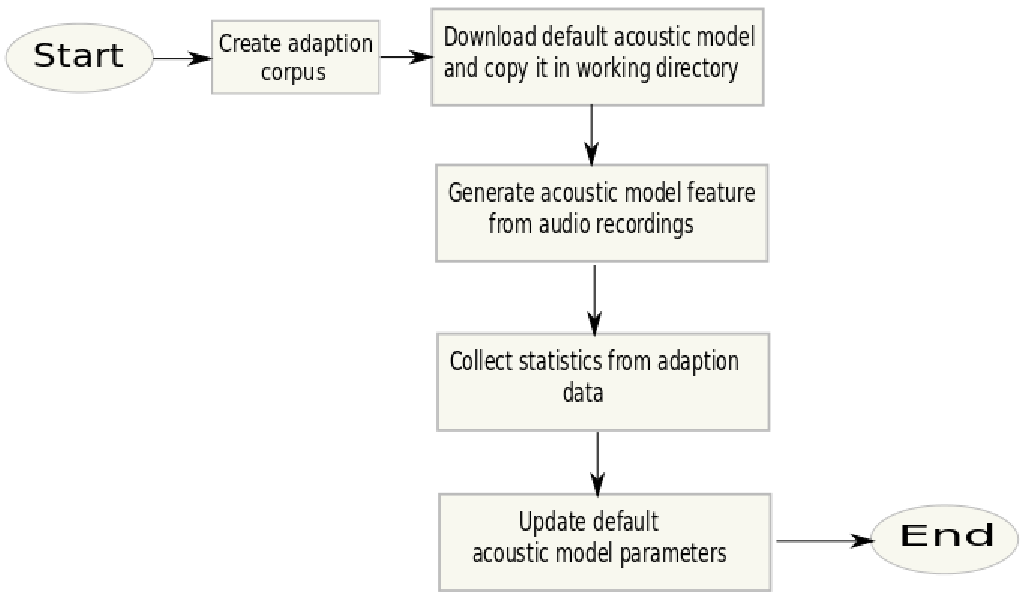

Start
Create adaption corpus
Download default acoustic model and copy it in working directory
Generate acoustic model feature from audio recordings
Collect statistics from adaption data
Update default acoustic model parameters
End

Figure 4.3: Adaption of an acoustic model

Acoustic models are trained from speech corpus, which consist audio recording with their timed aligned transcript at word or sentence level. As you adapt the model, it just improves the fit between adaption data and the model. Still, acoustic model alone are insufficient to achieve acceptable accuracy. There is dis-ambiguity between the word which have similar pronunciation sound.

It is more robust to adapt the existing acoustic model with the limited amount of training data rather than create a new acoustic model. Supervised adaption is a process of training a acoustic model with the knowledge of target speaker and domain. For example, adapt an acoustic model with a list of sentence and their corresponding recording. We have used supervised adaption for this project. Figure 4.3 represents the process of adapting an acoustic model. In unsupervised adaption, training task is performed iteratively. Output data from a first pass recognition is used to train the model for subsequent recognition process.

Language Model

A language model assigns probability to a sequence of words. While a n-gram language model gives the probability that a given word will appear in a sentence following the (n-1) proceeding words. It is most commonly used in speech to text application and machine translation system.

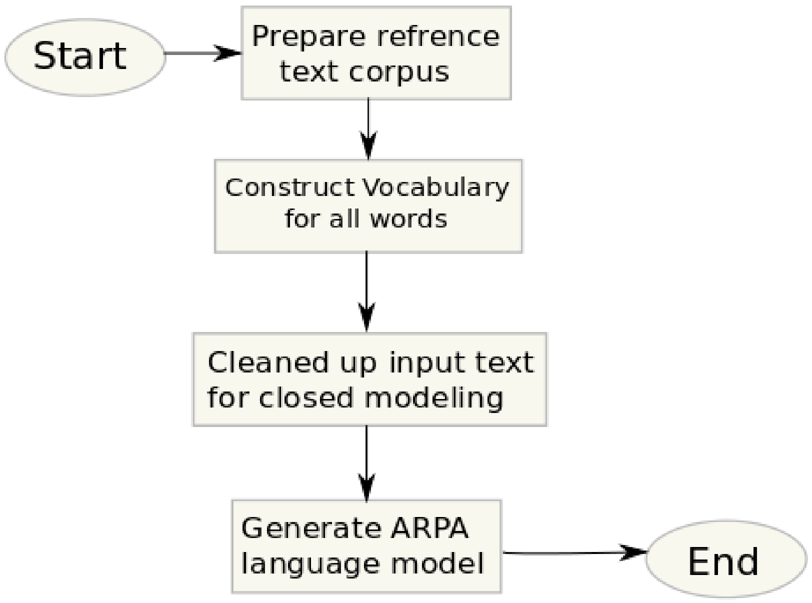

Start
Prepare refrence text corpus
Construct Vocabulary for all words
Cleaned up input text for closed modeling
Generate ARPA language model
End

