台語朗讀資料庫之自動切音技術應用於音文同步有聲書之建立

Automatic Time Alignment for a Taiwanese Read Speech Corpus and its Application to Constructing Audiobooks with Text-Speech Synchronization

黃偉杰 Wei-jay Huang
長庚大學資訊工程研究所

林志柔 Jhih-rou Lin
長庚大學資訊工程研究所

呂仁園 Ren-yuan Lyu
長庚大學資訊工程研究所, CS Dept. Chang Gung University
renyuan.lyu@gmail.com

江永進 Yuang-chin Chiang
清華大學統計所, Institute of Statistics, Tsing Hua University

張智星 Jyh-Shing Roger Jang
清華大學資訊所, CS Dept., Tsing Hua University

高明達 Ming-Tat Ko
中研院資訊所, Institute of Information Science, Academia Sinica

摘要

本篇論文是運用語音辨識中的自動切音技術，來建立有聲書之音文同步的功能。目前我們使用台灣教育部網站公開的閩南語朗讀文章共 140 篇，處理了約 11 個小時的語音，將近有 83%的文字之對應語音的時間點可被切出。最後將這些帶有時間點的文字放到 Youtube 與自己架設的網站上來呈現音文同步的效果，以做為進階的語言訓練教材之應用。

關鍵詞：語音辨識，自動切音，有聲書，音文同步，音文對齊，強制對齊技術

一、緒論

台灣是多語的社會，其中華語、台語、客語、原住民語言、甚至日語以及英語皆有一定的族群使用之。在本論文中我們針對台語做為研究與應用的標的語言。近年來因本土化的影響，台語研究漸受到重視，台語的語音資料可以由網路上擷取或者由廣播電視之台語新聞以及戲劇節目取得，蒐集到的語音資料需要經過一番整理才能作為後續使用，近年來電腦有聲書逐漸流行，有了聲音和文字連結的輔助，應能讓學習台語的人有更好的學習效果，但是製作有聲書的過程中有一部分工作相當麻煩，亦即聲音與文字的連結。一般而言需要花費大量人工作切音以及音文對齊，由於成本考量，一般有聲書至多僅做到句子層級的音文同步，本論文嘗試運用語音辨識技術中強制對齊技術(force alignment)，來

令台語語音資料庫達到在字或音節層級的音文對齊，以做為進階的語言訓練教材之應用。

本論文語料蒐集為台語，我們所擷取的資料庫為「閩南語朗讀文章選輯」[3]，資料庫收錄文章 140 篇（含重複者 7 篇），資料庫的內容為教育部特別邀請學者專家選錄文章，並聘請專人將文章內容改寫成適合朗讀之文稿，名為「閩南語朗讀文章選輯」，以供各界學習參考，在文字標音方面則是依據臺灣閩南語羅馬字拼音方式標注，另外也請國立教育廣播電臺錄製聲音檔每一篇文章配置一個聲音檔，盼望藉此提升大眾學習母語的興趣，資料庫第一篇列在圖一以為參考。

001 牛墟（hi） //紀傳洲（18/04/07thk改寫）

古早，牛是台灣人上重要的作穡（tsoh-sit）伴，牛會犁田、拖車、駛石碾、挨塗壟（e-thôo-lâng）...便若較粗重的空課攏愛伊鬥做。伊攏是恬恬仔揹力（kut-làt）去做，予（hōo）人真感心。這種性嘸真成（sîng）咱台灣人，毋才講咱是「台灣牛」。

圖一、「閩南語朗讀文章選輯」第一篇：「牛墟」部分內容

整個資料庫的資訊列表如表一。

表一、資料庫資訊

錄音人數	2 人
聲音檔 Wav 總容量	1.21G
聲音檔 Sample rate	16kHz
總時間	681.37 分鐘
總篇數	133 篇
總句數	15923 句
總字數	139271 字

由於文字格式的不同所以要經過格式的整理及編碼的轉換，後續才是拼音系統的轉換，本實驗都轉換為福爾摩莎 ForPA 拼音系統，傳統切音方式多為使用 Transcriber 軟體輔助以人工切割使得句子與聲音能夠做連結，然而人工切割不僅耗時且無法達到精細層面的切割。因此本論文提出一個精細層面的自動切音之機制以降低切割所需的時間與人力。此機制利用 HTK 訓練模型和辨識，使得電腦於音節(Syllable)的層次上自動找到最佳的切割時間點，接著再使用 Transcriber 呈現結果，經過效能分析的結果可知經過語音切割的方法錯誤平均值從 0.16 秒提升到 0.06 秒並且可知使用 HTK 切割效能比等切效能較佳。本論文採用 HTK 的音文對齊技術，針對台灣教育部出版的「閩南語朗讀文章選輯」之台語朗讀語料，做到聲音與文字在「音節」層級的對齊，音節對齊精確度可達 95%以上。這項技術應用於「音文同步有聲書」之建立，已將整套「閩南語朗讀文章選輯」建立成網頁應用程式，將在取得教育部授權後，開放給各界自由瀏覽聆聽。

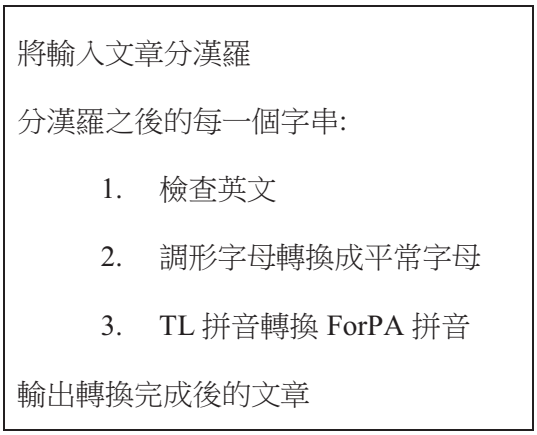
二、台語文字處理

在「閩南語朗讀文章選輯」中出現的台語文字，是台語漢羅文，也就是漢字以及羅馬字並存的文字，其中羅馬字的部分採用教育部建議的「台羅拼音」。以下是一些典型的例句：

例 1.「牛是台灣人上重要的作穡（tsoh-sit）伴。」

例 2.「牛會犁田、拖車、駛石碾、挨塗壟（e-thôo-lâng）。」
例 3.「人來客去濟 kah 若魩（but）仔魚。」
例 4.「規車疊 kah 滇涔涔（tīnn-khó-x）。」

