本專案的Kaldi執行腳已完成，我們也根據執行腳的規定建立好了目錄及大部分需要的檔案。同學需要完成的部分是：

1. 錄製語料，並將語料放至指定的位置。
2. 手動建立語言數據 (transcript.txt)，並將之放至指定的位置。
3. 手動建立語者label資訊檔 speaker.info ，並將之放至指定的位置。

繳交項目：上傳wav_data整個資料 (但不包含在wav_data/wav 下的所有語料集)。

期末專案 說明

Kaldi語音辨識 — 語音點餐系統

（語音錄製參考：語音期末錄音說明.pptx）

註：Kaldi教學.pptx 所提供的run.sh執行腳本 與 這裡期末專案的run.sh執行腳本 不同。
(事實上，Kaldi提供許多不同的執行腳本。它們的執行步驟基本上都是大同小異的，
主要差異可能是：檔案的命名不同，目錄的結構不同，檔案放置的位置不同等。)

期末專案準備(1/3)

- 資料夾final_project: 是所給的期末專案內容，包含有：二個目錄 (/conf 與 /local)，以及三個檔案 (cmd.sh, path.sh 以及run.sh)。
- 資料夾wav_data: 用以放置錄製的語料集，還有放置其他需要同學修改的檔案 (transcript.txt, speaker.info) 等。

~表示/home/kaldi/

- 將期末專案 (kaldi_SpeechRecognition.zip) 內的兩個資料夾**final_project** 和 **wav_data**，都放到virtualBox的共用資料夾 ~/share 。
- 爲了避免因權限不足等未知問題所造成的限制，我們將在共用資料夾 /home/kaldi/share/ 內的/final_project資料夾之所有檔案 複製到另一個目錄下，例如： /home/kaldi/ →
  - 把資料夾**final_project** 從虛擬機的共用資料夾 ~/share， 複製到資料夾~：

```
cp -r ~/share/final_project/ ~
```

- 現在，我們所要執行之期末專案的所有腳本，都在 **~/** 路徑下的 final_project資料夾內。然而， **wav_data**資料夾不是在**~/** 路徑下，故執行腳本run.sh會找不到**wav_data**資料夾，這問題可以藉由**軟連結**的方式來解決。

期末專案準備(2/3)

~表示/home/kaldi/

- 進入複製的final_project資料夾 (~/)，對需要的檔案建立**軟連結 (symbolic link) (即捷徑)**

/home/kaldi/final_project

```
cd final_project/
ln -s ~/share/wav_data/
ln -s ~/kaldi/egs/wsj/s5/steps/
ln -s ~/kaldi/egs/wsj/s5/utils/
```

使用kaldi使用者名稱登入後，所在的根目錄爲：
kaldi@kaldi:~$ 即→ /home/kaldi/，其爲登後所在的目錄。
第一個kaldi是系統名稱
第二個kaldi是用戶登入的帳號名稱

所以，/home/kaldi/**kaldi**/ 是指：在登入時所在的目錄下，安裝**kaldi**套件後，Kaldi套件的存放位置。

將~/share/wav_data/整個目錄的資料 連結到目前所在的路徑 /home/kaldi/final_project。
另外兩個會用到的 kaldi工具目錄（steps與utils）也要軟連結進來。

期末專案準備(3/3)

- 使用腳本install_kaldi_lm.sh來安裝 kaldi的語言模型套件 kaldi_lm：

install_kaldi_lm.sh是放在 /home/kaldi/kaldi/tools/extras 路徑下

```
cd                  # 回到 /home/kaldi/ 目錄
cd kaldi/tools
extras/install_kaldi_lm.sh
source env.sh
```

