CS221 Autumn 2016 Homework [3]

SUNet ID: [06033811]
Name: [Chao Xu]

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my original work.

Problem 1: Word Segmentation

In word segmentation, you are given as input a string of alphabetical characters ([a-z]) without whitespace, and your goal is to insert spaces into this string such that the result is the most fluent according to the language model.

(a) Consider the following 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.

Show that this greedy search is suboptimal. In particular, provide an example input string on which the greedy approach would fail to find the lowest-cost segmentation of the input.

In creating this example, you are free to design the n-gram cost function (both the choice of n and the cost of any n-gram sequences) but costs must be positive and lower cost should indicate better fluency. Your example should be based on a realistic English word sequence don't simply use abstract symbols with designated costs.

Solution: I can construct an example as following:

Visualizingandunderstandingconvolutionalnetworks

and assume we have the following cost set {visual: 5, izing: 100, izinga: 150 visuali: 100, and: 5, understand: 6, understanding: 5, ing: 100, convolution: 5, convolutional: 5, convolutiona: 100, networks: 5, net: 4, work: 5, netw: 100, works: 10}
For this example, the greedy algorithm would pick **visual** rather than **visualization** since the weights of **visual** is only 5 and **visuali** is 100. Then the algorithm will pick **izing** rather than **izinga** since the former one has lower weights. However, this will lead to false word segmentation: **visual izing and** using the greedy algorithm

Problem 2: Vowel Insertion

Now you are given a sequence of English words with their vowels missing (A, E, I, O, and U; never Y). Your task is to place vowels back into these words in a way that maximizes sentence fluency (i.e., that minimizes sentence cost). For this task, you will use a bigram cost function. You are also given a mapping possibleFills that maps any vowel-free word to a set of possible reconstructions (complete words).[6] For example, possibleFills('fg') returns set(['fugue', 'fog']).

(a) Consider the following greedy-algorithm: from left to right, repeatedly pick the immediate-best vowel insertion for current vowel-free word given the insertion that was chosen for the previous vowel-free word. This algorithm does not take into account future insertions beyond the current word.

Show, as in question 1, that this greedy algorithm is suboptimal, by providing a realistic counter-example using English text. Make any assumptions you'd like about possibleFills and the bigram cost function, but bigram costs must remain positive.

Solution: Consider the following example:

ths ppl hs nc bg

with the following bigram cost: {'these people': 6, 'this apple': 5, 'people have': 6, 'apple has': 20, 'has nice': 10, 'have nice': 4, 'nice bag': 10} Using greedy algorithm, the program will pick **this apple** rather than **these people** and then it has to pick **apple has**, **has nice** and **nice bag**. The total cost would be 45 using greedy algorithm rather **these people have nice bag** with cost of 22.

Problem 3: Putting It Together

We'll now see that it's possible to solve both of these tasks at once. This time, you are given a whitespace- and vowel-free string of alphabetical characters. Your goal is to insert spaces and vowels into this string such that the result is the most fluent possible one. As in the previous task, costs are based on a bigram cost function.

(a) Consider a search problem for finding the optimal space and vowel insertions. Formalize the problem as a search problem, that is, what are the states, actions, costs, initial state, and end test? Try to find a minimal representation of the states.

Solution: States: the index of character in the string, and the rest part of the string
Actions: A word picked after space and vowel insertion.
Costs: Bigram score of two sequential words.
Initial State: 0, begin of the string.
End test: if the search reach the end of the string.

(b) Let's find a way to speed up joint space and vowel insertion with A*. Recall that having to score an output using a bigram model b(w,w) is more expensive than using a unigram model u(w) because we have to remember the previous word w in the state. Given the bigram model b (a function that takes any (w,w) and returns a number), define a unigram model ub (a function that takes any w and returns a number) based on b. Now define a heuristic h based on solving a simpler minimum cost path problem with ub, and prove that h is consistent.
To show that h is consistent, construct a relaxed search problem. Recall that a search problem P with cost function Cost(s,a) is a relaxation of a search problem P with cost function Cost(s,a) if they have the same states and actions and Cost(s,a)Cost(s,a) for all states s and actions a. Explicitly define the states, actions, cost, start state, and end test of the relaxation.
Solution: Define the heuristics h(w) = $u(w) = min_i b(w_i, w)$. As the minimum function defined in heuristics, the future cost would be $\sum u(w)$ which is less than the actual cost $\sum b(w, w)$. Also, the heuristics presents the minimum cost for choosing a word as successor which is less than the original cost $b(w\prime, w)$