Mini Project

Guide - Prof. Ankit Chaudhary

- Akshay Gidwani
- Abhilash Pani

Automatic Subtitle generation

Why this topic ?

- Past decade has witnessed huge surge in audio and visual data
- Hearing impaired people missing out the important leisure activities.
- Utilised by farmers for listening the news.
- Regional language - speaking people.
- Utilising available technologies/utilities and making them available to people in need.

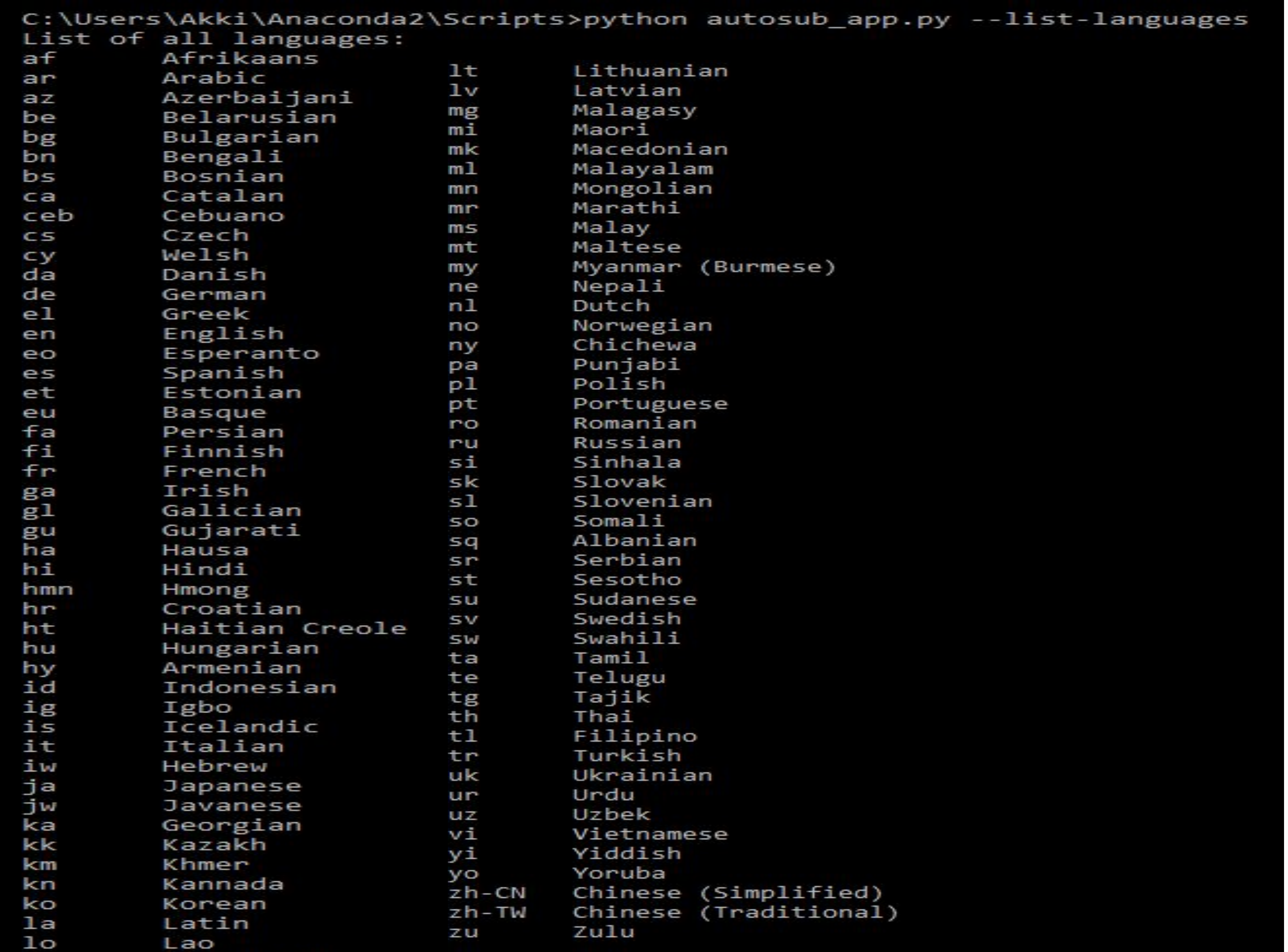
Previous Work

- Audio is extracted from a video sample.
- Audio is further clipped into various segments for effective subtitle generation
- Subtitles (timed-texts) are generated for audio clips using AutoSub.
- Autosub is a utility for automatic speech recognition and subtitle generation.
- It takes a video or an audio file as input, performs voice activity detection to find speech regions, makes parallel requests to Google Web Speech API to generate transcriptions for those regions.