Figure 4.4: Building a language model

Training or building a language model requires large amount of corpus data. Figure 4.4 represents the process of building a language model. However, Accuracy of a speech recognition engine depends on the size of language model. Our main goal is to optimize the accuracy of transcript.

For generating text corpus, we are using wikipedia as a linguistic resource. Wikipedia contain lot of extra information, grammatical, spelling or typography errors. To work with this data, we need to extract only useful information from wikipedia dump. Wikipedia dump contains set of articles with their detail information. Figure 4.5 represent the process of creating text corpus from wikipedia dump.

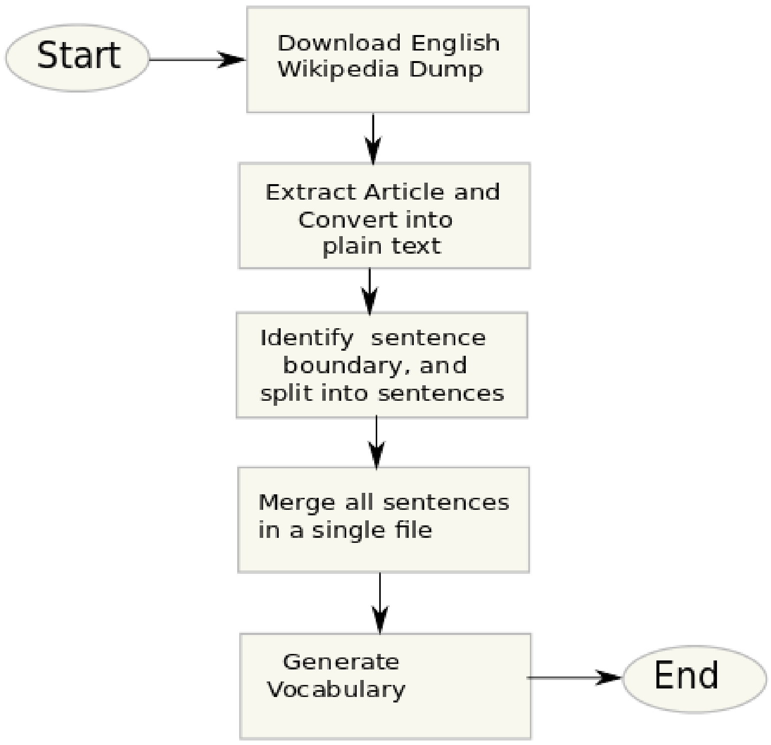

Start
Download English Wikipedia Dump
Extract Article and Convert into plain text
Identify sentence boundary, and split into sentences
Merge all sentences in a single file
Generate Vocabulary
End

Figure 4.5: Generating text corpus from WikiPedia

However, our goal is not to create larger language model, but a well adapted language model. A larger language model increases the search space for the recognizer. So, the accuracy of recognition can be reduced. To overcome this, we have created topic level language model to reduce the search space. For that, all the topic related articles are extracted from wikipedia. Gwtwiki tool is then used to process the wikipedia article and convert it into plain text.

Figure 4.6 shows the process of constructing a topic level language model from wikipedia dump. Initially, We have collected text corpus from lecture notes, presentation slides and text-books. But it is not sufficient to construct a well adapted language model. So, we have extracted topic keywords from lecture slides and lecture notes. These topic keyword are then used to extract all related article from wikipedia using gwtwiki tool.