→ 執行在extras目錄下的install_kaldi_lm.sh腳本

→設置環境變數

- 使用vim編輯器將heldout_sent的值修改為10 (heldout_sent=10)，可參考SRILM安裝教學裡的vim用法

```
vim kaldi_lm/train_lm.sh
```

kaldi的語言模型套件是放在 /home/kaldi/kaldi/tools/kaldi_lm 目錄下

```
. = (~/final_project/)
~ = /home/kaldi/

.
├── conf
│   ├── decode.config
│   ├── mfcc.conf
│   └── pitch.conf
├── local
│   ├── data_prep.sh
│   ├── score.sh
│   ├── prepare_dict.sh
│   └── train_lms.sh
├── cmd.sh
├── path.sh
└── run.sh
```

~表示/home/kaldi/

將 虛擬機之共用資料夾 ~/share 內的資料夾 final_project 複製到資料夾 ~ 後，目錄結構如左圖示。

對應到Kaldi教學之**語言數據 (linguistic data)的** data/local/dict/lexicon.txt檔案，需手動產生。 **待執行期末專案的**run.sh腳本後，會把lexicon.txt複製到**data/local/dict**目錄下，且自動產生語言數據的另外三個檔案到相同的**data/local/dict**目錄下。

speaker.info對應到Kaldi教學之**聲學數據 (acoustic data)** 的spk2gender 檔案。 在Kaldi教學中的run.sh腳本是將train與test的spk2gender檔案分開放置個自的目錄下：data/train/spk2gender、data/test/spk2gender。

但在這個期末專案所提供的run.sh腳本裡， train與test的spk2gender資訊 都全部放到wav_data/resource/speaker.info 內。

transcript.txt對應到Kaldi教學之聲學數據 (acoustic data)的 “text檔案”。 在Kaldi教學中的run.sh腳本是將train與test的”text檔案”分開放置個自的目錄下： data/train/text 與 data/test/text。

但在這個期末專案所提供的run.sh腳本裡， 先將train與test的”text檔案”全部放到wav_data/transcript/transcript.txt內。

待執行期末專案的run.sh腳本後，根據wav_data\wav\train 與wav_data\wav\test的目錄資訊，才將transcript.txt內train與test的 text資訊分別分開放至： data/train/**text** 、 data/test/**text**。

執行**期末專案**腳本run.sh後，會自動產生 對應到Kaldi教學之聲學數據 的其他檔案 {wav.scp, utt2spk}，且放至： data/train 與data/test 。期末專案所提供的run.sh腳本裡，不需要corpus.txt。

```
.
├── steps
├── utils
├── wav_data
│   ├── resource
│   │   ├── lexicon.txt
│   │   └── speaker.info
│   ├── transcript
│   │   └── transcript.txt
│   ├── wav
│   │   ├── train
│   │   └── test
│   └── result
│       ├── mono
│       │   └── decode.1.log
│       └── tri1
│           └── decode.1.log
├── data
├──mfcc
└── exp
    ├──mono
    │   └── decode_test
    │       └── log
    │           └── decode.1.log
    └──tri1
        └── decode_test
            └── log
                └── decode.1.log
```