Languages

Following languages can be converted from speech to text using the Autosub API.

NOTE: The languages can not be translated from one to other. Only the source language can be converted to text in the script of the same language.

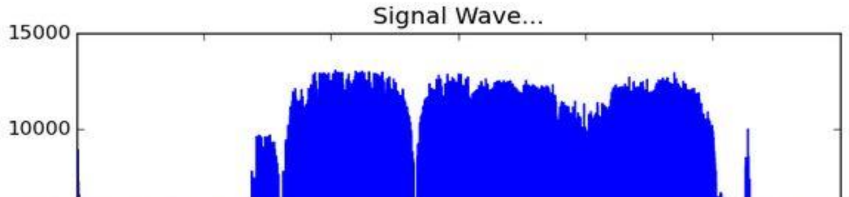


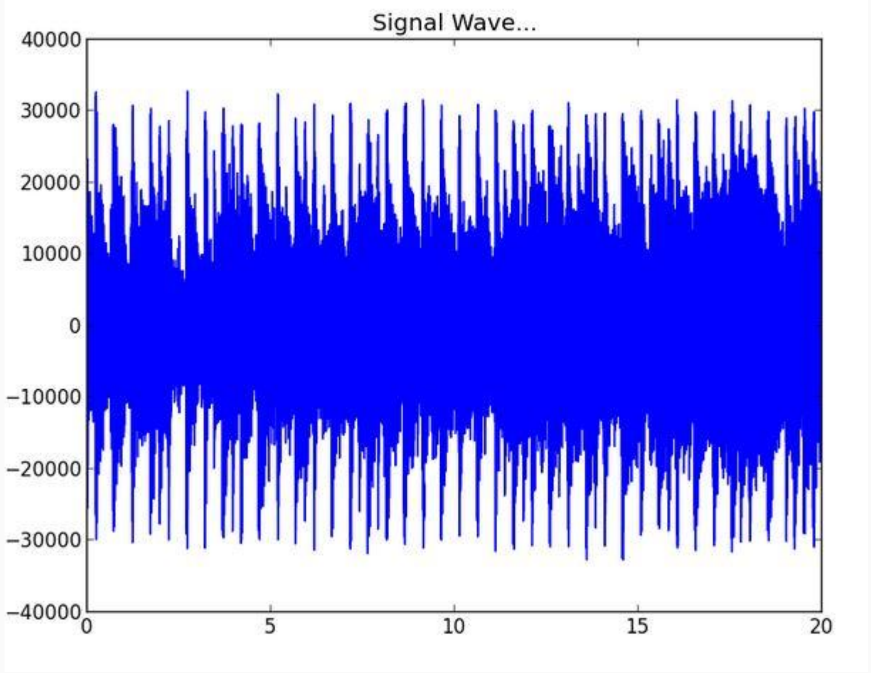
```
C:\Users\Akki\Anaconda2\Scripts>python autosub_app.py --list-languages
List of all languages:
af      Afrikaans        lt      Lithuanian
ar      Arabic           lv      Latvian
az      Azerbaijani      mg      Malagasy
be      Belarusian       mi      Maori
bg      Bulgarian        mk      Macedonian
bn      Bengali          ml      Malayalam
bs      Bosnian          mn      Mongolian
ca      Catalan          mr      Marathi
ceb     Cebuano          ms      Malay
cs      Czech            mt      Maltese
cy      Welsh            my      Myanmar (Burmese)
da      Danish           ne      Nepali
de      German           nl      Dutch
el      Greek            no      Norwegian
en      English          ny      Chichewa
eo      Esperanto        pa      Punjabi
es      Spanish          pl      Polish
et      Estonian         pt      Portuguese
eu      Basque           ro      Romanian
fa      Persian          ru      Russian
fi      Finnish          si      Sinhala
fr      French           sk      Slovak
ga      Irish            sl      Slovenian
gl      Galician         so      Somali
gu      Gujarati         sq      Albanian
ha      Hausa            sr      Serbian
hi      Hindi            st      Sesotho
hmn     Hmong            su      Sudanese
hr      Croatian         sv      Swedish
ht      Haitian Creole   sw      Swahili
hu      Hungarian        ta      Tamil
hy      Armenian         te      Telugu
id      Indonesian       tg      Tajik
ig      Igbo             th      Thai
is      Icelandic        tl      Filipino
it      Italian          tr      Turkish
iw      Hebrew           uk      Ukrainian
ja      Japanese         ur      Urdu
jw      Javanese         uz      Uzbek
ka      Georgian         vi      Vietnamese
kk      Kazakh           yi      Yiddish
km      Khmer            yo      Yoruba
kn      Kannada          zh-CN   Chinese (Simplified)
ko      Korean           zh-TW   Chinese (Traditional)
la      Latin            zu      Zulu
lo      Lao
```