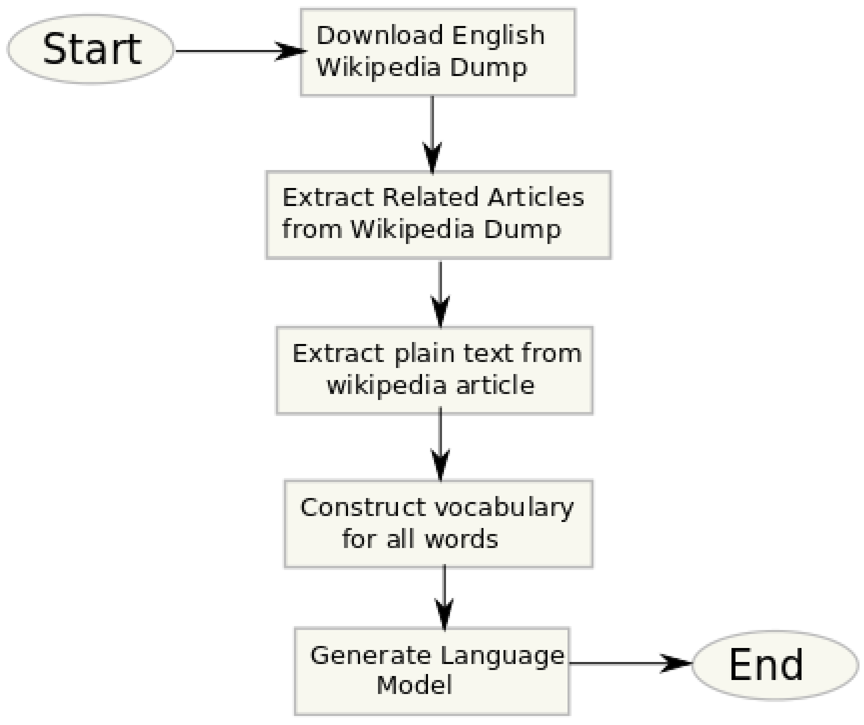

Start
Download English Wikipedia Dump
Extract Related Articles from Wikipedia Dump
Extract plain text from wikipedia article
Construct vocabulary for all words
Generate Language Model
End

Figure 4.6: Crating Text Corpus for Topic Modeling

Most of the speech recognition engine expected an audio file in a certain format. Sphinx4 requires following file format:

- Sampling Rate: 16 Khz
- Bit Depth: 16 bit
- Audio Channel: mono
- Byte Order: little-endian

To work with sphinx4, we need to convert lecture video in the format described as above. After processing the audio file, sphinx4 gives the transcript with start and end timestamps(in seconds) of each words. For example:

```
as(12.89,13.18) more(13.18,13.33)
the(15.2,15.65) carrier(15.65,16.21) committing(16.21,16.41)
```

4.2.2 Slide Extraction

Lecture videos are different from general videos. A Lecture video has some slide appearance in between. Normally, A slide will have a title or subtitle which represent the whole content of slide. Our main objective here is to extract title of each slide appeared in lecture video with timestamps. Figure 4.7 represents the process of extracting the title of slide from a lecture video.

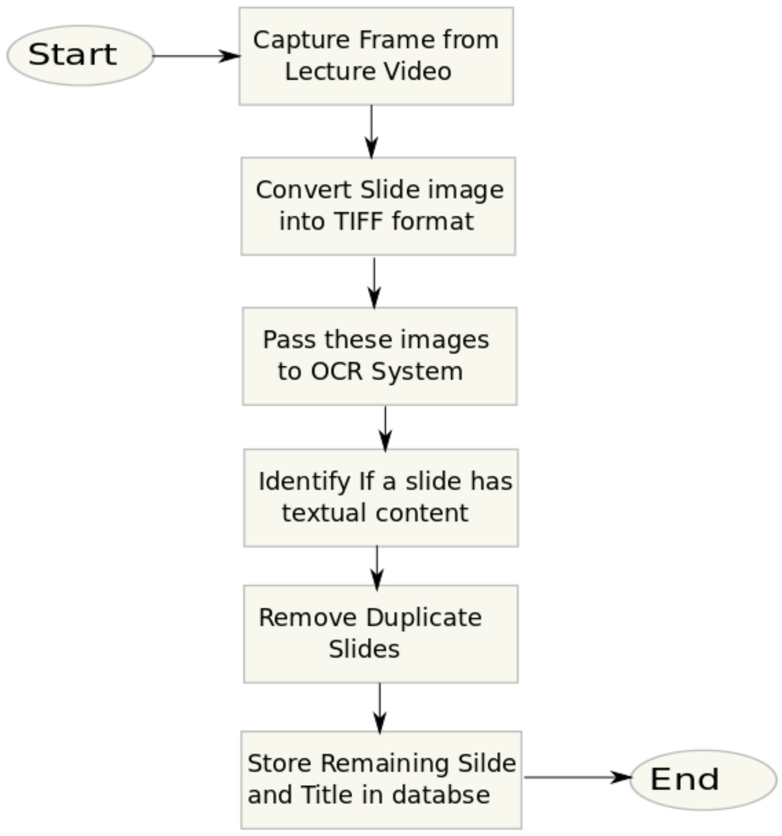

Start
Capture Frame from Lecture Video
Convert Slide image into TIFF format
Pass these images to OCR System
Identify If a slide has textual content
Remove Duplicate Slides
Store Remaining Silde and Title in databse
End

