CS221 Fall 2014 Homework 3

SUNet ID: rolfom01
Name: Mathieu Rolfo
Collaborators: None

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

Problem 1

(a) Consider the following greedy algorithm for space insertion: 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.
An example that shows this algorithm is suboptimal is the string ”Onomatopoeiaisrare.” Because the word ”on” is far more common than ”onomatopoeia,” the algorithm would return a smaller cost for ”on” and insert a space directly after it. However, this would leave the string ”omatopoeiaisrare,” which contains no prefixes that are words in the English language. This means the second space insertion will have a very high cost, and the algorithm is unable to find the minimum cost solution ”Onomatopoeia is rare.”

Problem 2

(a) Consider the following greedy algorithm for vowel insertion: from left to right, repeatedly pick the immediate-best (according to bigram cost) vowel insertion for the current vowel-free word given the insertion for the previous (i.e., not taking into account future insertions).
An example that shows this algorithm is suboptimal is the vowel-free word ”y y cptn.” The best reconstruction is clearly ”Aye aye captain.” However, the algorithm will select the first word to either be ”you” or ”eye” both of which are more common than ”aye” as the first word in a sentence (probably ”you”). However, this means that the second word has very limited reconstruction options, none of which are particularly favorable, and so the optimal reconstruction will not be selected.

Problem 3

(a) The characteristics of the vowel and space insertion problem are:
State = The state consists of the previous word formed and the current index of the input query string.
Actions(s) = Each action consists of two subactions. You select the next index value,

which determines your space insertion and creates a new query word. Then, you set each candidate vowel reconstruction of that query word as the new previous word formed.
Cost(s, a) = The cost of the action is the bigram cost of the original previous word and the candidate from possibleFills that is the next previous word.
S_{start} = (wordsegUtil.SENTENCE_BEGIN, 0)
isGoal(s) = You know if you've reached the goal state if your index is equal to the length of the input query string..

(b) ——————

(c) Let's define $u(w) = \min_{w' \in W} b(w', w)$, where W is the set of all words in the corpus. This can be calculated in a pre-processing step outside of the UCS. In other words, u(w) is the smallest bigram cost that exists for the tuple (precedingWord, w) where precedingWord is some word in the corpus. In this problem, our state = (previousWord, indexInQuery). Our action is to take some prefix of the query, and get some possible vowel reconstruction candidateWord from possibleFills. u(candidateWord) is strictly less than or equal to b(previousWord, candidateWord), because u(candidateWord) will select the smallest value of the bigram cost in the set. Therefore, our estimate of the cost, Cost'(s, a) = u(candidateWord) is strictly less than or equal to the value of Cost(s, a). Define $h_u(s)$ = FutureCost'(s, a), using Cost'(s, a). Because Cost'(s, a) $\leq$ Cost(s, a) for each step along the path, FutureCost'(s, a) $\leq$ FutureCost(s, a). This means we have a relaxation of our search problem P. By the theorem of consistency of relaxed heuristics, $h_u(s)$ is a consistent heuristic.