CCAligner

An open source tool for word by word subtitle - audio synchronisation.

CCExtractor Development, Google Summer of Code 2017

Saurabh Shrivastava
Institute of Engineering and Technology
Devi Ahilya Vishwavidyalaya, Indore
India
Email : saurabh.shrivastava54@gmail.com
Phone : +918871468353

TABLE OF CONTENT

PURPOSE

To build a tool for word by word synchronisation of subtitles with audio present in the video by tagging each individual word as it is spoken.

INTRODUCTION

Closed captioning (CC) and subtitling are both processes of displaying text on a television, video screen, or other visual display to provide additional or interpretive information. The usual subtitle files (such as SubRips) have line by line synchronisation in them *i.e.* the subtitles containing the dialogue appear when the person starts talking and disappears when the dialogue finishes. This continues for the whole video.

```
1274

01:55:48,484 --> 01:55:50,860

The Force is strong with this one
```

In the above example, the dialogue #1274 - The Force is strong with this one appears at 1:55:48 remains in the screen for two seconds and disappears at 1:55:50.

The aim of the project is to tag the word as it is spoken, similar to that in karaoke systems.

E.g.

```
The            [6948484:6948500]
Force          [6948501:6948633]
is             [6948634:6948710]
strong         [6948711:6949999]
with           [6949100:6949313]
```

In the above example each word from subtitle is tagged with beginning and ending timestamps based on audio. The output can be generated in various formats as per requirement. Note : The example above is illustrative and is not actual output.

From technical viewpoint, the high level workflow for this task basically involves extracting audio from video file, pre-processing if required, do audio analysis - mainly ASR, use existing subtitles as reference and then generate timestamps.

WHY THIS PROJECT?

Subtitles and captions are an integrated part of lives of numerous people like me. People use them for variety of reasons ranging from learning new language to as a aid for hearing impaired. I joined CCExtractor community in November last year, and ever since I have come across several people and organisations using subtitles and CC for the purposes I never thought of before. There's more to subtitles than these common usages.

Out of numerous such uses, some are :

- Archive.org uses CC data as a transcript to various audios and video files for indexing and classification.
- Our sister organisation The Distributed Little Red Hen Lab specialises in conducting transdisciplinary research in multimodal

communications, linguistics, computational linguistics, cognitive science, neuroscience, education, statistics, media effects studies, political communication, and library science where subtitles and CC data play a huge role.

- The projects such as Speaker diarization, Gender detection, Emotion detection and identification etc. make extensive use of CC and subtitles.

The capability which this project will add makes me extremely excited. This technique of tagging each word to audio unlocks completely new avenues like far better subtitle synchronisation, audio video editing (e.g. Videogrep), commentary and podcasts analysis, accurate transcription, enhancement of aforementioned projects et cetera.

I look at this technology from the perspective of enabling a whole new dimension of use cases. So the aim is to provide it to users/developers in as flexible form as possible so that people can build upon it for their personal use cases that they can think of.

PROPOSED DELIVERABLES

1. A generalised tool with the following functionality:
    a. Subtitles parsing and processing.
    b. Words and audio tagging.
    c. Generation of timestamps for each word.
    d. Audio based subtitle synchronisation.
    e. Speech Recognition based transcription.
    f. Generation of difference between transcript and subtitles.
    g. Indication of missing / extra words.
    h. Generation of output in various formats.
2. Provision for extending the tool so user can define the method being used to perform the action (e.g. if user want to perform subtitle synchronisation by offset calculation method or by thorough analysis).
3. Minimal and functional UI.
4. Tests for various logical units.
5. Documentation within the tool.
6. Setup scripts.

7. Blog posts on weekly basis + in case of any special event.

IMPLEMENTATION DETAILS

The tool will be primarily in C++ or C as is its parent program CCExtractor with some probable Python components.

The tools that will be leveraged in the overall workflow are:

1. Automatic Speech Recognition system :
    a. PocketSphinx : PocketSphinx is a lightweight large vocabulary, speaker-independent continuous speech recognition engine written in C++. They are under BSD-style license.

    b. Kaldi : Kaldi is a toolkit for speech recognition written in C++ and licensed under the Apache License v2.0.

    We will make use of this to create and locate phonemes or word from the audio files. PocketSphinx is the preferred choice, but options are kept open for discussion and testing.

2. FFmpeg: This tool will be used to extract audio from video files and for format conversions and processing of audio files when needed.

3. Forced Aligners : The tool may use code inspired by / taken from several available open source tools such as Gentle, Rhubarb Lip-Sync and PocketSphinx's inbuilt continuous recognisor.