Figure 4.7: Extract Slide and Title from Lecture Video

Optical character recognition is used to extract text from a image file. An optical character recognition expects a file in tiff format. If a slide has some textual content, it will be stored in a text file. Title is then extracted from this slide content. For considering the accuracy of recognition, we are only extracting the title of the slide.

Chapter 5

Implementation Detail

This chapter includes details of various open source tools, which we have used in our system. It describes all the implementation detail of system architecture and how they are integrated.

5.1 Speech Transcript Extraction

There are various open source tool available for speech recognition. Some of them I have listed below with their description.

1. **Sphinx-4:** Sphinx-4 is a speech recognition engine written entirely in java programming language. It requires a language model and acoustic model of a language for recognizing it correctly. It also provide training tool and language modeling toolkit, which is very useful in preprocessing phase of speech recognition.

2. **Julius:** Julius is an open source speech recognition engine. It is a language independent speech recognition system, can work with any language model and acoustic model. It provide real-time speech detection based on two phase recognition. By default, It supports only Japanese language.

3. **Kaldi:** Kaldi is a toolkit for speech recognition, licensed under the Apache License v2.0. It is written entirely in c++, that is easy to modify and extend. It is mostly used by researchers.

As each of them have it's own feature, but the most versatile is CMU Sphinx4. It provide following features:

- Capable of recognizing continuous and discrete speech

- Generalize language and acoustic models
- Utilities for post processing recognition result

5.1.1 CMU Sphinx [7]

CMU Sphinx is an open source speech recognition engine. It is an HMM based speech recognizer. We have used sphinx version 4, which is written in java. It is highly customized speech recognition engine. Figure 5.1 represents the Sphinx4 architecture and important module.

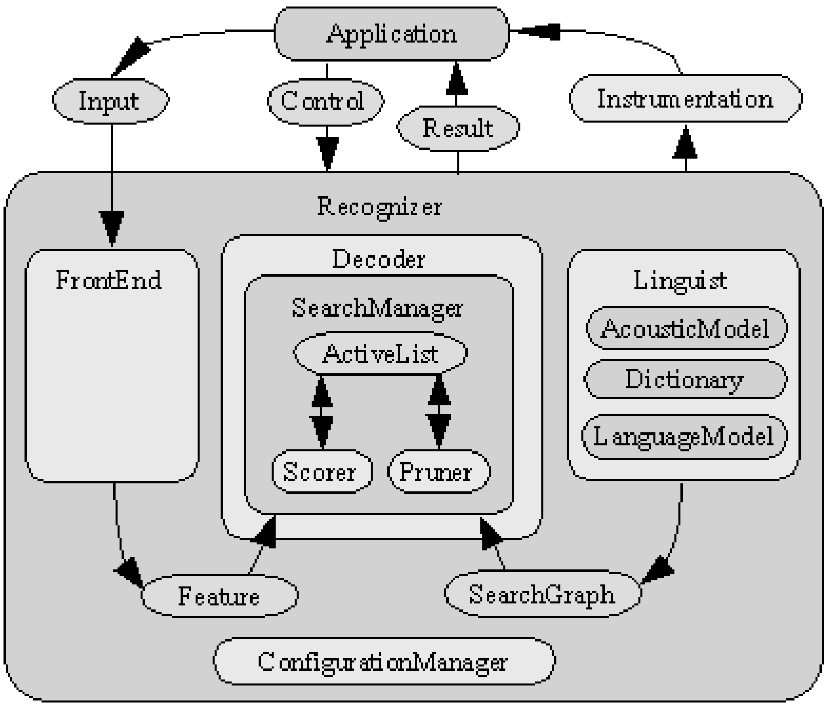


Figure 5.1: Sphinx-4 Architecture and Main Components[7]

Each module has its subcomponent. Sphinx4 constructs the module according to the configuration provided by user. These component then construct their subcomponent.