紅色部分的三個資料夾是建立軟連結時，所連結到~/final_project/路徑而有的：

ln -s ~/kaldi/egs/wsj/s5/steps/

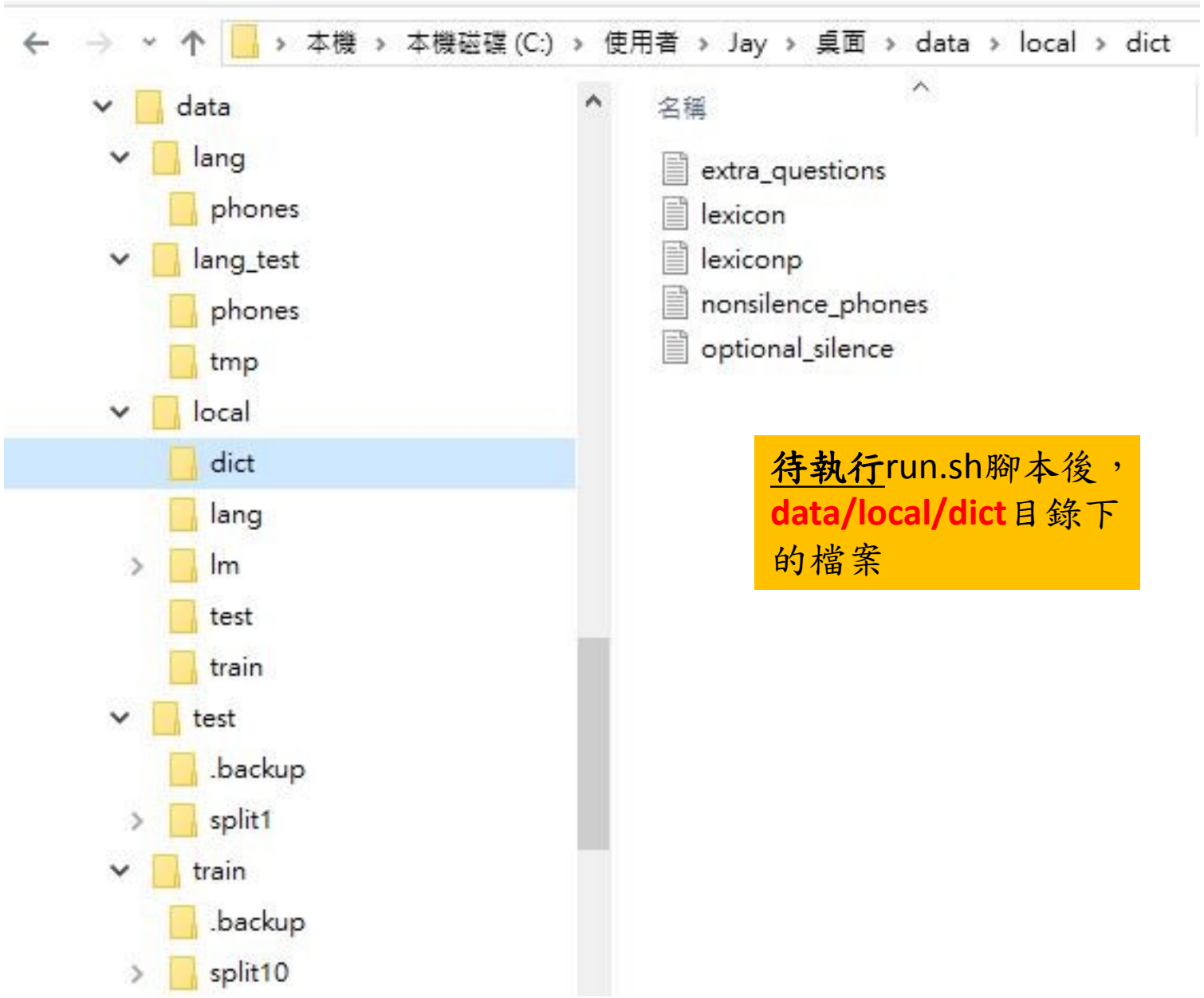
