Reconstruct Hw3

SUNId: Benma

1a.

greedy algorithm:

Begin at the front of the string. Find the ending position for the next word that minimizes the language model cost. Repeat, beginning at the end of this chosen segment.

For example, input “be my best friend forever”

Space removed “bemybestfriendforever”

A* cost (Optimal):

be 1, my 2, best 4, friend 6, forever 7

cost = 1+2+4+6+7 = 20

greedy algorithm (Suboptimal):

be 1, my 2, be 2, et 3, friend 6, for 3, ever 5

cost = 1+2+2+3+6+3+5 = 22

Therefore, greedy algorithm will choose shorter words for less weighted, so “be” could be chosen instead of “best” and “for” could be chosen instead of “forever” because “be” and “for” is much more common than “best” and “forever”. Eventually greedy have the bigger cost than the optimal algorithm in above example, thus greedy algorithm is a suboptimal algorithm.

2a.

For example “My good got damaged”

“M gd gt dmgd”

My god 3, get damaged 8

Greedy algorithm Cost = 3 + 8 = 11

Greedy algorithm will give “god” instead of “good” as the cost is lower by giving one “o” and give “get” instead of “got” because it deems the cost is lower. However “My god get damaged” is distorted, and that’s why it is suboptimal.

Optimal algorithm Cost: My good 4, got damaged 6 = 4+6 = 10

Therefore greedy algorithm is suboptimal because a low cost bi-grams is assigned with total high cost for greedy bigrams.

3a.

States:

Previous word and current index that space or vowels inserted

Actions:

Choose to insert space or vowels between letters

Costs:

Bigram cost for previous word and next word

Initial state:

Index starts with the 0 or '-BEGIN-'

End test:

Index reaches the last word len(query) or len(wordquery)

3c.

To speed up the joint space and vowel insertion using A*, we will need to have a good estimation of the successor with heuristic and we need a good approximation of future cost.

{cost'} = b(w',w) + u(w) => $U_b(w)$ = Min{ b(w', w)} => $U_b(w)$ <= b(w',w)

State:

Index of insertion

Cost: $U_b(w)$

Start state:

0 or '-BEGIN-'

End state:

Reach the end of the len(query)

Consistency proof:

The first inequality follows because h(s) = Future $Cost_{rel}(s,a)$ and all future costs correspond to the minimum cost paths, taking action from state s better be no better than taking the best action from state s.

Since b(w', w)>=0, then U_b >= 0 because the heuristic of the end state is 0 So the minimum cost from the end state to the end state is trivially 0, so just use 0 -- no future cost associated, therefore h(s) is consistent.

3d.

Is UCS a special case of A*? Explain why or why not.

Yes, UCS is a special case of A*, because A* search essentially runs UCS with modified edge costs, intuitively, A* is a bias of UCS towards exploring states which are closer to the end state.

Is BFS a special case of UCS? Explain why or why not.

No, UCS is to compute the past costs in order of increasing past cost. BFS is to search all the paths consisting of one edge, two edges, three edges, etc., until it finds a path that reaches an end state. BFS and UCS are essentially different.