International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 02 | Feb-2016 www.irjet.net p-ISSN: 2395-0072

Automatic Subtitle Generation for Sound in Videos

Kanika Mishra[1], Prerak Bhagat[2], Prof. Atiya Kazi[3]

Kanika Mishra, Department of Information Technology, University of Mumbai, Finolex Academy of Management & Technology, Ratnagiri, Maharashtra.

Prerak Bhagat, Department of Information Technology, University of Mumbai, Finolex Academy of Management & Technology, Ratnagiri, Maharashtra.

Prof.Atiya Kazi, Department of Information Technology, University of Mumbai, Finolex Academy of Management & Technology, Ratnagiri, Maharashtra.

Abstract – ***The last two decades have seen a vast growth in the amount of video content being generated across various platforms and spanning several genres. Apart from the convenient availability of cheap portable media devices, the emergence of dedicated websites tailored specifically for storage and sharing of videos has increased the value of audio - visual content in eyes of the general populous.***

However as with any technology, the emergence of audio-visual content as the preferred method of information sharing has brought along with it certain hurdles. Prominent among those is the existence of numerous accents and dialects of any particular language. Another is the sheer number of languages that the content is generated in. Also the hearing impaired audience is unable to receive the audio part of the content and thus, unable in most cases, to completely or accurately comprehend the information being presented to them. This situation can be rectified with aid of audio transcripts and necessary translations. The software's currently available require for most part an active user participation and also have a notable lack of accuracy. The sheer quantity of audio - video information therefore makes it impossible to provide transcripts for all relevant data encapsulated in such formats. Hence, the need for an automated concept is vividly felt.

This thesis report analyses a specific path to automatically generating subtitles for videos using Google's free API's. Due to lack of necessary resources, the translation component has not been included in this specific paper. We propose a three step model to realize our process. The first stage will be extracting the audio to be transcribed from the video file and then converting it into a format compatible with our second stage if the need be. The second stage uses a speech recognition engine to convert the information from audio format to text format. The third and final stage will be the encoding of the generated text into a subtitle format which will include adding time frames and necessary pauses and punctuation marks.

The results of the resulting works have been encouraging though there exists a tremendous room for improvements and adjustments.

***Key Words*: Audio Extraction, Speech Recognition, Subtitle Generation.**

1. INTRODUCTION

Video is one of the most popular multimedia used on PCs. So it is essential to make information encompassed in form of sound in videos comprehendible to people with auditory problems. The most natural way lies in the use of subtitles. However, manual subtitle creation is a tedious activity and requires active participation of the user. Hence, the study of automatic subtitle generation presents as a valid subject of research.

2. LITERATURE SURVEY

This is overview of concept we got from some papers about automatic subtitle generation for sound in videos. There are many programming languages that can be used in the creation of AutoSubGen. A brief study on internet suggests

focus on C/C++ and Java. We now analyze the positive points of the two languages based on the research of J.P. Lewis and Ulrich Neumann [8]. On the one hand, C++ provides remarkable advantages in speed, cross systems performances, and well-tested packages. On the other hand, Java offers an intuitive syntax, portability on multiple Operating Systems and trusted libraries. They are used in the Sphinx speech recognition engine. Java provides the Java Media Framework (JMF) API that allows developers to deal with media tracks (audio and video) in an efficient way. In Lewis's articles, it appears Java performances are quite similar to C/C++. Besides, in theory, Java performances should be higher. The absence of pointers, the use of efficient garbage collector and the run-time compilation plays a key role in Java and their extent should soon permit to go beyond C++ performances.

The JMF API features facilitate audio extraction. We present an overview of the JMF API and focus a bit deeper on insightful functions in relation to audio extraction. It exists an interesting tutorial [4] to start with the JMF and hence it is used in the following parts. The official documentations are also pertinent resource [4]. Furthermore, a complete book describing JMF API [3] was published in 1998. JMF gives applications the possible capability to output multimedia content, capture audio and video through microphone and camera, do real-time streaming of media over the Internet, process media and store media into file.