ln -s ~/kaldi/egs/wsj/s5/utils/

ln -s ~/share/wav_data/

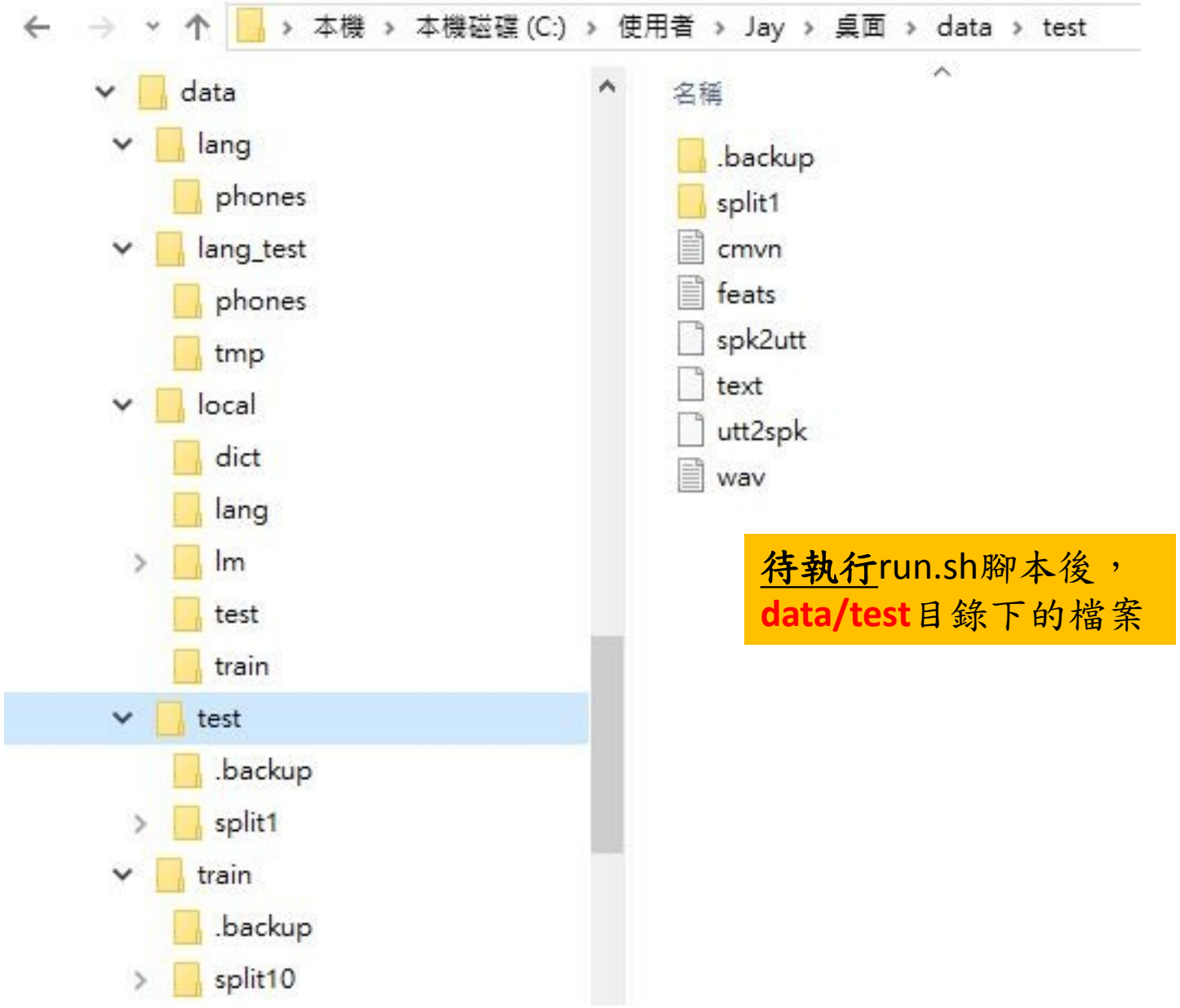
這裡 **wav_data/wav** 對應到Kaldi教學的 **waves_digits**目錄