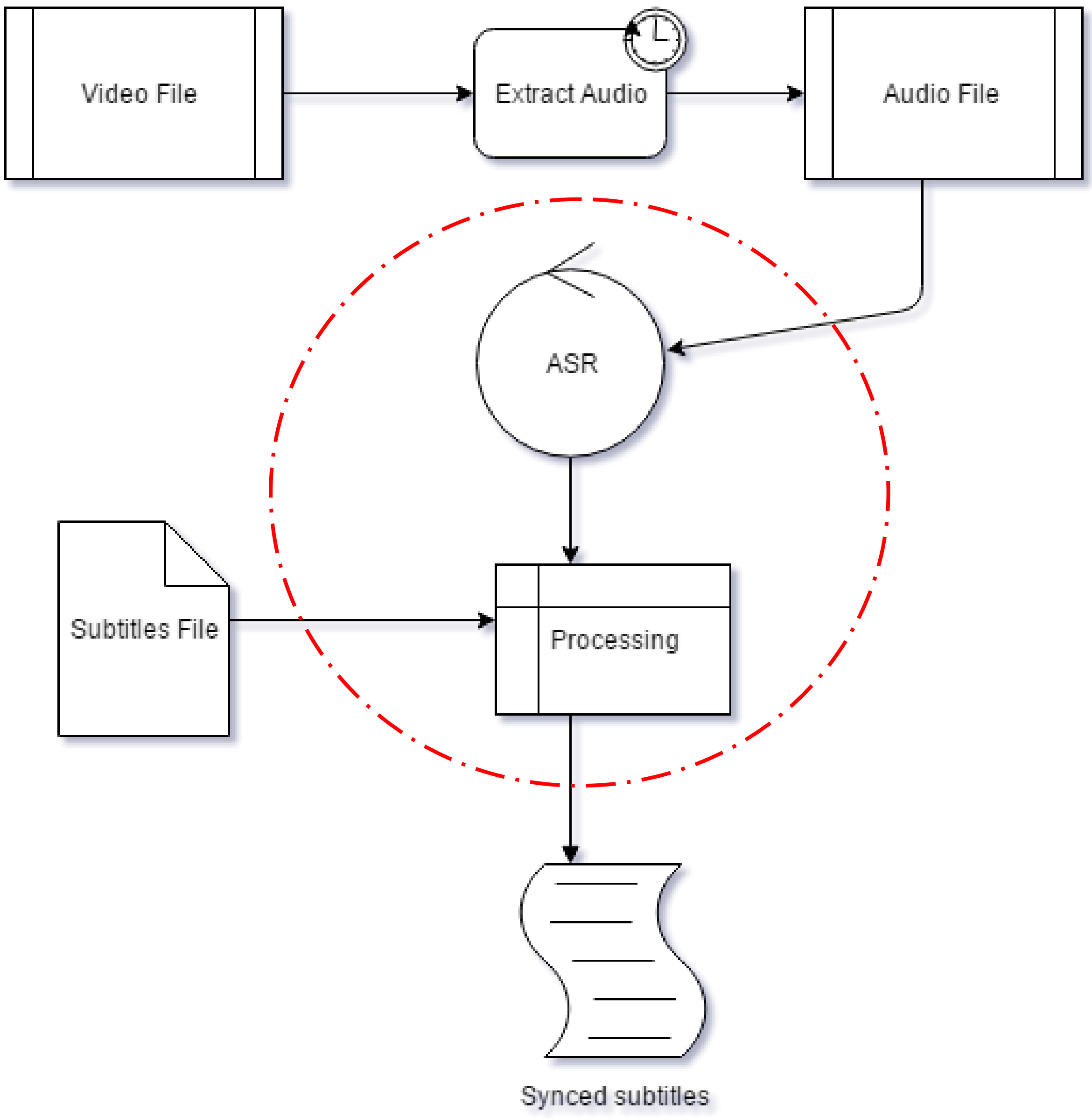
4. CMUCLMTK : Used for training language models.

    More open source tools maybe used as per need.

Note :
I experimented with PocketSphinx and CMUSphinx a lot but couldn't use Kaldi (I'll post the comparisons at the end of proposal) due to its high resource requirement. Pre GSoC I'll definitely spend time using and learning Kaldi as well so that I can compare and build robust backbone to

our tool. For now the preferred tool is PocketSphinx for its small size, fast processing and less resource using capability.

While the main workflow is pretty straightforward :



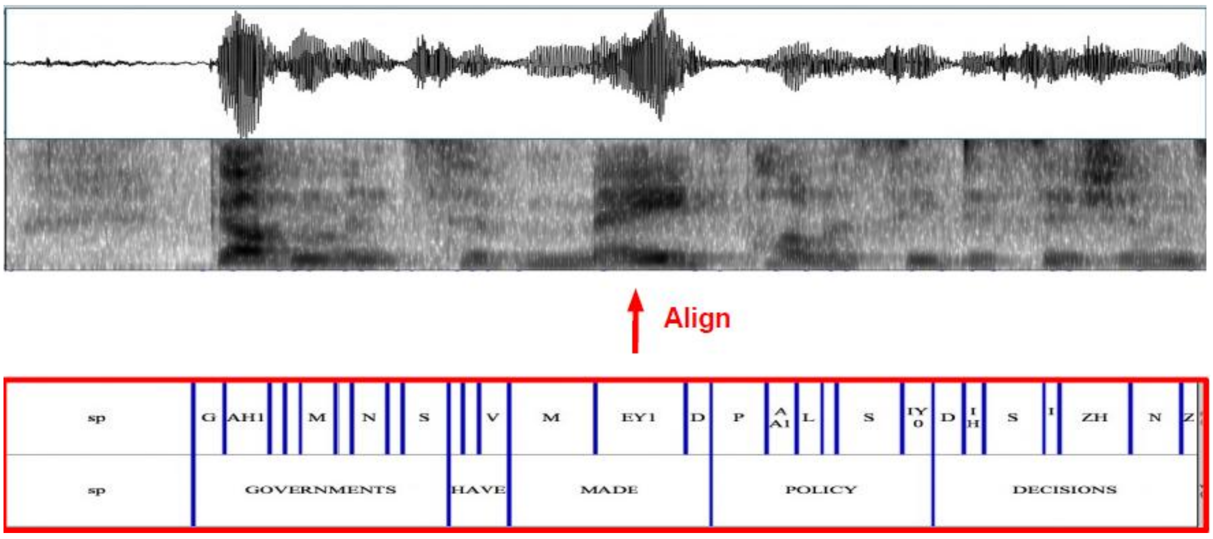
The main focus and time dedication will be towards the encircled region.

ASR technology (either PocketSphinx or Kaldi) will be used as a toolkit to perform Speech Recognition and Forced Alignment for lexical transcription or timestamps.

