DSP hw3 Report

Name: 賴昭蓉, ID: B07502165
Date: 2021/12/17

Execution

- `make map` # create mapping file ZhuYin-Big5.map by given Big5-ZhuYin.map
- `make` # compile mydisambig.cpp for running Viterbi with bigram
- `make SRC=disambig_trigram.cpp` # compile mydisambig_trigram.cpp for running Viterbi with trigram
- `./mydisambig <input_seg_file> <mapping_file.map> <language_model> <output_file>` # run
- To create language model, reference to homework spec dsp_hw3.pdf

Implementation

I. Bigram

用Bigram實作Viterbi algorithm，定義變數$\delta_t(q_i)$ 如下，由下列公式推導(W_i為第i個字的random variable) [1]，因為有遞回、optimal substructure關係，使用dynamic programming方法實作。（ initialize: $\delta_1(w_i) = P(W_1 = w_i)$。）

$$\begin{aligned}\delta_t(w_i) &= \max_{W_{1:t-1}} P(W_1, ..., W_{t-1}, W_t = w_i) && (1)\\ &= \max_{W_{1:t-1}} P(w_i|W_{t-1})P(W_{1:t-1}) && (2)\\ &= \max_{q_j} P(w_i|w_j)\delta_{t-1}(w_j) && (3)\end{aligned}$$

- Result: average 15 sec for given test data, run at provided docker on CSIE Workstation. All outputs are the same as `disambig` output.

II. Trigram

用Trigram實作Viterbi algorithm，公式推導如下[1]，initialize： $\delta_1(w_i) = P(W_1 = w_i)$ ；$\delta_2(w_i, w_j) = \max_{w_k} P(w_i|w_j)\delta_1(w_j, w_k)$。

$$\begin{aligned}\delta_t(w_i, w_j) &= \max_{W_{1:t-2}} P(w_i|w_j, W_{t-2})P(W_1, ..., W_{t-2}, W_{t-1} = w_j) && (4)\\ &= \max_{w_k} \max_{W_{1:t-3}} P(w_i|w_j, w_k)P(W_1, ..., W_{t-3}, W_{t-2} = w_k, W_{t-1} = w_j) && (5)\\ &= \max_{w_k} P(w_i|w_j, w_k) \max_{W_{1:t-3}} P(W_1, ..., W_{t-3}, W_{t-2} = w_k, W_{t-1} = w_j) && (6)\\ &= \max_{w_k} P(w_i|w_j, q_k)\delta_{t-1}(w_j, w_k) && (7)\end{aligned}$$

我嘗試了以下三種方法：1. Brute force 2. Threshold 3. Beam search

1. Brute Force Method

- 一開始用dynamic programming暴力解，發現如果遇到test_data中有連續三個注音字時，因為注音有太多種可能的組合，search space太大導致程式跑很久(more than 30 min)。但對於test_data中的1~3，因為沒有遇到連續2個以上注音，每次要考慮的組合很少，所以能在一分鐘內跑完。
- Time complexity: $O(S^3T)$, where S is max size of possible observations(big5) given state(注音or big5), T is the input sequence length

2. Threshold Method

- 因為第一個方法不成功，我觀察了一下$\delta_t(q_i, q_j)$大致機率分佈，希望能設個合理threshold，讓機率小於該threshold值的candidate都直接prune掉，下一次iteration只考慮上一組trigram跑完後機率有高於threshold者。然而這個作法不太理想，因為不同詞群之間機率分佈的情形差很多，設定好的threshold，在某些case可以prune得很好，某些case則會prune掉太多。我也試過使用兩組threshold，若觀察到比較大的threshold prune完之後candidates太少，就用比較小的threshold prune的candidate，但仍十分不理想。
- Time complexity: same as brute force in worst case.

3. Beam Search Method

- 最後使用dynamic programming + priority queue 實作 beam search，width設為2000，每次對某個time frame of sequence計算所有$\delta_t(w_i, w_j)$時，只考慮$\delta_{t-1}(w_j, w_k)$ 在所有time frame = t -1 至多前2000名大的。這個方法很好，有成功在1分鐘內跑完所有test data，輸出來的文字讀起來十分通順。
- Time complexity: $O(bS^2T + S^2logS^2T) = O(S^2T(db + logS))$, where b = beam width (=2000)

Result

- using beam search method with beam width = 2000.
- average 40 sec (up to 55 sec) for given test data. All outputs are the same as `disambig` output.

III. Some Challenges and Observations

- 在寫mapping.py時，發現用”big5” encode / decode會導致某些國字難字不見，所以改用”cp950” encode / decode.

ref [1] : Notation - HackMD