在上述例句中，「作穡（tsoh-sit）」、「挨塗壟（e-thôo-lâng）」、「魩（but）」、「滇涔涔（tīnn-khó-x）」等，即是漢字後面括弧加註「台羅拼音」式的「音標」，把音標移除並不影響文義的表達；而「濟 kah 若魩仔魚」、「疊 kah 滇涔涔」之中的「kah」則本身是「文字」，是文句中不可或缺的一部份。由於台羅拼音包含字母修飾符，如上述例子中的「ô」、「â」、「ī」、「ó」等，用來做為台語的聲調記號，因此我們必須採用 Unicode 做為編碼集，不能沿用一般的 Ascii/Big5 編碼，否則會有資訊遺失及顯示出現亂碼的現象，此為台語文字處理必須特別留意之處。下圖為轉換流程圖。



圖二、轉換流程圖

1. 分漢羅：

所謂分漢羅，意思是將文句中的漢字與羅馬字分離出來。舉例來說，要將如下字串分成子字串序列：

'這種性嘛真成（sing5）咱台灣人。'
⇩
['這', '種', '性', '嘛', '真', '成', '（', 'sing5', '）', '咱', '台', '灣', '人', '。']

分成子字串之後，進一步的處理、轉換較容易進行。
分漢羅可以用“正規表示法”(regular expression)實現。較細節之處暫時不管，分漢羅的 python 表示可以是：

```
re.split('([一-鶴]|[a-zA-Z]+\d*)',漢羅字串)
```

re 是 Python 的 regularexpress 模組，詳細語法請見 Python 參考書。 其中漢字“一”的 unicode 編碼是 \u4E00 漢字“鶴”的 unicode 編碼 是\uFA2D，中間包括大多數現在台灣使用的漢字。以前例而言，Python 執行結果如下：

```
>>> jj='這種性嘛真成（sing5）咱台灣人。'
>>> re.split('([一-鶴]|[a-zA-Z]+\d*)',jj)
['', '這', '', '種', '', '性', '', '嘛', '', '真', '', '成', '（', 'sing5', '）', '咱', '', '台', '', '灣', '', '人', '。']
```

(注意到分漢羅的子字串包括幾個空白字串，但那些不造成處理轉換的困難。)
不幸的，上述的分漢羅，對有調形的拼音字，會產生錯誤：

```
>>> ii='真成（sîng）咱台灣人。'
>>> re.split('([一-鶴]|[a-zA-Z]+\d*)',ii)
['', '真', '', '成', '（', 's', 'î', 'ng', '）', '咱', '', '台', '', '灣', '', '人', '。']
```

注意到 調形拼音字“sîng”， 沒有分正確。

“讓格書寫的 Python 工具箱(LGO.py)”中的 hunHL 函式[2]，正是基於類似的想法，考慮了更多的細節，正合乎我們的需要：

```
>>> ii='這種性嘛真成（sîng）咱台灣人。'
>>> hunHL(ii)
['這', '種', '性', '嘛', '真', '成', '（', 'sîng', '）', '咱', '台', '灣', '人', '。']
```

由上述的執行結果可看出，調形拼音字“sîng”，已正確完成分開。

2. 調形字母的轉換：

所謂調形字母的轉換，是將調形字母轉換成對應的數字加在字母後面做代表，譬如 e-thôo-lâng， 需要轉換成數字拼音字 e1-thoo5-lang5。台羅拼音所使用的調形字母，及其對應的拼音、調號，我們用三個字串表示，如下表二對應表。

表二、對應表

```
調形字 = 'ô  â  a̍  ō  î'
平常字 = 'o   a   a   o   i'
聲調   = '5   5   8   7   7'
```

有了這三個字串，我們容易製作出調形字母的轉換函數，轉換需要想法是，針對每一字母，檢查是不是調形字母，若是，則轉換平常字母，並且記憶該聲調數字，最後才將數字串接在後。

```
>>> toBSR('siâ')
'sia5'
>>> toBSR('dâ')
'da5'
```

圖三、轉換例子

3. 轉換 ForPA 拼音：

將 TL 拼音字 替換成 ForPA 拼音字，至少要經過兩個步驟：

(1) 第一步是分聲韻(Python 函式 hunSU)

(2) 第二步是聲韻分別替換(使用 Python Dict 直接替換) 再串接

以 TL 拼音'thoo5'為例，轉換成 ForPA 拼音的過程主要如下。

```
>>> s,u,d = splitSUD(‘thoo5') #S,U,D 分別表 聲、韻、調
>>> [S2S.get(s,s), U2U.get(u,u), D2D.get(d,d)]
'to5'
```

其中 splitSUD 是 分聲韻調， S2S, U2U, D2D 分別是 執行聲韻調轉換的 Python 資料結構(Python 的 dict)。以下分為四個部分說明這個過程。

(2.1)分聲韻調：

對音節有母音時，如下方的正規表示法可拆開聲韻調，

re.split('([aeiou][a-zA-Z]*)(\d*)',syllable)

譬如：

syllable = song4 => ['s', 'ong', '4', '']