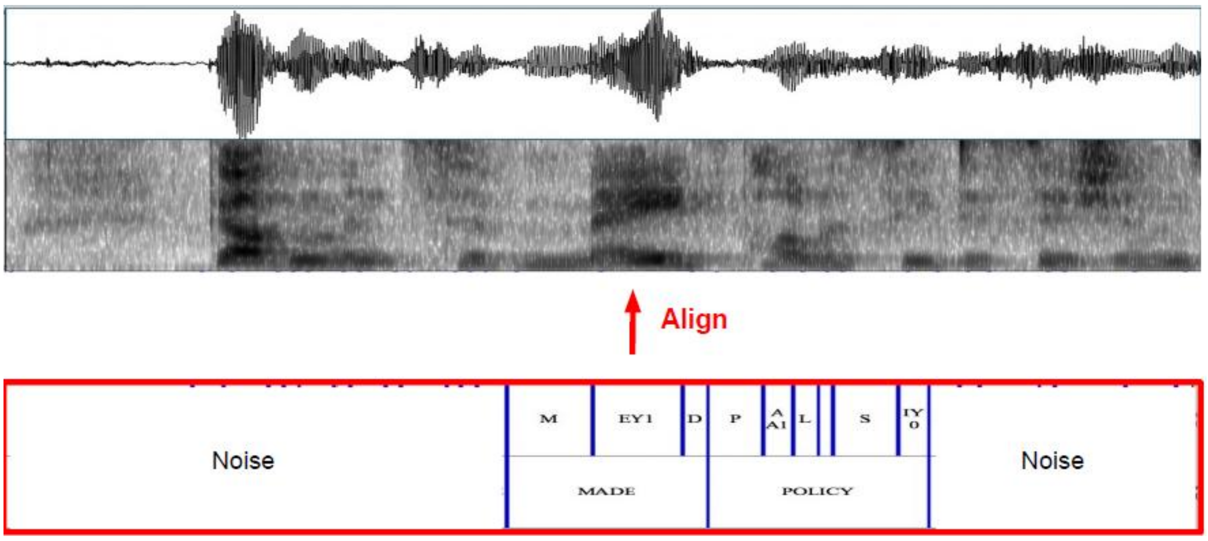
There are two primary cases that are possible during speech recognition :

1. Forced Alignment

- This is the case where the transcription is already available and we know what exactly is in the audio.



- This is the case where some transcription is available and we know what is in some of the audio.



This is the case which for which our tool needs ASR. While we have subtitles available sometimes they don't match audio 100%. The tool aims to provide a perfect audio-subtitle synchronisation for those words that do match. For those words in the audio that don't appear in the subtitles, add a different indicator or maybe the word itself (as will be depicted later in the second option).

E.g.

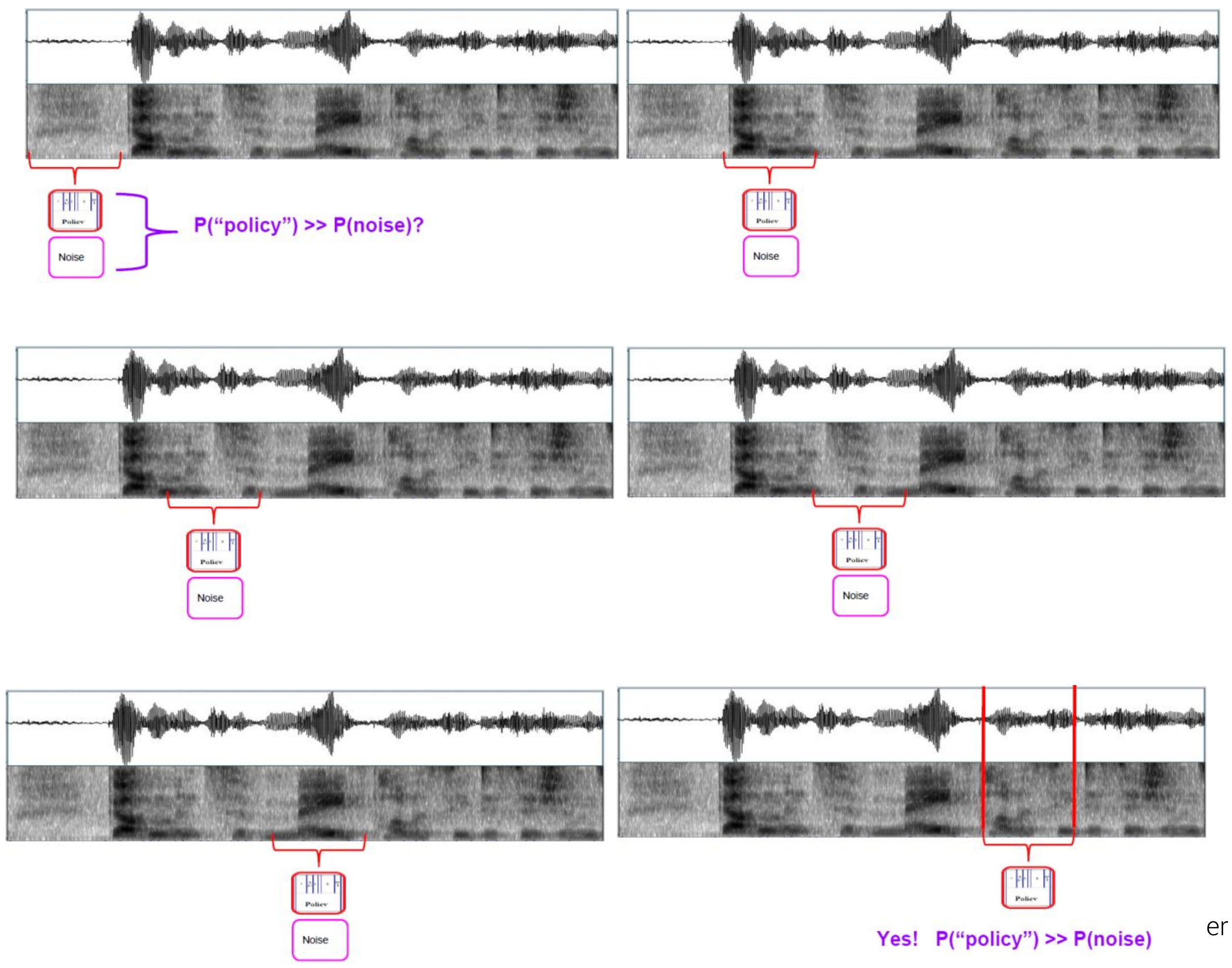
```
+ Added                         [10110:10180]

  there                         [11470:11580]

  are                           [11670:11710]

- missing
```

