# **Automatic Speech-text Synchronization**

# **Using Speech Recognition**

# **and Text-to-Speech Technology**

Chun-Han Lai, Renyuan Lyu, ,Chang Gung University ,Taiwan.

## Abstract

Using Python to package HTK(Hidden Markov Toolkit) to create a speech-text synchronization file, named timed-text file, with Cloud Text-to-speech and Speech Recognition. This timed-text file can be used to language learning which is called Computer-assisted Language Learning(CALL). One of language learning technique is Shadowing. Shadowing technique is that, while listening, you attempt to repeat – to “shadow” what you hear as quickly as you hear it. In order to easily Shadowing, we must get word by word timed-text file from HTK. To browse this timed-text file, we use JavaScript to build a simple website. This website can read timed-text file, and present a word by word speech-text synchronization. In addition, which provide checking dictionary online by Google Translate.

## Description

Using Cloud TTS (Text-to-speech) Technology, such as Google Translate, iSpeech ... etc, and simple Speech Recognition Technology, can be made to establish a method to create a Speech-text Synchronization file from original Text-only file.

That can be used for Computer-assisted Language Learning(CALL) to do a language learning skill, named "Shadowing".

## Research Motivation

With the advent of the Global Village trend, with a variety of language ability to be an indicator of competitiveness, especially the listening and speaking of expression ability are more important.

Digital textbooks under the computer and the Internet generally and widely used, and this trend makes Computer-assisted Language Learning(CALL) be a hot topic in recent years.

## Related Work

### Timed-text File :

Python have excellent string processing ability, so we can use Python to make timed-text file be converted between various formats. In generally, there are more popular timed-text fomat, like .lrc file, .srt file…etc. Such as these timed-text formats, often are used on song lyrics, movie subtitle.

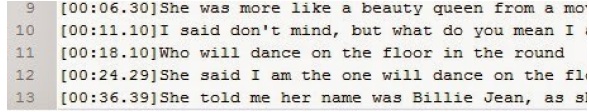


Figure 1. Lrc File Format

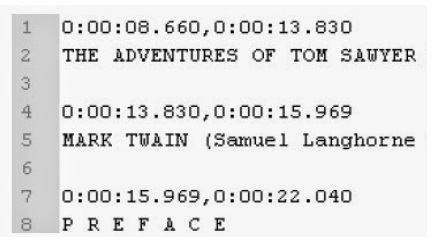


Figure 2. Sbv File Format

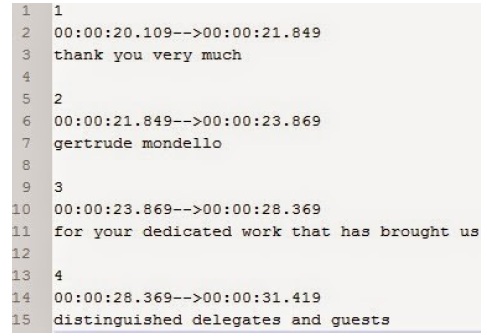


Figure 3. Srt File Format

### Google Translate –TTS :

Google Translate has a TTS(Text-to-speech) service. That makes we can easily get a speech from Google Translate by Google’s URL.

Here, we use Python built-in library, ‘urllib.request’ and ‘urllib.parse’, to communicate with Google Translate. That can send HTTP Request to Google Translate. The URL of Google Translate TTS is that -- <http://translate.google.com/translate_tts>

To use this URL, must have its URL parameters.

|  |  |
| --- | --- |
| parameters | Meaning |
| tl | Target Language, The language you want to convert |
| q | Query, Query you want to Text to Speech |
| total | Total number of text segments. |
| idx | Index of text segments. |
| textlen | String length in this segment. |

Figure 4. Google Translate TTS Parameters

Example :

* Query= I am a Chang Gung University Student
* tl = en (English), total = 1, idx = 0, textlen = 36.
* URL : http://translate.google.com.tw/translate\_tts?q=I%20am%20a%20Chang%20Gung%20University%20Student&tl=en&total=1&idx=0&textlen=36

### HTK (Hidden Markov Model Toolkit) :

HTK is referred to as the Hidden Markov Model Toolkit. HTK is primarily used for speech recognition research although it has been used for numerous other applications including research into speech synthesis, character recognition and DNA sequencing. HTK is in use at hundreds of sites worldwide.

