Digital Speech Processing Homework #3

ZhuYin Decoding

楊宗賢　B06901031

SRILM 提供的 disambig 可以指定 language model 的 order，
亦即同一個程式可以處理 bigram、trigram、4-gram…。
我撰寫的 MyDisambig 設計為 trigram decoding，order 固定是 3。

比較 SRILM 與 MyDisambig 在 11 筆測試資料上的輸出結果，
發現絕大多數的句子兩者結果完全一致，
但有一些句子兩者恰僅有前二個字不同，
其中有些句子 SRILM 較合理，有些句子 MyDisambig 較合理。

推測在求每句第三個字 $\delta_3(q_i,q_j)=\max_{q_k} P(q_i \mid q_j,q_k)\cdot\delta_2(q_j,q_k)$ 時，
較容易發生多種前二個字對應的 $\delta_2(q_j,q_k)=P(q_j \mid q_k)\cdot P(q_k)$ 相等，
因此 backtrack pointer 指到任一個 q_j,q_k 上皆正確，端看程式如何實作搜尋方法。

```
if(prob > m_prob) {
    m_prob = prob;
    m_ptr = j;
}
```

我的程式中，若搜尋到的機率**等於**之前記錄到的最大機率，pointer **不會**被取代。
因此多個字機率相等時，我的程式會輸出在 map 裡排在最前面的字。

以 python 實作的 mapping 將 Big5-ZhuYin.map 分解成行，
將每個中文字分別以**自己**為 key 及以**所有種注音的首字**為 key 存入一個 dict，
最後將 dict 存成一個 ZhuYin-Big5.map。

以 C++實作的 mydisambig 依序執行下列動作：

- 將 language model 讀入 Ngram
- 將 ZhuYin-Big5.map 讀入 unordered_map
- 讀入每行句子，將句子拆分成 vector of word
- 將每個 word 以 map 展開成 list of states，並存成 vector of VocabIndex
- 以 VocabIndex 求 log probability，據此進行 Viterbi algorithm (次頁詳述)
- 以 vector of word 及 map 將求得的 state sequence 轉換成句子
- 將句子存入檔案

Trigram decoding 的 Viterbi algorithm 流程(虛擬碼)如下：

```
for t from 1 to T
  switch (t)
    case 1:
      for i from 1 to I
        δ[1][i] = P(qi)
    case 2:
      for i from 1 to I
        for j from 1 to J
          δ[2][i,j] = P(qi|qj)*δ[1][i]
    case 3…T:
      Beam = {δ[t-1][j,k] if it is δ[t-1]中最大的前(I÷131072)名}
      for i from 1 to I
        for j from 1 to J
          δ[t][i,j] = -∞
          for k from 1 to K
            if δ[t-1][j,k] not in Beam: continue
            δ[t][i,j] = max{δ[t][i,j], P(qi|qj,qk)*δ[t-1][j,k]}
            if δ[t][i,j] is replaced: ψ[t][i,j] = k
(q[T],q[T-1]) = argmax(i,j) {δ[t][i,j]}
for t from T-2 to 1
    q[t] = ψ[t+2][i,j]
```

由於 Trigram 的迴圈範圍$O(I \times J \times K)$在三個注音連續出現時可高達 10 億，
我採用限制 beam width，即$J \times K < (131072 \div I)$的 Beam Search 來降低計算時間。
採用 Beam Search 將使 MyDisambig 的正確率些許下降，但還在可接受的範圍。

(我不採用限制 threshold 的方式是因為無法保證能將計算時間降到一定值以下。)