但台語還有無母音音素的音節，如下表三。

表三、ForPA 及 TL 的 子音音節

ForPA 子音音節				TL 子音音節			
m	ng			m	ng		
hm	bng	png	mng	hm	png	phng	mng
	dng	tng	nng		tng	thng	nng
	gng	kng	hng		kng	khng	hng
	zng	cng	sng		chng	chhng	sng

因為沒有母音，只好小步進行：

先拆音節後面的聲調 (若無則聲調設為空字串)，聲韻部份再試用母音拆開，若無母音則試拆(ng|m)，若再無，就整個字串當韻母。例子如下圖四。

>>> splitSUD('song4')	>>> splitSUD('oai')
['s', 'ong', '4']	['', 'oai', '']
>>> splitSUD('siong2')	>>> splitSUD('oe2')
['s', 'iong', '2']	['', 'oe', '2']
>>> splitSUD('sng5')	>>> splitSUD('dfgsfd3')
['s', 'ng', '5']	['', 'dfgsfd', '3']
>>> splitSUD('sng')	>>> splitSUD('字')
['s', 'ng', '']	['', '字', '']
>>> splitSUD('ng')	
['', 'ng', '']	

圖四、聲、韻、調例子

(2.2)聲韻分別轉換：

接著我們分別轉換聲韻調，從台羅拼音到 ForPA 拼音。下圖五分別是台羅拼音及 ForPA 拼音的聲、韻、調對照表。從這三組對應字串，Python 的 dict 很容易製作出需要的替換功能。

```
TLSVOR    ='p ph m b  t th n l  k kh ng g  ch chh s j h'.split()
ForPaSVOR ='b p  m bh d t  n l  g k  ng gh z  c  s r h'.split()
```

(a)、聲、韻、調例子

```
TLUVOR ="""
a  e  i  oo u  o
ai au   ia io iu iau    ua ue ui uai
am im om iam    an en in un uan ian   ang ing ong iang iong uang
ann enn inn onn unn
ainn aunn  iann iunn iaunn   uann uenn uinn uainn
m  ng
ah  eh  ih  ooh uh  oh
aih auh   iah ioh iuh iauh    uah ueh uih uaih
ap ip op iap    at et it ut uat iat   ak ik ok iak iok uak
annh ennh innh onnh unnh
ainnh aunnh  iannh iunnh iaunnh   uannh uennh uinnh uainnh
mh  ngh
""".split()
```

```
ForPaUVOR ="""
a  e  i  o u  er
ai au   ia ior iu iau    ua ue ui uai
am im om iam    an en in un uan ian   ang ing ong iang iong uang
ann enn inn onn unn
ainn aunn  iann iunn iaunn   uann uenn uinn uainn
m  ng
ah  eh  ih  oh uh  erh
aih auh   iah iorh iuh iauh    uah ueh uih uaih
ap ip op iap    at et it ut uat iat   ak ik ok iak iok uak
annh ennh innh onnh unnh
ainnh aunnh  iannh iunnh iaunnh   uannh uennh uinnh uainnh
mh  ngh
""".split()
```

(b)、TL韻母 對應 ForPA韻母

```
TLDiau    = '1 7 3 2 5 8 4'.split()
ForPaDiau = '1 2 3 4 5 6 7'.split()
```

(c)、TL對應ForPA聲調表

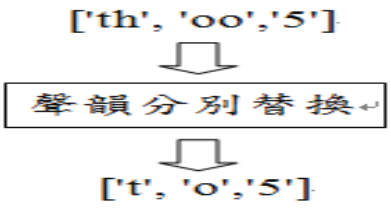
圖五、聲、韻、調例子

依據上圖使用 Python 容易製作出對照表，並且容易使用，程式碼如下圖六。

```
S2S = {k:v for k,v in zip(TLSVOR, ForPaSVOR)}
U2U = {k:v for k,v in zip(TLUVOR, ForPaUVOR)}
D2D = {k:v for k,v in zip(TLDiau, ForPaDiau)}
    #dict{...} will map from key to value
>>> [S2S.get(S,S), U2U.get(U,U), D2D.get(D,D)]
```

圖六、Python 容易製作出對照表

聲韻分別替換，輸入為['th', 'oo','5'] 使用 Python Dict 直接替換輸出為['to5']，如下圖七。



圖七、聲韻分別替換

但是，直接聲韻調替換，並沒有解決所有問題，因為台羅拼音還有“另類拼音”以及“條件變讀”問題。以下分(2.3)(2.4)進一步討論。

(2.3)台羅拼音的另類拼音法：

多少因為歷史原因，某些音素的台羅拼音，拼音法有變異，如下表另類拼音法幾個例子。

表四、另類拼音法幾個例子

TL 另類拼寫法	TL 拼音	ForPA 拼寫法
ts	ch	z
tsh	chh	c
oe	ue	ue
oa	ua	ua
oai	uai	uai

實作上， 如下圖，Python 字典可以輕易加入這些另類拼寫法：

```
S2S.update({'ts':'z','tsh':'c'})
U2U.update({'oe':'ue','oa':'ua','oai':'uai'})
```