Visualising audio using python

```
import matplotlib.pyplot as plt
import numpy as np
import wave
import sys


spf = wave.open('wavfile.wav','r')

#Extract Raw Audio from Wav File
signal = spf.readframes(-1)
signal = np.fromstring(signal, 'Int16')


#If Stereo
if spf.getnchannels() == 2:
    print 'Just mono files'
    sys.exit(0)

plt.figure(1)
plt.title('Signal Wave...')
plt.plot(signal)
plt.show()
```

you will have something like:



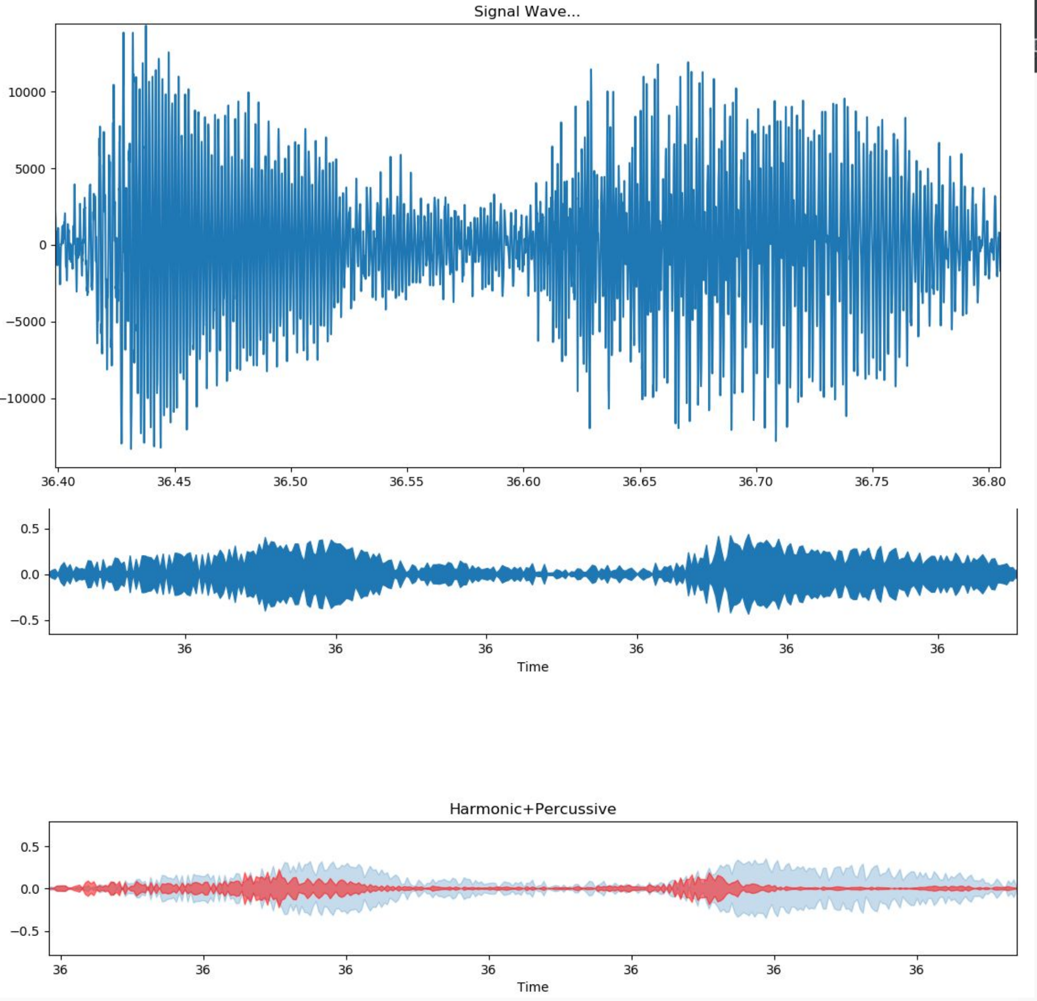


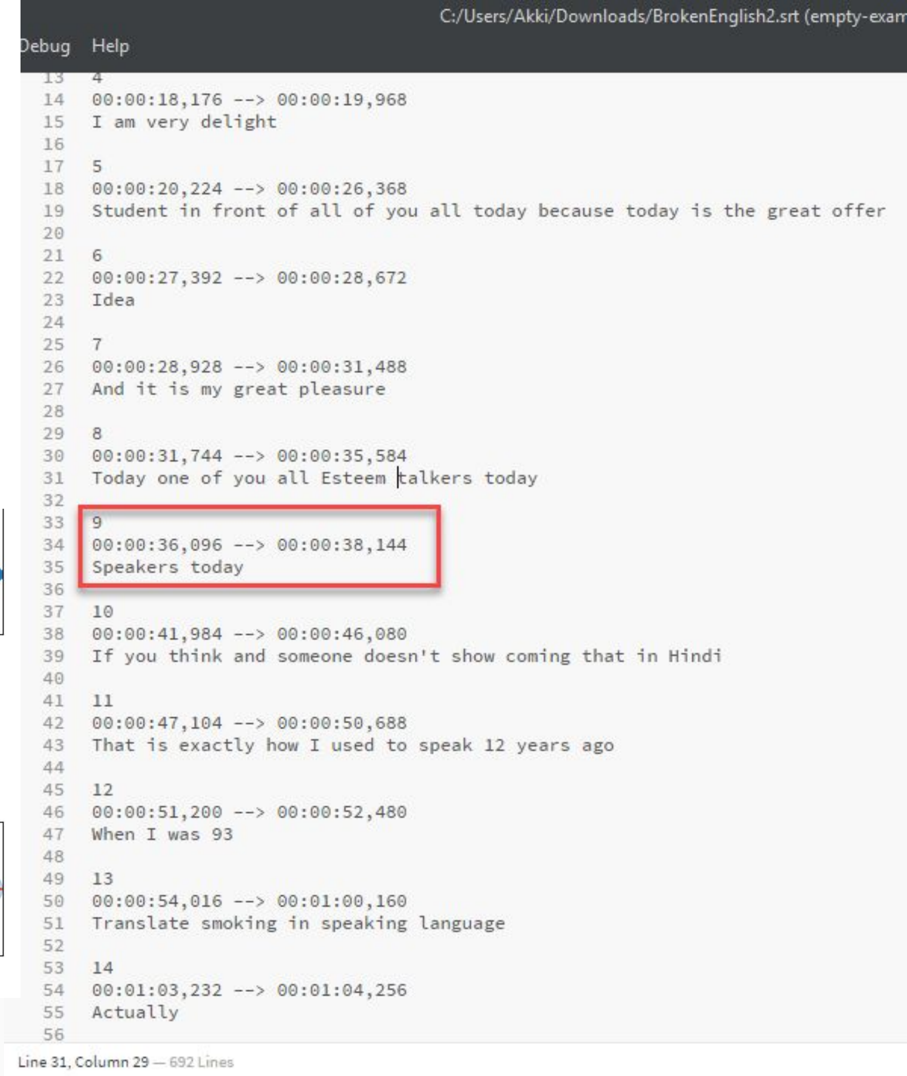
Output comparison at different bitrates

Output for one file “BrokenEnglish.mp4” was compared at varying bitrates.

It was observed that the o/p at faster bitrate(705 kbps) was inaccurate as compared to that with the slower one(192 kbps).

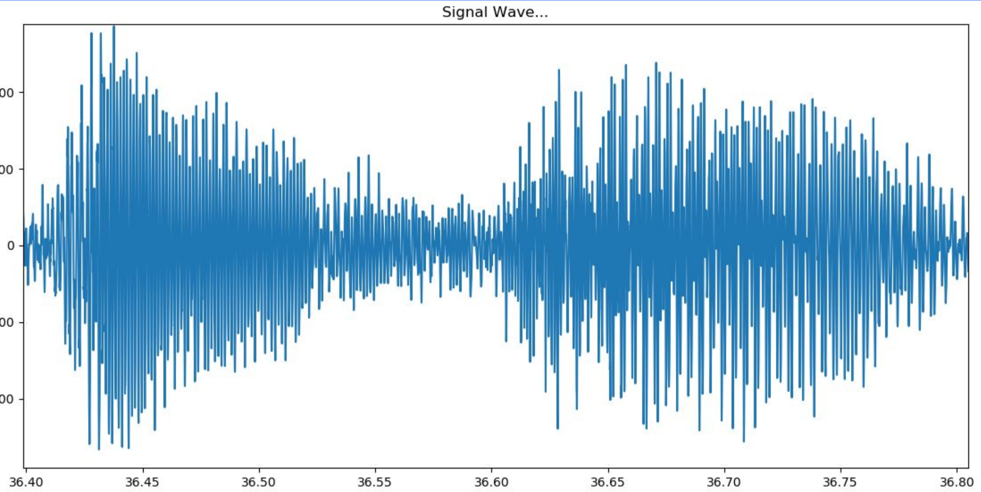
Following are the signal and output at 705 kbps and 192 kbps, respectively:

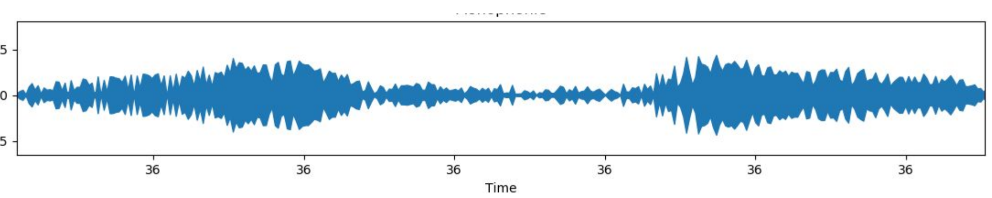


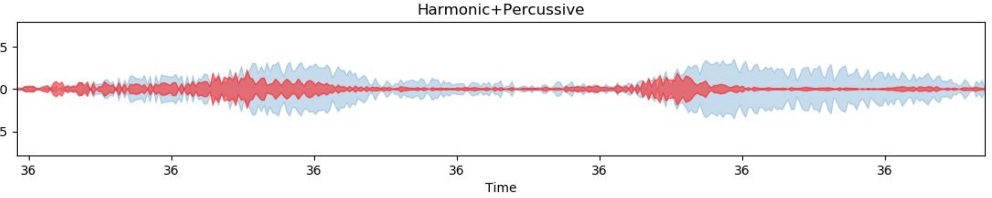


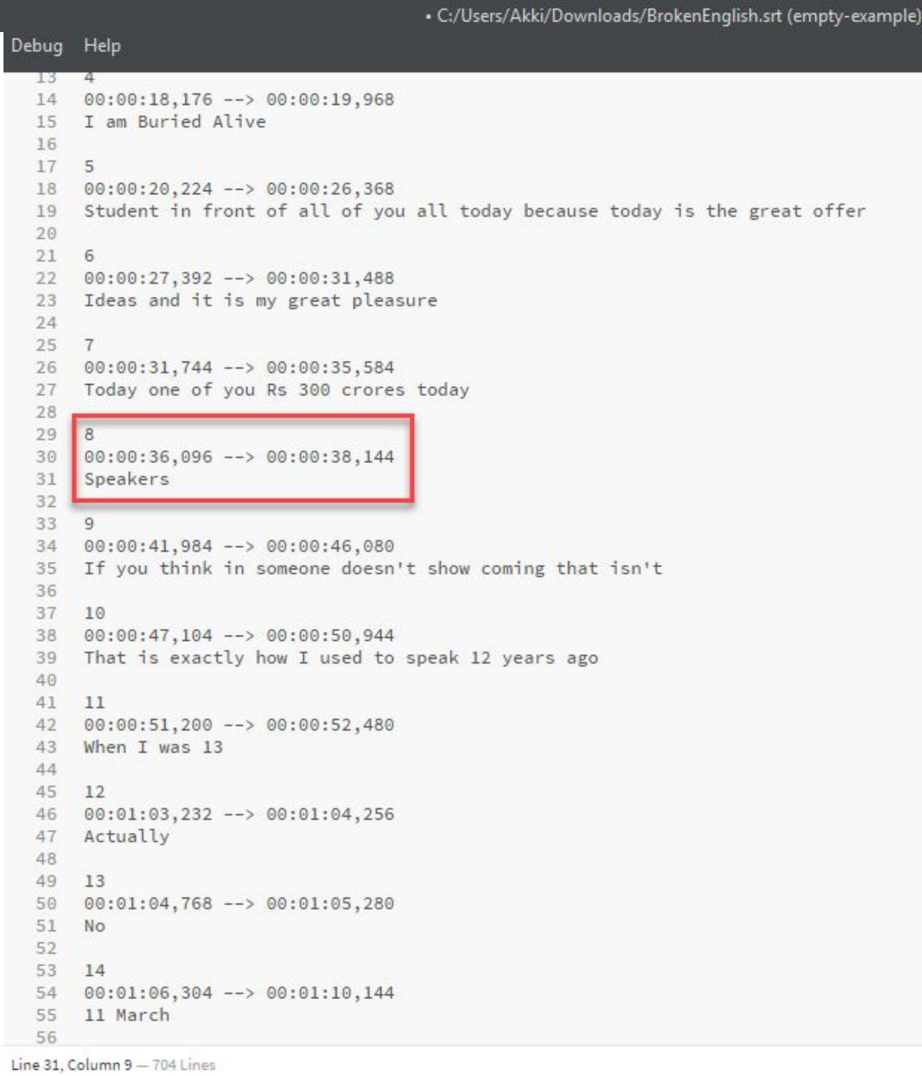
```
4
00:00:18,176 --> 00:00:19,968
I am very delight

5
00:00:20,224 --> 00:00:26,368
Student in front of all of you all today because today is the great offer

6
00:00:27,392 --> 00:00:28,672
Idea

7
00:00:28,928 --> 00:00:31,488
And it is my great pleasure

8
00:00:31,744 --> 00:00:35,584
Today one of you all Esteem talkers today

9
00:00:36,096 --> 00:00:38,144
Speakers today

10
00:00:41,984 --> 00:00:46,080
If you think and someone doesn't show coming that in Hindi

11
00:00:47,104 --> 00:00:50,688
That is exactly how I used to speak 12 years ago

12
00:00:51,200 --> 00:00:52,480
When I was 93

13
00:00:54,016 --> 00:01:00,160
Translate smoking in speaking language

14
00:01:03,232 --> 00:01:04,256
Actually
```

