Homework 3: Solutions

Junkun's Notes Revised from a Previous Rubric
CS 539

1 Part-of-Speech Tagging

1. Based on the following dymanic program, we output a best POS, given a sentence input.

$$best[i, t_j] = \max_{t_k \in P(t_j)} best[i-1, t_k] * p(t_j|t_k) * p(w_i|t_j)$$

   Where $P(t_j)$ is the previous possible tags of t_j. The base case is $best[0,' < s >'] = 1.0$.

2. We can easily extend the algorithm to trigram.

$$best[i, t_j] = \max_{t_{k_1} \in P(t_j) t_{k_2} \in P(t_{k_1})} best[i-1, t_{k_1}, t_{k_2}] * p(t_j|t_{k_1}, t_{k_2}) * p(w_i|t_j)$$

2 Decoding Katakana to English Phonemes

1. Decoding algorithm

   This part is to decode Japanese (Katakana) to English phonemes using Viterbi algorithm and chose the best solution. The idea is using a tag trigram model to represent the relationship among English phonemes. Since each English phoneme corresponds to 1-3 Japanese phonemes, we have to choose the most probable English phoneme sequence using Viterbi based on the combination of the current jprons with at most two previous jprons.

   Suppose we are decoding English phoneme from current Katakana phoneme j_i and at most two previous phonemes j_{i-1} and j_{i-2}. Using Viterbi algorithm, the cost of the shortest path ending with phonemes j_k (where $k \in \{i-2, i-1, i\}$) getting assigned the English phoneme e. Let e' and e'' be the phonemes before e. We have:

$$best[i][e', e] = \max_{e''} \max_{1 \le k \le 3} best[i-k][e', e'] * P(e|e'', e') * P(j_{i-k+1}, ..., j_i|e)$$

   To find the path that generates the smallest cost, we use the following formula:

$$back[i][e', e] = arg \max_{e'', 1 \le k \le 3} best[i-k][e', e'] * P(e|e'', e') * P(j_{i-k+1}, ..., j_i|e)$$

   The time complexity is $O(nT^3)$ and space complexity is $O(nT^2)$.

```
$ echo -e 'P I A N O\nN A I T O' | python decode.py epron.probs epron-jpron.probs

P IY AA N OW # 1.489806e-08
N AY T # 8.824983e-06
```

3 K-Best Output

1. extend our Viterbi algorithm to output k-best sequences by using a heap. Now the definition of the subproblem changed, $best[i, e', e]$ contains K-best solution for katakana phoneme sequence at i is tagged with english phonemes e, where the previous phoneme is e'. and the recursion becomes:

$$best[i][e', e] = Topk_{e'', 1 \leq k \leq 3} best[i-k][e', e'] * P(e|e'', e') * P(j_{i-k+1}, ..., j_i|e)$$

   By implementing K max, we use heap, such that we first push all possible 1-best solutions to generate current (i, e', e), and we pop the top one at a time. Whenever we pop one element, we have to push an additional element from the next value in the previous popped solution list.

```
$ cat jprons.txt | python decode_kbest.py 5 epron.probs epron-jpron.probs
HH IH R AH L IH K L IH NG T AH N # 2.443851e-17
HH IH R AH L IH K L IH N T AH N # 1.946566e-17
HH IH L AE R IH K L IH NG T AH N # 1.698078e-17
F IH L AE R IH K L IH NG T AH N # 1.407611e-17
HH IH L AE R IH K L IH N T AH N # 1.352546e-17
D N AH L D T R AE M P # 2.177145e-19
D N AH L D T R AH M P # 9.417660e-20
D N AH L D AH T R AE M P # 7.483929e-20
D AA N AH L D T R AE M P # 5.441892e-20
D OW N AH L D T R AE M P # 5.324423e-20
V IH D IY OW T EY P # 3.534682e-16
B IH D IY OW T EY P # 3.389997e-16
V IH D IY AH T EY P # 2.670477e-16
B IH D IY AH T EY P # 2.561167e-16
B IY D IY OW T EY P # 1.727004e-16
HH OW M ER SH IH M P S AH N # 2.127905e-17
HH AA M ER SH IH M P S AH N # 5.076107e-18
HH AH M ER SH IH M P S AH N # 3.591805e-18
HH OW M ER S IH M P S AH N # 3.436763e-18
F OW M ER SH IH M P S AH N # 2.655452e-18
R AE P T AA P # 4.498555e-12
L AE P T AA P # 3.563637e-12
R AH P T AA P # 2.742879e-12
R AE P T OW P # 2.436263e-12
L AE P T OW P # 1.929944e-12
```