3. METHODOLOGY

3.1 Audio Extraction

This module aims to output an audio file from a media file. An activity diagram of the module is presented in Figure 1. It takes as input a file URL and gives audio format of the output. Next, it checks the file content and creates a list of individual tracks composing the initial media file. An exception is thrown if the file content is irrelevant. Once the tracks are separated, the processor analyzes each track, selects the first audio track found and discards the rest. Finally, the audio track is written in a file applying the default or wished format. In Audio Formats Discussions, it was previously talked about the audio formats that Sphinx accepts. At the moment, it principally supports WAV or RAW files. Hence, the task of obtaining the preferred format is complicated. It often requires various treatments and conversions to reach the mentioned purpose. Audio extraction provides a way to extract audio from a media file but does not offer conversion tools. Otherwise, it is advisable to use the facilities of VLC for the purpose of getting suitable audio material. It specially uses the java media framework (jmf) for audio extraction. Because JMF provides a platform-neutral framework for handling multimedia [4].

Java Media Framework (jmf): JMF is used for handling streaming media in Java programs. JMF enables Java programs to -

- Present multimedia contents,
- Capture audio through microphone and video content through camera,
- Do instant streaming of media over the Internet,
- Process media(such as changing media format, adding special effects),
- Store media into a file.

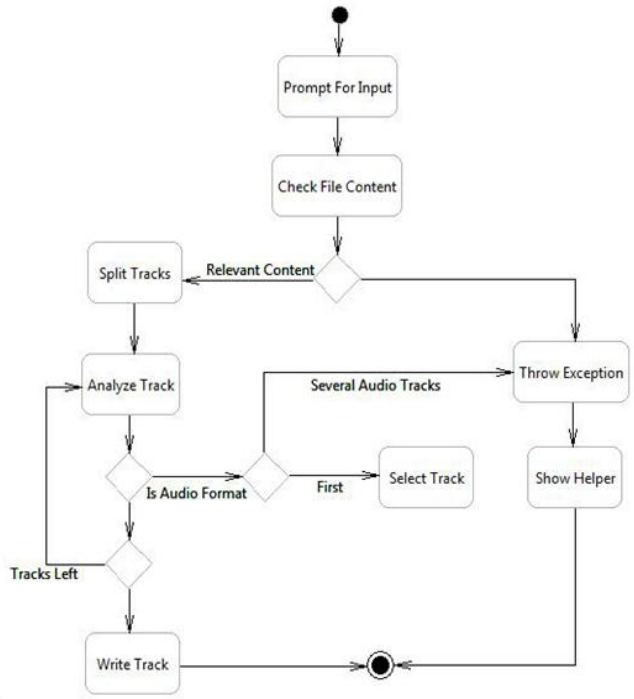

Figure 1. Activity Diagram for Audio Extraction.

3.2 Speech Recognition

Some speech recognition systems make use of "source-independent speech recognition" while some others use "training" in which a single speaker reads sections of text into the SR system. These system analyze the person's specific voice and use it to find the recognition of that individual's speech, which results in more accurate transcripts. Systems that do not make use of training method is called as "speaker-independent" systems. Systems that

make use of such training is called as "speaker-dependent" systems. Here in our Project we have used the Line playback device, which lets the voice to the google transcriptor in the chrome web app and the text is generated from it. Google Transcriptor gets the voice from the audio file which is provided by the user, through the LINE playback device and starts converting that audio into the text format and displays on the front end. We can create a file .txt or .rtf using that text. Using different Sphinx systems [6], for nearly two decades they have developed and proved their reliability within several ASR systems where they were linked in. It is therefore natural we opted for the Sphinx-4 decode. Sphinx-4 has been entirely written in Java [7], making it totally portable and able to provide a modular architecture, that allows us to modify configurations with ease. The architecture overview was shown in the background section. We aim to procure advantage of Sphinx-4 features in order to satisfy the needs of the Automatic Subtitle Generation Speech recognition module [5]. Subtitle Generation is expected to select the most suitable models considering the audio and the parameters (category, length, etc.) given in input and thereby generate a precise transcript of the audio speech. The Figure 2 resumes the organization of the speech recognition module. Speech Recognition uses the Hidden Markov Model (HMMs) as an algorithm.

