Automatic Subtitle Generation for English Language Videos

Akshay Jakhotiya, Ketan Kulkarni, Chinmay Inamdar, Bhushan Mahajan, Alka Londhe

Department of Computer Engineering
PimpriChinchwad College of Engineering, Pune, India

Abstract- *The use of videos for the purpose of communication has witnessed a phenomenal growth in the past few years. However, non-native language speakers or people with hearing disabilities are unable to take advantage of this powerful medium of communication. To overcome the problems caused by hearing disabilities or language barrier, subtitles are provided for videos. The subtitles are provided in the form of a subtitle file most commonly having a .srt extension. Several software have been developed for manually creating subtitle file, however software for automatically generating subtitles are scarce. In this paper, we introduce a system that we have envisioned will generate subtitles automatically through a 3-stage process: Audio extraction, Speech recognition and Subtitle synchronization.*

Keywords- *Subtitles, .srt file, Audio Extraction, Speech Recognition, ffmpeg, JAVE, CMU Sphinx*

I. INTRODUCTION

```
1582
01:58:40,417 --> 01:58:42,780
Are they gone?

1583
01:58:48,746 --> 01:58:51,676
No, they're not gone.

1584
01:58:51,744 --> 01:58:53,369
And maybe they
never will be.

1585
01:58:53,443 --> 01:58:55,170
But I've gotten used
to ignoring them
```

Fig. 1. A Sample SRT File

Subtitles are text translations of the dialogues in a video displayed in real time during video playback on the bottom of the screen. The subtitles may be in the same language as the video or other language. These subtitles are meant to assist people suffering from auditory problems, people not familiar with the language or even people who are learning to read. A subtitle file is the backbone of these subtitles. The several subtitle file formats are .srt,.ssa,.sub etc. A typical .srt file has the following structure:

1. Subtitle number -A number indicating which subtitle it is in the sequence.
2. Subtitle timing -The time that the subtitle should appear on the screen, and then disappear.
3. Subtitle text -The subtitle itself.
4. A blank line indicating the start of a new subtitle [5].

All the individual units containing one subtitle number, timing and the subtitle text will be referred as a 'subtitle entry' in this paper.

Several subtitle editors are available for the purpose of creating subtitles.These software allow the user to watch the video and to insert subtitles using the timeline of the video. One of the most popular subtitle editors is GNOME subtitles.

This software involves selecting start time instant and end time instant of each subtitle entry and then manually typing the subtitles. [6]. Several other software like Subtitle Editor [13]and Gaupol [14] are built on the same line.These software require the user to manually type the subtitles. Our paper details a system in Java that will automate this process of typing subtitles by means of speech recognition. Section II details the user interface and working from user's point of view. Section III details the mechanism behind automatic subtitle generation.

II. USER INTERFACE

The main components of the user interface are:

A. Video Panel:

The user will select the desired video for which he/she wants to create subtitles. The video will start playing in the video panel. This will facilitate the user to have a preview of the video before generating the subtitles for a particular portion.The video panel will have all types of video controls like play/pause, volume up/down etc.

B. Start-Stop Buttons:

A start and a stop button will be provided to let the user select the start time instant and end time instant for a subtitle entry. The user will pause the video at the desired time instant and click on start button to record the start time. The same procedure will be followed for recording end time. After the user will click on OK button, the speech in the video between

the start and end time instants will be converted to text .

C. Text Area:

The output of speech to-text will be displayed in the text area. The user can edit the text output if there are some errors. Below the text-area will be font formatting options like font colour, font type etc.Font formatting can be done to the subtitles by use of simple HTML tags. This is particularly useful in case of emotional subtitles where font types, sizes, colours are used to indicate the emotional nuances of the dialogues[1]. An OK button will be provided to confirm the subtitle entry and add it to the database.

D. History Panel:

A panel will be provided which will display all the subtitle entries already added in the database. This panel will be used by the user for reference.

E. Create Subtitles Button:

Finally after all the subtitle entries have been added, the user will click on 'Create Subtitles' button. A dialog window will appear prompting the user for subtitle file format. After the user will select the desired file format, the subtitle file containing all the subtitle entries will be created.