Line 31, Column 29 — 692 Lines









```
4
00:00:18,176 --> 00:00:19,968
I am Buried Alive

5
00:00:20,224 --> 00:00:26,368
Student in front of all of you all today because today is the great offer

6
00:00:27,392 --> 00:00:31,488
Ideas and it is my great pleasure

7
00:00:31,744 --> 00:00:35,584
Today one of you Rs 300 crores today

8
00:00:36,096 --> 00:00:38,144
Speakers

9
00:00:41,984 --> 00:00:46,080
If you think in someone doesn't show coming that isn't

10
00:00:47,104 --> 00:00:50,944
That is exactly how I used to speak 12 years ago

11
00:00:51,200 --> 00:00:52,480
When I was 13

12
00:01:03,232 --> 00:01:04,256
Actually

13
00:01:04,768 --> 00:01:05,280
No

14
00:01:06,304 --> 00:01:10,144
11 March
```

Further implementation

- The future objective was to embed this subtitle generation inside a video player which was successful . (Python-VLC media player)
- Further implementation included emotion annotation of the obtained subtitles and labelling them into pre-defined set of emotions.
- For now, instead of implementing on generated subtitles, we did it on web-scraped movie subtitles as they were easily tested with original results.

Approach used in brief -

<u>Subtitle generation and a video player</u> -

- Developed a video player(works as a GUI) using python-vlc module and python-gi module.
- Generated subtitles from autosub utility of python which uses Google API for speech recognition and returns timed-texts.
- The video player is embedded with play/pause, stop functionalities and Hindi & English Subtitle generating buttons.

Resources used

Dependencies for subtitle generation and the video player are -

- Google API Client and a paid API key for translation into different languages
- Autosub utility
- VLC Media Player and Python-vlc module
- Python-gi repository for GUI building of video player
- Ubuntu-based Linux Systems

Paper used as resource

- Whose line is it anyway? “Automatic Multilingual Emotion Annotation of Movie Dialogue by Wojciech Stokowiec
- RankLyrics by Shuo Zhang: A Ranking-based System for generating Song Lyrics
- Subtitling and dubbing songs in musical films - Martha Garcia (Spanish paper) discussing about constraints varying in different regions.
- Options and Choices in Musical Film Lyrics Translation - Masaryk University (Post - Diploma Thesis)

Approach used in brief -

Emotion Annotation for generated movie subtitle -

- BabelNet API was used for semantic tree generation for 8 emotional states like (love, hate, anger, sad etc.)
- Semantic tree is similar to decision tree and hence we obtained 8 different trees.
- Subtitles transformed to vector representations.
- Trees constructed for group of 10 sentences acc. to closest distance of vector to the 8 semantic tree root node obtained.