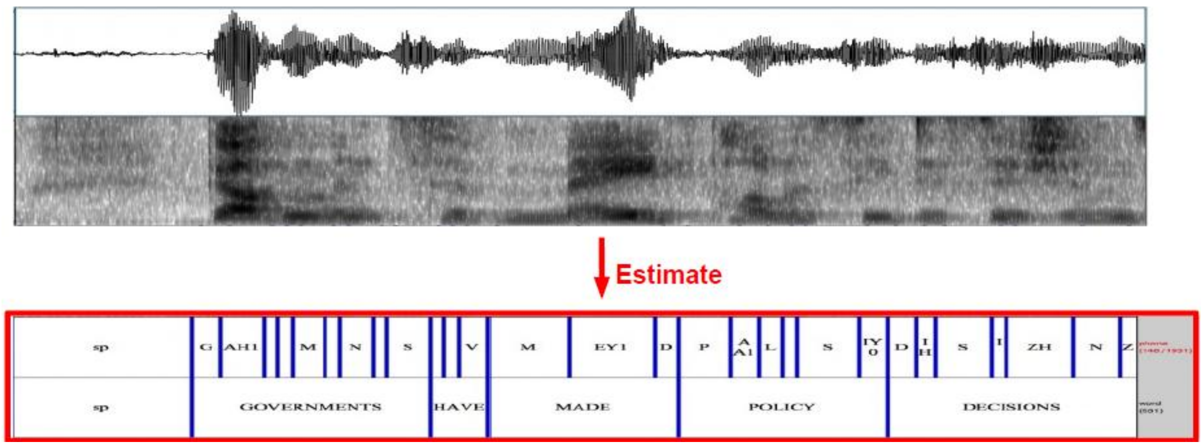
Now since we already have ordered list of words (obtained from subtitles), we can spot the word in audio as a means of possible transcription



er

2. Speech Recognition

- This is the case where the transcription is missing and we do not know what is in the audio.



The project in future can be extended to very well support this and automatically transcript a video without subtitles and in fact is one my post GSoC plans.

PROPOSED ARCHITECTURE

1. Approximation and Probability Based : While using this method, the tool will calculate timestamps of the words based on the probability of them appearing within a given timeframe. This can be better visualised from the demonstration below :

Consider a part of input subtitle :

347
00:34:15,220 --> 00:34:17,764
A young Jedi named Darth Vader...

This means that all the words [A, young, Jedi, named, Darth, Vader] are definitely falling within the timeframe of [00:34:15,220] and [00:34:17,764] inclusive. Thus, taking a broad assumption - if each word was pronounced for equal amount of time, the time taken by each word will simply be `( {end_timestamp} -{ start_timestamp} ) / {no_of_words}`

This of course will be erroneous because in real life each word takes different time to be spoken.

This can be improved by considering the probability of duration of time a word could have taken. For example one criteria could be the length of the word, higher the length more time it'll take to pronounce.

So, using lengths, calculating the percentage of time a word probably have taken and then assigning it the timestamp. I wrote the following to give a quick idea

```
    /* Basic word position approximation algorithm */
    initial_timestamp
    final_timestamp
    no_of_words
    total_length_of_sentence //except_space

    sentence_time = final_timestamp - initial_timestamp

    for 1 to no_of_words
        perc_length_of_current_word = length_of_current word / to-
tal_length_of_sentence_except_space
        current_word_time = perc_length_of_current_word * sentence_time
        current_word_beg_time = prev_word_timestamp_end
        current word_end_time = current_word_beg_time + current_word_time
        prev_word_timestamp_end = current word_end_time
```

This will still not be perfect, but will be a fast solution. This can later be combined using VAD (Voice Activity Detection) to produce even better and maybe close to perfect results.

With the help of VAD, we can locate the "empty" or non-voice zones and also find number of such zones within a given timeframe and then modify the above algorithm to incorporate those. This should result in comparatively higher accuracy.

2. Using phoneme detection : Both Kaldi and PocketSphinx are capable of generating phonemes based on audio provided a trained model and suitable dictionary are supplied. One of the tasks will be to figure out a method to quickly generate relevant dictionary and model based on

subtitles provided (since they are generally small) which should give more accuracy. The principle on which the working will be based is described in the previous section of this proposal [IMPLEMENTAION DETAILS : FORCED ALIGNMENT].

3. Using word detection : Another approach is to directly perform speech to text from audio and based on the probability of the word being detected (we will use the supplied subtitles for this), and create a timeline of words - thus getting word by word audio subtitle synchronisation. Frames from which words begin and end can be very well used to calculate starting and ending timestamps.

All the other details such as breaking audio into smaller chunks with the help of silence zones or VAD to process them more accurately, making the tool practically usable than just an academic tool et cetera are also kept in mind.