III. MECHANISM

The automatic subtitle generation process is a 3 stage process:

A. Audio Extraction:

The speech recognition engine requires a .wav audio file as input. Hence, it is necessary to convert the video into .wav format [2]. Ffmpeg will be used to accomplish this. Ffmpeg is a complete, cross-platform solution to record, convert and stream audio and video[7]. Ffmpeg is used by giants like VLC Media Player, Google Chrome, Blenderetc[8]. In order to use ffmpeg for converting a video into audio, JAVE will be used. The JAVE (Java Audio Video Encoder) library is Java wrapper on the ffmpeg project. Developers can take take advantage of JAVE to transcode audio and video files from a format to another [9].Using JAVE library, only the portion of the video between the start and end time selected by the user will be converted to audio .wav format suitable for speech recognition.

B. Speech Recognition:

The .wav file obtained from the audio extraction phase will be passed forward for speech recognition. An open source speech recognition engine called CMU Sphinx will be used in this stage. Sphinx-4 is an open source project led by Carnegie Mellon University, Sun Microsystems Inc. and Mitsubishi Electric Research Laboratories.It is completely written in Java. It offers a highly modularized and flexible architecture as well as versatile APIs, supports any acoustic model structure and handles most types of language models[3].CMU Sphinx requires following inputs:

1) **Acoustic model**: An acoustic model contains acoustic properties for each senone.There are context-independent models that contain properties (most probable feature vectors for each phone) and context-dependent ones (built from senones with context)

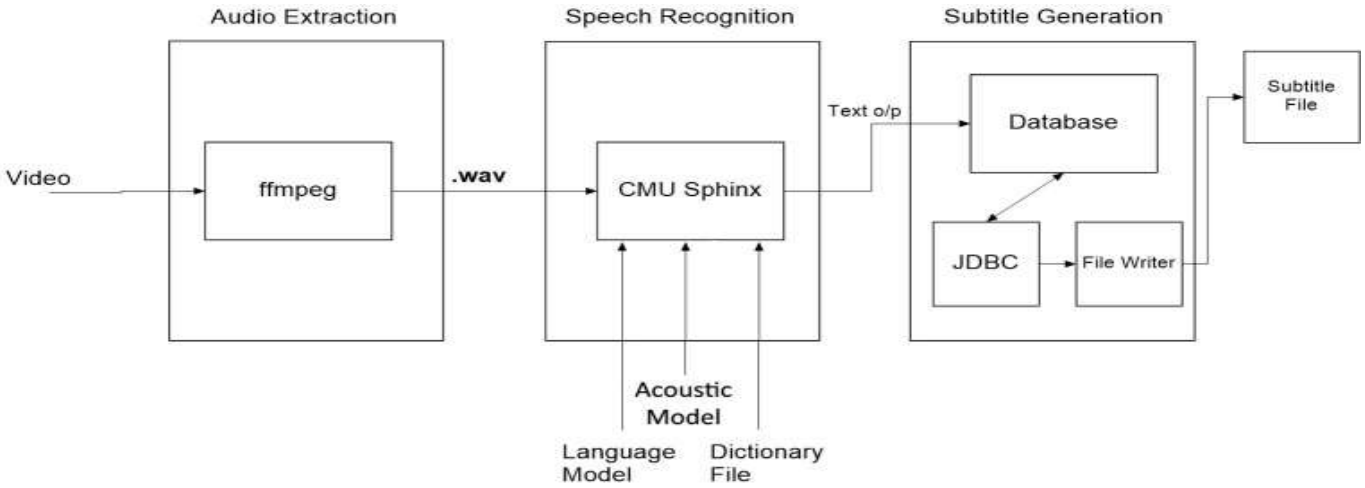

Fig. 2. 3-Stage Process

2) Language model: A language model is a grammar file for an entire language. It defines which word could follow previously recognized words and helps to significantly restrict the matching process by stripping words that are not probable.
3) Dictionary file: A dictionary file contains the list of all words in the language along with their pronunciations.
4) *Audio input:* An audio input in the form of .wav is required [10].