Speech Recognition involves quite profound mathematical, the code is not easy to write, high barriers to entry, the complexity is not easy to control.

But since HTK from 2000 into freeware open source, significantly reducing barriers to entry, rapidly enhance the development of Speech technology.

## Proposal

Here is split into three parts:

1. TTS Process
2. CGUALIGN
3. Website presentation.,

As shown below Figure 5.

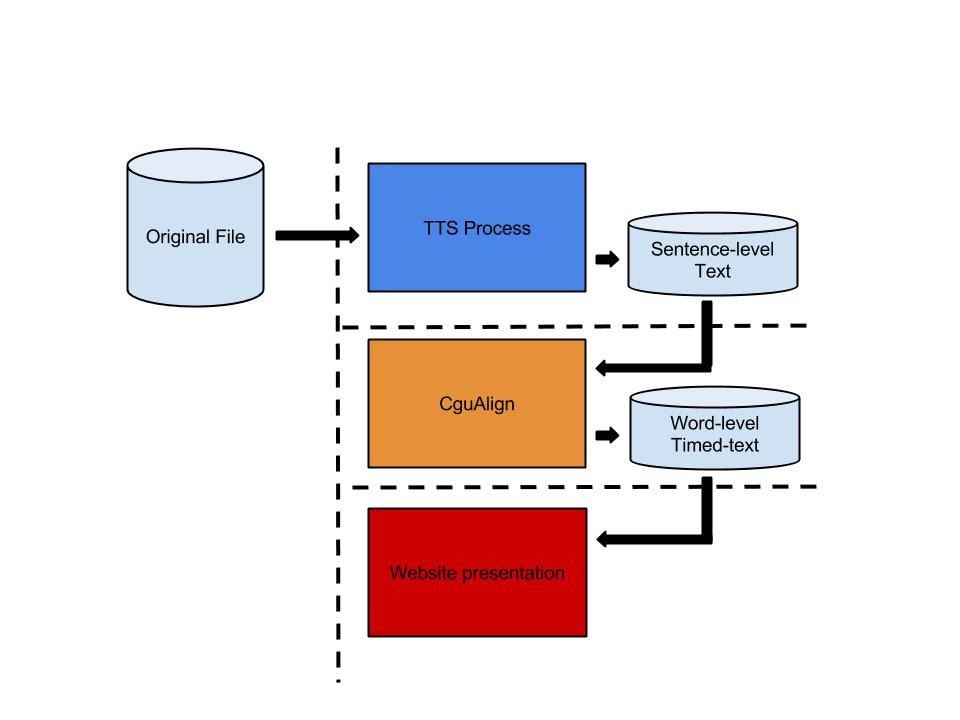


Figure 5 . Flow Diagram

### TTS Process

Here is split into five parts:

* Text Spliter
* Upload to Google Translate
* Download TTS Audio file
* Audio Converter
* Timed-text file Converter

As shown below Figure 6.