加上這些另類拼寫，聲韻調轉換更加完整。

(2.4)台羅拼音的條件變讀問題：

台羅拼音(及教會拼音) 不幸的有條件變讀的問題，如下表：

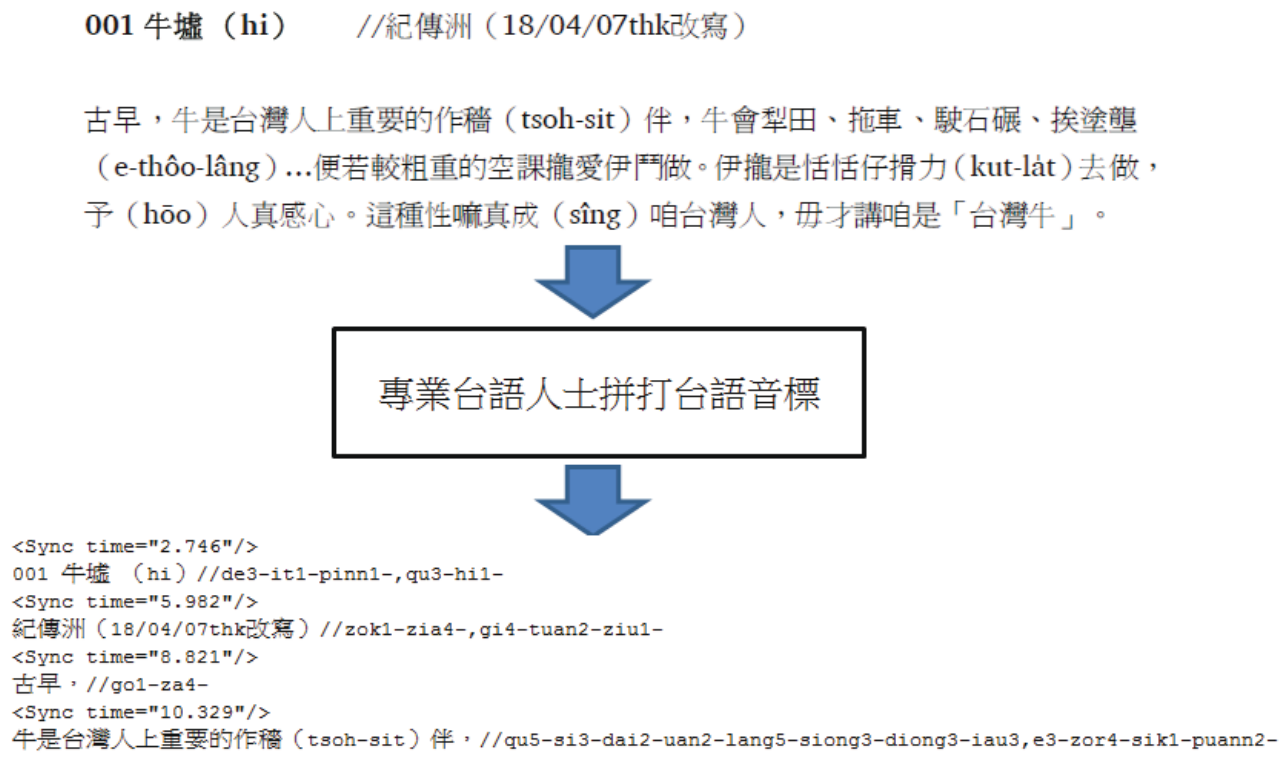
表三、條件變讀的問題

台羅拼音		ForPA 拼音	
soo	oo 表ㄛ	so	o 表ㄛ
so	o 表ㄜ	ser	er 表ㄜ
mo	o 表ㄛ	mo	o 表
no	o 表ㄛ	no	o 表
ngo	o 表ㄛ	ngo	o 表

在此變讀是符號 o 在不同音節中代表不同的音素。所以，如果直接替換將 o 直接替換成 er，則 mo/no/ngo 換成 mer/ner/nger，將發生錯誤。因此聲韻調替代時，應考慮周遭音境，如下 Python 聲韻調替代程式。

```
S,U,D = splitSUD(syllable)
if S in {'m n ng'.split()} and U is 'o':
    rr = [S,U,D]
else :
    rr = [S2S.get(S,S), U2U.get(U,U), D2D.get(D,D)]
```

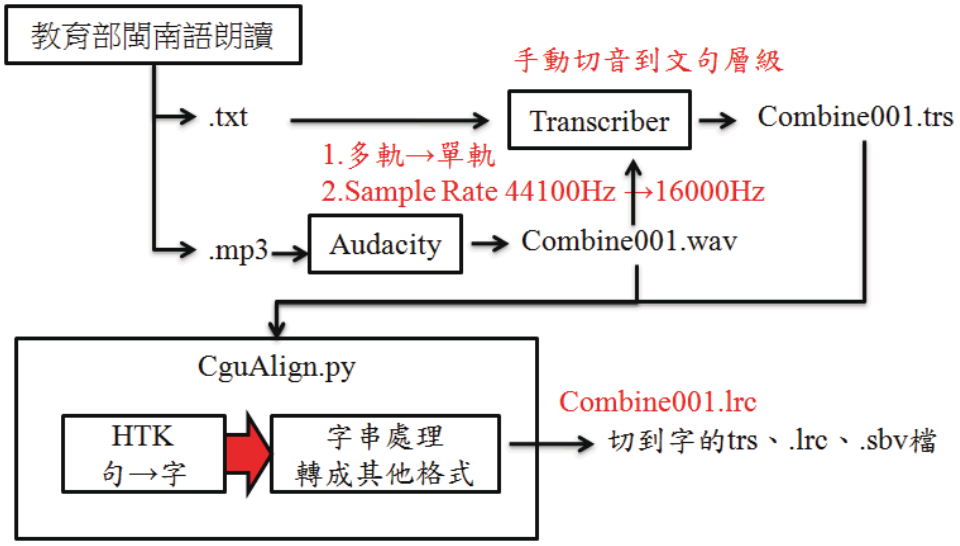
最後我們從教育部網站上取得「閩南語朗讀文章選輯」，再由台語的專業人士幫忙將漢字拼打成台語音標，接著我們依照上述的台語文字處理，將每句的漢字與台語音標使用 Transcriber 來表示如下圖八。