The language model, acoustic model and dictionary file for US English and few other languages are available for download on CMU sphinx website [11]. This system deals with English language videos. Hence, the .wav file generated in previous phase is passed to CMU Sphinx engine along with the US English language model and dictionary file. The text output for the given audio will be generated and displayed to the user. User will add this subtitle entry to the database after desired editing.

C. Subtitle Synchronization:

This is the final phase in the automatic subtitle generation process. Java DB is Oracle's supported distribution of the Apache Derby open source database. It is based on Java, JDBC and SQL standards and can be embedded in Java applications[12].The subtitle entries generated in previous phase will be added one by one to a table in Java DB through JDBC connection.The table will have the following format:

TABLE I. SUBTITLES TABLE FORMAT IN JAVA DB

START TIME	END TIME	SUBTITLE
01:58:40,417	01:58:42,780	Are they gone?
01:58:48,746	01:58:51,676	No, they're not gone.
01:58:51,744	01:58:53,369	And maybe they never will be.
01:58:53,443	01:58:55,170	But I've gotten used to ignoring them
01:58:55,243	01:58:58,638	and I think as a result they've kind of given up on me.

Once the user has finished adding all the subtitle entries in the table, he/she will select the format of the subtitle file. The data stored in the table will be retrieved row by row, one at a time and written to the subtitle file according to the specifications of the subtitle format. This write operation to the subtitle file will be achieved through the use of FileWriter class in Java.

IV. CONCLUSION

We have tweaked the manual typing part in conventional subtitle editing software and made it automatic by use of speech recognition. This system comes with a limitation though that the accuracy of speech recognition can never match the accuracy in manual typing by humans. This loss in accuracy may occur due to inability to punctuate the text output, inability to process the words not present in the dictionary, difference in speaking accents or noise in video[4]. Nonetheless, this system also provides the user with facility to edit subtitles before creating subtitle file. This method can save a lot of time and increase productivity.

ACKNOWLEDGEMENTS

The authors would like to thank Dr K Rajeswari and Prof. SonaliTidke for their useful suggestionsand support during the preparation of this technical paper.

REFERENCES

[1] J.O. Djan and R. Shipsey, “E- Subtitles: Emotional Subtitles as a Technology to assist the Deaf and Hearing-Impaired when Learning from Television and Film”, in Sixth International Conference on Advanced Learning Technologies, pp.464-466, 2006
[2] A. Mathur, T. Saxena, R. Krishnamurthi, “Generating Subtitles Automatically using Audio Extraction and Speech Recognition”, 2015 IEEE International Conference on Computational Intelligence & Communication Technology, pp.621-626
[3]Suman K. Saksamudre1 and R. R. Deshmukh, “Isolated Word Recognition System for Hindi Language.“, International Journal of Computer Science andEngineering, Vol. 3,Issue.7,2015, pp. 110-114.
[4]Sukhandeep Kaur1 and Kanwalvir Singh Dhindsa, “Speaker Recognition System Techniques and Applications.“, International Journal of Computer Science andEngineering , Vol. 3,Issue.8,2015, pp. 101-104.
[5]http://www.matroska.org/technical/specs/subtitles/srt.html, accessed October 2015
[6] http://gnome-subtitles.sourceforge.net, accessed October 2015
[7] https://www.ffmpeg.org, accessed October 2015
[8] https://trac.ffmpeg.org/wiki/Projects, accessed October 2015
[9] http://www.sauronsoftware.it/projects/jave/index.php, accessed October 2015
[10] http://cmusphinx.sourceforge.net/doc/sphinx4, accessed October 2015
[11]http://sourceforge.net/projects/cmusphinx/files/Acoustic%20and%20Language%20Models/, accessed October 2015
[12]http://www.oracle.com/technetwork/java/javadb/overview/index.html, accessed October 2015
[13]http://home.gna.org/subtitleeditor, accessed October 2015
[14]http://home.gna.org/gaupol, accessed October 2015