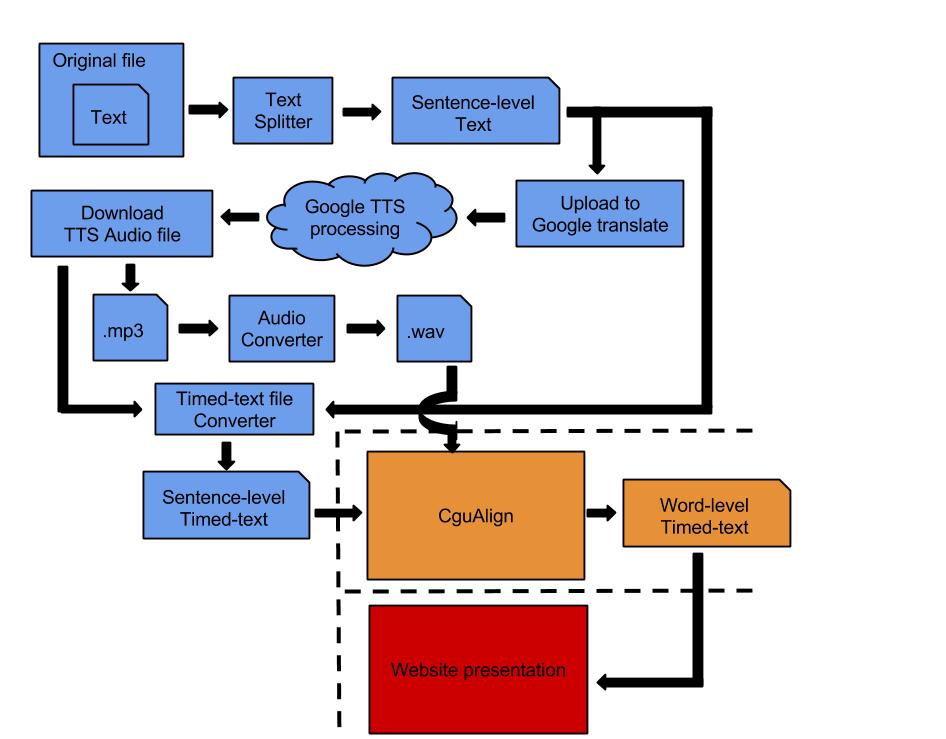


Figure 6. TTS Process Flow Diagram

* **Text Splitter**

If we want to do a Long article TTS, we need to do the Text Splitter. Because Google Translate TTS can not directly enter the query string when the string length greater than 100.

Here we use Python built-in String Function – String.replace(). The rules of split is following :

(1) In accordance with "full stop" to do the cutting, if the string length are less than 100, the cut end, or continue to cut.

(2) The "question mark" do the cutting, if the string length are less than 100, the cut end, or continue to cut.

(3) In accordance with "exclamation point" to do the cutting, if the string length are less than 100, the cut end, or continue to cut.

(4) In accordance with "dash" to do the cutting, if the string length are less than 100, the cut end, or continue to cut.

(5) In accordance with "colon" do the cutting, if the string length are less than 100, the cut end, or continue to cut.

(6) The "comma" to do the cutting, if the string length are less than 100, the cut end, or continue to cut.

(7) If the final string length still has more than 100 from more than 100 the string will be in the middle of the "space" cut.