藍色部分是執行腳本run.sh後，**自動產生的**

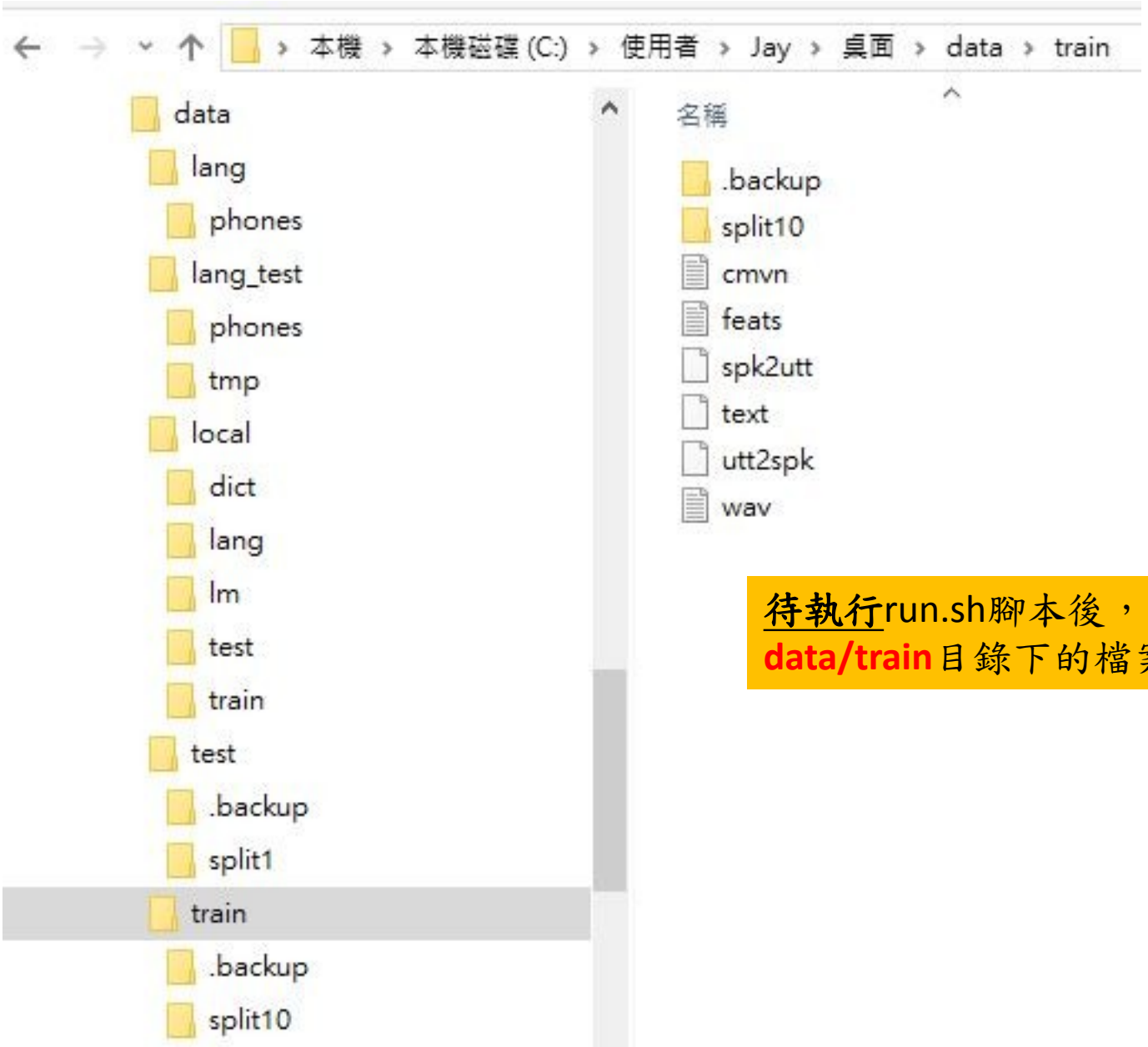
data目錄會存放聲學與語言數據 ；
mfcc目錄是存放抽取出來的語料聲學特徵。 ；
exp目錄用來存放：1. Kaldi訓練出來的語言與聲學模型 (這裡我們沒有列出其目錄)。2. 測試的輸出結果 (執行腳本run.sh也會將輸出結果複製到wav_data/result目錄下)。...