圖八、台語文字處理流程

三、台語語音處理

我們運用 HTK 以及 Python 程式語言寫成 CguAlign 程式，可以執行自動切音過程。配合部分手動的處理，本篇論文以教育部閩南語朗讀 1~10 篇(Combine001)[3]為例，我們將整個手動的資料處理到自動切音的過程以下圖一來說明：



圖九、系統流程圖

手動的資料處理：

從教育部閩南語朗讀上取得 txt 文字檔以及對應的 mp3 語音檔，由於需要使用到 HTK 的幫助，而 HTK 在語音檔的部分只能處理 wav 的檔案，所以使用 Audacity[6]將原來 mp3 檔轉成 wav 檔，這裡需要將語音檔的 Sample rate 換成 16000Hz 以及雙軌變成單軌。接著將 txt 文字檔與轉好的 wav 語音檔傳入 Transcriber[5]中，以手動的方式將整段文章切到文句的層級，輸出一個 trs 檔。手動資料處理的部分告一段落，以下為 CguAlign 自動切音過程。

CguAlign 自動切音過程：

CguAlign 自動讀取一個切到文句的 trs 檔與一個 wav 語音檔，Output 出切到字層級的 trs 檔、lrc 檔與 sbv 檔。以下我們分為兩個部分來詳細介紹自動切音的過程。下圖二為 CguAlign 的主程式碼。

```
def CguAlign主程式():

    原文檔名集=['Combine001',#]
    '''
    '''

    建立資料夾()

    for x in 原文檔名集:

        trsFn = 'Input/切到句的trs檔/'+ x +'.trs'
        wavFn = 'Input/wav/'+ x +'.wav'


        語音總時間長度=將trs檔的時間與對應字串轉成多個lab檔(trsFn)

        將長語音檔依照lab檔所對應的時間來切割成多個短語音檔(wavFn)

        製造各個HtkTool所需的參數檔()

        處裡語音標籤及詞典()

        擷取語音特徵及訓練語音模型()

        將語音文字做對齊()

        轉成切到字的trs檔(trsFn,語音總時間長度)

        對齊原文格式並轉成切到字的lrc與sbv檔(x)

if __name__=='__main__':
    CguAlign主程式()
```

1.HTK 切音

2.字串處理
轉其他格式

圖十、CguAlign 主程式碼

1. 使用 HTK[1]來切音：

(1) 將 trs 檔的時間與對應字串轉成多個 lab 檔
抓取切到句層級 trs 中的時間與對應字串，在時間部分轉換時間的格式；在相同時間的字串部分，將字串頭尾加上 sil，並在字與字中間空白處替換成底線來連接成句。

(2) 將長語音檔依照 lab 檔所對應的時間來切割成多個短語音檔
將長語音檔依照 lab 檔中每行的時間進行切音，並轉存成多個短語音檔。

(3) 製造各個 HtkTool 所需的參數檔
這裡製造出 7 個參數檔，分別為 hLed.led、hLed00.led、hCopy.conf、hInit.conf、hRest.conf、hErest.conf、hVite.conf。

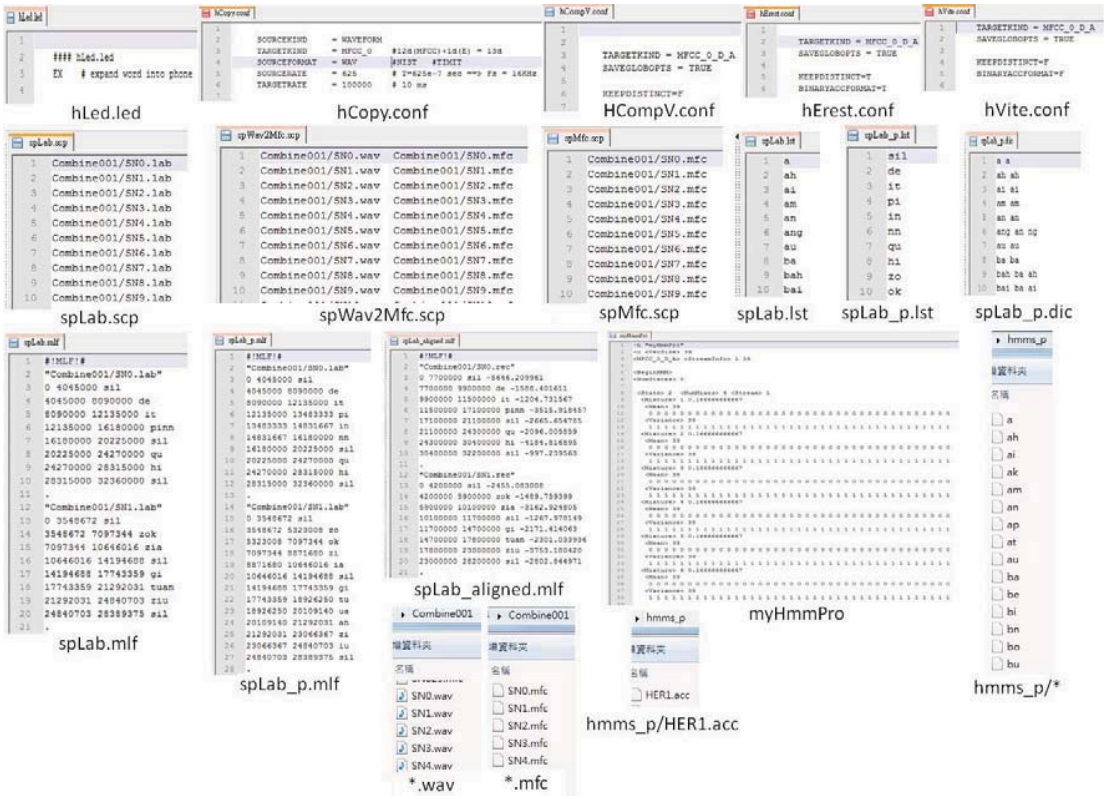
(4) 處裡語音標籤及詞典
主要在製作 mlf 檔。這裡開始介紹語音模型訓練及切割過程，使用 HTK 的 hled 程式，輸入為 scp 檔經過 hled 程式可以得到一個為 lst 檔另一個為 mlf 檔，這兩個檔案的差別在於 mlf 檔裡的每句話使用等切的方式記錄開始及結束的時間點。輸入為 scp 及 mlf 及 dic 檔 經過 hled 程式把 mlf 轉換成「雙音素」(Biphone)的格式，在 dic 檔中為音節對應「雙音素」的格式。