All the methods have their pros and cons and my aim will be to implement these with as much efficiency as possible. The project has wide scope. All these methodologies, with their different combinations can yield some super good results. I will try to make the tool as extensible and flexible as possible so that this is possible. I will continue to work on this even after GSoC and with more new minds (maybe from next GCI) and ideas hope to incorporate more better and new ideas into it.

BRIEF TIMELINE

- Phase 0 (Till 4th May) : Pre - GSoC period.
- Phase 1 (4th May - 30th May) : Community Bonding period.
- Phase 2 (30th May - 26th June) : Coding Phase 1.
- Phase 3 (26th June - 30th June) : Phase 1 Evaluations.
- Phase 4 (30th June - 24th July) : Coding Phase 2.
- Phase 5 (24th July - 28th July) : Phase 2 Evaluations.
- Phase 6 (28th July - 29th August) : Finishing, Testing, Final Documentation & Submission.
- Phase 7(29th August - 5th September) : Final Evaluation.

TENTATIVE PROJECT TIMELINE

Pre - GSoC Period (Till 5th April) :

Setting up the test - environment as required. Learning more about PocketSphinx , Kaldi, Gentle and testing each to understand output formats for given inputs. preparing and working on proposal. Simultaneously working on fixing bugs at CCExtractor. I will also try to achieve 15 points so that I can utilise the initial two weeks for project as well.

Pre - GSoC Period (6th April - 6th May) :

Unavailable due to tertiary mid semester exams , minor project submission and ultimately end - semester examinations in college. Here is college's official notice stating this : (http://ietdavv.edu.in/images/Notices/Revised_Academic_Calendar_Session_Dec-16_to_Apr2017.pdf).

Community Bonding Period (6th May - 27th May) :

In addition to creating accounts in all systems, updating all the relevant pages , setting up server & development environment and performing other necessities, I'll spend this time fine tuning the milestones of my deliverables and getting more clear about project implementation with the help of my mentors. I also want to spend this time in collecting and processing samples and create a repository for the same.

28th May - 31st May :

Begin designing the skeleton of the workflow. Also, come up with a design basic operation scheme and testing plan.

1st June - 7th June :

Create testing environment : Writing subtitle processing tool to convert subtitles into our required format and vice - versa i.e. remove styling tags, convert timestamps to milliseconds et cetera.

E.g.

```
00:05:35,936 --> 00:05:41,407
<font color="ffff00"> This has font style attached to it. </font>
```

In this we are only interested in the text and timestamps, not the style tags.

Also, writing a difference showing script to compare the outputs generated by various proposed architecture. I worked on improving the sample-platform's difference generator which already works for subtitles and I will port it to meet our needs. Simultaneously start building basic skeleton of tool.

8th June - 15th June :

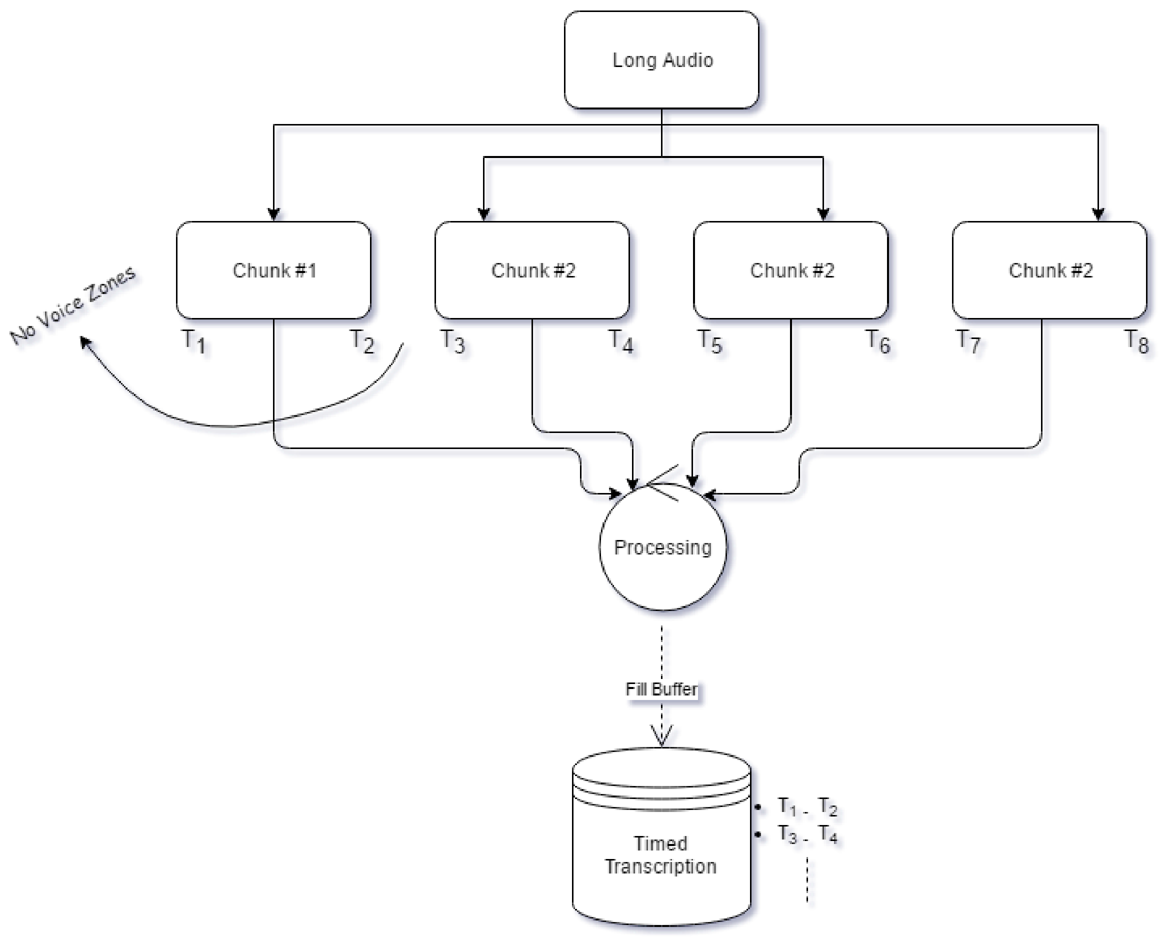
Write the implementation of proposed architecture 1 and test the results using the environment set-up. Make a report on its accuracy and document the usage case scenarios.