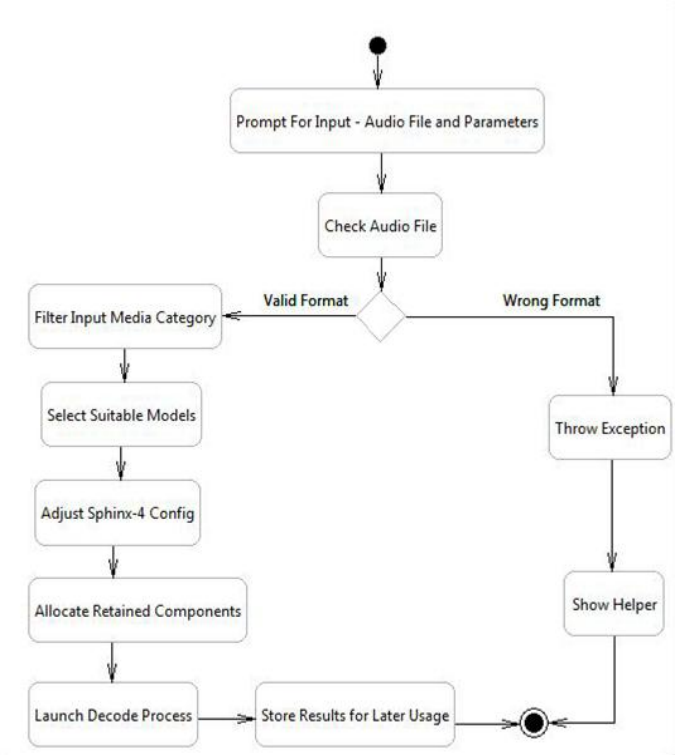

Figure 2.Activity Diagram for Speech Recognition.

3.3 Subtitle Generation:

This module is expected to get a list of words and their respective speech time-frames from the speech recognition module and then produce a .srt subtitle file. To do so, the module must look at the list of words and use silence (SIL) spoken words as a boundary between for two consecutive sentences. The Figure 3 offers a visual representation of the steps to follow for the subtitle generation for sound in videos. However, we face up to some limiting rule. Indeed, it will not be able to define punctuation in our system since it involves much more speech analysis and deeper design.