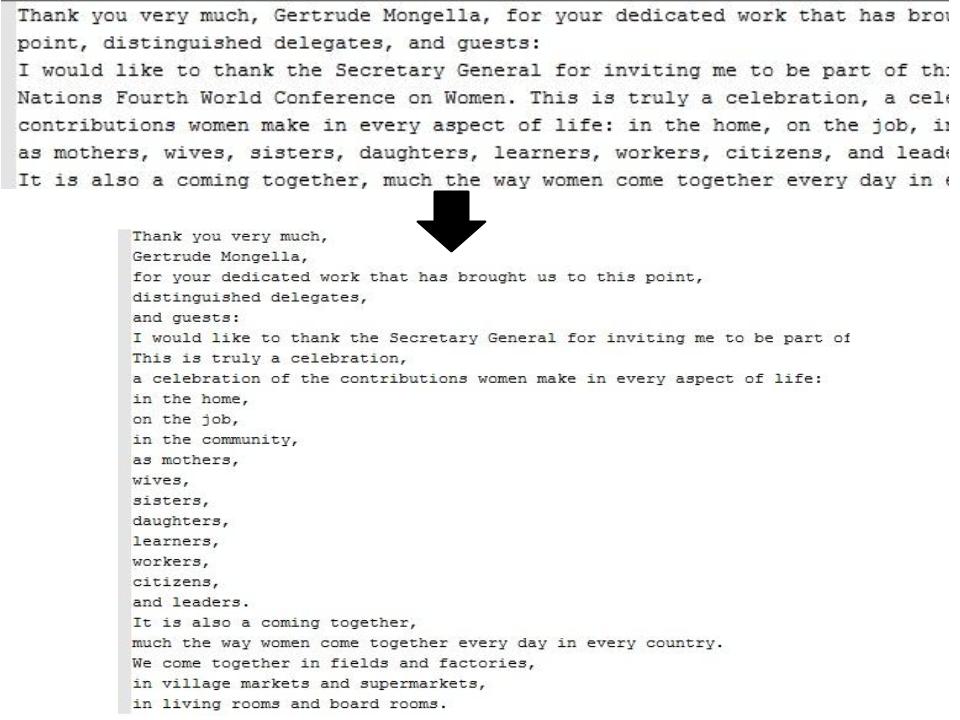


Figure 7. Text Splitter Example

* **Upload to Google Translate**

Here, we use Python built-in library, ‘urllib.request’ and ‘urllib.parse’, to communicate with Google Translate. That can send HTTP Request to Google Translate. The URL of Google Translate TTS is that -- <http://translate.google.com/translate_tts>

The Example Code is following :

**im**p**ort** urllib**.**request

**import** urllib**.**parse

savefile**=**"./TTS.mp3"

f**=** open**(**savefile**,** 'wb+'**)**

文字**=** "Chung Gung University Student"

GOOGLE\_TTS\_URL**=** 'https://translate.google.com.tw/translate\_tts?'

payload **=** **{** 'ie'**:** 'utf-8'**,**

'tl'**:** 'en'**,**

'q'**:** 文字**,**

'total'**:** 1**,**

'idx'**:** 0**,**

'textlen'**:** len**(**text**)** **}**

**try:**

hdr **=** **{**'User-Agent'**:**'Mozilla/5.0'**}**

data **=** urllib**.**parse**.**urlencode**(**payload**)**

req **=** urllib**.**request**.**Request**(**GOOGLE\_TTS\_URL**+**data**,**headers**=**hdr**)**

r **=** urllib**.**request**.**urlopen**(**req**)**

byte**=** r**.**read**()**

f**.**write**(**byte**)**

byteNum= len(byte)

**except** Exception **as** e**:**

**raise**

f**.**close**()**

Figure 8. Upload to Google Translate Example Code

* **Download TTS Audio file**

Follow Figuare 9. line 4,20,21,22.

Line 4 and 21 are open a file and save the TTS file to mp3 by binary write.

Line 20 is decode the HTTP Response from Google Translate.

Line 21 is record every TTS file capacity by byteNum. This can be used for the next step ‘Timed-text file Converter’.

**im**p**ort** urllib**.**request

**import** urllib**.**parse

savefile**=**"./TTS.mp3"

f**=** open**(**savefile**,** 'wb+'**)**

文字**=** "Chung Gung University Student"

GOOGLE\_TTS\_URL**=** 'https://translate.google.com.tw/translate\_tts?'

payload **=** **{** 'ie'**:** 'utf-8'**,**

'tl'**:** 'en'**,**

'q'**:** 文字**,**

'total'**:** 1**,**

'idx'**:** 0**,**

'textlen'**:** len**(**text**)** **}**

**try:**

hdr **=** **{**'User-Agent'**:**'Mozilla/5.0'**}**

data **=** urllib**.**parse**.**urlencode**(**payload**)**

req **=** urllib**.**request**.**Request**(**GOOGLE\_TTS\_URL**+**data**,**headers**=**hdr**)**

r **=** urllib**.**request**.**urlopen**(**req**)**

byte**=** r**.**read**()**

f**.**write**(**byte**)**

byteNum= len(byte)

**except** Exception **as** e**:**

**raise**

f**.**close**()**

Figure 9. Download TTS Audio File Example

* **Timed-text file Converter**

The use of the previous step collected byteNum size of each segment, and the sum of the calculated byteNum, it is possible to calculate the length of each segment in the total period of the speech length of time formula is as follows,



SegmentLength (i) : the length of segment i.