待執行run.sh腳本後，**data/local/dict**目錄下的檔案



待執行run.sh腳本後，
data/test目錄下的檔案



待執行run.sh腳本後，
data/train目錄下的檔案

本地端路徑 wav_data/**wav** (音檔目錄資訊)

- 訓練與測試用的音檔，分別放置到train與test資料夾內：
  - 在train目錄下 (wav_data/wav/train/)，建立全班個別的學號資料夾，然後放入個別的語音 (從雲端抓取)。
  - 在test目錄下(wav_data/wav/test/) ，建立自己的學號資料夾，放入**自己的語音**來做測試。
- 某個其他同學的音檔xxx.wav之存放路徑一定是這樣：
  - wav_data/wav/**train**/某個同學的學號/xxx.wav
- 自己的音檔yyy.wav之存放路徑一定是這樣：
  - wav_data/wav/**test**/某個同學的學號/yyy.wav

注意：音檔位置一定不能放錯，以免之後發生錯誤。

本地端路徑 wav_data/**resource** (「詞」與「音素」的對照表 lexicon.txt 、語者label資訊檔 speaker.info)

- 在lexicon.txt 內，我們已建立好：「詞」對應「羅馬拼音 (最小單位音素)」的對照表。
- 在speaker.info內，記錄了所有語者的label，其中我們將「學號」當作語者的label (在train與test資料夾內，有出現的學號都要填寫)。

```
1  B104170AA
2  B104170BB
3  B104170CC
4  B104170DD
5  B104170EE
6
```

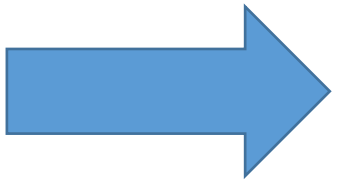
- 注意：在speaker.info內，我們省略了性別的資訊。

本地端路徑 wav_data/**transcript** (文本與檔名的對應資訊 transcript.txt)

- 在transcript.txt內，建立了 所有訓練與測試用的音檔 之每一「文本句子」對應的「錄音檔名」。文本需要做斷詞，我們已經完成之，同學只需要修改錄音檔名。
- 打開我們給你的transcript.txt檔 (含有400句文本)，利用 取代 功能將每一個同學的400句文本，取代爲其學號檔名。注意：**如果**每位同學都完成這400句的錄製，完成後的transcript.txt文件，其總行數爲：400句*(訓練與測試的總人數) 。

```
381 v2_n19_k01 外帶 十九 個 漢堡
382 v2_n19_k02 外帶 十九 個 雞翅
383 v2_n19_k03 外帶 十九 塊 雞腿
384 v2_n19_k04 外帶 十九 份 雞塊
385 v2_n19_k05 外帶 十九 份 沙拉
386 v2_n19_k06 外帶 十九 杯 可樂
387 v2_n19_k07 外帶 十九 杯 紅茶
388 v2_n19_k08 外帶 十九 包 薯條
389 v2_n19_k09 外帶 十九 包 蘋果派
390 v2_n19_k10 外帶 十九 支 冰淇淋
391 v2_n20_k01 外帶 二十 個 漢堡
392 v2_n20_k02 外帶 二十 個 雞翅
393 v2_n20_k03 外帶 二十 塊 雞腿
394 v2_n20_k04 外帶 二十 份 雞塊
395 v2_n20_k05 外帶 二十 份 沙拉
396 v2_n20_k06 外帶 二十 杯 可樂
397 v2_n20_k07 外帶 二十 杯 紅茶
398 v2_n20_k08 外帶 二十 包 薯條
399 v2_n20_k09 外帶 二十 包 蘋果派
400 v2_n20_k10 外帶 二十 支 冰淇淋
401
```

