SC Final Project Report

Singing transcription in AI-Cup competition

我們同時進行RNN和Rule-based兩種方法，在嘗試各種改進RNN的方法沒有得到好結果後，我們最後的最高分是由Rule-based產生，因此這份report會著重在Rule-based的嘗試與改進上。

RNN

一開始我們在training data上使用RNN獲得model，來進行預測，但經過調整net、超參數和optimiizer、loss function、改用更加複雜的LSTM和GRU後，即使loss逐漸下降，在比賽要求的score上仍沒有得到滿意的結果。比對預測結果和答案，發現問題在onset和offset的位置不夠準確，推測其原因可能為:

1. 我們所選的features和所求之間的關聯性不足。
2. 所使用的loss function(cross entropy和binary cross entropy)和評分使用的F1-score有落差。

Rule-based

由sample code改寫，使用onset generate決定每個note開始和結束的位置，再用中位數、平均數方法預測每個note的音高。

onset generate

onset detector

由於我們發現導致分數低的主因，是onset的位置錯誤(若開始的位置就錯(COn)，那麼剩下的COnP和OnPOff也都拿不到分數)，於是我們從onset detector上著手:

```
onset_times= librosa.onset.onset_detect
(onset_envelope= onset_strength
sr=sr,
hop_length= h_length,
backtrack= False,
units='time', pre_max= 5, post_max= 5,
pre_avg= 5, post_avg= 5, wait= 0)
```

觀察發現，答案中的pitch之間，通常都有0.05~0.2秒之間的間隔，而我們預測的pitch之間通常都沒有任何間隔，這顯然是不符合邏輯的。助教提示合理的output.json檔案大小應在28MB左右，而我們對testing data預測的結果則只有17MB左右，檢查後發現是因為pitch普遍太長，於是我們嘗試了以下改進:

1. peak_pick的參數:pre_max, post_max, pre_avg, post_avg, wait: 這幾個參數是產生detect時，reference的frame數量，pre和post分別表示向後參考和向前參考；max和avg分別表示最大能參考幾個和平均能參考幾個；wait表示產生一個pitch後要跳過幾個frame再開始detect。我們將pre_max和post_max設為7後，發現pitch普遍變得更長、檔案變得更小(13MB)了，score自然相當低，但將post_max、post_max設為5以下，其他參數也一起調整，卻始終沒有得到>20MB的檔案，score和原來設定的參數相當接近。
2. backtrack: 由於前面提到pitch普遍過長，希望透過backtrack來更精細地分割pitch。然而將backtrack設為True後，反而分割得太細，在testing data上只得到0.1455 score。

Shift

另外，我們還發現使用函式所算出來的onset，會比實際上的onset往前移一點。所以我們把出來的onset都往後移動一點，shift0.32秒得到0.275，shift0.2秒得到0.289，而這也是我們最後的最佳score。

Reverse Offset Detection

曾突發奇想，試著將音檔反轉，然後用上述相同的方法找出 onset，再將 `歌曲長度 - n-th onset`，將此結果當成歌曲的 offset，再取 onset 及 offset 之間 pitch 的中位數或眾數當作這個區間的音高，但結果不如預期，只有0.265，可能音頻的 offset 不能用此方法來找出。

pitch generate

將每個part內pitch是0的frame去掉，就變成最後的一個note，再算出這個note的pitch就結束。 我們也嘗試過許多種計算pitch的方式，使用平均、眾數、中位數，後來發現出來的結果差不多，可能是因為一個人在唱歌的時候，一個音儘管有上下起伏，但不會有非常大的變化，基本上在某個區間之內，所以算出來的值也相去不遠。

音檔優化

> https://www.google.com/search?client=ubuntu&channel=fs&q=python3+vocal+sepataion&ie=utf-8&oe=utf-8

我們也覺得音檔內的樂器伴奏可能會干擾我們的運算，為了避免伴奏的干擾，我們上網找了人聲分離的程式，先將原始音檔的人聲抽取出來，再去進行onset detection。 抽取前為0.289，抽取後得到0.181，結果沒有改進，後來詢問其他用類似方法的組別，他們得到的結果都有變好，所以應該是我們找的人聲抽取程式不夠好，並沒有很有效的將人聲分離出來。

改進方向與發現

在過程中，我們花費許多時間以及上傳次數在嘗試計算onset的函式的不同參數，但其實結果並不會比原本的好。既然要用rule based去完成報告，那重點應該放在取得onset之後要怎麼去微調，像shift這樣的行為。如果我們能發現更多資料需要微調的地方，並針對這些地方進行修改，應該可以取得更好的結果。