Resources used

Dependencies used for emotion annotation of movie subtitles -

- Pre-trained word vector Google model.
- BabelNet API
- BabelNet Semantic Tree stored as a pickle (python object)
- Gensim
- Closest distance calculation of word vectors to root nodes of semantic trees
- Plotting and comparison tables for testing the precision of the proposed model.

Visuals-

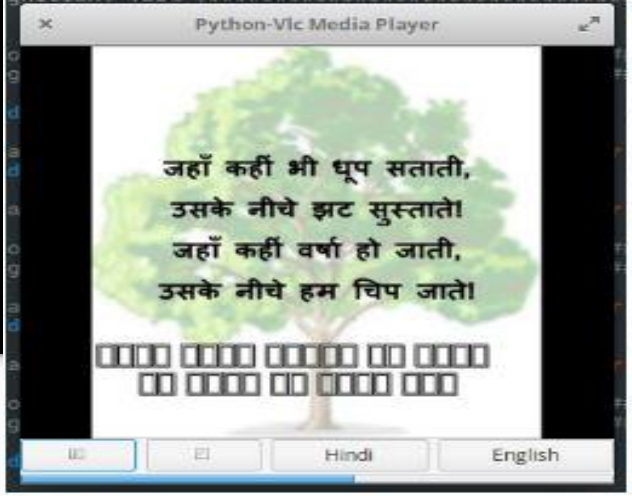


Fig. 3 Screenshot of the Hindi Subtitles on Media Player

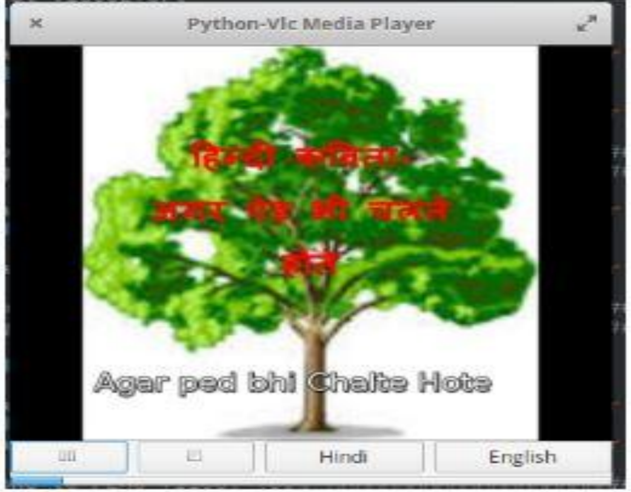
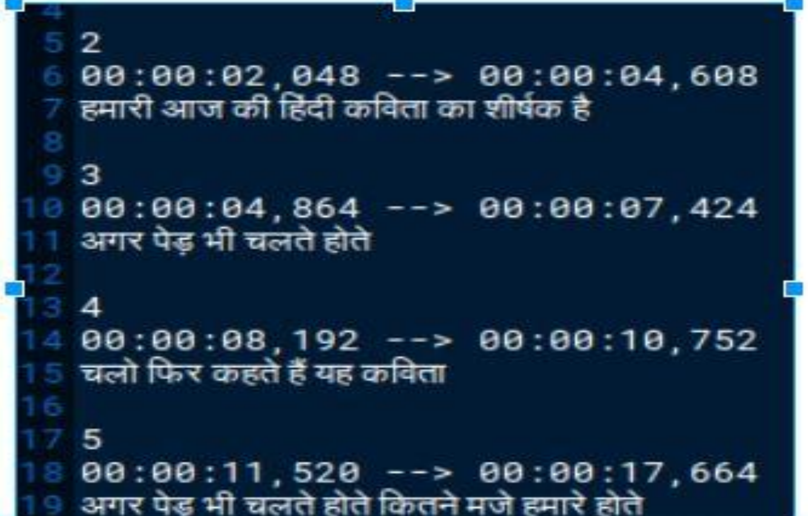
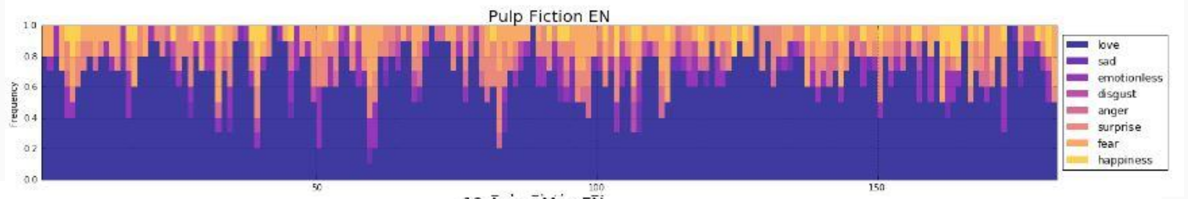