v 取代成 學號_v



```
381 M104170AA_v2_n19_k01 外帶 十九 個 漢堡
382 M104170AA_v2_n19_k02 外帶 十九 個 雞翅
383 M104170AA_v2_n19_k03 外帶 十九 塊 雞腿
384 M104170AA_v2_n19_k04 外帶 十九 份 雞塊
385 M104170AA_v2_n19_k05 外帶 十九 份 沙拉
386 M104170AA_v2_n19_k06 外帶 十九 杯 可樂
387 M104170AA_v2_n19_k07 外帶 十九 杯 紅茶
388 M104170AA_v2_n19_k08 外帶 十九 包 薯條
389 M104170AA_v2_n19_k09 外帶 十九 包 蘋果派
390 M104170AA_v2_n19_k10 外帶 十九 支 冰淇淋
391 M104170AA_v2_n20_k01 外帶 二十 個 漢堡
392 M104170AA_v2_n20_k02 外帶 二十 個 雞翅
393 M104170AA_v2_n20_k03 外帶 二十 塊 雞腿
394 M104170AA_v2_n20_k04 外帶 二十 份 雞塊
395 M104170AA_v2_n20_k05 外帶 二十 份 沙拉
396 M104170AA_v2_n20_k06 外帶 二十 杯 可樂
397 M104170AA_v2_n20_k07 外帶 二十 杯 紅茶
398 M104170AA_v2_n20_k08 外帶 二十 包 薯條
399 M104170AA_v2_n20_k09 外帶 二十 包 蘋果派
400 M104170AA_v2_n20_k10 外帶 二十 支 冰淇淋
401
```

執行腳本run.sh

- 於路徑 /home/kaldi/final_project 下，執行

  sh run.sh

- 該腳本進行 資料預處理、MFCC抽取、聲學模型與語言模型之訓練，最後會分別輸出mono和triphone聲學模型的準確率到自動建立的目錄 ~/final_project/exp/ 下 (同時在腳本中，我們也另外將結果複製到~/final_project/wav_data/result/ 下)：

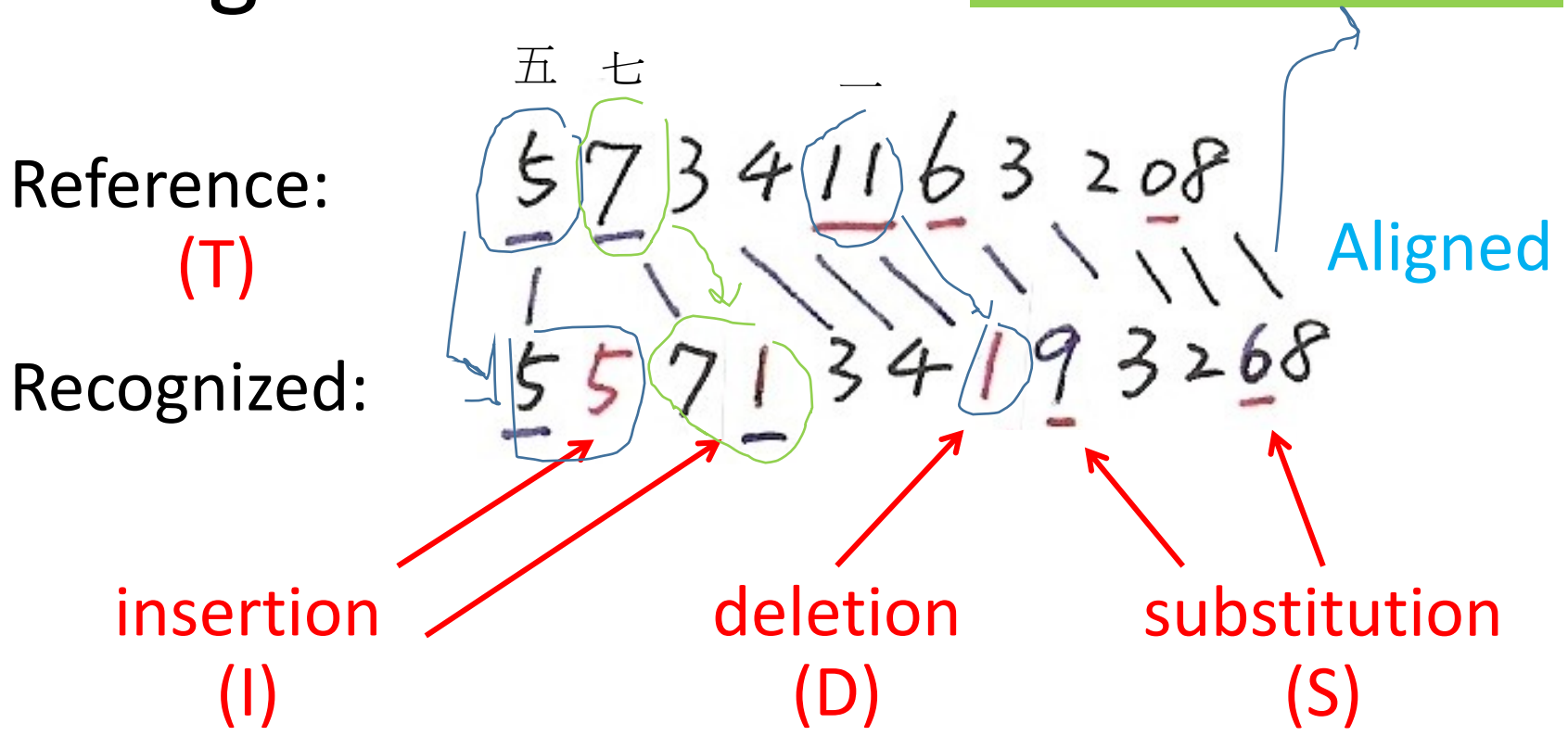
~表示/home/kaldi/

```
%WER 28.17 [ 755 / 2680, 27 ins, 173 del, 555 sub ] exp/mono/decode_test/cer_16_0.0
%WER 34.40 [ 922 / 2680, 59 ins, 482 del, 381 sub ] exp/tri1/decode_test/cer_17_0.0
```