ByteNum (i) : the size of segment i.

Sum (ByteNum) : total file size.

TotalLength : total speech length.

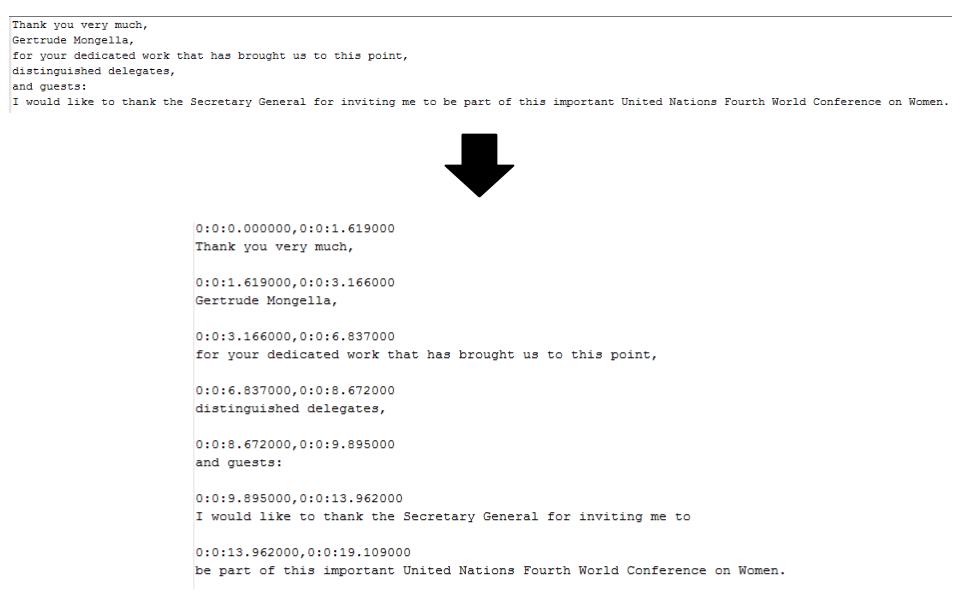
When calculated the length of every segment ,we can make the Sentence-level of Timed-text files by Sentence-level text file and every segment's length .

Figure 10. Timed-text file Converter Example

* **Audio Converter**

In order to use HTK, we need to convert TTS mp3 file to wav file. Here we use another free software – ‘FFmpeg ’, it produces libraries and programs for handling multimedia data. To use FFmpeg in Python, we import Python built-in module, named ‘os’, which can help us to call External program, like FFmpeg, HTK…etc. The function we need in os is os.system().

**def** ffmpeg\_AudioDuration**(**filename**):**

os**.**system**(**"ffmpeg -report -y -i ./TTS-MP3/{0}.mp3" **+**

"./FFmpeg-WAV/{1}.wav"**.**format**(**filename**,**filename**))**

dirlist**=** os**.**listdir**()**

**for** i **in** dirlist **:**

**if** i**.**find**(**'ffmpeg'**)!=-**1 **and** i**.**find**(**'.log'**)** **!=-**1 **:**

report\_name**=** i

**break**

f**=**open**(**report\_name**,**"r"**)**

**for** i **in** f**:**

**if** i**.**find**(**"Duration:"**)** **!=** **-**1**:**

duration**=** i**.**split**(**" Duration: "**)[**1**].**split**(**","**)[**0**]**

hour**=** int**(**duration**.**split**(**":"**)[**0**])**

min **=** int**(**duration**.**split**(**":"**)[**1**])**

sec **=** float**(**duration**.**split**(**":"**)[**2**])**

total\_ms**=** int**(**hour**\*** 3600000 **+** min**\***60000 **+** sec**\***1000**)**

**print(**total\_ms**)**

f**.**close**()**

os**.**system**(**"copy "**+**report\_name**+**" .\\FFmpeg-WAV\\"**+**report\_name**)**

os**.**system**(**"del "**+**report\_name**)**

**return** total\_ms

Figure 11. Audio Converter Example Code

### CGUAlign

Here is split into five parts:

* Hled
* Hcopy
* HCompV
* HERest
* HVite

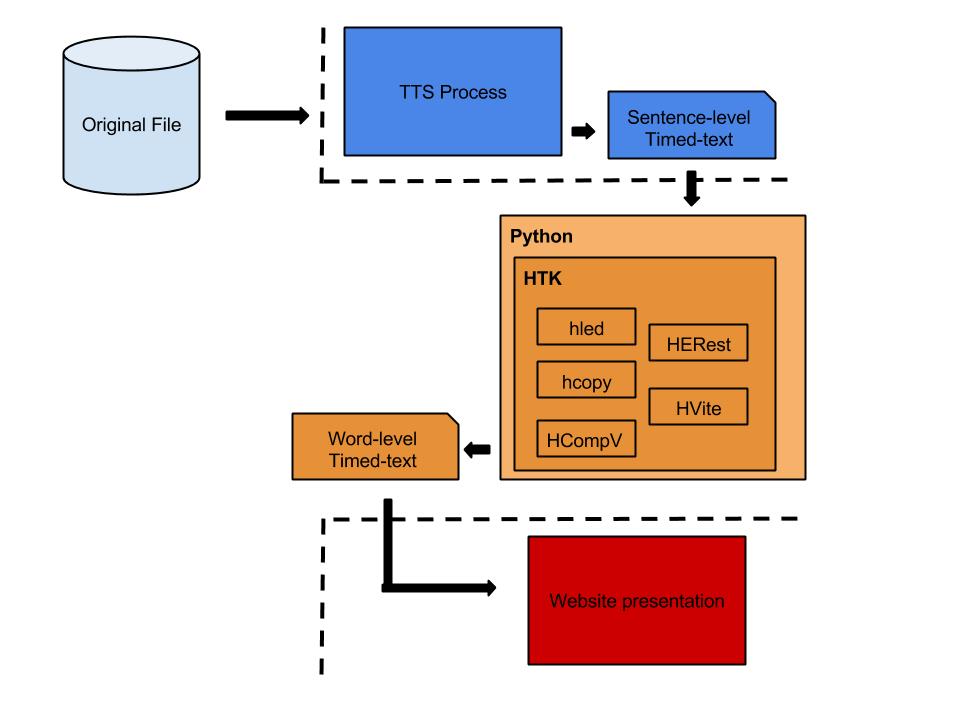


Figure 12. CGUAlign Flow Diagram

* Hled

The main purpose of Hled is to deal with the linguistic labels.

Here we call os.system() function to call Hled.exe .



Figure 13. Hled Processing Example

* Hcopy

The main purpose of Hcopy is to extract speech feature.

Here we also call the function, os.system() , to call Hcopy.exe .



Figure 14. Hcopy Processing Example

* HCompV

The main purpose of Hcopy is to train proto language model.

Here we also call the function, os.system() , to call HCompV.exe .



Figure 15. HCompV Processing Example

* HERest

The main purpose of HERest is to embedded train language model.

Here we also call the function, os.system() , to call HERest.exe .



Figure 16. HERest Processing Example

* HVite

The main purpose of HVite is to forced alignment.

Here we also call the function, os.system() , to call HVite.exe .