- FrontEnd
  - Audio File Data Source
  - Feature Extraction

- Decoder
  - SearchManager
  - Scorer
  - Pruner

- Linguist
  - Acoustic Model
  - Dictionary
  - Language Model

The frontend manages the audio resource and is configured to read an WAV file with mono audio channel with the pulse-code modulation at 16 khz, 16 bit. The task of frontend is to parameterized an input signal into a sequence of output features.

The linguist manages the acoustic model, dictionary and language model. The dictionary and language model must contain all the words that has to be recognized. It generates the search graph that is further used by decoder during the search.

The decoder uses the feature from frontend and search graph form linguist to generates the recognition hypothesis and results. It also provides utilities for generating the confidence score of result hypothesis. This information is retain by scorer. Search Manager uses this information to generate accurate result based on the given configuration.

5.1.2 Sphinx4 Configuration

There are two ways for configuring Sphinx4.

- **Configuration Management:** In this, the configuration is described by an XML file. It offers the advantage of keeping the source code separate with configuration. A user can change the configuration of sphinx4 without even touching the source code of application.

- **Raw Configuration:** It is used when configuration is not easily described by static XML structure. It provides extremely complex and dynamic configuration. It is also preferred when writing scripts. Scripts are used to automate this process.

In this project, We have used configuration management which is a static XML file. It defines the name and type of all components of the system. We have described the configuration for main component of sphinx4. Listing 5.1 represents the XML configuration for Sphinx4 dictionary component. Listing 5.2 represents the language model and Listing 5.3 representations the acoustic model configuration for sphinx4 application. We have used cmudict0.6 version along with newly crated dictionary based on language model. Dictionary component also has element which describe the noise dictionary, contains the word that we don't want to recognize.

Listing 5.3 defines the language model component based upon the dictionary define in listing 5.1. Frond End component defines the input source for sphinx application. It can be a microphone or audio file. The complete configuration for sphinx4 application is described in appendix A.

5.1.3 Adapting Default Acoustic Model

This section describes, how to adapt an acoustic model to improve speech recognition for our system. It takes transcribed data and improves the fit between the adaption data and the model. We have used a tool called sphinxtrain for this purpose. It comes with the sphinx4 provided by CMU. Adaption of an acoustic model is done in following steps:

1. **Creation of adaption corpus:** It will consist following files:
   - **corpus.txt:** list of sentences

     this is something very core

     single database and everything is the interface

   - **corpus.dic**: pronunciation of all words present in corpus.txt

THIS	DH IH S
IS	IH Z
...	...

   - **corpus.transcription:** sentences with their recordings

     <s> this is something very core </s> (file_001)

     <s> single database and everything is the interface </s> (file_002)

   - **corpus.fileids:** file name of recordings

     file_001

     file_002

2. **Adapting the acoustic model:** First we have to copy the default acoustic model into current directory. hub4 acoustic model is comes with pocketsphinx and it can be found in $/usr/local/share/pocketsphinx/model/hmm/en\_US/hub4$ directory.

   ```
   # cp -a /usr/local/share/pocketsphinx/model/hmm/en_US/hub4 .
   ```

   - Now, generate acoustic model feature from audio files. It can be done by sphinx_fe tool from SphinxBase.

Listing 5.1 Dictionary Configuration

```
<component name="dictionary" type="..dictionary.FastDictionary">
       <property name="dictionaryPath" value="resource:/dict/cmudict.0.6d"/>
       <property name="fillerPath" value="resource:/noisedict"/>
       <property name="addenda"
          value="file:/home/ravi/Desktop/dbms-l10-rev.dic"/>
       <property name="addSilEndingPronunciation" value="false"/>
       <property name="allowMissingWords" value="false"/>
       <property name="createMissingWords" value="true"/>
       <property name="wordReplacement" value="&lt;sil&gt;"/>
</component>
```

Listing 5.2 Language Model

```
<component name="ngramModel" type="..language.ngram.large.LargeNGramModel">
       <property name="location"
          value="file:/home/ravi/lm/dbms-l10-rev.lm.DMP"/>
       <property name="unigramWeight" value="0.7"/>
       <property name="maxDepth" value="3"/>
       <property name="logMath" value="logMath"/>
</component>
```