此準確率是懂利用800句的訓練文本得到的，且發現triphone很明顯比mono-phone差，這是因為：在我們文本中的句子，基本上 字與字之間 沒有協同構音 (Coarticulation) 的問題，故此mono-phone會產生較佳的結果。

Recognition Errors

Alignment 可藉由DTW來達成



$$\frac{T - D - S - I}{T} \times 100\% = \text{Accuracy}$$

本地端路徑下 wav_data/**result** (輸出結果)

- **wav_data/**result/mono/decode.1.log 與 **wav_data/**result/tri1/decode.1.log 分別爲monophone與triphone模型的預測結果，對應之正解存放在 wav_data/transcript/transcript.txt。

```
1  # gmm-latgen-faster --max-active=7000 --b
2  # Started at Wed May 29 17:47:51 CST 2019
3  #
4  gmm-latgen-faster --max-active=7000 --bea
5  add-deltas ark:- ark:-
6  apply-cmvn --utt2spk=ark:data/test/split1
7  M104170AA_v1_n01_k01 內用 十一 個 漢堡
8  LOG (gmm-latgen-faster[5.5.313~1-203c]:De
9  M104170AA_v1_n01_k02 內用 個 雞翅
10 LOG (gmm-latgen-faster[5.5.313~1-203c]:De
11 M104170AA_v1_n01_k03 內用 塊 雞腿
```

decode.1.log 記錄“第一位測試語者”的辨識結果。同理，若有第二位測試語者，則有decode.2.log。

因爲wav_data目錄是放在共用資料夾，故當執行腳本run.sh也將輸出結果複製到wav_data/result目錄下時，同學可以直接在本地端將輸出結果讀出來。

```
41 M104170AA_v1_n01_k01 內用 一 個 漢堡
42 M104170AA_v1_n01_k02 內用 一 個 雞翅
43 M104170AA_v1_n01_k03 內用 一 塊 雞腿
```

transcript.txt 記錄原本 (正確) 的文本句子