Figure 17. HVite Processing Example

### Website presentation

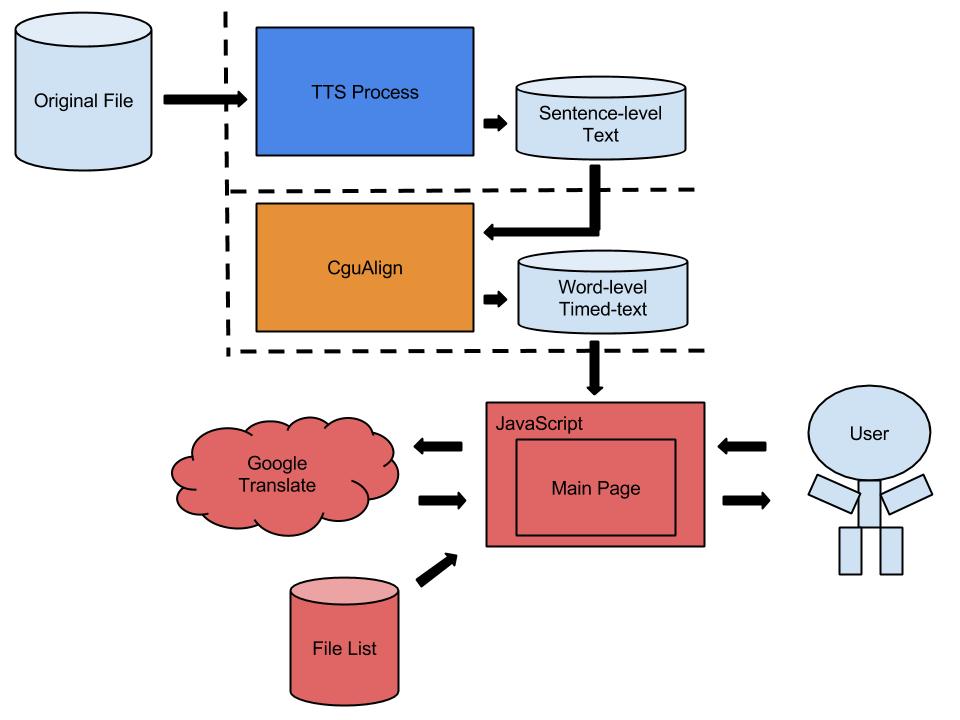
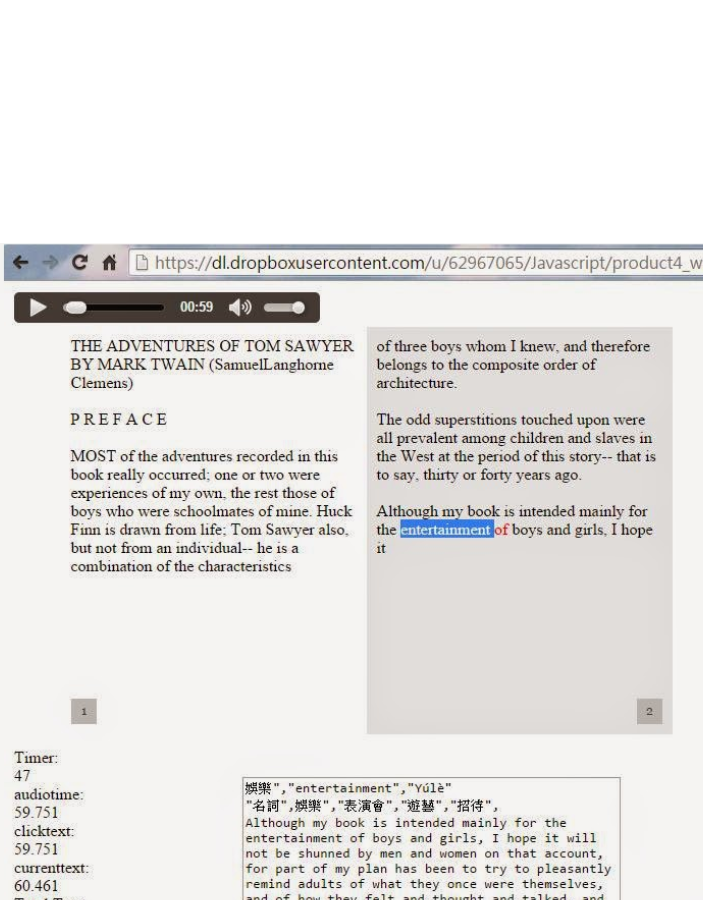


Figure 18. Website Presentation Flow Diagram

Here we use simple JavaScript to build a website to browse the word-level timed-text file.



Here we can take easy shadowing technique to language learning. In bottom of the page, show the meaning of the word you choose.