NLTK Outstanding Work

Chapter 2:

25. My implementation of find\_language() is included below. As per the exercise specifications, it limits its searches to Latin-1 encoded languages in the udhr corpus. I included an additional step of standardizing word case to avoid any possible errors due to capitalization.

def find\_language(word):

from nltk.corpus import udhr

ret = []

for l in udhr.fileids():

if l[-7:] == "-Latin1":

if word.lower() in udhr.words(l):

ret.append(l)

return ret

Chapter 3:

39. The code below is my implementation of the “non-SQL” Soundex algorithm. My method gives correct codes for all the example words on the algorithm Wikipedia page (i.e. “Robert,” “Rupert,” “Rubin,” “Ashcraft,” “Ashcroft,” “Tymczak,” “Pfister,” and “Honeyman”).

def soundex(w):

word = w.lower()

code = word[0]

word\_list = list(word)

consonants = {"b":1, "f": 1, "p":1, "v":1, "c":2, "g":2, "j":2, "k":2, "q":2, "s":2, "x":2, "z":2, "d":3, "t":3, "l":4, "m":5, "n":5, "r":6}

for let in range(1, len(word)):

if word\_list[let] in consonants:

word\_list[let] = str(consonants[word\_list[let]])

for let in range(1, len(word)):

if len(code) == 4:

break

if let == 1 and word[0] in consonants and str(consonants[word[0]]) == word\_list[let]:

continue

if let > 1:

if word\_list[let] == word\_list[let-1]:

continue

if let > 2:

if word\_list[let] == word\_list[let-2] and (word\_list[let-1] == "h" or word\_list[let-1] == "w"):

continue

if word\_list[let].isdigit():

code += word\_list[let]

if len(code) < 4:

code += "0"

return code.upper()

41. I rewrote the nested for-loops and if-statement as a list comprehension within a set comprehension.

vsequences = {"".join(ch for ch in word if ch in "aeiou") for word in words}

Chapter 4:

35. Included below along with some sample output, my word-square solver uses simple backtracking and the words in the NLTK Corpus. Its algorithm and program structure are very similar to those I wrote for the first (i.e. unoptimized and lacking in artificial intelligence) parts of Sudoku and Crosswords. As expected of a basic brute-force algorithm, its runtime increases exponentially. The largest n value that it solves in under 60 seconds is 6.

As a side note, my code is deterministic and only finds one possible solution for any given n. Due to its strictly alphabetic processing of words during initial lookup generation, the solutions it generates typically include several words beginning with the first few letters of the alphabet.

from nltk.corpus import words

def coor\_to\_index(row, col, n):

return int(col+n\*row)

def index\_to\_coor(index, n):

return int(index/n), int(index%n)

def all\_n\_words(n):

all\_words = words.words()

n\_words = list()

for w in all\_words:

if len(w) == n and w.lower() not in n\_words:

n\_words.append(w.lower())

return n\_words

def find\_poss(d, blank):

comp = "".join(["-" for k in range(len(blank))])

if blank == comp:

return d

ret = []

for w in d:

isMatch = True

for i in range(len(blank)):

if blank[i] == "-" or blank[i] != w[i]:

if blank[i] != "-":

isMatch = False

break

if isMatch:

ret.append(w)

return ret

def fill\_blank(board, blank\_num, word):

b\_list = list(board)

for i in range(0, len(word)-blank\_num):

row\_i = coor\_to\_index(blank\_num+i, blank\_num, len(word))

col\_i = coor\_to\_index(blank\_num, blank\_num+i, len(word))

b\_list[row\_i] = word[i+blank\_num]

b\_list[col\_i] = word[i+blank\_num]

return "".join(b\_list)

def backtrack(board, d):

if "-" not in board:

return board

n = int(len(board)\*\*0.5)

num = index\_to\_coor(board.index("-"), n)[0]

poss = find\_poss(d, board[num\*n:(num+1)\*n])

for w in poss:

new\_board = fill\_blank(board, num, w)

result = backtrack(new\_board, d)

if result is not None:

return result

return None

|  |  |  |
| --- | --- | --- |
| **n** | **Solution** | **Time Required (s)** |
| 1 | a | 2.810000000008639e-05 |
| 2 | aa  aa | 0.00011799999999984045 |
| 3 | aal  aba  lab | 0.0021567000000000114 |
| 4 | aani  abac  nace  iced | 0.018339399999999895 |
| 5 | aalii  abash  lazar  isawa  ihram | 5.137635100000001 |
| 6 | ababua  babery  abidal  bedene  uranus  ayless | 7.677439100000001 |