(5) 擷取語音特徵及訓練語音模型
這裡分為聲音處理與模型訓練兩個部分。
聲音處理：當我們把 lab 處理完後，下一步為自動電腦語音切音之訊號處理的層次，在訊號處理的層次來說語音辨識技術使用 mfc 可達到不錯語音辨識之效果，波型轉換到 mfc 過程中我們使用 HTK 工具，在 HTK 中要做特徵擷取的動作為 hcopy 這程式，主要觀念為輸入聲音檔經過 hopy 輸出為 mfc，在過程中需要提供 scp 檔案告訴 hcopy 什麼樣的 wav 對應到什麼樣的 mfc，另外 hcopy 本身也需要一些參數例如 windows 寬度及 windows shift 的長度…等等，hCopy.conf 檔案提供這些參數給 hcopy，執行這指令 os.system('hcopy -A -C hCopy.conf -S spWav2Mfc.scp') 來做上述的事情。
模型訓練：轉換成 mfc 後，接下來就是模型訓練因為在後續切割時需要用到，模型訓練使用 HMM 技術來製作，我們使用 HTK 的 HCompV 這程式執行以下指令為模型訓練的第一步 os.system('HCompV -A -C HCompV.conf -S spMfc.scp -m -I spLab_p.mlf -M hmms_p/ -o '+m+' myHmmPro') 。
輸入為 mfc 經過 HCompV 輸出為那些音標的 HMM 模型，並且為「雙音素」的模型，以細節來說在 spLab_p.mlf 檔案中如上圖語音標籤(lab)之處理過程為「雙音素」的型式儲存且切割的時間點為等切的時間點，spMfc.scp 檔案為在目錄下所蒐集的特徵檔列表，myHmmPro 檔案為未訓練之前先給定一個原型模型檔，內容為有幾個模型及 Mixture 及幾個 State，此檔案可以程式製作及手動製作，最後經過指令可得到初始的模型。
模型訓練的第二步讓模型更精緻化，在 HTK 中所使用的程式為 HERest，輸入為 mfc 檔經過 HERest 輸出為 HMM Phone 的模型，因為更精緻化的關係所以多提供了一個模型列表檔案，並且在跑了 5 次的迴圈，在我們程式中設定 N=5，經過更精緻化的訓練得到一組雙音素的模型，第一步及第二步做完後就把模型訓練完成，上述過程中把有加標籤的語音訊號已訓練成模型，下一步為語音切割。

(6) 將語音文字做對齊
主要為語音切割的部分。HTK 中使用語音切割的程式為 HVite，HVite 可以用在語音辨識，在賜此我們使用強制對齊(Forced alignment)這種功能，因為我們已知道音標但是不

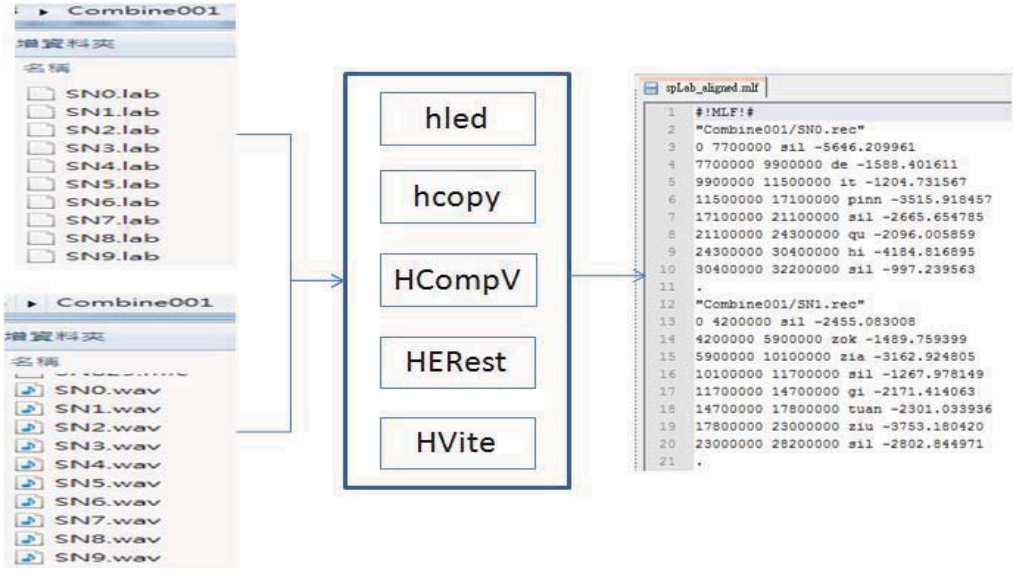
知道斷點，例如 SN0.mfc 的內容為 de_it_pinn_sil_qu_hi 對應到 lab 檔這是我們已知的語言標籤，這邊不使用。

spLab_p.mlf 檔是因為最後我們只需要切割至音節的層次不需要切割到雙音節的層次，最後在 HVite 中輸入為 mfc 及未切割時間點的 mlf 檔，經過 HVite 輸出為經過語音辨識切割出的時間點結果。請參考下圖十格式內容。



