DSP HW3 Report

B08202033 楊智凱

What have I done

這次作業中，我試著實作了 Viterbi decoding 去解讀注音文。其中包含兩大步驟：

1. **mapping.py 之實作**

   這部分的目標是為了建立 ZhuYin-Big5 mapping。實作的方式很簡單，我先將檔案讀進來(注意 encoding)，接著建立一個字典，並以注音符號為 key，一個 list 為 value 儲存以該注音為首的漢字。最後每個漢字也有自己的 list 儲存自己以利後續寫檔案比較輕鬆。最後將儲存的東西寫進去 output file 即可。

2. **mydisambig.cpp 之實作**

   這部分我將以下列兩點分別描述：

   (1) 建立注音與其可能的 candidate 之關係：

   這裡我使用 C++ STL 中的 map 來處理。其中每個注音的 candidates 都用 C++的 vector 儲存。使用 vector 的理由是可以直接讀出其 size，由於在過程中會多次使用到 size，因此使用 vector 比較方便。

   (2) Viterbi algorithm：

   這部分是這次作業的重點。在做 dynamic programming 時，我會記錄當下最高的 log 機率值(和前一個字的各個 candidate 之 bigram log 機率值+該 candidate 往前查表的 log 機率值)。過程中，由於有不少運算會一直重複出現，因此我使用 global 的 map 儲存已經計算過的 VocabIndex，這樣遇到一個字時可以先查表看看，如果沒有算過再進行運算。

   至於 delta table 的初始化，我使用 unigram 的 log 機率值來初始化，同樣也會使用一個表格儲存已經被計算過的值。

What have I observed

做這次作業的過程中，我有幾點發現：

1. Testing data 裡面有許多注音文的注音很奇怪，例如快樂在 data 裡可能會被故意標成快ㄐ，導致 model 可能會 output 出奇怪的答案。
2. 在訓練語料中有些詞可能出現的頻率較低，但在 testing data 中卻頻繁出現，可能導致 model 判斷錯誤。例如ㄏ視，model 會很容易 output 成忽視，但在 testing data 中，華視才是正確的答案。
3. 和 disambig 相比，output 的結果雖然大致上是一樣的，可是仍有一點點差異。其中差異可能有 disambig output 正確而我的 model output 錯誤，或是反過來是我的 model 正確但 disambig 錯誤兩種。在我發現我的 output 結果和 disambig

有些差別後又回頭檢查了自己的 code，卻並未找出甚麼重大的錯誤。或許是實作演算法時有些小小的差異導致的，但由於我沒有去看 disambig 的 source code (事實上按照規定應該也是不能去看的？) 所以也無法斷言是哪些部分造成差異。

4. 按照前述，我會將已經計算過 VocabIndex 的字和結果儲存下來，按照在我自己的環境實驗的結果，這麼做可以帶來大約 2~3 秒的 speedup，代表這個運算是很花時間的。不過即使如此，和 disambig 的運算時間相比還是慢了一點，又再次暗示了可能實作的一些小細節有點差異導致運算速度的差別。當然也有可能是套件內部的 optimization 做得比較好導致的。