Listing 5.3 Acoustic Model

```
<component name="hub4" type="..acoustic.tiedstate.TiedStateAcousticModel">
       <property name="loader" value="sphinx3Loader"/>
       <property name="unitManager" value="unitManager"/>
</component>

<component name="sphinx3Loader" type="..acoustic.tiedstate.Sphinx3Loader">
       <property name="logMath" value="logMath"/>
       <property name="unitManager" value="unitManager"/>
       <property name="location" value="file:/home/ravi/am/hub4_cont_adapt"/>
       <property name="dataLocation" value=""/>
</component>
```

```
# sphinx_fe -argfile hub4/feat.params -samprate 16000 -c corpus.fileids
    -di .  -do .  -ei wav -eo mfc -mswav yes
```

- Next step is to collect the statistics from the adaption data. This is done using the bw program from SphinxTrain.

```
# ./bw -hmmdir hub4 -moddeffn hub4/mdef.txt -ts2cbfn .cont.
    -feat 1s_c_d_dd -cmn current -agc none -dictfn corpus.dic
    -ctlfn corpus.fileids -lsnfn corpus.transcription -accumdir .
```

- Now, the final step is to overwrite the existing model with the newly created files. It is done by the map_adapt tool from sphinxtrain.

```
# ./map_adapt
    -meanfn hub4/means
    -varfn hub4/variances
    -mixwfn hub4/mixture_weights
    -tmatfn hub4/transition_matrices
    -accumdir .
    -mapmeanfn hub4_adapt/means
    -mapvarfn hub4_adapt/variances
    -mapmixwfn hub4_adapt/mixture_weights
    -maptmatfn hub4_adapt/transition_matrices
```

After adaption, the acoustic model is located in the folder named "hub4_adapt”. It should have the following files:

- mdef
- feat.params
- mixture_weights
- means
- noisedict
- transition_matrices
- variances

Now, You have to just put the location of this folder in sphin4 configuration file.

5.1.4 Building Language Model

There are many ways to generate statistical language model. If your data is small, you can use some online web service. But for large data it’s not possible. We have used CMU SLM Toolkit to build a language model.

Text Preparation

We have collected lecture notes, text books, and slides to generate the language model. But, this data is not sufficient to recognize speech. We have described a process for generating text corpus from wikipedia in figure 4.6. Text corpus from wikipedia is then integrated to our lecture notes and text books data. CMU SLM Toolkit expects cleaned up text that is number are converted into text, all abbreviations are expanded, remove special characters.

Language Model Creation

The process of creating a language model is as follows:

- The language modeling toolkit expects it's input to be in certain format. The input should be a list of sentence that are bounded by start(<s>) and end(</s>) sentence markers. Example:

  <s> this is something very core </s>
  <s> single database and everything is the interface </s>

- Now generate a vocabulary file, which is list of all words present in the file. Text2wfreq utility is used to generate the most frequent word in a file and the result is passed to wfreq2vocab utility to generate vocabulary. By default it generate top 20000 most frequent words, but we can change it to 64000 words.

```
# text2wfreq < corpus.txt | wfreq2vocab > corpus.vocab
```

- To create a closed vocabulary language model, every word of a corpus.txt should be present in the vocabulary. If a word is not present in the vocabulary, remove it from input transcript file.

- Convert the text stream into an id n-gram stream using textidngram utility. And, then convert it into n-gram arpa language model.

```
# text2idngram -vocab corpus.vocab -idngram corpus.idngram < corpus.txt
# idngram2lm -vocab_type 1 -idngram corpus.idngram
    -vocab corpus.vocab -arpa corpus.arpa
```

- CMU Sphinx expects a language model in binary format. So, it is better to convert ARPA language model into binary format.

```
# sphinx_lm_convert -i corpus.arpa -o corpus.lm.DMP
```

5.1.5 Create Phonetic Dictionary

There are various tools for creating the phonetic dictionary. We can also use online web services to produce a dictionary which matches it's language model. We can also use offline variant of lmtool, called Logios. Logios allow you to compile a vocabulary file to generate phonetic dictionary, which can be used for sphin4 speech recognition engine.

5.1.6 Generating Speech Transcript from Lecture Video

To generate speech from lecture video, first we need to convert the lecture video into an audio file. Sphinx4 speech recognition engine expects an audio file to be in certain format. Audio file should be a binary with 16 bit/sample, at 16000 samples/second sampling rate.

Video to Audio Conversion

There are many open source video processing tool available. Some of them I have listed here:

- MPlayer
- FFMpeg
- AviSynth
- MKVToolnix
- MP4Box

Each of them have its own feature, but the most versatile utility is FFmpeg. Only FFmpeg provide Java wrapper for performing operation on multimedia files. This is main reason for choosing FFmpeg for converting video to audio file. It support all multimedia file formats for audio and video file.

```
# ffmpeg -i input.mp4 -ar 16000 -ac 1 output.wav
```

Here, we are using lossless compression for better quality audio. FFmpeg used ar and ac option for defining audio frequency and audio channel respectively.