圖十一、Input、Output 格式內容

如下圖九為整個處理過程從所有語音檔及所有 lab 檔經過 hled、hcopy、HCompV、HERest、HVite 的指令，最後產生出切割好的時間點。



圖十二、整個處理過程

以上為 CguAlign.py 程式中的 HTK 自動切音的步驟，其中在製作聲學模型時，我們使用一個特別的方式，利用原文所出現的字當作字典，每個字一兩兩字母的方式連接做為此單字的音標，接著進行擷取語音特徵及訓練語音模型。

2. 字串處理轉其他格式：

在這部分主要將帶有切到字層級的音標與原文單字做對齊，以上述主程式碼中的 function(對齊原文格式並轉成切到字的 lrc 與 sbv 檔)來進行處理，以下圖十為此 function 的詳細說明。

```
def 對齊原文格式並轉成切到字的lrc與sbv檔(音標檔案名稱):
    音標檔內容 = 讀取檔案('Output/切到字的lab檔/'+音標檔案名稱+'_aligned.lab')
    (音標時間,音標字串,開始時間對應結束時間字典) = 將sil接到上一個音標(音標檔內容)
    (句子時間,句子字串) = 將trs檔時間與文字存成字典('Input/切到句的trs檔/'+音標檔案名稱+'.trs')
    (對齊後時間,對齊後字串) = 原文句字對齊音標(音標時間,音標字串,句子時間,句子字串,開始時間對應結束時間字典)
    轉成lrc檔(音標檔案名稱,對齊後時間,對齊後字串)
    轉成切到字的sbv檔(音標檔案名稱,對齊後時間,對齊後字串)
```

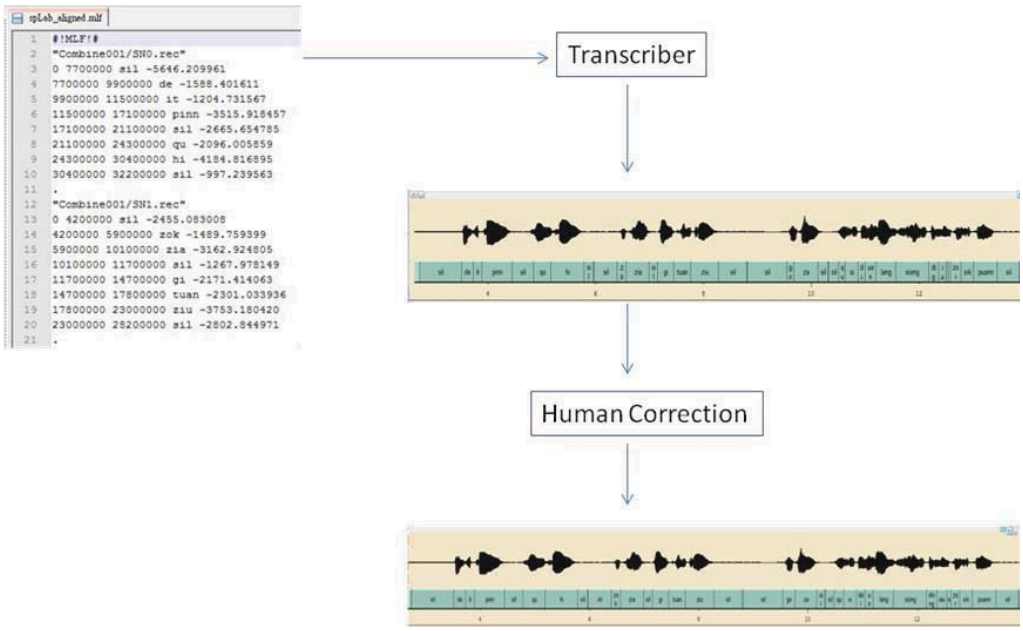
圖十三、對齊原文格式並轉成切到字的 lrc 與 sbv 檔

對齊原文格式並轉成切到字的 lrc 與 sbv 檔程式碼，將語音對齊文字 output 一個.lab 檔，其中.lab 為切到字的時間點，由於此 lab 檔所切出的字串為音標，所以將音標與原文做對齊，再轉成 lrc 與 sbv 檔以供音文同步效果之呈現，此音文同步效果可由以下網址來呈現 http://dl.dropbox.com/u/33089565/ryEx007_3.html。

四、台語切音實驗結果

（一）、切割效能分析

如下圖十一需把切音出來的結果轉回到 Transcriber 上面，因為 Transcriber 為圖型介面，很明顯的可以看出斷點，因此我們就利用人工根據主觀的判斷並且設定好螢幕解析度為 1280 x 800 一次大約可看 5 秒做調整，需要調整的地方為有明顯切割到別的音節地方才需做調整，經過人工修正我們稱為標準答案。

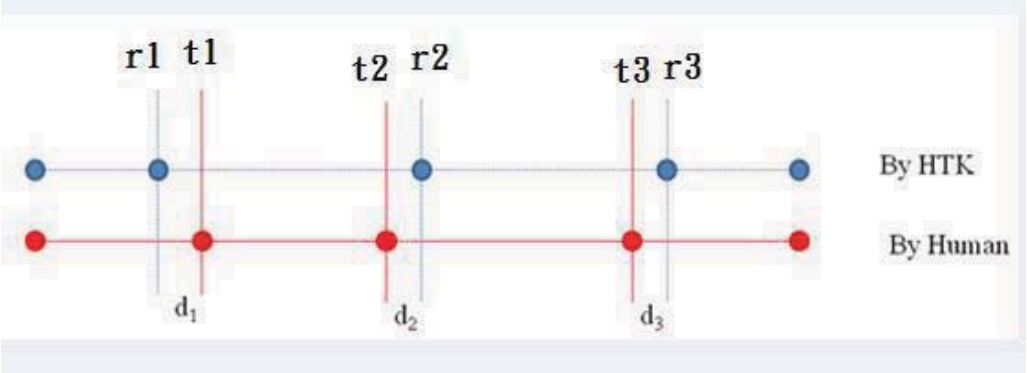


圖十四、轉回 Transcriber

有了標準答案分別再跟 HTK 切割和等切做比較，做為這邊的效能分析。如下圖十二，虛線為使用 HTK 切割出的結果，實線為標準答案，分別計算出互相對應之切割點之時間距離公式 $d_i = |r_i - t_i|$再依下