Work load has been kept less in the beginning two week so that I can practice other crucial techniques such as adding test cases and understanding code review work flow by mentors.

16th June - 23rd June :

Implement VAD using either WebRTC or PocketSphinx (currently WebRTC is preferred choice as it gives more reliable data).
Use VAD to make timestamps more accurate by ignoring the silence zone within a time span and then applying the synchronisation algorithm.

Also, if it increases speed, cut videos into smaller chunks based on VAD and silence regions. The small chunks of video will be processed separately, adaptive to their content and their results then later combined to form a single transcription. The different timestamps T1, T2, .. can be used as a placeholder, against which calculated offset (time in millisecond) will be added to find timestamp of each word.



Thus for example if we obtain a timestamp for word present in chunk #2, it's actual timestamp relative to the audio will be T_3 + obtained timestamp.

24th June - 29th June :

Complete pending work, fix bugs, look into ways of improving accuracy of both - the tool and VAD. This may imply better algorithm, faster string operations, better tuning of VAD. Also, prepare report for first evaluation. Test the output against sample repository and test environment.

PHASE 1 DELIVERABLES :

- Tool for subtitle processing and basic testing architecture
- Sample repository.
- Algorithmic and Probability based word - audio matching.
- VAD implementation.

29th June - 6th July :

Start full blown ASR work. Begin word detection, and write code for intelligently assigning timestamps on basis of frames and also probability. Try different setting and combinations to achieve maximum accuracy. The challenging part will be to incorporate missing and additional words present in subtitle. I will try to create a logic based on fuzzy search that shall look for words approximately ahead and behind the set domain.

```
static void print_word_times()
{
    int frame_rate = cmd_ln_int32_r(config, "-frate");
    ps_seg_t *iter = ps_seg_iter(ps, NULL);
    while (iter != NULL) {
        int32 sf, ef, pprob;
        float conf;

        ps_seg_frames(iter, &sf, &ef);
        pprob = ps_seg_prob(iter, NULL, NULL, NULL);
        conf = logmath_exp(ps_get_logmath(ps), pprob);
        printf("%s %.3f %.3f %f\n", ps_seg_word(iter), ((float)sf /
frame_rate),
               ((float) ef / frame_rate), conf);
        iter = ps_seg_next(iter);
    }
}
```

The above code is mirror to the option -time which is used in command line tool pocketsphinx_continuous and will be adapted for our use case.

7th July - 14th July :

Continue completing remaining work on word detection. Look into creating and fine tuning language models and dictionaries based on subtitles to restrict recognition to specific words. I plan to discuss and come up with the best plan for this after discussion in PocketSphinx forum or IRC.

Language Model Training and Adaptation :
[http://cmusphinx.sourceforge.net/wiki/tutoriallm]
[http://cmusphinx.sourceforge.net/wiki/tutorialadapt]

15th July - 21st July :

Fix bugs and complete missed milestones. Work on improving functionality, speed and accuracy.

Alongside, begin phoneme detection and tracing timestamps. A very good implementation of this is available in an open source tool Rhubarb Lip-Sync which uses this information to produce different mouth positions. I expect to receive some really good pointers for the same. I have had contact with the creator of that tool, and even contributed a small fix to that repository.

Phoneme Recognition :
[http://cmusphinx.sourceforge.net/wiki/phonemerecognition]

This will be done by aligning utterance and get phoneme timestamps. ps_alignment API will have to modified to implement this.

Also will use CMUDict and create custom dictionaries from the method figured out previously and begin forced alignment.

22nd July - 25th July :

Look into ways of improving the recognition (like segmentation, noise reduction) and apply them to both word based and phoneme based approach. Begin preparing for midterm evaluation.

26th July - 30th July :

Prepare report for midterm evaluations. Finish remaining documentation if any.

PHASE 2 DELIVERABLES :

- Word based timed transcription.
- Tuned language models and dictionaries.
- Adaptation script and implementation for custom models and dictionaries based on subtitles.
- Phoneme recognition.

1st August - 7th August :

Finish both phoneme and word based alignment and aim for auto-transcription. Work for the cases where the words are not present in subtitles and insert them(though it would be tried to do it alongside each step) and choose and implement appropriate marker to indicate the same. Work on various output formats such as XML, JSON, text to store and dump data in. Work on missing documentation and missed milestones.

8th August - 15th August :

Clean up the code. Fix bugs, memory leaks. Improve speed and work on cross platform compatibility. Include several unit and build tests and catch up on any milestones if missed.

15th August - 21st August:

Buffer week in case of some work getting delayed, in case of any emergency or otherwise. The time saved here will be distributed as required. Additionally, devote this time to take feedback from the community and iterate on the designs and improvise on use cases. Ensure code quality by adding more test cases and working with more videos.

PHASE 3 DELIVERABLES :

- Tool capable of word by word audio - subtitle synchronisation.
- Complete project documentation.
- Reports containing comparison of applied techniques and some benchmarks.

21st August - 29th August

Finalisation and completion of project. Submit code, project summaries, and final evaluations of mentors. I also am inspired by ex CCExtractor GSoC students and like them, I would also like to write a detailed technical report (like a research paper) describing the specific work which I have done over the summer and the various techniques which I have used to achieve my objective.