Generate Speech Transcript

Now, we have everything, an acoustic model, language model and phonetic dictionary. We just have to properly configure sphinx4 and input the audio file. Code snippet for generating speech from an audio file is given below:

```
// Creates the URL of audio file
URL audioURL = LVBrowser.class.getResource("dbms-l10.wav");
...
//Creates the URL od configuration file
URL configURL = LVBrowser.class.getResource("config.xml");
...
ConfigurationManager cm = new ConfigurationManager(configURL);
Recognizer recognizer = (Recognizer) cm.lookup("recognizer");
...
//allocate the resource necessary for the recognizer
recognizer.allocate();
...
// configure the audio input for the recognizer
AudioFileDataSource dataSource = cm.lookup("audioFileDataSource");
dataSource.setAudioFile(audioURL, null);
...
//Loop until last utterance in the audio file has been decoded, in which case
    the recognizer will return null.
Result result;
while ((result = recognizer.recognize())!= null) {
  String resultText = result.getTimedBestResult(false, true);
  System.out.println(resultText);
}
```

Output of this code is

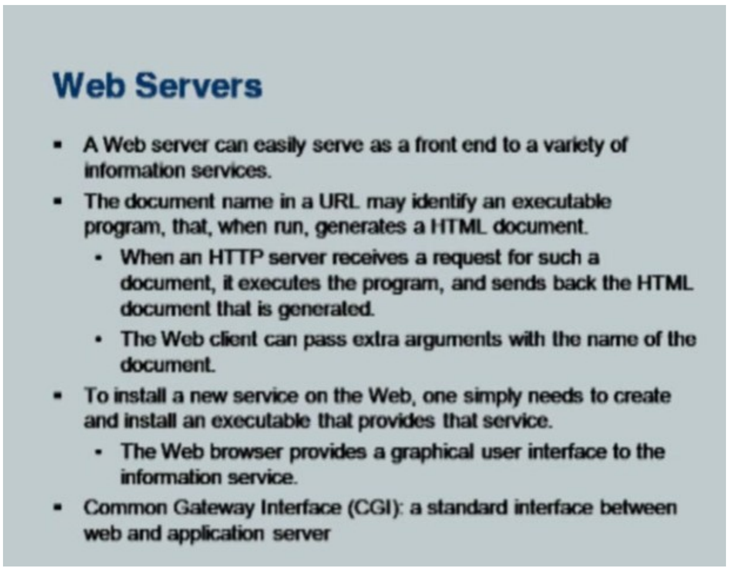
```
as(12.89,13.18) more(13.18,13.33) and(13.33,13.37)
to(13.62,13.67) produce(14.05,14.55)
the(15.2,15.65) carrier(15.65,16.21) committing(16.21,16.41) a(16.55,17.08)
...
```

5.2 Slide Extraction

Slides contains most of the part of a lecture video. So, it would be better if we utilize it to generate topics for lecture video. First, Frames are extracted from a lecture video. We are using ffmpeg library for video processing. Obviously, there are many video processing tool available. But, ffmpeg offers lossless transformation, it also provide java wrapper, which provide an easy integration to the java application. Command for extracting frame from a lecture video is given below:

```
# ffmpeg -i input.mp4 -r 1 -f output_%d.jpeg
```

A slide frame can be divided into two parts: title and description. In most of the slides, the top most 20-25% part is always represents title of the slide. We have extracted these title from each slides.



Web Servers

- A Web server can easily serve as a front end to a variety of information services.
- The document name in a URL may identify an executable program, that, when run, generates a HTML document.
  - When an HTTP server receives a request for such a document, it executes the program, and sends back the HTML document that is generated.
  - The Web client can pass extra arguments with the name of the document.
- To install a new service on the Web, one simply needs to create and install an executable that provides that service.
  - The Web browser provides a graphical user interface to the information service.
- Common Gateway Interface (CGI): a standard interface between web and application server

Figure 5.2: A sample slide

Figure 5.2 represents a sample slide. I have extracted top 23% part of this slide to extract title. Figure 5.3 represents the title part of the slide.