式時間距離平均$\mu = \frac{\sum_{i=1}^{N-K} d_i}{(N-K)}$為錯誤平均值，其中$r_i$為 HTK 切割出的結果，為人工調整之結果即是

標準答案，N 為時間點總數，K 為人工認定正確之切割點，所以不納入錯誤平均值之計算，而 N-K 為人工認定錯誤，有待調整之時間點。



圖十五、效能分析

表四、效能分析比較

	HTK 切割	等切
錯誤平均值	0.06032 秒	0.164033 秒

此實驗的數據由第一篇文章總共 759 句產生，如上表一，以錯誤平均值來說 HTK 跟標準答案比相差 0.06 秒，等切跟標準答案比相差 0.16 秒由此可知經過語音切割的方法從 0.16 秒提升到 0.06 秒並且可知 HTK 切割其效能較佳。

（二）、切出率

我們使用切出率當做實驗結果的數據，所謂的切出率為一個時間單位只有一個單字，這裡我們分兩種方式來計算切出率，其方式一為以單字來計算切出率，方式二為以時間來計算切出率。切出率公式如下：

$$\text{切出率}(\%) = \frac{(1 - \text{未切出個數})}{\text{總數}} * 100$$

未切出個數：在方式一中代表未切出單字的數量；在方式二中代表未切出時間長度。

總數：在方式一中代表文章中全部單字的數量；在方式二中代表語音時間總長度。

表二為 CguAlign 計算教育部閩南語朗讀全集的切出率：

表五、CguAlign 計算教育部閩南語朗讀全集的切出率

	教育部閩南語朗讀全集
方式一(以單字來計算)	78.67%
方式二(以時間來計算)	82.93%

CguAlign 分析未切出之問題：語音中出現的字，文章中卻沒有出現此文字，如下圖十三教育部閩南與第一篇中在語音時間 5.176 到 5.962，出現作者但文章中卻未出現：

```
[3.506]0
[3.746]0
[3.896]1
[4.456]牛
[5.176]墟
[5.982]紀傳洲
```

圖十六、CguAlign 未切出之問題

五、音文同步有聲書系統

在音文同步效果呈現的部分，目前以兩種方式來進行，分別在 Youtube 上與我們自己所架設的網站(CguTASync)，請連結此網址：https://dl.dropbox.com/u/36364100/wj.html ，以下為兩種方式的呈現說明：

（一）、Youtube[4]平台呈現：

我們將帶有時間點的文字放到 Youtube 平台上呈現，以 Combine001(教育部閩南語 1~10 篇)為例，下圖十四為音文同步效果的呈現，紅色框框中的牛為音文同步字幕效果呈現，點選下方紅色框框中的 CC 可以選擇你要呈現的字幕名稱，而我們就是在這裡提供帶有時間點的文字。

圖十七、Combine001 在 Youtube 平台上呈現音文同步效果

（二）、CguTASync(CguTextAudioSynchronization)呈現：

這是我們所架設的網頁，用多種方式來呈現音文同步的效果，因為某種套件的關係需由 Firefox 瀏覽器來開啟，以下圖十五為功能的介紹：

圖十八、CguTASync 音文同步效果呈現與功能說明

六、結論

台語文字在處理上較為複雜，蒐集語料庫時要注意格式、拼音的問題，蒐集的資料格式必須統一之後在進行統計，台語拼音中常使用特殊符號，所以就要使用 UTF8 來編碼，這裡特別要注意編碼的問題。把語料庫的聲音及文字利用 Transcriber 軟體做到以句子切割為時間點，以人工手動切割句子，一篇文章大約需要二十到三十分鐘總共有 140 篇文章如果要手動切割到音節的層次相對的需要花更多倍的時間，因此我們利用 HTK 工具幫我們找出音節的時間點在過程中需要訓練模型之後進行語音切割最後找出時間點並且轉回 Transcriber 格式，經過語音切割的方法從 0.16 秒提升到 0.06 秒並且可知 HTK 切割其效能較佳。在音文同步的部分，由於經過語音辨識音文對齊後，切出帶有時間的音標以與原來文章的文字不同，除之外還有失去標點符號原有的位置以及原文段落的格式，我們將這些帶有時間點的音標與原文做對齊，並還原原有文章的標點符號與段落格式。

目前我們處理了台灣教育部所提供的閩南語朗讀文章共 140 篇，處理了約 11 個小時的語音，將近有 83%的文字之對應語音的時間點可被切出。最後我們將教育部閩南語朗讀語料所處理完之帶有時間點的文字放到 YouTube 與自己架設的網站上來呈現音文同步的效果。

參考文獻

[1] S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason,D. Povey, V. Valtchev, and P. Woodland, “The HTK book(for HTK version 3.4.1),” Cambridge University Engineering Department,Tech. Rep., March. 2009.

[2] 江永進(2011). 讓格書寫 Python 工具箱。 新竹：清華大學統計所。 (程式檔案 LGO.py)

[3] 全國語文競賽臺灣閩南語朗讀參考資料使用說明 http://140.111.34.54/MANDR/minna/first.html

[4] Youtube，http://www.youtube.com/

[5] Transcriber，http://trans.sourceforge.net/en/presentation.php

[6] Audacity，http://audacity.sourceforge.net/