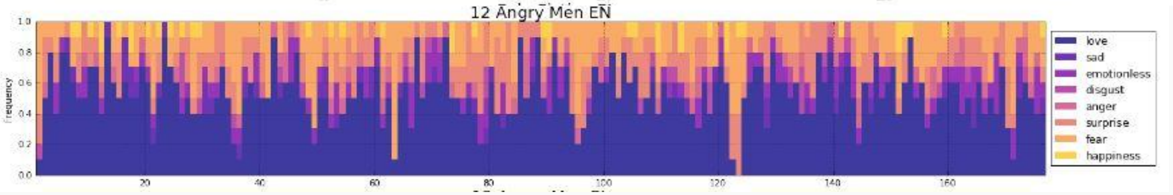
Fig. 4 Screenshot of the English Subtitles on Media Player

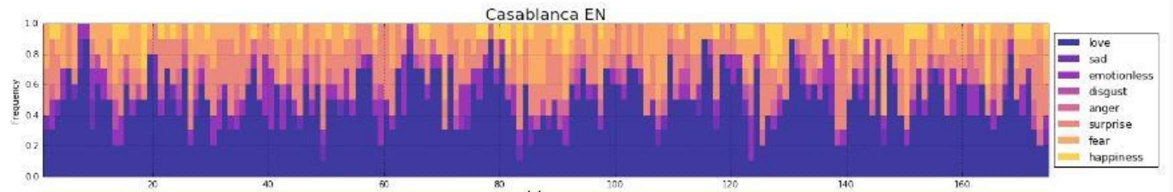


```
2
00:00:02,048 --> 00:00:04,608
हमारी आज की हिंदी कविता का शीर्षक है

3
00:00:04,864 --> 00:00:07,424
अगर पेड़ भी चलते होते

4
00:00:08,192 --> 00:00:10,752
चलो फिर कहते हैं यह कविता

5
00:00:11,520 --> 00:00:17,664
अगर पेड़ भी चलते होते कितने मजे हमारे होते
```

Visuals -







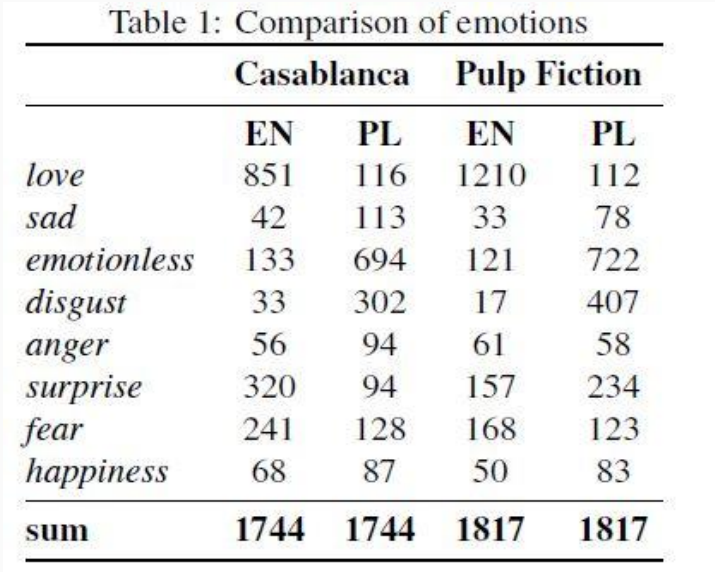


Table 1: Comparison of emotions

	Casablanca		Pulp Fiction	
	EN	**PL**	**EN**	**PL**
love	851	116	1210	112
sad	42	113	33	78
emotionless	133	694	121	722
disgust	33	302	17	407
anger	56	94	61	58
surprise	320	94	157	234
fear	241	128	168	123
happiness	68	87	50	83
sum	**1744**	**1744**	**1817**	**1817**

Problems faced -

While implementation of the project, we faced numerous problems such as -

- Maintaining high accuracy of generated subtitles.(less for languages other than english, hindi).
- Encoding issues for hindi subtitles in the video player.
- Parsing of one file arguments to other.
- Training word-vector model required high maintenance.
- Safety of storage of obtained semantic trees.
- Validating results for generated subtitles emotion annotation.

Future Objective

- Generation in real-time.
- Menu-driven package.
- Further improvement of transcription of dialogues, sound effects and other material audio information.

Thanks!