SOME NOTES REGARDING TIMELINE :

1. I haven't dedicated specific weeks for documentation as I believe documentation is a part of coding itself. I'll try to keep my code as documented as possible while writing the code itself. Also if I miss documentation somewhere, it will be covered as mentioned in timeline.

2. I will be writing weekly blog posts about the milestone at my blog, address of which is mentioned in the later sections of the proposal.

3. I realise that open source is all about communication and collaboration and I like it that way. Hence, I will be in continuous contact with mentor throughout the program describing him the progress and problems I find.

4. The timeline above is tentative. I will prepare a detailed and more fine checklist, which I will host as a Github gist on the above mentioned blog after or during community bonding period. Thus, in this way checklist will be available to everyone with easy accessibility. It will also become easier for mentor to add task directly to that checklist and asses them.
5. I have mentioned my working hour/ environment / availability et cetera later in the proposal.

6. During pre-GSoC period, as per my plan - I will learn more about the used systems such PocketSphinx and hence may incorporate/change existing plan. I will be prompt to report everything to my mentor.

7. If everything is on schedule or I complete some milestone early, I can proceed to the next task ahead of time. If working ahead of time makes my project finish earlier than expected, then I will move on to doing one of the spare tasks which I have listed.

TENTATIVE SPARE TASKS :

In the case that I am able to finish my project before time, I will use the remaining time to work on a spare task of my interest. Also, if my project is not selected, I am comfortable working on one of these projects. My ideas for this spare task are :

1. Sample-Platform : I absolutely love working on sample-platform and in fact that is where I began contributing in the first place. I am already a little bit familiar with its codebase and I would love to complete the tasks laying in its issue tracker and add more functionality.

2. Creating Cross Platform Integrated GUI : While the command line version is super easy to use, thanks to all the contributors, download statistics clearly show that the GUI version is downloaded the most. We currently have GUIs separate for each platform and it makes sense to make single cross platform integrated GUI. Not only will it be easy to maintain, it will become easier to make user experience same and amazing for all users. I have done a lot of work on front end and am quite familiar with designing clean and minimal UI. I recently also made changes to already existing windows GUI, and will be happy to do this task.

3. DTMB Support : I have become quite familiar with CCExtractor codebase and general flow of work, and have relatable comfort in reading specs and implementing them. Not much is known about this, but good research has been done by the team to lay foundation of its support. I would love to begin implementing and discover things on the go.

REQUIREMENTS

1. High Speed Server : While I have a good working development environment available to me, the samples on which we work are very large in size. Plus, working on a high speed server drastically improves efficiency. I do my most of the work on CCExtractor's gsocdev3 server, and it is definitely more productive. Additionally it will be a good choice to train language models.

2. Large number of sample : Samples containing audio and it's transcript. I will try to create a good repository during community bonding period.

Any more requirement would be conveyed during pre-GSoC period while fine tuning the milestones.

PERSONAL INFORMATION

Planned Work Schedule, Environment and Absences :

- I intend to work on all weekdays (Monday to Friday) and a bit on weekends too, and will easily able to work 40+ hours a week.
- The checklist on the blog will cover all weekly milestones , and they shall be called complete on the day when I update my progress on the developer blog.
- My local time zone throughout the summer will be IST (Indian Standard Time) which is GMT + 05:30.
- As far as the times of the day are concerned, I am pretty flexible, and am willing to adjust with what suits my mentor the best. I have already been working with them since past 4 month.

I will be working from either my home or college hostel. I have access to a stable internet connection at both places along with an active 4G connection. Internet outages are rare, although sometimes I experience momentary disconnections for a few minutes or so but the 4G should suffice at those times.

I have no planned absences over the summer, but in case any arises - I'll be prompt to inform my mentor about the same and use the buffer week to catch up. I shall be available to work full time with CCExtractor throughout the period of GSoC. My summer vacation will end near July end, maybe August (the official dates aren't announced yet). For the last weeks of the program, I will also be attending college, but the burden is minimum in initial weeks, so I shall be able to dedicate enough time for project.

Why CCExtractor Community?

The reason is simple - I absolutely love CCExtractor community. While I have been on Github since around last year, I had not made any "real world" contribution until I came across CCExtractor. I was way too hesitant to even strike a conversation and now thanks to CCExtractor I actively participate in discussions, bug fixes, PR reviews. I have learnt a lot in these 4 months.

The community is extremely helping - any query's resolution is just a slack message / email away. I was even given access to a high speed server with my own account to ease my work. Most of the past GSoC students are still actively involved and now serving as mentor, even the GCI winners are mentoring illustrating how appreciated is everyone in the community. I have also had an incredible opportunity to learn a lot from one of the CCExtractor's GSoC veteran and it was super fun to work with GCI students (whom now I've made great friends with). I definitely feel I am a part of this community and I am thankful to it.

In future, I would definitely love to remain and be part of CCExtractor. CCExtractor community is a hidden gem not many know about it. One of my post GSoC plan which I have mentioned later in this proposal is related to this.

I cannot think of a better platform than CCExtractor to spend my summer with. Not to forget that this project is of my upmost interest, plus who to better implement it with than the experts who made the de-facto subtitle tool.

BUG FIXES AND CONTRIBUTION

I feel proud to be one of the active contributors in CCExtractor (currently #7 according to Github Graphs). The following contributions are listed in chronological order.