Web Servers

Figure 5.3: Title of a slide

Tesseract, A optical character recognition engine is used to generate text from a image file. It expects an image to be in TIFF (Tagged Image File Format). The commands for extracting title from a slide is given below:

```
# convert input.jpeg -crop x23% outputT.jpeg

# convert outputT.jpeg output.tif

# tesseract output.tif slidetext
```

Chapter 6

Experiments and Results

This chapter presents various evaluation matrices and result throughout our implementation process. We have used database management system lecture video by Prof. S Sudarshan for our experiments. Initially, we have used hub4 acoustic model and language model to generate the speech transcription. But, the accuracy of transcription is very low. Hub4 works well for american english and broadcasting news. It can not be used for lecture video. Therefor, We have adapted the default acoustic model for Prof. S Sudarshan's voice. We have also constructed a new language model specifically for database management system lecture video.

6.1 Speech Recognition Evaluation

There are various metrics available for evaluating speech recognition of a lecture video. WER(Word Error Rate) is most commonly used metric for performance of speech recognition.

6.1.1 Word Error Rate

It is derived from Levenshtein Distance, working at the word level. It first aligned the speech transcript with the reference text using dynamic string alignment. Then, Computes the following:

- S: Number of substitutions
- D: Number of deletions
- I: Number of insertions
- C: Number of correct words
- N: Total number of words in reference text

Now, Word Error Rate can be computed as follows:

$$WER = \frac{S+D+I}{N} \tag{6.1}$$

or

$$WER = \frac{S+D+I}{S+D+C} \tag{6.2}$$

6.1.2 Word Accuracy

Performance of speech recognition is usually represented by word accuracy. The Word Accuracy of speech recognition is can be calculated from word error rate.

$$WA = 1 - WER \tag{6.3}$$

or

$$WA = \frac{N-(S+D+I)}{N} \tag{6.4}$$

Chapter 7

Conclusion and Future Work

Appendix A

Sphinx4 Configuration

Appendix B

Source Code

Bibliography

[1] Academic earth. Retrieved: October 15, 2012, `http://www.academicearth.org`.

[2] Centre for distance engineering education programme. Retrieved: October 15, 2012, `http://www.cdeep.iitb.ac.in`.

[3] Coursera. Retrieved: October 15, 2012, `https://www.coursera.org`.

[4] Free video lectures. Retrieved: October 15, 2012, `http://freevideolectures.com`.

[5] Mit open course ware. Retrieved: September 30, 2012, `http://ocw.mit.edu/courses/audio-video-courses`.

[6] NPTEL. Retrieved: October 15, 2012, `http://www.nptel.iitm.ac.in/`.

[7] Sphinx-4 - a speech recognizer written entirely in the java(tm) programming language. Retrieved: May 15, 2013, `http://cmusphinx.sourceforge.net/sphinx4/`.

[8] Videolectures.net. Retrieved: October 15, 2012, `http://videolectures.net`.

[9] J. Glass, T.J. Hazen, S. Cyphers, I. Malioutov, D. Huynh, and R. Barzilay. Recent progress in the mit spoken lecture processing project. In *Proc. Interspeech*, volume 3, 2007.

[10] Marti A. Hearst. Texttiling: segmenting text into multi-paragraph subtopic passages. *Comput. Linguist.*, 23(1):33–64, March 1997.

[11] V.K. Kamabathula and S. Iyer. Automated tagging to enable fine-grained browsing of lecture videos. In *Technology for Education (T4E), 2011 IEEE International Conference on*, pages 96 –102, july 2011.

[12] M. F. Porter. Readings in information retrieval. chapter An algorithm for suffix stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[13] S. Repp, A. Gross, and C. Meinel. Browsing within lecture videos based on the chain index of speech transcription. *Learning Technologies, IEEE Transactions on*, 1(3):145–156, july-sept. 2008.

[14] S. Repp and C. Meinel. Segmentation of lecture videos based on spontaneous speech recognition. In *Multimedia, 2008. ISM 2008. Tenth IEEE International Symposium on*, pages 692 –697, dec. 2008.

[15] J.C. Reynar. *Topic segmentation: Algorithms and applications*. PhD thesis, University of Pennsylvania, 1998.

[16] Harald Sack. Integrating social tagging and document annotation for content-based search in multimedia data. In *In Semantic Authoring and Annotation Workshop (SAAW)*, 2006.

[17] Harald Sack and JÃűrg Waitelonis. Automated annotations of synchronized multimedia presentations. In *In Workshop on Mastering the Gap: From Information Extraction to Semantic Representation, CEUR Workshop Proceedings*, 2006.