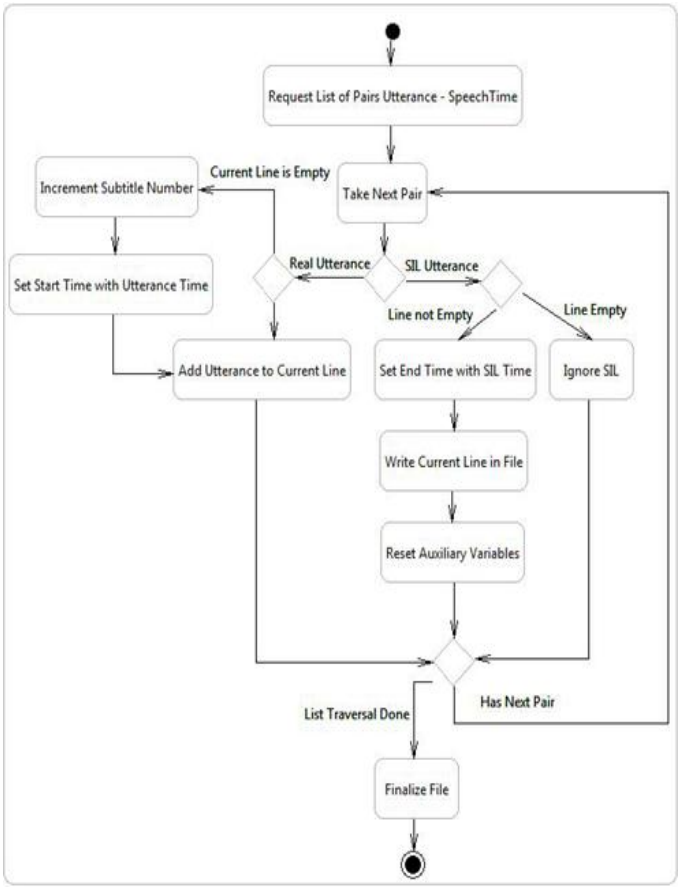

Figure 3. Activity Diagram for Subtitle Generation.

4. REQUIREMENTS OF PROPOSED SYSTEMS

The requirements of our proposed system is expressed through Table I.

TABLE I. REQUIREMENT SPECIFICATIONS

No	Technology used	Description
1.	1GB RAM	Storing
2	Java Speech API	It allows to incorporation of speech technology into user interfaces for Java based applets and applications. This API defines a cross-platform interface to help command and control recognizers and speech synthesizers. Sun [12] has also developed the JSGF(Java Speech Grammar Format) to supply cross-platform grammar of speech recognizers
3	JDK	For writing Java applets and applications and run them into java language.
4	Apache Tomcat	To manage sessions as well as applications across the network
5	MySQL Database	Storing audio and video related text files
6	Net beans	It is an IDE use to run the whole project.

5. CONCLUSIONS

A complete system including the required modules could not be realized since the audio conversion need more resources. VLC gave a suitable solution but a custom coded Java component is expected in future projects so that portability and installation of the system become uncomplicated. Nonetheless, the expected output for each phase has been reached. The audio extraction module supply a suitable audio format that can be used by the intended speech recognition module. Speech recognition produce a list of recognized words and their corresponding time-frames in the audio content, although the accuracy cannot be guaranteed. The former list is then used by the intended subtitle generation module to create standard subtitle file that is readable by the most common media players available.

As of Now subtitles are being generated for the specific format of the Audio files, But In future this restriction of the format will be removed. English Subtitles of the Regional Languages will be generated, and the accuracy of the project will be more precise. Offline content will be web-based in coming future, and the User can access and generate subtitles for his video in a feasible way. Overall it will be Accurate, less time consuming, No language barrier, Web-based.

REFERENCES

[1] Boris Guenebaut, "Automatic Subtitle Generation for Sound in Videos", Tapro 02, University West, pp. 35, 2009.

[2]Zoubin Ghahramani,"An introduction to hiddenmarkov models and Bayesian networks", in World Scientific Publishing Co., Inc. River Edge, NJ, USA, pp.30, 2001.

[3] Robert Gordon, Stephen Talley, and Rob Gordon.Essential Jmf: Java Media Framework. Prentice Hall PTR, pp. 17, 1998.

[4] Java tm media framework api guide, 1999. URL:http://java.sun.com/javase/technolog/desktp/mdia/jmf/2.1.1/guide/index.html

[5] Frederick Jelinek, Statistical Methods for Speech Recognition. MIT Press, pp. 7, 1999.

[6] L. Besacier, C. Bergamini, D. Vaufreydaz, and E. Castelli,"The effect of speech and audio compression on speech recognition performance". In IEEE Multimedia Signal Processing Workshop, Cannes:France,2001.URL :http://hal.archives ouvertes.fr/docs/00/32/61/65/PDF/Besacier01b.pdf.

[7] Sphinx-4 a speech recognizer written entirely in the javatm programming language, pp. 5, 1999-2008. URL http://cmusphinx.sourceforge.net/sphinx4/.

[8] J.P. Lewis and Ulrich Neumann. Performance of java v/s\ c++ URL:http://www.idiom.com/~zilla/Computer/javaCbenchmark.html.