Contributions to main CCExtractor repository

- [MERGED] Cleaning and replacing redundant code with already existing functions. #714

- [MERGED] Fixes for teletext in .bin files (-unixts param, proper return code, -ucla param). #700

- [MERGED] Fixed missing tpage number in UCLA from BIN. #694
- [MERGED] Changes regarding output file name and minor print_usage corrections. #679
- [MERGED] Ability to extract Chapters from MP4 files. #655
- [MERGED] Prevent usage of stdin with MP4 files. #635
- [MERGED] SMPTE-TT : Removed appearance of garbage value in color code. #625
- [MERGED] SMPTE-TT : Added support for font color. #590
- [MERGED] Adobe Premiere Pro Compatibility. #582
- [MERGED] Filename corrections. #539
- [MERGED] Printing exact error for clean file. #533
- [MERGED] Fixed missing directory in Makefile for linux. #532
- [CLOSED] Completed TODO: Get from framerates_values[]. #495
- [MERGED] Added some dictionaries. #490
- [MERGED] Minor correction in README.md. #488

All my commits to main CCExtractor repository may be viewed at : [https://github.com/CCExtractor/ccextractor/commits?author=saurabhshri]

Contributions to Sample Platform repository

- [MERGED] Fix the result status of individual tests. #42

- [MERGED] For non-zero return code, ignore checking output file. #38
- [MERGED] Only considering checking for file where rc is 0. #37
- [MERGED] Installation instructions added. #36
- [MERGED] Added missing dependencies and minor directory name fixes. #32
- [MERGED] Proper handling of html tags while displaying. #30
- [MERGED] Fixed creation of blank table divisions. #29
- [MERGED] Improvements to the diff generator. #28
- [MERGED] Minor build script fixes. #14
- [MERGED] Finished TODOs in the Installer script. #11
- [MERGED] Corrected timezone handling. #8
- [MERGED] Display Timezone in Sample Platform. #7
- [CLOSED] Corrected SampleNotFoundException. #5

All my commits to Sample-Platform repository may be viewed at : [https://github.com/canihavesomecoffee/sample-platform/commits?author=saurabhshri]

Contributions to CCExtractor's Windows GUI repository

- [MERGED] Drop a folder into the Input Files to process complete Dir hierarchy. #2

All my commits to CCExtractor's Windows GUI repository may be viewed at :
[https://github.com/CCExtractor/CCExtractorWindowsGUI/commits?author=saurabhshri]

I also helped in finding out some bugs :

- Reported issues in CCExtractor repository :
  [https://github.com/CCExtractor/ccextractor/issues?utf8=%E2%9C%93&q=is%3Aissue%20author%3Asaurabhshri%20]

- Reported issues in Sample-Platform repository :
  [https://github.com/canihavesomecoffee/sample-platform/issues?utf8=%E2%9C%93&q=is%3Aissue%20author%3Asaurabhshri%20]

My other contributions may be viewed at my Github profile listed later in the proposal.

I have **successfully earned 7+ GSoC points**, deeming me qualified for applying (the actual point counts will be evaluated later, hence I am not mentioning the exact count).

Personal Details

I am Saurabh Shrivastava, a 3rd year Information Technology Engineering undergrad at IET DAVV, Indore, India. My interests lie in the field of computer science. I was introduced to programming in 11th standard and scored a perfect 100 in my 12th Computer Science Board Examinations. I love programming in C and C++ and am constantly learning new things, python being the latest addition to club. I am good in data structures and algorithm and try to write clean, documented and properly tested code.

I am comfortable using Linux and Windows, working with Debuggers and IDE, and version control using git. My past work with community should reflect the same.

I recently got introduced to the world of open source and I haven't looked back since. I absolutely love the idea of open source software and am thoroughly enjoying it. I am trying my part to contribute as I learn. It feels amazing to know that my contribution is actually out there and people are using it. I also started competitive coding at CodeChef but haven't been active since sometime. I have participated and qualified in many coding competitions including ACM-ICPC and Facebook Hackercup.

Accounts and Contact

- I am available on Email, Hangouts and Google Plus under address : *saurabh.shrivastava54@gmail.com* .
- I am available on Call, SMS, WhatsApp, Allo, Duo with number : *+918871468353* .
- My alternate phone number is *+918770069479* .
- My Github handle is *@saurabhshri* and link to profile is *https://github.com/saurabhshri* .
- I hangout on Slack and IRC under nickname *saurabhshri* .
- My personal blog resides at *https://*saurabhshri.github.io .
- My Twitter handle is *@shubh_saurabh* and CodeChef handle is *@shubhshri* .

Other GSoC Organisations

I have not applied to any other organisations except CCExtractor. This is also my first GSoC.

Post GSoC Plans

In addition to continuing working and improving on the project (like completing the pipeline, add more language support , stream support et cetera) , I would also love to keep contributing to CCExtractor and other open source projects and learn as I grow.

I realise that CCExtractor is a small and modest organisation. While there might not be million CCExtractor users, but there are millions of people who use CCExtractor's output. Almost all the subtitles files available on internet to download are courtesy to CCExtractor. I plan on bringing CCExtractor to light and make people aware about it.

I will begin by writing a manual for step by step usage of CCExtractor in regard to all its feature. This will ensure that people actually find and come across CCExtractor. I will also work on spreading the word by writing Quora answers, making blog posts on platforms such as Medium. The main part of this improvement is the redesign of website. The information in the current website is extremely scattered. There is a need to reorganise everything and add more information. Maybe next GCI we can build a new website altogether.

I would also like to add Windows support on Sample - Platform and implement the features which are currently missing.

I will definitely try to make my juniors and batch mates in college aware about open source and encourage them to start contributing as well.

Reference :

- The flowcharts and figures are drawn by me with the help of *www.draw.io*

- Almost all the remaining images are the courtesy of *Forced Alignment and Speech Recognition Systems, Oxford University.*

I am no speech recognition or software developing expert, but I am a firm follower of what my mentor says - always follow *the 50%-50% rule... 50% of the required work you should know how to do or at least how to approach, the other 50% you should have no idea about but be really eager to learn.* I have carefully planned my work and I am more than just eager to learn.